Reference architecture for distributed systems management

by M. A. Bauer
P. J. Finnigan
J. W. Hong
J. A. Rolia
T. J. Teorey
G. A. Winters

Management of computing systems is needed to ensure efficient use of resources and provide reliable and timely service to users. Distributed systems are much more difficult to manage because of their size and complexity, and they require a new approach. A reference architecture for distributed systems management is proposed that integrates system monitoring, information management, and system modeling techniques. Three classes of system management—network services and devices, operating system services and resources, and user applications—are defined within this framework, and a detailed hospital application is presented to clarify the requirements for managing applications. It is argued that the performance management of distributed applications must be considered from all three perspectives. Several management prototypes under study within the COnsortium for Research on Distributed Systems (CORDS) are described to illustrate how such an architecture could be realized.

Distributed computing systems typically consist of large numbers of heterogeneous computing devices connected by communication networks, various operating system resources and services, and user applications running on them. These resources and applications are becoming indispensable to many enterprises, but as distributed systems get larger and more complex, more things can go wrong, potentially interrupting or crippling critical operations. Thus, management support is often cited by end users as the single most important aspect required in such a system. Unavailability of services, incorrect services, or inefficient operations of services and applications could mean the loss of valuable customers and revenues.

While some monitoring of systems is conducted today for computing environments, an integrated approach to system management is needed for more sophisticated distributed computing environments. As an example of such an approach, let us hypothesize the actions of a distributed systems manager of the future who must maintain acceptable performance of a complex system as the set of active users and applications rapidly change.

In our proposed scenario the distributed system is running smoothly when suddenly user application response times increase to unacceptable levels. The manager initiates a control program to start monitoring system resources and to store the data in local information bases. The control program

collects packet (workload) data associated with parts of a network, node resource utilizations, and loads on specific processes. The information bases are queried by the control program and aggregate data are sent to a modeling tool that simulates the actual system configuration. The model is then executed with the live workload data, and bottlenecks detected by the model are validated using the console information the systems manager originally gathered.

At this point the manager is confident that the model predicts performance accurately for the current system load and uses the model to reconfigure resources in various ways to test hypotheses for improving system performance. The model results show that some reconfigurations cause further performance degradation, and those options are discarded. Other configurations do result in better performance, but the cost of implementing them is excessive. Finally, a balance between cost and performance is reached and the manager makes the changes dynamically. Typical changes involve redistributing workloads, processes, or devices, inserting more devices in a particular part of the configuration, or purchasing faster devices.

The advantage of this type of management system is that reconfiguration can be done with extreme confidence that the performance of the system will improve significantly and service to the user will return to normal quickly. Thus the integration of network monitoring, system monitoring, application monitoring, information management, and modeling techniques can potentially lead to improved distributed system reliability and performance. This is the major theme of this paper.

The management of distributed systems needs to involve not only managing network services and devices, but also system services and resources and user applications. Most of the research and implemented systems to date have focused on network management, but in order to provide true manageability for users, all three classes of distributed systems management must eventually be included. We examine the functional requirements of managing distributed systems and propose a reference management architecture that satisfies those requirements.

The paper is organized as follows. A typical distributed application—a hospital information system used to illustrate the requirements for many applications—is first described. Major functional requirements for managing distributed systems are summarized, along with the proposal for an integrated management architecture to satisfy those requirements. The hospital example is revisited and it is shown how the management of this environment fits into our proposed architecture. Finally, several prototype management subsystems are discussed, along with how the functional integration works in practice.

The prototypes under study are part of research within the COnsortium for Research on Distributed Systems (CORDS). The CORDS project brings together researchers from four IBM research laboratories, six Canadian universities, four United States universities, and other international research centres.

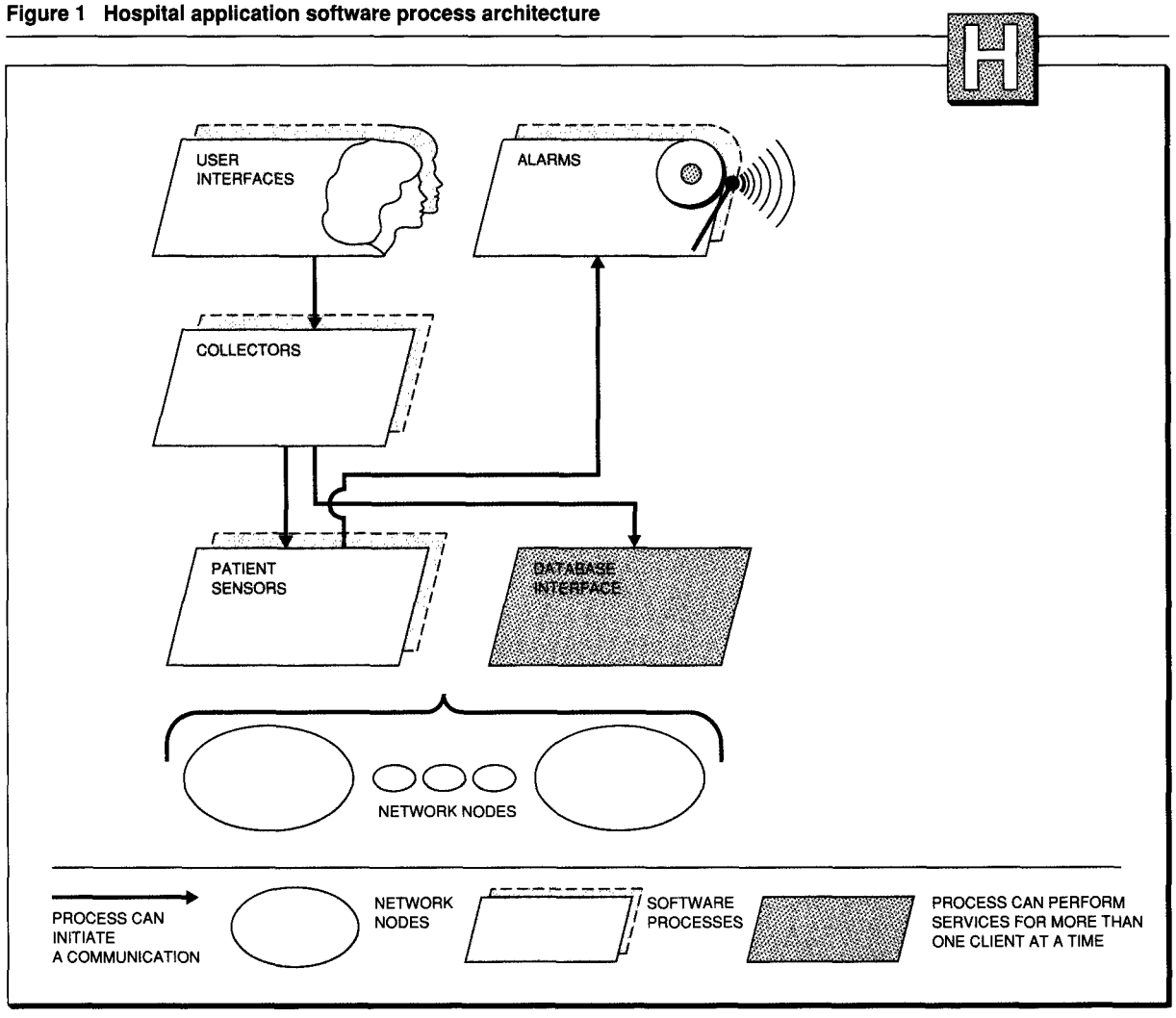
Distributed application example

In this section we introduce a simple distributed application and discuss its performance requirements and deployment on a local area network. Several issues are described that must be addressed by a systems manager when deploying or managing the performance of distributed applications. These include the allocation of processes to network nodes and, when necessary, the choosing of an appropriate level of internal concurrency for processes. The example helps to provide a rationale for the management requirements discussed in the next section. Later we show how the proposed management architecture interacts with the application and how management tools could be used to analyze the performance of the application, host, and network.

Consider the example of a distributed application used to support the real-time monitoring of critical care patients' vital signs in a hospital.[1] Medical sensors measure each patient's temperature, blood pressure, and skin tension. If any readings fall outside of a safe range, an alarm is displayed at the patient's nurse station. Summaries of sensor measurements are saved in a database and nurses and doctors can later query the database or append further patient status information.

The distributed system that supports the application has the following performance requirements:

Figure 1 Hospital application software process architecture



- The system must be capable of supporting up to one hundred patients and summarize each patient's sensor readings in the database every minute.
- If a sensor value falls outside its safe range, the nurse station must display an alarm quickly. Ninety percent of the alarms must be displayed within three seconds, and 99 percent within five seconds.
- No more than 60 percent of the host database processing power must be used by the application.
- The load on the network from the application must not exceed 30 percent of the capacity of the network.

In general, a distributed system can have many performance requirements. Each requirement applies best within the application, host, or network subsystem domain. Application requirements span the network and are monitored at the application level. Host requirements are most easily monitored at the operating system level, whereas network subsystem requirements pertain to the actual network.

A software process architecture that can be used to implement the system is shown in Figure 1. It illustrates several performance features of distributed applications using a notation based on Buhr's machine charts.[2] Each parallelogram rep-

resents a software process. Overlapped parallelograms indicate that the application may include more than one instance of a process. The lines with arrowheads show that a process can initiate a communication with another process. The shaded parallelograms indicate that the process can perform services for more than one client at a time. Ellipses are used to illustrate the presence of network nodes. To deploy the application, we must allocate each process to a specific network node. We now describe the purpose of the processes in the architecture and the network nodes required to support them.

In this example, nurse station user-interface processes are implemented on workstations with one workstation per nurse station. Each workstation is a network node. The processes provide operations, or services, that support the request and update of (1) patient information, (2) requests to allocate and deallocate medical sensors and their collectors to patients, and (3) start and stop commands to the collector processes that facilitate patient monitoring. Each nursing workstation also has an alarm process that manages a status window used to indicate whether the sensor values for the patient are within an acceptable range.

The database is configured to reside on a specific host in the network and the database process accepts requests to store and retrieve sensor data and other patient information. The process permits many requests for service to be active concurrently. Controlling the number of clients that can be served concurrently affects the load on the resources for the process, and the delays for service incurred by the clients of the process.

Patient sensor processes reside on special-purpose processors close to their sensors. These processors are connected to the network and are network nodes. Each sensor process supports a single sensor by obtaining data from the sensor at regular intervals, converting the results into appropriate measurement units, and maintaining a buffer of recent measurements. If measurements are outside of an acceptable range, the process notifies the appropriate nurse workstation alarm process.

Control processes start, stop, and poll a single patient's sensor processes, summarize the sensor data, and periodically write a summary to the database. Nurse station user-interface processes issue commands to collector processes. They cause the collector processes to be associated with sensors, and to start and stop patient monitoring. The collector processes are target-independent and can execute on either nurse workstations or on the database host.

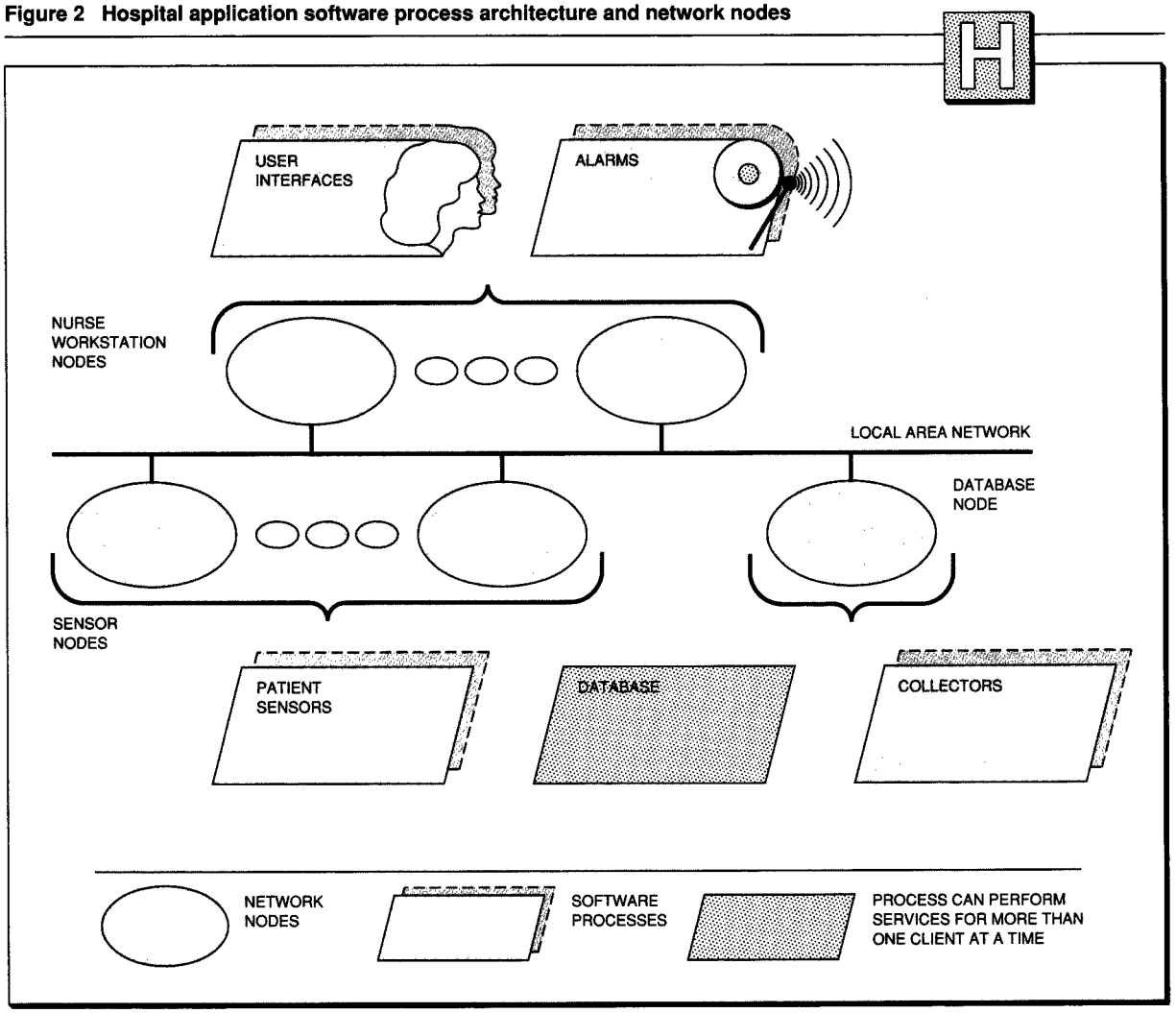
Before the application can be deployed, the collector processes must be assigned to nodes and the level of concurrency within the database process must be decided. For now, we associate the collector processes with the database host and show the allocation of processes for the deployed application across network nodes in Figure 2. Alternate placements are discussed in the later section "prototype implementations." The choices affect contention for application resources, such as processes and their internal elements of concurrency, node operating system resources, such as processors and input-output subsystems, and network communication subsystems. The interactions lead to complex performance behaviour that often defies intuition. Management systems and applications are needed to help improve our ability to understand and control the behaviour of distributed systems.

A distributed computing management system must characterize the performance behaviour of distributed system features separately and also provide a means to correlate their interactions. The features include distributed application components such as processes, nodes, and network subsystems. Since performance requirements can be expressed for each of these features separately, the management system should help us verify whether their respective requirements are indeed satisfied. Performance managers may also attempt to control or improve the system performance. When this is the case, it can be helpful to correlate management information. For example, we may wish to correlate network load with application processes so that it is possible to consider the impact of alternative process placements on network performance.

Reference architecture for distributed systems management

As discussed in the previous section, it is our view that the primary objective of distributed systems management is to ensure the required behaviour of distributed applications; since it is the

Figure 2 Hospital application software process architecture and network nodes



applications that are critical to the end users of the organization. This is not to say that the management of networks and systems is not important, rather it implies that their management is a critical factor in the overall management objective. Clearly, a malfunctioning network, whether providing poor performance or many faults, results in poor behaviour of distributed applications operating over that network.

From the perspective of managing the performance of a distributed application, it is evident that information about the behaviour of the underlying systems and networks is essential. It is our premise that management of networks, systems, and applications in isolation is too narrow a focus, and that an integrated approach is required. Our research has begun to explore this idea and we present a reference architecture for integrated management services. Before doing so, however, we briefly review what we consider are the key requirements for distributed systems management.

Requirements for distributed systems management. In order to provide an effective distributed systems management environment, management services that can satisfy the following requirements are needed.

- It must be possible to collect data about the behaviour and performance of devices, networks, systems, and applications throughout the distributed computing environment.
- It must be possible for a system administrator to statically and dynamically specify which components are to be monitored, what data should be collected, and how those data are subsequently analyzed or stored.
- There must be facilities to enable an administrator to control the behaviour of components based on the collected and analyzed information; such facilities might involve direct intervention by the administrator or may be predefined management components specified by the administrator to act in an automatic or semiautomatic manner.
- The management tools and services must support multiple, perhaps independent, administrators and yet provide consistent information, e.g., the definition of managed object attributes, across the entire distributed computing environment.
- The overhead of the management services (such as storage of collected data, management protocol traffic, processor load for monitoring) should be kept as minimal as possible. Redundant data collection, storage, processing, etc., should be avoided.

The above key requirements also suggest several secondary ones:

- There should be support for the development and operation of management applications. For example, if it is necessary to access multiple sources of information for analysis, the underlying management services should facilitate it.
- It will be necessary to maintain information on the location, origin, interconnection, and dependencies among various entities in the computing environment. For example, it will be necessary to know the processes constituting a particular distributed application, the hosts on which those processes execute, and perhaps the communication dependencies among those processes.
- Distributed support for the collection, storage, and access of data will be required. Since data collected could be distributed across numerous sites, and may itself be dynamically allocated, information about the sources of management data must be maintained.

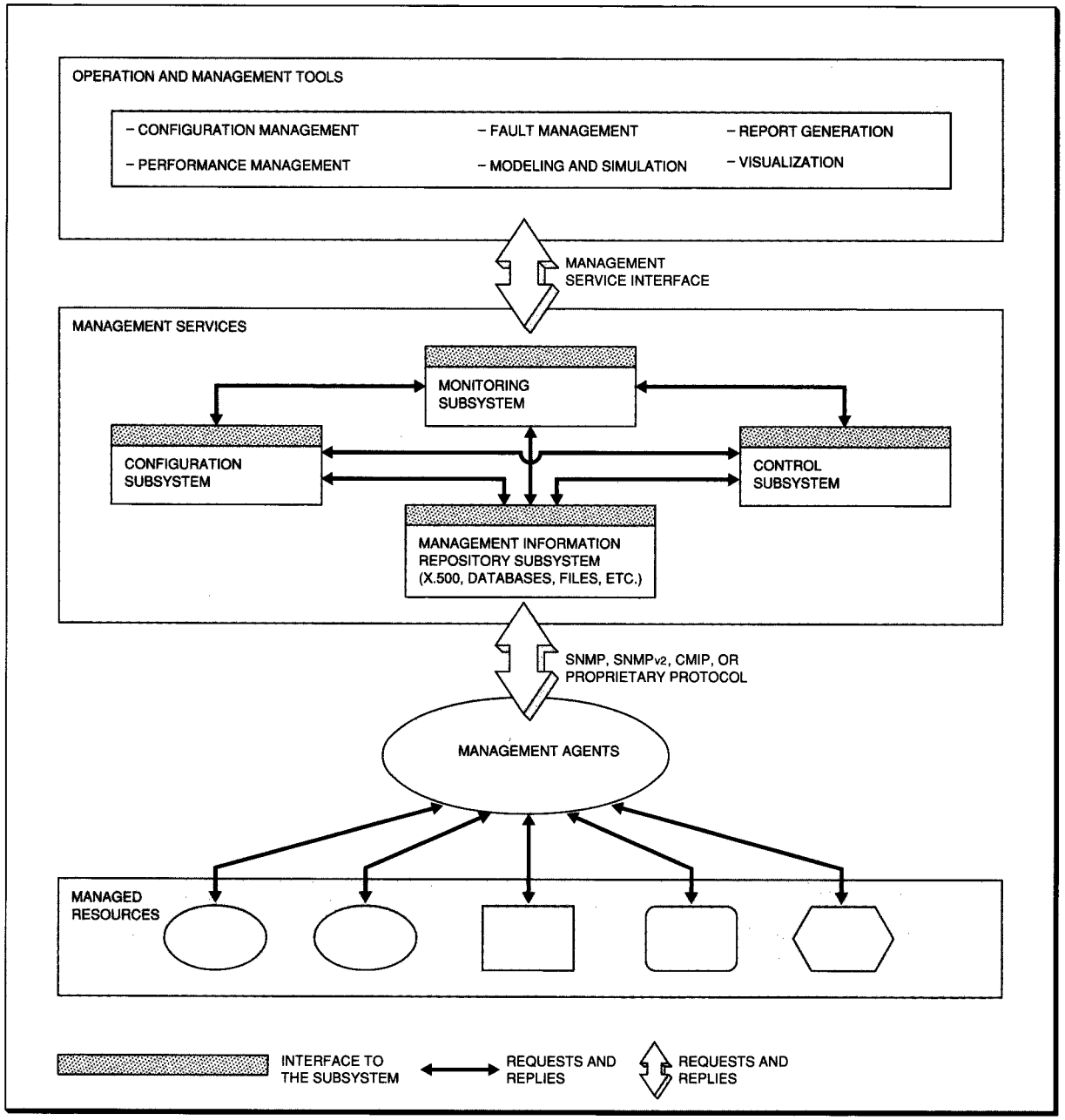
These requirements are key whether one is focusing specifically on network management or on the management of distributed applications. In either case, an integrated approach for providing management services can be beneficial, resulting in simpler interfaces, consistent data, elimination of redundancy, and greater flexibility in accommodating new standards and tools. Of course, these gains become more significant within the context of distributed applications management because of the increased complexity and numerous potential sources of information.

Integrated management architecture. We propose an integrated management architecture that accommodates the requirements discussed above and supports existing and emerging management standards. The architecture (illustrated in Figure 3) decomposes the collection of necessary management services into a logical collection of subsystems and components. It assumes the interaction between the operation and management tools (or management applications) used by system administrators and the several management subsystems that provide the needed management services. The major components of the architecture are briefly discussed next.

Management tools and services. Management tools are used to perform various activities, such as configuration management, analysis of performance bottlenecks, report generation, visualization of network or system activity, simulation, and modeling. These activities would be initiated either manually by systems administrators using appropriate management tools, or automatically via predefined setups that trigger management requests to the underlying management services. These services, in turn, may utilize various management agents to carry out the requests. Interactions between management tools and services may take the form of simple procedure calls, remote procedure calls, or message passing, depending on how the management tools and services are implemented.

Examples of these management tools are currently available commercially.[3-6] However, most of these packages currently do not provide management tools or capabilities to manage all three classes of distributed systems management: network, operating system, and applications.

Figure 3 An integrated management architecture for distributed systems



The heart of our integrated management architecture is a set of management services that are logically organized as four subsystems, namely, *configuration, monitoring, control,* and *management information repository* subsystems. The management service interface used by management tools consists of the union of all the services available through the individual service interfaces (illustrated by shaded portions of the boxes on top of individual subsystems in Figure 3). Interac-

tions between the management subsystems (i.e., exchanging service requests and replies) may depend on the implementation platform as well. They may be done via local procedure calls if the subsystems are implemented as a single unit of management service, or remote procedure calls or some type of message passing if the subsystems are implemented as independent servers or available remotely. Each of these subsystem services is briefly described next.

Monitoring service. The monitoring service is responsible for monitoring the behaviour of managed objects in distributed systems. The monitoring activities are carried out by interacting with management agents. Monitored management data are collected by or from management agents and stored in the management information repository. Subsequently, management data may be retrieved from the repository for analysis. Results may be returned to the repository for other uses, such as for subsequent display or for further analysis.

The monitoring subsystem must be able to determine appropriate agents or information in response to current and anticipated requests for information. Such requests could originate from management applications via administrators or come from other management services. Accordingly, the monitoring subsystem is responsible to initiate the collection of information at appropriate times, to delegate monitoring requests to remote monitoring components or to subordinate systems and devices, to coordinate the collection of data from multiple agents, and to ensure the ongoing monitoring activities even under failures.

The amount of potentially useful data about components during execution can be very large. It would be desirable for the monitoring service to make decisions, perhaps with administrator input, as to where particular collected data should be stored based on the anticipated volume, importance, anticipated access, or use. Such decisions could also take into consideration multiple storage mechanisms (such as X.500,[7,8] databases, or files). The monitoring service should allow a means for selecting (or filtering) in advance what data are to be collected. Furthermore, the monitoring service should allow the user (or tools) to define certain conditions on managed resources so that when these conditions are met, an appropriate action can be triggered automatically. A simple example of this is to monitor the size of a queue in a server and generate an exception notification when it exceeds a specified limit.

Control service. The monitoring service (previously discussed) enables data on the run-time activities of the system components to be collected. These data can be analyzed or filtered and turned into useful information that can trigger some control actions to be taken by a control service. Such control actions may be required to prevent disasters on the network, device, service, or application. They may also be required to fine-tune the components to improve performance. For example, when a server gets overloaded, the resulting response time may be degraded considerably. If more requests arrive and the arrival rate of requests is greater than the service rate, the server may eventually fail to work. Such a condition may be detected by the monitoring service, and the management application could instruct the control service to carry out appropriate actions, such as instructing the server to drop requests or instructing the clients to submit fewer requests until the size of the request queue in the server goes below a threshold.

The control service encompasses the set of components responsible for controlling the behaviour of managed objects. Control activities may also be carried out by interacting with management agents. The control service requests may come from various user management tools, from the monitoring subsystem, or from the configuration subsystem. For example, the monitoring subsystem or user management tools may trigger appropriate control actions to be taken when exceptions on managed objects arise. The management information repository may be used for the storage and retrieval of control information.

Configuration service. Distributed applications and services typically consist of a number of components (such as clients and servers) running on various computing nodes dispersed throughout a distributed system. The configuration service supports component (e.g., device, server, process) tracking, component starting or termination, tracking of topological configurations and version information (and facilitates changes to them as required), and migrating of components to other nodes for reasons such as load balancing for increased performance. These activities are essential for ensuring reliable and efficient ser-

vices as well as for simplifying the tasks that system administrators have to perform daily.

Configuration service requests may come from user configuration management tools, from the monitoring subsystem, or from the control subsystem. This subsystem uses the management information repository for the storage and retrieval of configuration information.

Information repository service. The information repository provides a logically centralized view of the management information and provides a single interface to access the data and data sources. The repository itself will consist of many different collections of management data sources. For example, some may be kept in the X.500 directory, some in relational databases, some in files, and other information in processors themselves (such as routers or gateways). Such information would be distributed across multiple sites of the distributed environment. A unified interface provides a consistent way to access the data for management tools and other management services and agents and provides transparency of access to such information.

The repository must deal with both static and dynamic management information. Static management information includes such information as definitions of managed objects (as defined for SNMP [Simple Network Management Protocol],[9,10] CMIP [International Organization for Standardization standard for Common Management Information Protocol,[11] or Open Software Foundation Distributed Management Environment][12]), descriptions of collection agents, and information on applications and services (e.g., their configuration information). Dynamic management information includes such information as performance information (e.g., CPU load, network load, average response time), and faults information (e.g., status of servers, availability of services).

The information repository service may be used by the monitoring service to store data being collected from management agents. It may also be used by the configuration service to store the information pertaining to the initial network, device, system, or application configuration. The configuration information can be subsequently retrieved for various uses, such as reconfiguration or deployment of components. Further, analysis tools may retrieve stored data from the repository, process the data, and save the analyzed information in the repository. Such analyzed information may be retrieved by visualization tools and displayed to the user. Thus, the information repository service provides critical support for various management functions and other management services. An analysis of the essential requirements for management information and some experiences with prototype repositories can be found in Reference 13.

Management agents. Agents exist for carrying out management activities on behalf of management services and tools. For example, they may configure the managed resources, monitor their behaviour, and perform control actions on these resources. Such agents may vary from ones that constitute a collection of management interface routines (i.e., a library included as part of an application that was connected to sensors at the development stage), to ones that are active, independent processes, such as SNMP[14,15] or CMIP[16] agents. Management agents may reside in the systems and network layers to monitor and control the managed resources in those layers. Some agents may possess the capabilities of both monitoring and controlling functions. Others may even possess some analysis capabilities, which may be quite helpful to management services. Management services may communicate with these agents using SNMP, CMIP, or proprietary protocols.

Efforts such as OSF DME,[17] NMF OMNIPoint**,[18] UI Atlas-DM**,[19] and IBM SystemView*[20] represent efforts at defining some of the details of management services and, in particular, at defining how management components are related to other components of a distributed computing architecture. The DME work on the definition of managed objects, such as servers, is important and the results of the definitions represent the types of standardized managed object definitions to be captured in the information repository. In contrast to the other efforts, our work on the architecture is aimed at delineating the underlying management services required to support the management of distributed applications and the underlying network and systems resources.

To date, the architecture has been used as a framework to study the types of services, means of integration, and tools that should exist within

a management environment defined by the architecture. These experiments are discussed in the "Prototype implementations" section. The following section revisits the hospital example using the integrated approach suggested by the architecture.

A managed application

In this section, we revisit the hospital application to show how it could be managed by the proposed management architecture and tools. To effectively manage the application, management information is needed that can be used to verify that performance requirements are being satisfied, and to support management tools. In the example, there are application, host, and network performance requirements. Process operation response times, processor utilization, and network utilization are examples of management information. Their values can be compared with values determined by performance requirements to show whether requirements are met.

The managed objects of the distributed system include the processes for the hospital application, host operating systems, and the local area network. The distributed environment, operating systems, and network subsystems that support the application must make available management information about the managed objects within their domains. Management agents reside on each host to acquire this information and make it available to the management information repository and management tools.

The allocation of hospital processes and management agent processes to network nodes is shown in Figure 4. From the figure, we see how the management agents present in the distributed system are part of the management architecture. Management tools such as the Method of Layers (MOL)[21] and XNetMod**[22] (see also the next section) could use the proposed management architecture to acquire information needed to create performance models. Next we show how the tools can be used to deploy the hospital application, verify that its performance requirements are being satisfied, and perform a capacity planning exercise.

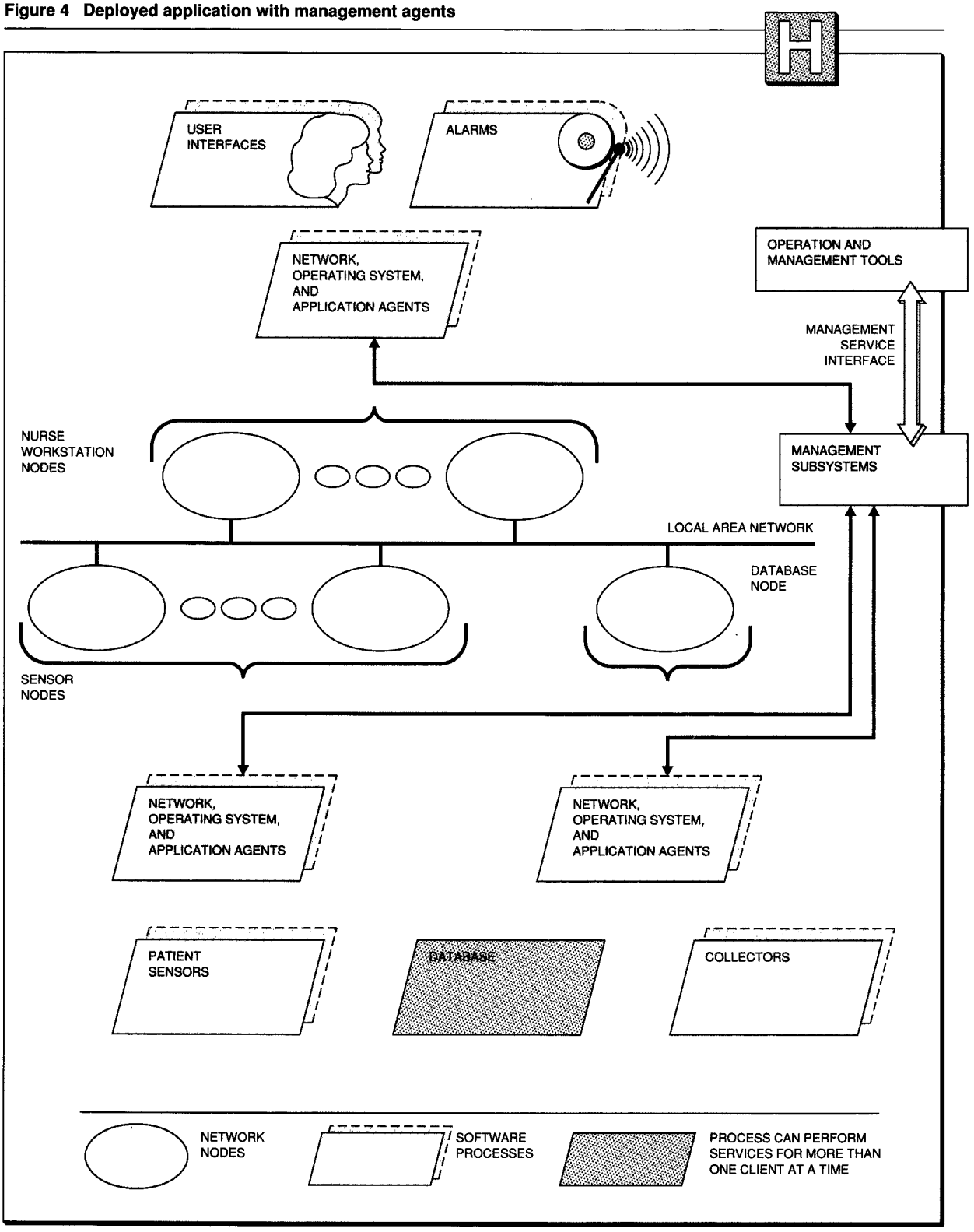
To configure a distributed application, many alternatives for process placement and selection of concurrency levels should be considered. A configuration that satisfies both performance and nonperformance requirements must be selected. Unfortunately, it is not always possible to deploy and measure the performance of all configurations of interest. Obtaining statistical significance for measurements is time-consuming and limits the number of configurations that can be considered. Analytical performance modeling tools can be used to help narrow the set of alternatives that should be measured. They can also be used for capacity planning studies when it is not practical to measure the performance of the system under consideration.

The systems manager for the hospital application has several degrees of freedom when deploying the application. The collector processes can be allocated to either the nurse workstations or the database host and the level of concurrency within the database process must be decided. With the proposed management architecture and management tools, a systems manager could undertake the study that is next discussed.

Two sets of analytic performance models are created that describe the distributed system required to support a workload of one hundred patients. The models are used to consider application and system performance behaviour. Initially, the resource requirement values used to create performance models are based on estimates provided by application developers. In the first set of models, the collector processes are allocated to the database host. With this configuration, collector and database processes do not require the use of the communications network to communicate. This keeps network communications to a minimum. In the second set, the collector processes are allocated to the associated nurse's workstations. The load of the application on the database host is lower in the second set of models. In each set of models, a range of levels for concurrency within the database process is considered. Performance estimates for the models are then obtained using analytic performance modeling tools.

The performance estimates suggest that allocating the collector processes to the database host will satisfy the performance requirements. The database process can serve up to six clients concurrently without the application exceeding its allocated processor utilization of 60 percent, so the level of concurrency for the database process is set to six. Analytic models help guide the systems

Figure 4 Deployed application with management agents



manager but only consider average behaviour. The distributed system management architecture must be applied to collect measurements and verify that the performance requirements are satisfied.

The application is deployed and subjected to a workload of one hundred patients. The management system helps to simplify the task of performing the measurement study by collecting the required management information and storing it in the management information repository. Measured values of process operation response times and host and network utilizations are compared with values determined by the performance requirements. In this example, the performance requirements are satisfied. The measurement system is also used to help undertake further performance tuning and to collect actual resource consumption metrics for future modeling exercises. It is found that the performance requirements are fulfilled and that the database process can support up to seven clients concurrently without exceeding the allocated capacity for the application on the database host.

As a capacity planning exercise, the systems manager considers the impact on system performance of doubling the number of patients and nurse stations. The system performance model is altered to reflect the new workload. Performance estimates for the models suggest the repository machine does not have the capacity to support the new work. The model is altered so that collector processes are allocated to nurse workstations. The load on the network, due to new sensor data and to the new network communications traffic between collector processes and the repository, exceeds the desired load on the network. Plans are put in place to investigate data compression mechanisms that will reduce the load on the network. The cost and impact on performance is compared with an alternate option of upgrading the network's capacity. The least expensive option is chosen.

Prototype implementations

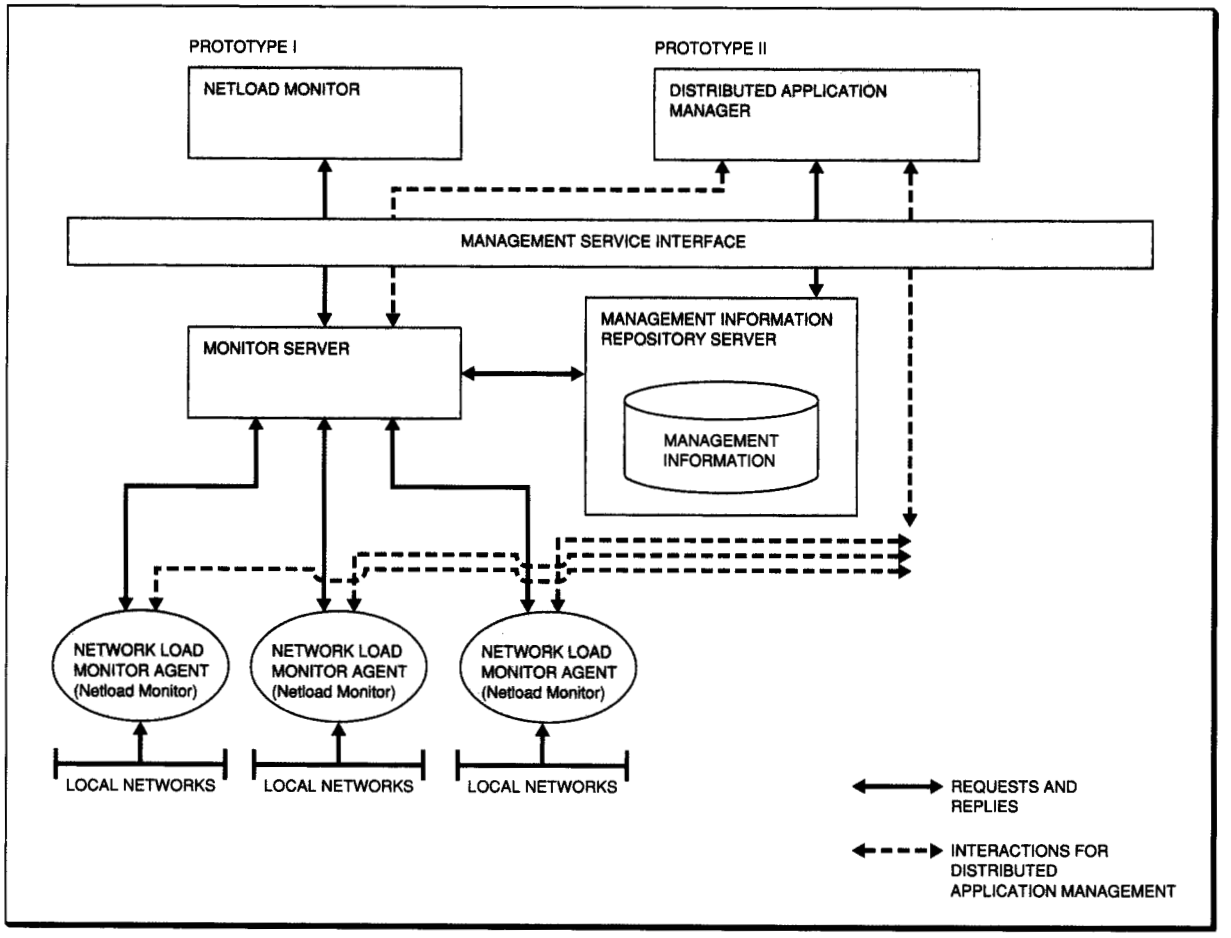
In this section we describe several prototype implementations of management tools and subsystems and how they fit into the management architecture presented earlier. The first two prototypes describe management agents that have been used to collect and store information in a repository. The next two prototypes describe management applications that can make use of information in repositories to model the performance behaviour of networks and distributed applications themselves. Lastly, a prototype that integrates a management application with a repository supported by management agents is described. The prototypes demonstrate the viability of the approach to distributed application modeling.

Prototype I—Network load monitoring. The University of Western Ontario (UWO) has developed two distributed systems monitoring prototypes, one for monitoring network load[13] and one for monitoring system load.[23] Both prototypes utilize the supporting service of a management information repository for storing and retrieving various management data and information. For example, the monitored systems and network loads are periodically collected and stored into the management information repository. This information can then be retrieved by other services or applications and used for other purposes, such as statistical analysis, load balancing, process migration, packet routing, and so on. Both prototypes also make use of monitoring servers and monitoring agents for collecting the load information.

Here, we elaborate one of those two prototypes. The network load monitoring prototype (Netload Monitor or NM) is a simple system that monitors the network loads of subnets in a heterogeneous internetwork environment. The monitoring tool consists of a graphical user interface, a monitoring server and a number of network load monitoring agents throughout the internetwork environment. Each network device runs agent software (network load monitor agent in Figure 5) for monitoring network traffic data from its interface. The number of packets that have been sent to the network by the device and the number of packets that have been read from the network by the device are examples of such traffic data. The network load monitoring server periodically polls these agents and collects the traffic data. The collected traffic data are then averaged out per subnet (per second, for example) and then stored in the management information repository.

The management information repository has been implemented using the X.500 Directory Service.[7,13] For interacting with the Directory Service, the prototypes used the Directory Access

Figure 5 The Netload monitoring system being managed by the Distributed Applications Manager

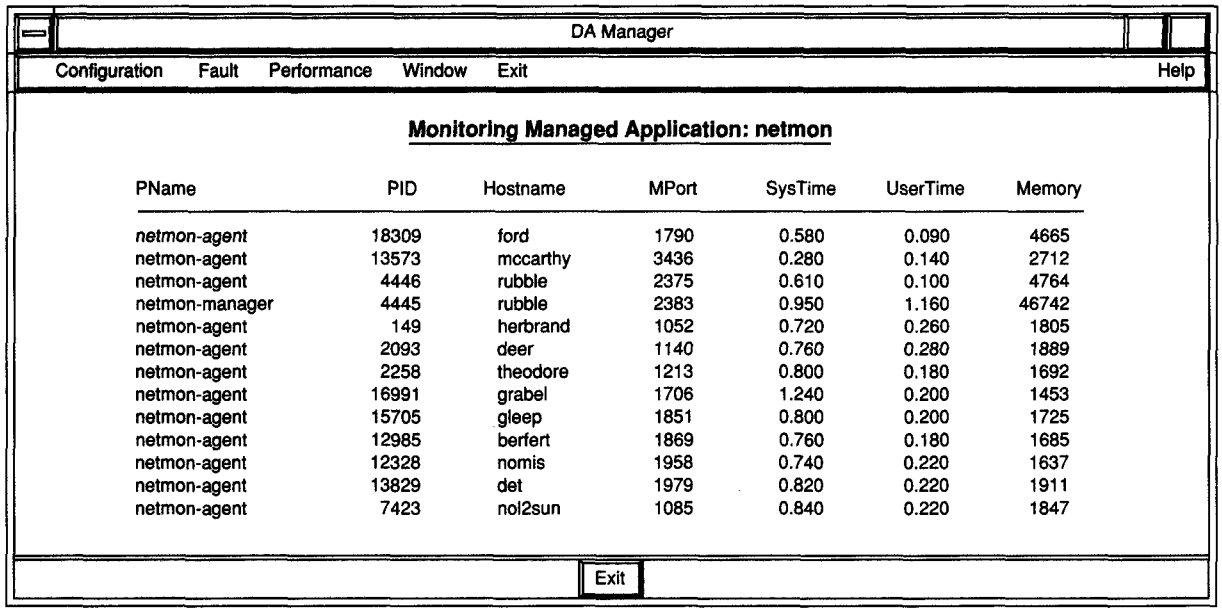


Protocol (DAP) interface. The X.500 implementation used in the prototype is QUIPU Version 7.0.[24]

Prototype II—DA manager. UWO has also developed a prototype (Distributed Applications Manager or DA Manager) for managing distributed applications.[25] The DA Manager consists of configuration, fault, and performance management tools. The configuration management tool allows the user to configure the components of a distributed application being managed. For example, it allows the user to allocate resources (such as processes) on appropriate hosts and to set their communication relationships. The configuration tool also allows the user to start up the component processes on remote hosts and shut them down when necessary. The fault management tool records the status of component processes by periodically checking the activity level of the processes. The performance management tool monitors the performance of the application by extracting appropriate performance data from the component processes and analyzing them.

In order for an application to be managed by the DA Manager, the application components must be instrumented with a management interface, which is a set of operations that can retrieve various information from managed processes and perform various actions on them (such as getting the resource usage of the process, setting the values on some data, and so on). Distributed applications come in various forms and sizes. Although each application possesses its own ap-

Figure 6 A display of monitored data of managed processes

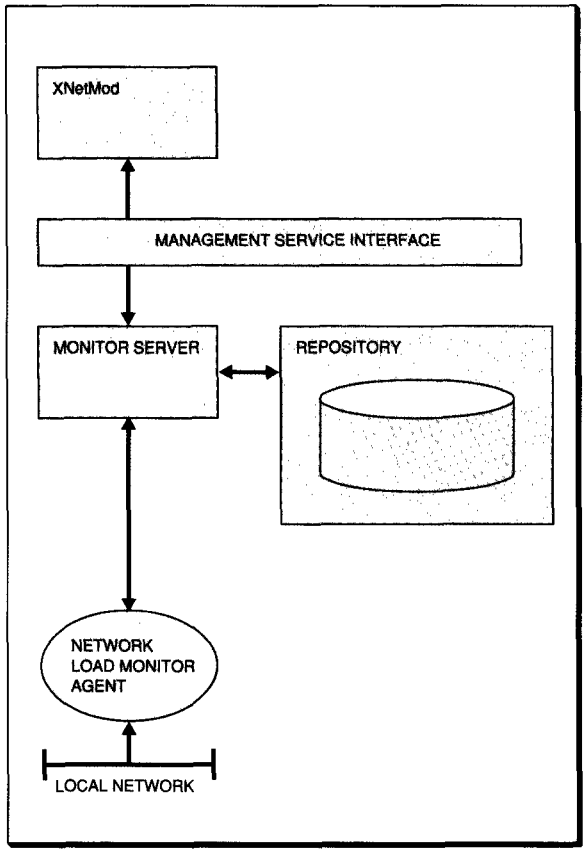


plication-specific data, there is a set of data that is common to most (if not all) distributed applications. Examples of such common data are process name, communication port, CPU usage, and memory usage. This common set of data and operations required to get or set their values constitute a generic management interface, which is what needs to be instrumented into applications so that they can be managed by the DA Manager.[26] That is, the DA Manager can manage any distributed application that is instrumented with the generic management interface.

To demonstrate the appropriateness of our management architecture, we have integrated the DA Manager prototype with the Netload Monitor prototype (described earlier). To be more precise, the DA Manager is being used to manage the Netload Monitor prototype, which happens to be a distributed application. This is also illustrated in Figure 5. The dashed lines joining the DA Manager and the managed components represent the interactions for the purpose of distributed application management. Each component process of the network load monitoring system is instrumented with the generic management interface, through which the DA Manager retrieves management information or performs actions on the managed process. Figure 6 shows a graphical display of a set of process-related generic information being collected from the managed processes by the DA Manager.

Prototype III—XNetMod. The University of Michigan has developed a network analytical modeling and performance analysis tool called the Network Modeling Tool (NetMod**).[22] The tool predicts the performance of new network technologies in a large-scale local network environment, possibly involving hundreds of local area networks and hundreds of thousands of sites. The principal application of the tool is to assist the network designer in configuring proposed network hardware and software components. The tool consists of analytical (queuing) models and a dynamic graphics user interface capability that allows users to visually design a network configuration and then call the tool to predict its performance. User workload is estimated from several basic user types that are related to generic applications such as CAD/CAM (computer-aided design/computer-aided manufacturing), word processing, database, etc. The workload can be customized to any environment.

Figure 7 A network management prototype example at the University of Michigan



The extended version of NetMod, XNetMod, has been developed for CORDS with extensions for network management. The extension allows XNetMod to use actual network traffic workload (rather than estimated workload) for the prediction of network performance. In the prototype implementation, illustrated in Figure 7, the Berkeley Packet Filter (BPF)[27] was used as the network monitoring agent. BPF strips off packet headers on the local area network and forwards them to the monitor server. The monitor server gathers statistical information about the local area network workload from the packet headers and stores this information into a repository (which is implemented using the Andrew File System**.[28]) XNetMod issues requests to the monitor server for workload information, which is satisfied by the server through the Andrew File System. Other requests by XNetMod to the monitor server include changing the rate at which workload information is written to the repository and controlling the amount of information to be saved.

Prototype IV—Method of Layers. The Method of Layers (MOL) is an analytical performance modeling tool developed at Carleton University and the University of Toronto that is used to study the performance behaviour of distributed applications.[21] It is based on approximate mean value analysis techniques originally developed for capacity planning in mainframe environments.[29,30] The techniques have been extended to reflect the impact on performance of software interactions. For example, client-server relationships as supported by the remote procedure call (RPC) and servers that can serve up to n (where $n >= 1$) clients concurrently are represented in the models.

A distributed application performance model takes as its input parameters:

- Process descriptions including a list of operations each process provides to other processes
- The scheduling disciplines of processes and devices
- The number of instances of each process
- The level of concurrency within each process
- The average service demand of each operation at its devices
- The average number of requests for service each operation makes to each other operation[21,31]

The MOL estimates average process operation response times and throughputs. It considers the queuing delays that processes incur when requesting service at devices and other processes. Parameters for the model can be altered to represent a change in system configuration. The tool then predicts the impact of the change on average operation response times and process throughputs.

It is a challenge to support the efficient collection of the above information for managed applications.[31] Sources of information include instrumentation used by application developers (as in the "Prototype II" section), distributed environment run-time systems,[32] and operating systems.[19] Once collected and stored in a repository, much of the work needed to create and validate

performance models for distributed applications can be automated.

Prototype V—Integration example. Another prototype implementation demonstrates the interoperability between the management tools and subsystems that are geographically dispersed. As presented in the sections on Prototype I and III, network management tools (that monitor network loads) were developed independently at UWO and at the University of Michigan (UofM) respectively. XNetMod analyzes the performance of networks at the UofM through its monitor server, which collects raw network traffic from its local networks (see Figure 7). UWO's Netload Monitor also analyzes its local networks by collecting raw network traffic from its local networks. The network traffic collected by the Netload Monitor is also stored into the management information repository (see Figure 5).

In this integration example, XNetMod is used to analyze the performance of networks located at UWO (which is located approximately 300 km from the UofM). This is achieved by the monitor server retrieving the UWO's network traffic data that are stored in the management information repository via a management information repository agent (MIR Agent in Figure 8). The dashed lines in Figure 8 represent the monitored data flow in the stand-alone Prototype III previously described. As explained earlier, the X.500 Directory Service has been used in the UWO Netload Monitor prototype to serve as a management information repository. The MIR Agent is basically a Lightweight Directory Access Protocol (LDAP)[33] server running on one of the machines at UWO accessing the repository on behalf of the monitor server running at the UofM. The monitor server at the UofM has been instrumented with the LDAP interface so that it can communicate with the LDAP server running at UWO using LDAP.

In this prototype we can label the key components as they are listed in our proposed architecture (Figure 3). Netload Monitor and XNetMod (monitoring and modeling applications) rely on the monitoring subsystem provided by UWO and the management information repository subsystem implemented through X.500 and LDAP. As illustrated by this prototype, the architecture can be successfully used in the support of distributed systems management. The key to the success of the integration relies heavily on the management information repository subsystem. The repository must be distributed to allow access by geographically dispersed applications. It must also have the data modeling capabilities to support XNetMod. Finally, the repository must possess reasonable performance and scalability characteristics. X.500 was shown to be suitable for this prototype integration,[34] although relational and object-oriented databases are also under consideration.

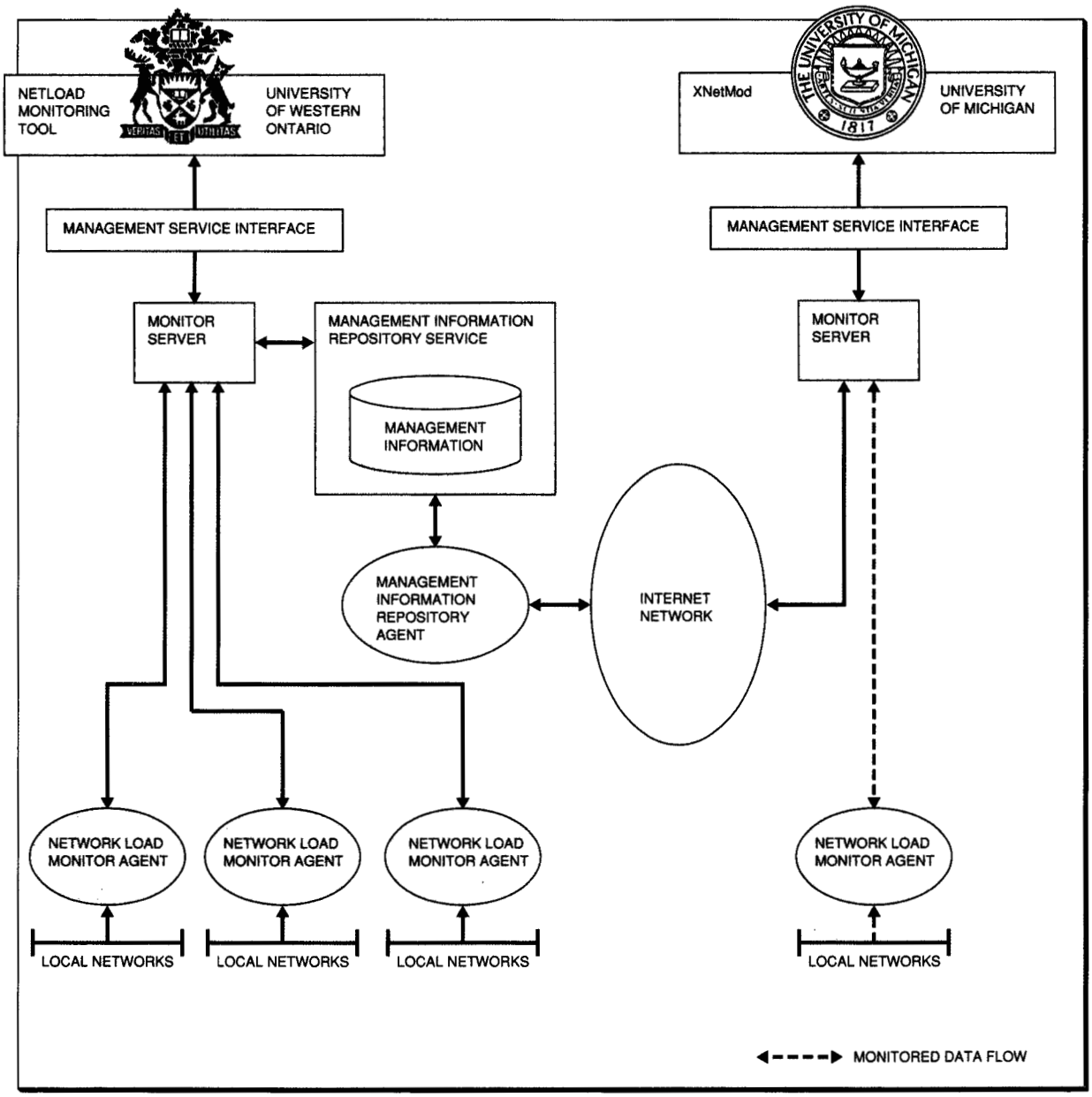
Concluding remarks

A sample application was described and used to show that network, operating system, and application performance behaviour are all dependent on one another. Combined information is needed for system managers to maintain efficient and reliable operations. As a result it is necessary to support an integrated approach to distributed systems performance management.

A reference architecture was presented that defines the components needed to support a distributed management system. We describe how the architecture supports the performance management of application, operating system, and network resources, and how the performance management tools we envision fit into the architecture. The relationship between components in the reference architecture and current technologies such as SNMP, CMIP, and DME is also described.

Several prototypes were presented that illustrate the integration of management agents and applications in distributed system management. The agents collect information regarding process resource consumption and network usage and store the information in repositories. Two management applications that make use of stored information are discussed along with a description of the management information on which they depend. The first, XNetMod, is a network performance modeling tool. The second, the Method of Layers, models the impact of process placements and software interactions on application performance. XNetMod supports the automatic creation of a network performance model based on the real load present on a network. Performance measures for the network are then estimated.

Figure 8 Integration example using the CORDS management architecture



Much has been done to define the measurement and control interfaces for application domains,[31,32] operating systems,[12] and network communication subsystems.[9,35] More work needs to be done to better understand the relationships between them. Existing management protocols provide the means to extract management information and to specify events. It is not yet clear whether they provide sufficient expressiveness so that events and conditions associated with the interacting environments can be expressed. Also, it is not clear whether the protocols can be used to delegate more functions, both monitoring and control, to more "intelligent" agents.[36] This is

necessary to support a scalable unified approach to distributed application performance management.

In the past few years much effort has been put into the management of networks and network devices. With the increase in the availability and use of distributed systems and applications, more effort is being directed toward the management of systems and applications. Efforts such as the Open Software Foundation DME,[17] the Network Management Forum OMNIPoint,[18] UI Atlas-DM,[19] and the IBM SystemView[20] are some examples that provide management architectures for the management of not only the networks and devices but also for system resources and applications. In general, most of these efforts are in early stages and are constantly going through many changes. We are well aware of these similar efforts and are currently following their activities closely. We intend to learn from their efforts, and we hope to influence their work as well.

Acknowledgments

This research was supported by the IBM Centre for Advanced Studies and the Natural Sciences and Engineering Research Council of Canada. We would also like to thank the referees of this paper for the excellent comments and suggestions for improving the paper.

*Trademark or registered trademark of International Business Machines Corporation.

**Trademark or registered trademark of Network Management Forum, Unix International, Regents of the University of Michigan, or Carnegie Mellon University.

Cited references

1. M. Alford, "SREM at the Age of Eight: The Distributed Computing Design System, *IEEE Computer* **18**, No. 4, 36–46 (April 1985).
2. R. Buhr, *Systems Design with Ada*, Prentice-Hall, Inc., Englewood Cliffs, NJ (1983).
3. J. H. Chou, C. R. Buckman, T. Hemp, A. Himwich, and F. Niemi, "AIX NetView/6000," *IBM Systems Journal* **31**, No. 2, 270–285 (1992).
4. H. S. Klemba, "Open View's Architectural Models," *Integrated Network Management*, I, Elsevier Science Publishers, New York (1989), pp. 565–572.
5. C. K. Law, "SunNet Manager," *SunWorld* **4**, No. 3, 60–68 (March 1991).
6. *DualManager Programmer's Reference Manual*, 1st Edition, NetLabs, Inc., Los Altos, CA (July 1991).
7. "The Directory—Overview of Concepts, Models and Services," *CCITT X.500 Series Recommendations*, CCITT (International Telegraph and Telephone Consultative Committee), (December 1988).
8. "The Directory—Overview of Concepts, Models and Services," *Draft CCITT X.500 Series Recommendations*, CCITT (International Telegraph and Telephone Consultative Committee), (December 1991).
9. K. McCloghrie and M. T. Rose, "Management Information Base for Network Management of TCP/IP-Based Internets: MIB-II, *Internet Request for Comments 1213* (March 1991).
10. K. McCloghrie and M. T. Rose, "Management Information Base for Network Management of TCP/IP-Based Internets," *Internet Request for Comments 1156* (May 1990).
11. Information Processing Systems—Open Systems Interconnection, *Structure of Management Information, Part 1: Management Information Model*, International Organization for Standardization, International Standard 10165-1 (1991).
12. *Catalog of OSF Managed Object Definitions, Volume 0.4*, Open Software Foundation, OSF-SIG-MAN-SD-7, Cambridge, MA (July 1992).
13. J. W. Hong, M. A. Bauer, and J. M. Bennett, "Integration of the Directory Service in the Network Management Framework," *Proceedings of the Third International Symposium on Integrated Network Management*, San Francisco, CA (April 1993), pp. 149–160.
14. J. Case, K. McCloghrie, M. Rose, and S. Waldbusser, "Introduction to Version 2 of the Internet-Standard Network Management Framework," *Internet Request for Comments 1441* (April 1993).
15. J. D. Case, M. S. Fedor, M. L. Schoffstall, and J. R. Davin, "A Simple Network Management Protocol," *Internet Request for Comments 1157* (1990).
16. Information Processing Systems—Open Systems Interconnection, *Management Information Protocol Specification, Part 2: Common Management Information Protocol*, International Organization for Standardization, International Standard 9596 (1991).
17. *The OSF Distributed Management Environment (DME) Architecture*, Open Software Foundation, Cambridge, MA (May 1992).
18. "Network Management Forum," *Discovering OMNIPoint*, Prentice-Hall, Inc., Englewood Cliffs, NJ (1993).
19. J. Herman, "UI-Atlas Distributed Management," *Distributed Computing Monitor* **7**, No. 12, 3–19 (December 1992).
20. *SystemView Structure*, SC31-7038, IBM Corporation (May 1993); available through IBM branch offices.
21. J. A. Rolia, *Software Performance Modelling*, CSRI Technical Report 260, University of Toronto, Canada (January 1992).
22. D. W. Bachmann, M. E. Segal, M. M. Srinivasan, and T. J. Teorey, "NetMod: A Design Tool for Large-Scale Heterogeneous Campus Networks," *IEEE Journal on Selected Areas in Communications* **9**, No. 1, 15–24 (January 1991).
23. J. W. Hong, M. A. Bauer, and J. M. Bennett, "Integration of the Directory Service in Distributed Systems Management," *1992 International Conference on Parallel and Distributed Systems*, Hsin Chu, Taiwan (December 1992), pp. 142–149.
24. C. J. Robbins and S. E. Kille, *The ISO Development Environment: User's Manual, Volume 5:QUIPU* (July 1991).
25. J. W. Hong and M. A. Bauer, "Design and Implementa-

tion of a Generic Distributed Applications Management System," *Proceedings of the GLOBECOM '93*, Houston, TX (November 1993), pp. 207–211.
26. J. W. Hong and M. A. Bauer, "A Generic Management Framework for Distributed Applications," *Proceedings of the IEEE First International Workshop on Systems Management*, Los Angeles, CA; IEEE Computer Society Press (April 1993).
27. S. McCanne and V. Jacobson, "The BSD Packet Filter: A New Architecture for User-Level Packet Capture, *"Proceedings of USENIX Conference*, San Francisco, CA (January 1993), pp. 259–269.
28. J. H. Howard, "An Overview of the Andrew File System," *Proceedings of USENIX Conference*, Dallas, TX (February 1988), pp. 23–26.
29. *MAP User's Guide*, Quantitative System Performance, Inc., Seattle, WA (1982).
30. *BEST/1 User's Guide*, BGS Systems, Inc., Waltham, MA (1982).
31. J. A. Rolia, "Distributed Application Performance, Metrics and Management," *Proceedings of the ICODP '93* (September 1993), pp. 205–216.
32. R. Friedrich, "The Requirements for the Performance Instrumentation of the DCE RPC and CDS Services," OSF DCE RFC 32.0 (June 1993).
33. W. Yeong, T. Howes, and S. Hardcastle-Kille, *Lightweight Directory Access Protocol*, Internet Engineering Task Force OSI-DS Working Document 26 (August 1992).
34. G. Winters and T. Teorey, "Managing Heterogeneous Distributed Computing Systems: An Analysis of Two Data Repositories," *Proceedings of the 1993 CAS Conference*, Toronto, Canada (October 1993), pp. 691–706.
35. Information Processing Systems—Open Systems Interconnection, *Structure of Management Information, Part 2: Definition of Management Information*, International Organization for Standardization, International Standard 10165-2 (1991).
36. G. Goldszmidt, S. Yemini, and Y. Yemini, "Network Management by Delegation—the MAD Approach," *Proceedings of CASCON '91*, IBM Centre for Advanced Studies, Toronto (October 1991), pp. 347–361.

Accepted for publication April 5, 1994.

Michael A. Bauer *Department of Computer Science, University of Western Ontario, London, Ontario N6A 5B7, Canada (electronic mail: bauer@csd.uwo.ca).* Dr. Bauer is chairman of the Computer Science Department at the University of Western Ontario. He holds a Ph.D. from the University of Toronto in computer science. His research interests include distributed computing, distributed directories, and software engineering.

Patrick J. Finnigan *IBM Software Solutions Division, Toronto Laboratory, 1150 Eglinton Avenue E, Don Mills, Ontario M3C 1H7, Canada (electronic mail: finnigan@vnet.ibm.com).* Mr. Finnigan is a staff member at the IBM Toronto Software Solutions Laboratory. He received his M.Sc. in computer science from the University of Waterloo in 1994. His research interests include visualization for distributed applications and software engineering.

James W. Hong *Department of Computer Science, University of Western Ontario, London, Ontario N6A 5B7, Canada (electronic mail: jwkhong@csd.uwo.ca).* Dr. Hong is a research associate and an adjunct professor in the Department of Computer Science at the University of Western Ontario. He received his Ph.D. from the University of Waterloo in 1991. His research interests include distributed computing, operating systems, software engineering, and network management.

Jerome A. Rolia *Department of Systems and Computer Engineering, Carleton University, Ottawa K1S 5B6, Canada (electronic mail: jar@sce.carleton.ca).* Dr. Rolia is an assistant professor in the Systems and Computer Engineering Department at Carleton University. He received a Ph.D. in computer science from the University of Toronto in 1992. His research interests include performance modeling, distributed systems, and software design for performance.

Toby J. Teorey *University of Michigan, Ann Arbor, Michigan 48103-4943 (electronic mail: teorey@umich.edu).* Dr. Teorey is Professor of Electrical Engineering and Computer Science at the University of Michigan and is Associate Chair for Computer Science. He holds a Ph.D. from the University of Wisconsin in computer science. His research interests include data modeling, distributed databases, and network performance tools.

Gerald A. Winters *Department of Electrical Engineering and Computer Science, the University of Michigan, Ann Arbor, Michigan 48109-2122 (electronic mail: gerald@citi.umich.edu).* Mr. Winters is a Ph.D. student in the Department of Electrical Engineering and Computer Science at the University of Michigan. His research interests include databases, networks, and system management.

Reprint Order No. G321-5549.