A CORBA-based Management Framework for Distributed Multimedia Services and Applications

Ji-Young Kong and J. Won-Ki Hong*
Dept. of Computer Science and Engineering, POSTECH, Korea

Jong-Tae Park
School of Electronic and Electrical Engineering, Kyungpook National University, Korea

Dong-Jin Kim
Distributed Processing Section, ETRI, Korea

Abstract

This paper proposes a CORBA-based management framework for managing distributed multimedia services and applications. 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. As a proof of concept, we have developed a Web-based management system for a CORBA-based distributed multimedia system, MAESTRO. The prototype management system uses OrbixWeb to interface with the management server which is implemented as a CORBA object and monitors distributed multimedia services which are also implemented as CORBA objects.

[**Keywords:** distributed multimedia service management, management framework, distributed multimedia services MIB, CORBA, Web-based management]

*The contact person for this paper is Prof. J. Won-Ki Hong, Dept. of Computer Science and Engineering, POSTECH, San 31, Hyoja-Dong, Pohang, 790-784, KOREA, TEL: +82-562-279-2244, FAX: +82-562-279-5699, Email: jwkhong@postech.ac.kr

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 or dominant technology 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 [14, 16] and those carried out under the umbrella of IETF [4, 16]. 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 [7, 1, 2]. 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 [11], Mail Monitoring MIB [12], sysAppl MIB [15], and WWW MIB [10].

In this paper, we propose a CORBA-based management framework for managing distributed multimedia services and applications. 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.

Earlier, we have developed a CORBA-based distributed multimedia system called MAESTRO which supports the development and operation of distributed multimedia applications [17]. For validation of our management framework, we have attempted 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 [9] to interface with the management server which is implemented as a 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 an ordinary 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 an architecture for the management of distributed multimedia services. Section 3 defines a set of management services required for the management of MAESTRO. Section 4 presents the definition of DMS MIB. Section 5 describes our prototype implementation of Web-based management system. We summarize our work and discuss some possible future work in Section 6.

2 Architecture for Management of Distributed Multimedia Services

In this section, we present an architecture for the management of distributed multimedia services. Our architecture is based on an object-oriented modeling. We assume that all multimedia services and management services exist as objects. Figure 1 illustrates the architecture which consists of Service Objects (SOs), Management Service Objects (MSOs), Management Interface Objects (MIOs), multimedia applications and management application.

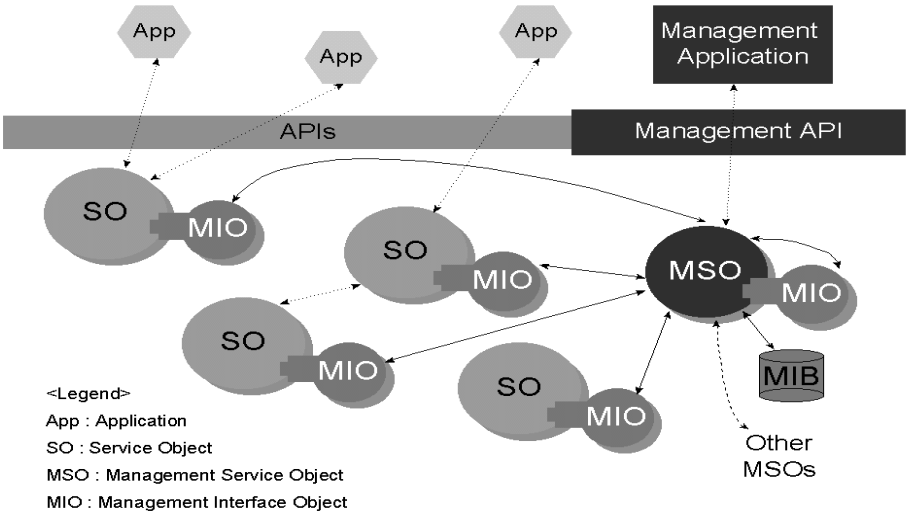

Figure 1: A Management Architecture for Distributed Multimedia Services

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.

MIO is an object which is instrumented in every managed SO so that SOs can be managed by MSO [6]. 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 technique. 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.

3 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 (SOs). Figure 2 illustrates the services included in the Management Service Object (MSO) and the relationships with the management application (shown as the Service Manager in the figure) and the Service Objects (SOs) being managed.

3.1 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 a MSO. A domain can be created, changed, and deleted by MSO upon the request of the service manager. The configuration management service is responsible for starting and shutting down one or more SOs. It keeps track of which SOs are available in a domain and their individual management information. Such

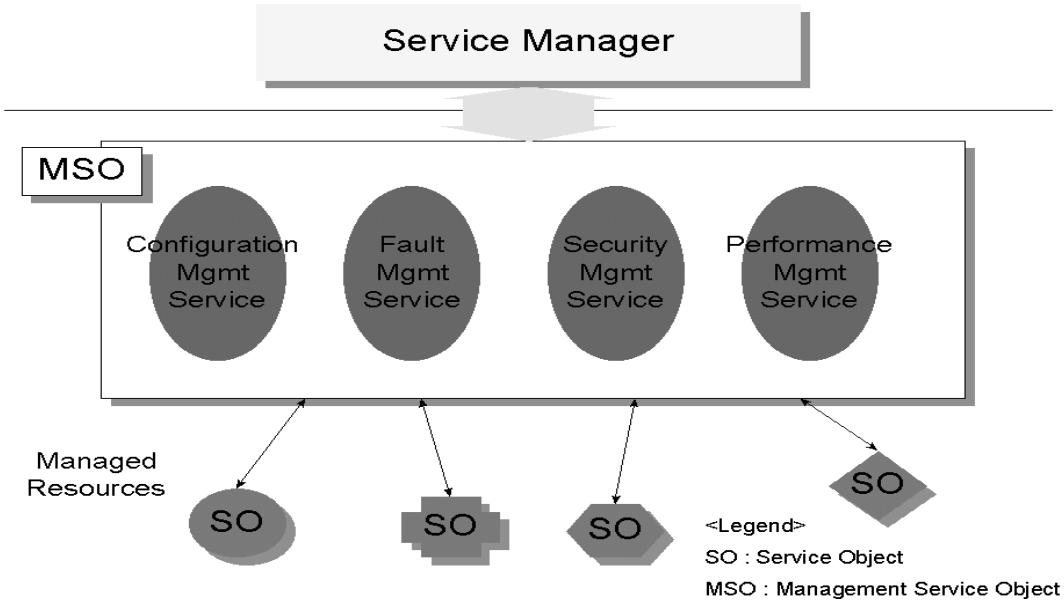

Figure 2: Management Services for Distributed Multimedia Services

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 service manager 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 provide 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 MSO_CMS {
    boolean add_host(in short token, in IPAddress ip);
    boolean delete_host(in short token, in IPAddress ip);
    IPAddressList list_hosts(in short token);
    RID register_resource(in string rname);
    boolean cancelregister(in RID rid);
    short detect_resources(in short token);
    RIDList list_resources(in short token);
```

```
    RInfo list_rinfo(in short token, in RID rid);
    AttrType get_attribute(in short token, in RID rid, in AttrType attr);
};
```

3.2 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 service manager 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 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 service manager.

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

```
interface MSO_FMS {
    LFilterID set_log_filter(in short token, in EventTypeList elist,
                        in char infoset, in string file);
    LID start_logging(in short token, in LFilterID filter);
    EventInfoList show_logs(in short token, in LID lid);
    boolean stop_logging(in short token, in LID lid);
    FFilterID set_fault_filter(in short token, in EventTypeList flist,
                        in char infoset);
    boolean start_detecting_fault(in short token, in TimeStamp interval);
    EventInfoList show_faults(in short token, in FFilterID filter);
    boolean stop_detecting_fault(in short token);
    boolean substitute_resource(in short token, in RID rid, in RID newrid);
    boolean report_event(in RID rid, in EventType event);
};
```

3.3 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 service manager 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 MSO_PMS {
    PMFilterID set_perf_filter(in short token, in PerfMetricList flist,
                        in string file);
    PMID start_perf_monitoring(in short token, in TimeStamp interval);
    PerfInfoList show_perf_metrics(in short token, in PMID pmid);
    PerfInfo show_current_metric(in short token, in PMFilterID filter);
    boolean stop_perf_monitoring(in short token, in PMID pmid);
};
```

3.4 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 performance management service defined in CORBA IDL is given below.

```
interface MSO_SMS {
    boolean login(in string id, in string passwd);
    boolean unlog(in short token, in string id);
    boolean set_ACL(in short token, in string id, in unsigned long sid,
                        in char permission);
};
```

4 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 our 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 cane be viewed as a logical extension to the network services monitoring (NSM) MIB [11]. 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 [15]. 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 [5] and its full definition is given in Appendix A.

4.1 General Information

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

- Index: Arbitrary integer. This integer yield 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 in which an SO are executed.
- Directory: Full path in which an SO's executable exists.
- Executable: An SO's executable name.
- 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: An SO's version
- Last Change Time: time when an SO's version is updated.
- Last Start Time: time when an SO start of late.
- Last Service Time: time when an SO served any application of late.
- Current Status: An 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. At last, 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, any application can not 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 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.

4.2 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 as like 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 SOs by which an SO are served currently.
- InRequest: The number of requests received by an SO.
- OutRequest: The number of requests sent to other SOs 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 spaces used by an SO.

4.3 Resources 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: It 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 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 a thread or a process, it has one of running and suspended.

4.4 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 interfaces format. The interfaces may be CORBA IDL [13], RPC IDL [3], 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 interfaces need an external specification, as like IDL file, this attribute is used. It has a directory of IDL file.
- File: Used at the same case as the above. It has a file name for IDL.

5 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 are developing a prototype management system to validate our concepts. The platform we are using is MAESTRO [17], 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

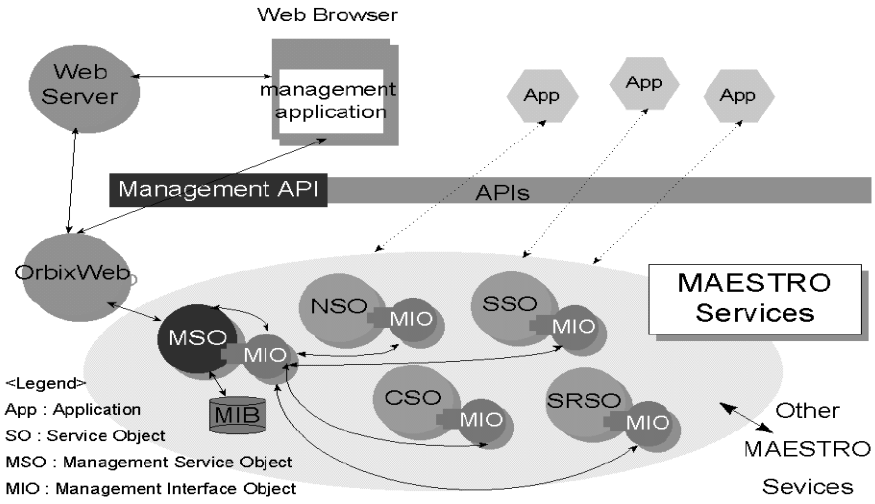

Figure 3: A Web-based Prototype Management System for MAESTRO

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 [13] platform, IONA Orbix 2.0 [8], 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 can be run.

Figure 3 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 [9] is used to allow the management application to invoke methods on MSO which is a CORBA object created with Orbix.

6 Conclusion and Future Work

In this paper, we have discussed the motivation behind our work on developing a CORBA-based management framework 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 MIB which can be easily extended to develop MIBs for specific distributed multimedia services and 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 run be 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 completing the implementation of the management system and evaluating its features. The multimedia applications we have developed (such as video multicasting application, video conferencing application and whiteboard application) on MAESTRO can be used to fully test the management services and the Web-based management system.

As more multimedia services and applications are developed and deployed for use at home and at work, their effective management is desperately needed to provide a reliable and efficient environment. We believe our work is a good start for tackling such formidable task.

References

[1] 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.

[2] 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.

[3] John Bloomer. *Power Programming with RPC*. O'Reilly & Associates, Inc., 1992.

[4] J. Case, M. Fedor, M. Schoffstall, and J. Davin. *Simple Network Management Protocol*, March 1990. RFC 1157.

[5] 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.

[6] J. W. Hong, G. Gee, and M. A. Bauer. Towards automating instrumentation of systems and applications for management. In *Proc. of the IEEE Global Telecommunications Conference*, pages 107–111, Singapore, November 1995.

[7] 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. In *Proc. of the International Conference on Computer Communication*, pages 215–220, Seoul, Korea, July 1995.

[8] IONA. *Orbix 2.* IONA Technologies Ltd., November 1996. Release 2.0.

[9] IONA. *OrbixWeb.* IONA Technologies Ltd., http://www.iona.com/Orbix/OrbixWeb/index.html, December 1996. Release 2.0.

[10] C. Kalbfleisch and H. Hazewinkel. *Definition of Managed Objects for WWW Servers.* http://ds.internic.net/internet-drafts/draft-ietf-applmib-sysapplmib-06.txt, January 1997. Internet Draft.

[11] S. Kille and N. Freed. *Network Services Monitoring MIB.* http://ds.internic.net/rfc/rfc1565.txt, January 1994. RFC 1565.

[12] S. Kille and N. Freed. *Mail Monitoring MIB.* http://ds.internic.net/rfc/rfc1566.txt, August 1996. RFC 1566.

[13] OMG. *The Common Object Request Broker: Architecture and Specification Revision 2.0.* OMG, July 1995. OMG TC Document.

[14] OSI. *Information Technology - Open Systems Interconnection - Systems Management Overview.* International Organization for Standardization, June 1991.

[15] J. Saperia, C. Krupczak, R. Sturm, and J.Weinstock. *Definition of Managed Objects for Applications.* http://ds.internic.net/internet-drafts/draft-ietf-applmib-sysapplmib-06.txt, November 1996. Internet Draft.

[16] William Stallings. *SNMP, SNMP-2, and CMIP, The Practical Guide to Network-Management Standards.* Addition-Wesley Publishing Company, INC, 1993.

[17] T. H. Yun, J. Y. Kong, and J. W. Hong. Object-oriented modeling of distributed multimedia services. In *Proc. of IEEE International Conference on Communications*, Montreal, Canada, June 1997. To be published.

Appendix A: DMS MIB Definition

```
DMS-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE,
    Counter32, Integer32                        FROM SNMPv2-SMI
    DisplayString, TimeStamp                    FROM SNMPv2-TC
    experimental                                FROM RFC1155-SMI;

dmsMIB MODULE-IDENTITY
    LAST-UPDATED "9702190000Z"
    ORGANIZATION "Distributed Processing Environment Lab., POSTECH"
    CONTACT-INFO
            "Ji-Young Kong
             Postal: Dept. of Computer Science and Engineering
                     Pohang University of Science and Technology
                     San 31, Hyoja-Dong, Pohang, 790-784, KOREA
                     TEL: +82-562-279-2244
                     FAX: +82-562-279-5699
                     E-mail: konga@nile.postech.ac.kr"
    DESCRIPTION
            "The MIB module for distributed multimedia services.
             It has generic attributes for distributed multimedia services"
    ::= { experimental 100 }

-- dmsObjectTable contains general information for distributed multimedia
-- service objects (SO).

dmsObjectTable  OBJECT-TYPE
    SYNTAX      SEQUENCE OF DmsObjectEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "The table of the SO present on the system."
    ::= { dmsMIB 1 }

dmsObjectEntry  OBJECT-TYPE
    SYNTAX      DmsObjectEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "Details of a particular SO"
    INDEX   { dmsObjectIndex }
    ::= { dmsObjectTable 1 }

DmsObjectEntry ::= SEQUENCE {
    dmsObjectIndex
```

```
    Integer32,
dmsObjectName
    DisplayString,
dmsObjectHostName
    DisplayString,
dmsObjectDirectoryName
    DisplayString,
dmsObjectExecutable
    DisplayString,
dmsObjectOwner
    DisplayString,
dmsObjectComm
    DisplayString,
dmsObjectVersion
    DisplayString,
dmsObjectLastChange
    TimeStamp,
dmsObjectLastStart
    TimeStamp,
dmsObjectLastService
    TimeStamp,
dmsObjectStatus
    INTEGER,
dmsObjectRole
    INTEGER,
dmsObjectPrimarySpareIndex
    Integer32,
dmsObjectInboundAssociations
    Counter32,
dmsObjectOutboundAssociations
    Counter32,
dmsObjectInRequest
    Counter32,
dmsObjectOutRequest
    Counter32,
dmsObjectInRequestErrors
    Counter32,
dmsObjectOutRequestErrors
    Counter32,
dmsObjectCPU
    Integer32,
dmsObjectMemory
    Integer32,
dmsObjectInterfaceType
    DisplayString,
dmsObjectInterfaceDirectory
    DisplayString,
dmsObjectInterfaceFile
    DisplayString
```

```
}

dmsObjectIndex  OBJECT-TYPE
    SYNTAX      Integer32 (1..2147483647)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
         "An index to uniquely identify the SO."
    ::= { dmsObjectEntry 1 }

dmsObjectName   OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         "The SO's name needed to access the SO's information."
    ::= {dmsObjectEntry 2}

dmsObjectHostName OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         "The host name in which the SO run."
    ::= {dmsObjectEntry 3}

dmsObjectDirectoryName OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         "The directory in which the executable of the SO exists."
    ::= {dmsObjectEntry 4}

dmsObjectExecutable OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         "An SO's executable name."
    ::= {dmsObjectEntry 5}

dmsObjectOwner OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         "The owner who possesses the process of the SO."
    ::= {dmsObjectEntry 6}
```

```
dmsObjectComm OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         "The communication protocol which is used between the SO
          and applications. TCP, UDP, etc"
    ::= {dmsObjectEntry 7}

dmsObjectVersion OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         "The version of the SO"
    ::= {dmsObjectEntry 8}

dmsObjectLastChange OBJECT-TYPE
    SYNTAX      TimeStamp
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The time when the version of the SO is updated lately."
    ::= {dmsObjectEntry 9}

dmsObjectLastStart OBJECT-TYPE
    SYNTAX      TimeStamp
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The time when the SO start lately."
    ::= {dmsObjectEntry 10}

dmsObjectLastService OBJECT-TYPE
    SYNTAX      TimeStamp
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The time when the SO serve any application lately."
    ::= {dmsObjectEntry 11}

dmsObjectStatus OBJECT-TYPE
    SYNTAX     INTEGER
        { not_reachable(1),
          runnable(2),
          running(3),
          not_available(4)
        }
```

```
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "the current status of the SO. It is one of ‘not_reachable’, ‘runnable’,
         ‘running’, and ‘not_available’. ‘not_reachable’ is the case when
         MSO (Management Service Object) 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.
         At last, ‘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, any application can not be served by the SO."
    ::= {dmsObjectEntry 12}

dmsObjectRole OBJECT-TYPE
    SYNTAX      INTEGER
        { primary(0),
          spare(1)
        }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "the SO is either ‘primary’ or ‘spare’. ‘primary’ is a running one
         currently and ‘spare’ is a waiting one to be substituted
         when ‘primary’ is failed."
    ::= {dmsObjectEntry 13}

dmsObjectPrimarySpareIndex OBJECT-TYPE
    SYNTAX      Integer32 (1..2147483647)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "When the role of the SO is PRIMARY, this is an index of its spare.
         When the role of the SO is spare, this is an index of its primary."
    ::= {dmsObjectEntry 14}

dmsObjectInboundAssociations OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         "The number of applications which are served by the SO, currently."
    ::= {dmsObjectEntry 15}

dmsObjectOutboundAssociations OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
```

```
    STATUS      current
    DESCRIPTION
        "The number of other SOs by which the SO are served currently."
    ::= {dmsObjectEntry 16}

dmsObjectInRequest OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of requests received by the SO."
    ::= {dmsObjectEntry 17}

dmsObjectOutRequest OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of requests sent to other SOs by an SO."
    ::= {dmsObjectEntry 18}

dmsObjectInRequestErrors OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of requests which are not served among requests received."
    ::= {dmsObjectEntry 19}

dmsObjectOutRequestErrors OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of requests which are not served among requests sent."
    ::= {dmsObjectEntry 20}

dmsObjectCPU OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Current CPU load used by the SO."
    ::= {dmsObjectEntry 21}

dmsObjectMemory OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-only
    STATUS      current
```

```
    DESCRIPTION
        "Current memory spaces used by an SO."
    ::= {dmsObjectEntry 22}

dmsObjectInterfaceType OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "the SO's interfaces format. The interfaces may be CORBA IDL,
         RPC IDL 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''."
    ::= {dmsObjectEntry 23}

dmsObjectInterfaceDirectory OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "In a case when the interfaces need an external specification,
         as like IDL file, this attribute is used. It has a directory of IDL
         file."
    ::= {dmsObjectEntry 24}

dmsObjectInterfaceFile OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Used at the same case as the above. It has a file name for IDL."
    ::= {dmsObjectEntry 25}

-- dmsResourceTable contains information about resources
-- used by managed SO. Each entry has its own index and
-- an index to dmsObjectTable to identify the SO which is
-- using the resource.

dmsResourceTable  OBJECT-TYPE
    SYNTAX      SEQUENCE OF DmsResourceEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The table of the resources which are used by SOs.
         Because several SOs can use the same resource,
         a resource may be listed in this table once or more."
    ::= { dmsMIB 2 }

dmsResourceEntry  OBJECT-TYPE
```

```
    SYNTAX      DmsResourceEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Details of a particular resource which is used
         by at least one SO."
    INDEX { dmsObjectIndex, dmsResourceIndex }
    ::= { dmsResourceTable 1 }

DmsResourceEntry ::= SEQUENCE {
    dmsResourceIndex
        Integer32,
    dmsResourceType
        INTEGER,
    dmsResourceName
        DisplayString,
    dmsResourceState
        INTEGER
}

dmsResourceIndex  OBJECT-TYPE
    SYNTAX      Integer32 (1..2147483647)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "An index to uniquely identify the resource"
    ::= { dmsResourceEntry 1 }

dmsResourceType OBJECT-TYPE
    SYNTAX      INTEGER
        { file(1),
          device(2),
          thread(3),
          process(4)
        }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Resource type. Its value may be ‘file’, ‘device’, ‘thread’, or ‘process’."
    ::= { dmsResourceEntry 2 }

dmsResourceName OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "It has a different content 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 Operating System. It allows a resource
          to be identified uniquely."
    ::= { dmsResourceEntry 3 }

dmsResourceState OBJECT-TYPE
    SYNTAX      INTEGER
        { open(1),
          reading(2),
          writing(3),
          ready(4),
          waiting(5),
          using(6),
          running(7),
          suspended(8)
        }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "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 a thread or a process, it has one of ‘running’ and ‘suspended’."
    ::= { dmsResourceEntry 4 }
END
```