

A CORBA-BASED FRAMEWORK FOR THE MANAGEMENT OF MULTIMEDIA SERVICES

Myung-Sup Kim and James Won-Ki Hong
Dept. of Computer Science and Engineering
POSTECH

ABSTRACT

This paper presents a CORBA-based framework for the management of multimedia service. The framework consists of an architecture, methodologies, services, and MIB for managing distributed multimedia services. The focus is on managing the software components that provide distributed multimedia services. In this context, the key components include managed systems which are instrumented with management interfaces and a managing system which consists of a set of management services and management applications. The management services have been developed using OMG CORBA and can be used to support the management of both CORBA-based

as well as non-CORBA-based multimedia services. The API provided by the management services can be used for quick and easy development of management applications. A generic distributed multimedia service (DMS) MIB has been defined for the management of various multimedia services and applications. The DMS MIB can be easily extended to develop MIBs for specific multimedia services and applications. As a proof of concept, we have developed a Web-based management system for a CORBA-based distributed multimedia system called MAESTRO.

KEYWORDS:

DISTRIBUTED MULTIMEDIA SERVICE MANAGEMENT, MANAGEMENT FRAMEWORK,
DISTRIBUTED MULTIMEDIA SERVICES MIB, INSTRUMENTATION METHODOLOGY, CORBA

1. INTRODUCTION

Recently, we have been seeing an explosive growth on the development and use of distributed multimedia systems and applications in every aspects of our lives, from education to entertainment to work. The quality of many of these multimedia system and applications depends not only on the underlying networks and the end systems but also on the effective management of the resources involved. Because multimedia data are temporal, large, and complex, they require more network and system resources as well as special devices. Further, it typically takes more time to process than non-multimedia data. Thus, service managers must be concerned about various conditions to provide users with reliable and efficient services. However, the management of multimedia services is a very complex and troublesome work such that a powerful management system for multimedia services is necessary. Currently, there does not exist any international standards for the management of multimedia services and applications.

Since the early 80's, there has been a lot of work done in the area of network management. This work includes those carried out under the umbrella of ISO and ITU-T [1, 2] and those carried out under the umbrella of IETF [2, 3]. However, there has been very little work done in the area of services and applications management. This is especially true for the area of managing distributed multimedia services and applications. Our current work on the management of multimedia services and applications is based on our earlier work on the management of distributed applications and systems [4, 5, 6]. Recent effort being shown by IETF on defining MIBs for internet applications can be considered similar to our effort. It includes NSM (Network Services Monitoring) MIB [7], Mail Monitoring MIB [8], sysAppl MIB [9], and WWW MIB [10].

This paper presents a CORBA-based management framework, architecture and methodology for managing distributed multimedia services and applications [11,12, 18, 22]. We have developed a set of management services needed to monitor and control distributed multimedia applications and their supporting services. These management services have been defined using CORBA IDL and can used for quick and easy development of management applications. A generic distributed multimedia service (DMS) MIB has been defined for the management of

various multimedia services and applications. The DMS MIB can be easily extended to develop MIBs for specific multimedia services and applications.

A basic assumption is that multimedia service objects must be instrumented with management interfaces so that they can be monitored and controlled by managing systems. The DMS MIB is realized as Management Interface Objects (MIOs) which are instrumented automatically by using inheritance when multimedia service objects are developed. For non-CORBA-based multimedia services, our framework can be still used but by using a gateway.

Earlier, we have developed a CORBA-based distributed multimedia system called MAESTRO which supports the development and operation of distributed multimedia applications [13, 14]. For validation of our work, we have used our framework to manage the multimedia services which are part of MAESTRO as well as applications running on MAESTRO. In this paper, we also describe our effort on the prototype implementation of a Web-based management system for MAESTRO. The prototype management system uses OrbixWeb [15] to interface with the management server which is implemented as a collection of CORBA object and monitors distributed multimedia services which are also implemented as CORBA objects. The result is that the administrator can manage the MAESTRO system with a popular Web browser such as Netscape or Internet Explorer which is a familiar user interface by most people these days.

The organization of the paper is as follows. Section 2 introduces our management framework. Section 3 introduces an architecture for the management of distributed multimedia services. Section 4 describes the methodologies used in the management. Section 5 defines a set of management services required for the management of distributed multimedia services. Section 6 presents the definition of DMS MIB. Section 7 describes our prototype implementation of a Web-based management system. We summarize our work and discuss some possible future work in Section 8.

2. FRAMEWORK FOR MULTIMEDIA SERVICES MANAGEMENT

Most traditional network management systems

use the manager-agent (or managing-managed system) paradigm. Here, an agent is typically closely located with real resources being managed and monitors and reports their activities. A manager performs various management functions on behalf of the human manager by obtaining management information from agents or requesting the agents to perform management actions on managed objects (MOs). Management information base (MIB) is a collection of MOs that an agent maintains. Typically, MOs are abstract representation of real managed resources. For multimedia services management, these real managed resources would be software components (such as servers). Figure 1 illustrates a general management paradigm that is applicable to both network and systems management.

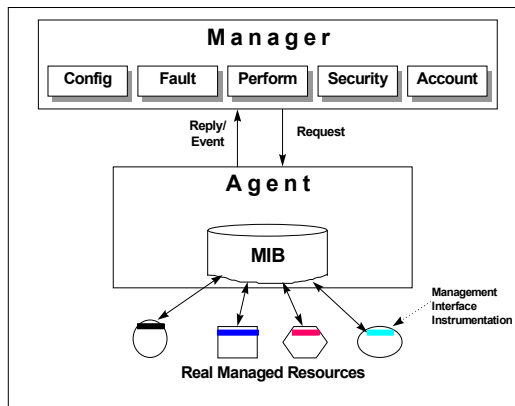


Figure 1. A General Management Paradigm

For multimedia services management, we can use the same management paradigm used in traditional network management. That is, we need a manager (managing system) and agent (managed system) with MOs and real managed resources they represent. Of course, there can be multiple managers and agents in a service environment.

In order to monitor and control multimedia service objects, managers must be able to extract management information from and perform management operations on them. This requires that the managed objects be instrumented with management interfaces [16]. Before management interfaces can be instrumented, they must be defined. This requires an in-depth analysis of what information and operations are important for the purpose of management. Another important requirement is that the instrumentation should be

made as simple and automatic as possible. This would reduce the burden from the system developers. In order to satisfy these requirements, we have defined distributed multimedia service (DMS) MIB and realization of the MIB in the form of CORBA object which can be inserted automatically by inheriting this interface in every managed service object. The details on these will be explained in later sections.

Also needed in a managing system is a set of management services and management applications. Management services are required to support the general management functional areas such as configuration, fault, performance, security and accounting management. Management services are also required to easily support a variety of higher-level management functions (such as QoS management, session management and so on). Management applications that are used by human managers to manage the services should be powerful but very user-friendly. The current trend is to provide a Web browser-based user interface for simplicity and ubiquity.

3. ARCHITECTURE FOR MULTIMEDIA SERVICES MANAGEMENT

In this section, we present an architecture for the management of distributed multimedia services. The architecture is based on an object-oriented modeling. We assume that all multimedia services and management services exist as objects. Figure 2 illustrates the architecture which consists of Service Objects (SOs), Management Service Objects (MSOs), Management Interface Objects (MIOs), multimedia applications and management application.

SO is a multimedia service object. For example, SO can be a multimedia communication service object, session service object, name service object, storage and retrieval service object, and so on. SOs are used by multimedia applications for their operations. In our architecture, SOs are the resources to be managed. MSO is an object which provides various management services to the management application. MSO receives management requests from the management application and performs the requested operations on the SOs via MIOs. Events can be generated by SOs when problems or predefined set of conditions are met and then notified to the MSO which in turn can send

notifications to the management application. The management services included in MSO are described in more detail in the next section.

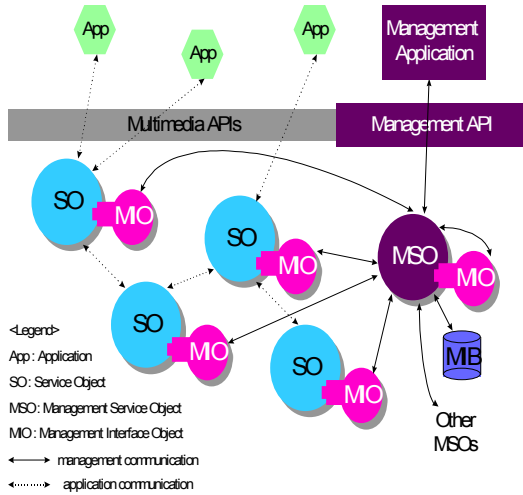


Figure 2. A Management Architecture for Distributed Multimedia Services

MIO is an object which is instrumented in every managed SO so that SOs can be managed by MSO [16]. MIO is defined as a set of management operations and data needed for managing SOs. Specific MIOs can be developed by extending the general MIO using the inheritance feature of the object-oriented modeling. Since MSO itself is a service object like other SOs, MSO can be managed by being equipped with a MIO of its own.

Since we are interested in managing multimedia services which are distributed in a possibly large internetwork environment, we may require more than one MSO. Multiple MSOs can be used to increase fault-tolerance of the management system. Further, MSOs may need to communicate each other to support distributed management of the distributed multimedia services.

4. METHODOLOGIES FOR MULTIMEDIA MANAGEMENT SERVICES

In this section, we describe the methodologies required for multimedia services management. In order for the managed objects to be monitored and controlled, management interface must be instrumented into managed

objects so that the managing system can extract appropriate information and perform operations on them. Thus, we describe the methodology for instrumenting management interface into managed objects. We then describe the methodology for management interaction between the managing system and managed system. Lastly, we describe a methodology for managing non-CORBA-based multimedia services.

4.1 Instrumental Methodology of Management Interfaces

Distributed multimedia services are provided to users by software components that may be scattered throughout a distributed environment. These software components are realized as clients and servers running on various computer systems. Typical server components may be name servers, directory servers, communication servers, session servers, accounting servers, user information servers, multimedia database servers and so on. In CORBA-based systems, these servers are realized as one or more objects and may provide services to client objects as well as to other server objects as illustrated in Figure 3.

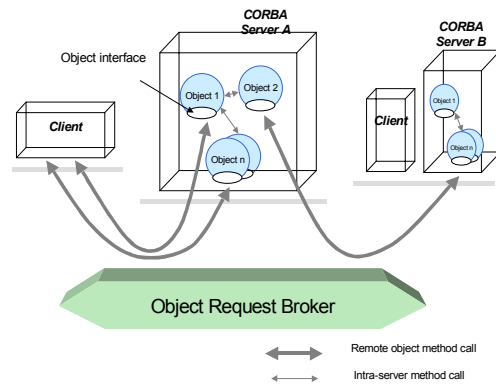


Figure 3. Interaction of Client-Server Objects in CORBA

Objects interact with each other by invoking methods defined as part of object interfaces. For example, the following is an abbreviated CORBA IDL definition of session service object which provides session service to clients. Clients can request a session service by invoking one of the methods defined as part of the Session_Service_Object interface. We call such an interface an application interface.

```

interface Session_Service_Object {
    boolean createSession (in string session_name, in
        string name, in string passwd, in string domain, in
        short sessionPermission)
        raises (Reject);
    boolean destroySession (in string session_name, in
        string passwd)
        raises (Reject, NotDescription);
    boolean joinSession (in string session_name, in
        string name, in string domain)
        raises (Reject, NotDescription);
    boolean leaveSession (in string session_name, in
        string name)
        raises (Reject, NotDescription);
};
    
```

In order to monitor and control multimedia services, we must be able to interact with these service objects. As application interface is available for providing intended multimedia service, we need an interface in each of these managed objects for the purpose of management. In CORBA, it is possible to have multiple interfaces in a single object (e.g., one for application and another for management). In order to instrument a management interface, we need to define its IDL definition and include it as part of the service object. The definition of management interface (which we call Management Interface Object, or MIO) will be described in Section 6. Using the inheritance feature of CORBA, we can define a session service object inheriting the MIO as shown below.

```

Interface Session_Service_Object : MIO {
    boolean createSession(in string session_name, in
        string name, in string passwd, in string domain, in
        short sessionPermission)
        raises (Reject);
    ....
};
    
```

In this way, as a Session_Service_Object is instantiated, it will have the application interface and MIO interface automatically.

4.2 Interaction Methodology between Managing and Managed Systems

In traditional network management systems, both polling (i.e., a manager solicits agents for management information) and event reporting (i.e., an agent sends an unsolicited management information to a manager when a reportable event occurs) are used. We can use the same interaction techniques for multimedia services management. For polling, a method invocation on an MIO object, which is embedded in the

service object being managed, to retrieve information from and to perform actions on it. For event reporting, the Event Service defined as part of the CORBA common object services [23] can be used.

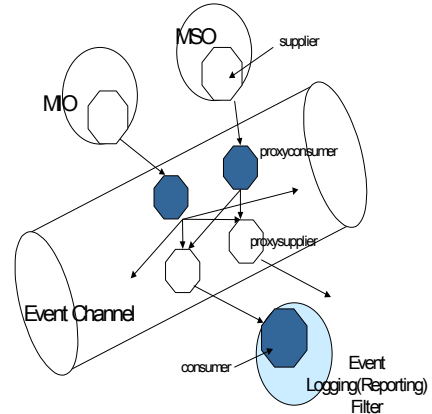


Figure 4. CORBA Event Service

The CORBA Event Service has been designed to provide basic event service to CORBA objects. It provides two event service models: the “push model” and “pull model”. Events are generated from “supplier” objects and then consumed by “consumer” objects. The “supplier” and “consumer” objects are combined with the “push” and “pull” event models to provide four possible types of supplier and consumer objects: the push supplier, pull supplier, push consumer and pull consumer. Also, the events can be supplied or consumed on behalf of the real supplier and consumer objects via proxy objects. All of this event communication (i.e., supply and consume) takes place with through a mediation object called “Event Channel”, as illustrated in Figure 4.

This is how it basically works. Every object can supply or consume events by creating and including supplier or consumer objects. Such an object can request the event channel to create supplier/consumer admin objects, which are responsible for administrating the activities of the supplier and consumer objects. Then it requests the supplier/consumer admin objects to create proxy supplier/consumer and connects is supplier/consumer to the proxy. After these actions are performed successfully, the object can supply events to all consumers or consume events from all suppliers which are connected to the event channel. An event channel allows events

to be multicasted from suppliers to all consumers. Event filter objects (which are described in detail in Section 5.2) then select only the events of interest and forward them to appropriate consumer objects.

4.3 Methodology for Managing Non-CORBA-based Multimedia Services

Although the framework, architecture and methodologies presented in this paper focus on managing CORBA-based multimedia services, they can be easily extended to manage non-CORBA-based multimedia services. The key aspect of making it possible is the use of CORBA/non-CORBA gateway as illustrated in Figure 5. This gateway would appear to be a CORBA object to the CORBA-based multimedia service management system and manage it like any other CORBA object. The gateway translates the management operations and data from the manager system into the appropriate ones of the target managed system. The replies and event reports from the managed system would also go through the gateway and the necessary conversions would have to be made. To the non-CORBA-based multimedia system, the gateway appears as a manager.

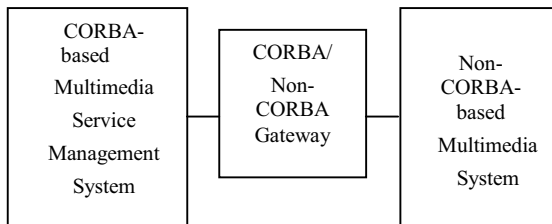


Figure 5. Management of Non-CORBA-based Systems via Gateway

As mentioned earlier, the interaction interface between the manager system and gateway is CORBA. The interaction interface between the gateway and the managed system depends on the interface the managed system can provide. For example, it may be an SNMP interface, CMIP interface or some proprietary interface.

5. MANAGEMENT SERVICES FOR DISTRIBUTED MULTIMEDIA SERVICES

In this section, we describe the management

services for distributed multimedia services. They are classified into four sets of services which include configuration management, fault management, security management and performance management. These services are used by the management application for the management of multimedia service objects. Figure 6 illustrates the service objects included as part of the Management Service Object (MSO) and the relationships with the management application and the application server objects being managed.

The MSO is composed of Configuration Management Service Object (cMSO), Fault Management Service Object (fMSO), Security Management Service Object (sMSO) and Performance Management Service Object (pMSO). These management service objects processes management requests from the management application and performs appropriate management operations on the application server objects (SOs) being managed through the management interface object (MIO). They also process replies or event reports generated from the managed application servers and forwards them to appropriate management applications. Typically, each management service object may be composed of multiple cooperating objects, providing the intended service.

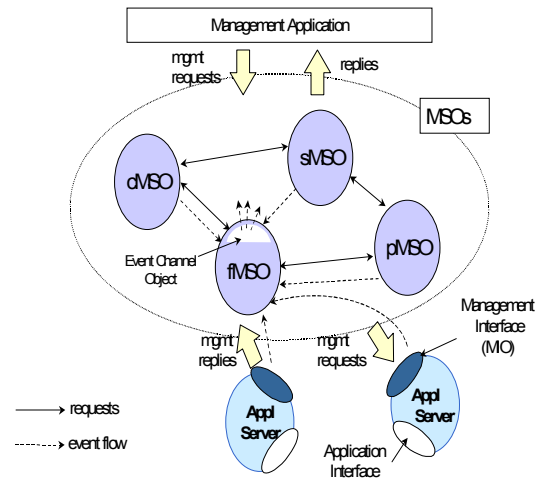


Figure 6. Management Services for Distributed Multimedia Services

5.1 Configuration Management Service

The configuration management service is

responsible for several things for managing the configuration of resources in a management domain. A management domain is a collection of resources which are managed by MSO. A domain can be created, changed, and deleted by MSO upon the request of the management application. The configuration management service is responsible for starting and shutting down one or more application server objects (SOS). It keeps track of which SOS are available in a domain and their individual management information. Such information can be registered by SOS to the configuration management service when they start up or can be polled by the configuration management service. The configuration management service is also responsible for adding, updating and maintaining the relationships among the SOS in a domain.

In addition, the configuration management service monitors the status of SOS during their operation. It can provide status information to the management application periodically during normal operations and notify problems when problems are detected. When problems are detected, the management application may request some actions to be taken by the configuration management service. This may include the change of the relationship of SOS, change of attributes, restarting a new SO and so on. The configuration management service also provides a simple information listing service such as listing the available SOS in a domain, their attributes and so on.

A subset of the configuration management service defined in CORBA IDL is given below.

```
interface cMSO {
    void detect_so(in Token token, in string
        domain_name, out SOIDSeq o_ids);
    void list_so(in Token token, out SOIDSeq so_ids);
    void get_attr(in Token token, in long token, in SOID
        so_id, in OID oid,
        out AttrType attr);
    void set_attr(in Token token, in long token, in SOID
        so_id,
        in AttrType attr);
    void init(in Token token, in SOID so_id);
    void terminate(in Token token, in SOID so_id);
    SOID register_so(in string so_name, in string object);
    void cancel_so(in SOID so_id);
};
```

5.2 Fault Management Service

The fault management service is responsible for providing a reliable service environment by handling faults gracefully when they do occur in

the resources of a management domain. When a fault occurs, the fault management service must notify the management application of the problem. It in turn must be able to determine where the problem lies so that appropriate action can be taken in order to isolate the problem and fix the problem as soon as possible. If the problematic SO can not be fixed immediately, a replacement SO must be made available right away. To achieve these, several functions must be provided in the fault management service.

First, a logging facility is provided. A service manager must be able to set a filter which specifies the kinds of events or faults and information to be logged. It also must be able to start a logging process according to the logging filter and stop the process. Moreover, the service manager must be able to specify the name of a log file. These log files can be examined by the service manager to determine the cause of problems in SOS.

Second, the fault management service must be able to detect and notify various faults. A fault filter can be used to specify the kinds of faults which are notified. Intelligence can be added to the fault management service so that minor problems can be handled by the fault management service itself without notifying to the service manager.

Finally, when a SO can not perform its functions, the SO is substituted by a new SO. This can be done automatically by sending a request to the configuration management service by the fault management service or done manually by the management application.

A subset of the fault management service defined in CORBA IDL is given below.

```
Interface Efilter {
    boolean set_name(in string name);
    boolean set_scope(in SOIDSeq soids);
    boolean set_eventlist(in EventTypeSeq eventtypes);
    boolean set_eventinfo(in char infoset);
    boolean start();
    boolean stop();
    string get_result(out EventInfoSeq eventinfos);
};

interface Efilter : Efilter { ... };
interface Efilter : Efilter { ... };

interface fMSO
    Efilter create_elfilter(in Token token);
    Efilter create_elfilter(in Token token);
    boolean destroy_elfilter(in Token token, in Efilter
        elfilter);
    void detect_faults(in Token token);
```

```

boolean substitue_SO(in Token token, in SOID
soid,in
                SOID new_soid);
};

```

5.3 Performance Management Service

The performance management service is responsible for providing an efficient multimedia service environment. To achieve this, the behavior of Sos must be monitored closely and appropriate performance data must be collected and analyzed. These metrics must be defined as part of the Management Interface Object (MIO) and Sos must be able to provide the needed data.

The performance management service may utilize the logging service to log performance-related data. This data can be used by the management application to analyze the performance levels of the SO in question. As a result of such analysis, the configuration management service can be invoked to perform appropriate control actions to improve the performance.

A subset of the performance management service defined in CORBA IDL is given below.

```

Interface pMSO {
    boolean start_perf_monitoring(in Token token, in
TimeStamp interval);
    PerfInfoSeq show_perf_info (in Token token);
    PerfInfo show_current_perf (in Token token);
    boolean stop_perf_monitoring(in Token token);
};

```

5.4 Security Management Service

The security management service is responsible for providing a secure environment for the operation and management of the resources in a domain. The configuration management service and fault management service may perform control operations that should be performed only by the authorized users. Otherwise, disasters can happen. Thus, the security management service must provide an authentication mechanism for checking valid users of the management application and management requests.

The monitoring function of the management services typically generate logs for keeping track of requests, replies and event reports. Such information must only be accessed by the authorized users as well. Access on such information as well as other important information

must be controlled by an access control mechanism, which must be also provided by the security management service.

A subset of the security management service defined in CORBA IDL is given below.

```

interface SMSO
    boolean add_user(in Token token, in string id, in
string passwd, in short authorization);
    boolean delete_user(in Token token, in string id);
    boolean change_passwd(in Token token, in string
oldpasswd, in string newpasswd);
    long login(in string id, in string passwd);
    boolean unlog(in Token token);
    boolean set_ACL(in Token token, in string id, in
SOID rid, in char permission);
    char get_ACL(in Token token, in string id,in SOID
rid);
    boolean is_authuser(in long token);
};

```

6. Distributed Multimedia Service MIB

In this section, we define a generic distributed multimedia service (DMS) management information base (MIB). We call it a generic DMS MIB because it contains management information common to most (if not all) multimedia service objects (SOs). Thus, the DMS MIB can be used to monitor and control most SOs. However, if specific management information must be obtained in order to manage a particular SO, the DMS MIB can be easily extended by including the SO specific management information.

In designing the DMS MIB, we have utilized a lot of work that was already done by several IETF working groups. That is, we have analyzed existing MIBs defined for the management of Internet applications and services and extracted important and relevant management information from them. A distributed multimedia service can be seen as a black box that supports multimedia applications on a network or internetwork. In this way, its management can be viewed as a logical extension to the network services monitoring (NSM) MIB [7]. Distributed multimedia services can also be viewed from their implementation as a collection of software processes, threads, files, etc. From this perspective, their management is an extension to the system application (sysAppl) MIB [9]. So, our DMS MIB includes parts of both NSM MIB and sysAppl MIB as well as new management information.

The DMS MIB is defined into four information areas: 1) general information, 2)

operational statistics, 3) resources information, and 4) service interface information. The DMS MIB has been defined using SMIV2 [17] and its full definition is given in [18].

6.1 General Information

The general information includes information common to most SOs. Each attribute is described below.

- Index: Arbitrary integer. This integer yields a unique key for each SO in a management domain.
- Object Name: SO's object name. Applications or MSO can find an SO and access its interfaces by the object name.
- Host Name: Host name on which an SO is executed.
- Executable: SO's executable name with full path.
- Owner: A user who executes a registering process of an SO.
- Communication: Communication protocol which is used between an SO and applications. (TCP, UDP, etc.)
- Version: SO's version.
- Last Change Time: Time when an SO's version was updated.
- Last Start Time: Time when an SO started.
- Last Service Time: Time when an SO served any application most recently.
- Current Status: SO's current status. It is one of NOT_REACHABLE, RUNNABLE, RUNNING, and NOT_AVAILABLE. NOT_REACHABLE is the case when MSO or applications can not reach the SO. The case may be caused by shutdown of a host in which the SO's executable exists or failure of network connection between the SO and applications. RUNNABLE is the case when all conditions of the SO go well, but there is no application which is served by the SO. RUNNING is the case when the SO is serving more than one application. Lastly, NOT_AVAILABLE is the case when the SO's host is alive, but some conditions are invalid. For example, when SO's executable is deleted suddenly, no application can be served by the SO.

- Role: An SO is either primary or spare. Primary is a running one currently and spare is a waiting one to be substituted when the primary is failed.
- Primary/Spare Index: When an SO's role is primary, this is an index of its spare. When an SO's role is spare, this is an index of its primary.

6.2 Operational Statistics

The followings are information about operational statistics. These are calculated while an SO is running. Using these statistics, the SO's performance metrics such as throughput and availability can be calculated.

- Current InboundAssociation: The number of applications which are served by an SO currently.
- Current OutboundAssociation: The number of other Soss by which an SO is served currently.
- InRequest: The number of requests received by an SO.
- OutRequest: The number of requests sent to other Soss by an SO.
- InRequestErrors: The number of requests which are not served among requests received.
- OutRequestErrors: The number of requests which are not served among requests sent.
- CPU Utilization: Current CPU load used by an SO.
- Memory: Current memory space used by an SO.

6.3 Resource Information

Each SO has its own list of resources it is currently using. For example, if an SO is using a device, a file, a child process, or a thread, they can be listed. Each entry in the list has the following attributes:

- Type: Resource type. Its value may be file, device, thread, or process.
- Name: This has a different form according to the type of a resource. For a file, it is a file name including directory path. For a device, it is a device name. And for a thread or a process, it is an identifier given by the operating system. It allows a resource to be identified uniquely.

- State: It has different values according to the type of resource, too. For a file, it has one of open, reading, and writing. For a device, it has one of ready, waiting, and using. For a thread or a process, it has one of running and suspended.

6.4 Service Interface Information

The service interface information is concerned with the SO's interfaces. Because they are static information which are not changed during run time, they are set once at the time when the SO starts.

- Type & Version: An SO's interface format. The interface may be CORBA IDL [19], RPC IDL [20], or application level protocol using message passing. It has also a version or a release number. For example, this attribute has a value, 'Orbix IDL 2.0'.
- Directory: In a case when the interface needs an external specification, such as IDL file, this attribute is used. It has a directory path of IDL file.
- File: Used like the Directory attribute but has a file name for IDL.

7. PROTOTYPE IMPLEMENTATION

In the previous sections, we have presented a framework for the management of distributed multimedia services including an architecture, a set of management services and the DMS MIB. Based on these, we have developed a prototype management system to validate our concepts. The platform we are using is MAESTRO [13,14], which is a CORBA-based distributed multimedia system.

MAESTRO is an object-oriented, distributed multimedia system, whose goal is to provide multimedia services needed to develop and operate a variety of multimedia applications. MAESTRO has several service objects including Name Service Object (NSO), Communication Service Object (CSO), Session Service Object (SSO), Storage and Retrieval Service Object (SRSO), and Management Service Object (MSO).

Because MAESTRO was developed on a CORBA [19] platform, IONA Orbix 2.0 [21], our management system is also implemented on the same platform. In order to provide a uniform interface and multi-platform management, our

management system is Web-based. This means that administrators can easily manage MAESTRO from any platform where a Java-enabled Web browser such as Netscape or Internet Explorer can be run.

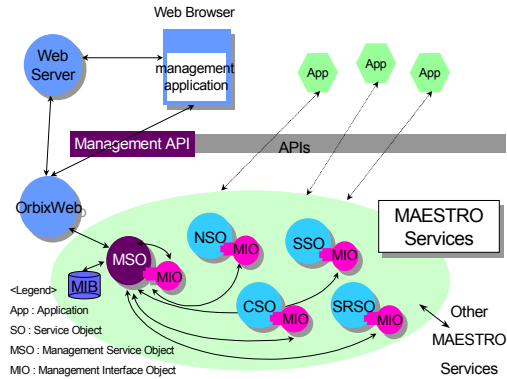


Figure 7. A Prototype Management System for MAESTRO

Figure 7 illustrates our Web-based prototype management system for MAESTRO. MSO manages service objects in a MAESTRO sever and provides management API using Java to develop a management application which is a Java applet. OrbixWeb [15] is used to allow the management application to invoke methods on MSO which is a CORBA object created with Orbix.

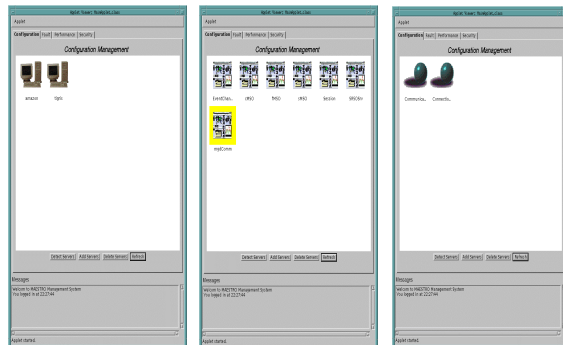


Figure 8. Configuration Management Application

Figure 8 illustrates a prototype configuration management application. The first window displays the hosts on which the managed application server objects are currently running. By clicking the mouse on one of the server machine icons, one can see the list of the application server objects being managed. By clicking the mouse on one of the application

A CORBA-BASED FRAMEWORK FOR THE MANAGEMENT OF MULTIMEDIA SERVICES

server objects, one can see its component CORBA objects. The attributes for individual objects can be monitored or modified through appropriate management operations defined as part of their MIOs. In this way, one can manage the configuration of the multimedia application server objects.

Figure 9 illustrates a prototype fault management application. The first window displays the hosts on which the managed application server objects are currently running. It also displays the managed server objects and events defined for each object. One can set the logging flag so that the events can be logged or have the events be reported using a desired reporting mechanism. One can create an event filter so that only certain events of interest can be captured. The events being captured are displayed in the second window.

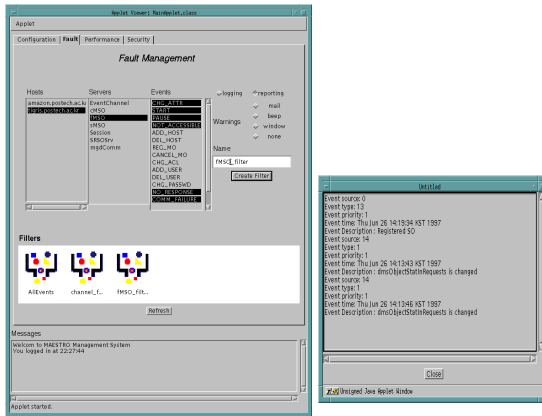


Figure 9. Fault Management Application

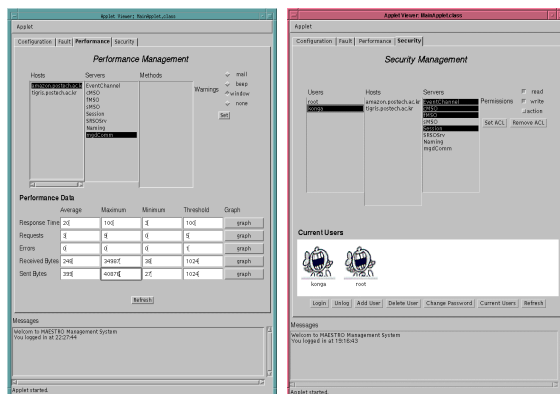


Figure 10. Performance and Security Management Applications

Figure 10 illustrates a prototype performance management application and security management application. The performance management application monitors the performance aspects of the managed application servers. For example, one can monitor the average, maximum, minimum values of response time, number of requests, received bytes and sent bytes. One can also set the threshold for each performance parameter and get notified if the monitored values go beyond the threshold values. The security management application shows a list of users and access control lists (ACLs), and options for setting ACLs.

8. CONCLUSION AND FUTURE WORK

In this paper, we have presented a framework, architecture and methodologies for managing distributed multimedia services. We have defined a set of management services using CORBA IDL, which is needed to support the management of distributed multimedia services and applications. Our other contribution is the development of the DMS MIBs which can be easily extended to develop MIBs for specific distributed multimedia services and applications. Although the work presented in this paper focuses on managing CORBA-based multimedia services, we have also shown how non-CORBA-based multimedia services can be managed using the gateway.

As a proof of concept, we have developed a set of Web-based management applications. We believe our decision to develop a Web-based management system is a good choice since it offers many attractive features. First, it provides a widely-used and familiar user interface -- the administrator does not have to learn another GUI when learning this system or when the system capabilities and features change. Second, the management console can be run from any platform which supports a Java-enabled Web browser -- such browsers are available on most UNIX and PC platforms. Third, adding/deleting/modifying the features of the management system does not involve change in the management console but only in the management service.

Our future work includes developing CORBA/non-CORBA gateways so that non-CORBA-based multimedia systems can be managed using our framework and management applications.

REFERENCES

-
- [1] OSI, Information Technology - Open Systems Interconnection - Systems Management Overview. International Organization for Standardization, June 1991.
- [2] William Stallings. SNMP, SNMP-2, and CMIP, The practical Guide to Network Management Standards. Addition-Wesley Publishing company, INC, 1993.
- [3] J. Case, M. Fedor, M. Schoffstall, and J. Davin. Simple Network Management Protocol, March 1990. RFC 1157.
- [4] J. W. Hong, M. J. Katchabaw, M. A. Bauer, and H. Lutfiyya. Modeling and management of distributed applications and services using the OSI management framework. *Proc. of the International Conference on Computer Communication*, pages 215-220, Seoul, Korea, July 1995.
- [5] M. Bauer, N. Coburn, D. Erickson, P. Finnigan, J. Hong, P. Larson, J. Slonim, D. Taylor, and T. Teorey. A distributed system architecture for a distributed application environment. *IBM Systems Journal*, 33(3):399-425, September 1994.
- [6] M. Bauer, P. Finnigan, J. Hong, J. Rolia, T. Teorey, and G. Winters. Reference architecture for distributed systems management. *IBM Systems Journal*, 33(3):426-444, September 1994.
- [7] S. Kille and N. Freed, Network Services Monitoring MIB, <http://ds.internic.net/rfc/rfc1565.txt>, January 1994. RFC 1565.
- [8] S. Kille and N. Freed, Mail Monitoring MIB, <http://ds.internic.net/rfc/rfc1566.txt>, August 1996. RFC 1566.
- [9] C. Krupczak and J. Saperia, Definitions of System-Level Managed Objects for Applications, <http://ds.internic.net/internet-drafts/draft-ietf-applmib-sysapplmib-08.txt>, Internet Draft, April 15, 1997.
- [10] C. Kalbfleisch, H. Hazewinkel, and J. Schoenwaelder, Definition of Managed Objects for WWW Server. <http://ds.internic.net/internet-drafts/draft-ietf-applmib-wwwmib-02.txt>, Internet Draft, March 26 1997.
- [11] MAScOTTE, White Paper, MAScOTTE Project (Management Services for Object oriented distributed systems) Esprit Project: 20804, November, 1996.
- [12] Monitoring of CORBA-based Applications, The CORBA-Assistant, White Paper, Fraunhofer Institute Informations and Data Processing IITB, May, 1997. <http://tes.iitb.fhg.de/corba-assistant/>
- [13] T. H. Yun, J. Y. Kong, and J. W. Hong, Object-oriented modeling of distributed multimedia services. *Proc. of IEEE International Conference on Communications*, Montreal, Canada, June 1997. pp. 777-781.
- [14] T. H. Yun, J. Y. Kong and J. W. Hong, A CORBA-based Distributed Multimedia System, *Proc. of the Fourth Workshop on Distributed Multimedia Systems*, Vancouver, Canada, July 1997, pp. 1-8.
- [15] IONA, OrbixWeb, IONA Technologies Ltd., <http://www.iona.com/Orbix/OrbixWeb/index.html>, December 1996. Release 2.0.
- [16] J. W. Hong, G. Gee, and M. A. Bauer, Towards automating instrumentation of systems and applications for management. *Proc. of the IEEE Global Telecommunications Conference*, pages 107-111, Singapore, November 1995.
- [17] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser, Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2), January 1996. RFC 1902.
- [18] J. Y. Kong and J. W. Hong, "A CORBA-based Management Framework for Distributed Multimedia Services and Applications", *Proc. of the Distributed Systems: Operations and Management*, Sydney, Australia, October 1997, pp. 132-144.
- [19] OMG, The Common Object Request Broker: Architecture and Specification Revision 2.0. OMG, July 1995, OMG TC Document.
- [20] John Bloomer, Power Programming with RPC, O'Reilly & Associates, 1992.
- [21] IONA, Orbix 2. IONA Technologies Ltd., March 1997, Release 2.2.
- [22] J. Y. Kong, T. H. Yun, S. W. Park, S. H. Kim, Y. M. Shin and J. W. Hong, "Design and Implementation of a Management System for CORBA-Based Distributed Systems", *Proc. of the Asian-Pacific Network Operations and Management Symposium*, Seoul, Korea, October 1997, pp. 473-485.
- [23] OMG, CORBA services: Event Service Specification,

<http://www.omg.org/corba/sectrans.htm>, March 1995.