Journal of Network and Systems Management, Plenum Press, Vol. 6, No. 2, June 1998.

Design and Implementation of TMN SMK System Using CORBA ORB

Jong-Tae Park, Su-Ho Ha
School of Electronic and Electrical Eng.
Kyungpook National University
Taegu, Korea
Tel: +82-53-950-5543
Fax: +82-53-950-5505
Email: {park,shh}@ee.kyungpook.ac.kr

James Won-Ki Hong
Dept. of Computer Science
POSTECH
Pohang, Korea
Tel: +82-562-279-2244
Fax: +82-562-279-5699
Email: jwkhong@postech.ac.kr

Abstract

In Telecommunication Management Network (TMN), the interworking of manager and agent requires the sharing of management information defined as Shared Management Knowledge (SMK) in ITU-T Recommendation M.3010. SMK includes information on the protocol knowledge, management functions, managed object classes and their instances, and authorization capabilities. We have developed a CORBA-based TMN SMK system in order to provide a dynamic and efficient distributed processing environment for the exchange of management information between managers and agents. In this paper, we discuss the design issues and present the design of a CORBA-based SMK system including the access protocol for obtaining the SMK information from MIB and for SMK context negotiations. Also, our effort on the prototype implementation of an SMK system using ORBeline and OSIMIS is presented. The preliminary performance results of our prototype implementation show that ORB-based SMK information access outperforms CMIS-based SMK information access, and that our approach is an efficient way of exchanging SMK information.

Keywords: TMN, SMK, CORBA, Distributed Management

1. INTRODUCTION

In Telecommunication Management Network (TMN) [1], the interworking of manager and agent requires the sharing of management information between managers and agents. This management information is defined as Shared Management Knowledge (SMK) in ITU-T Recommendation M.3010 [2]. SMK includes the information on the protocol knowledge, management functions, managed object classes and their instances. In addition, the authorization scope and the containment relationships between managed objects are defined in SMK. Several management knowledge object are also defined in ITU-T Recommendation X.750 [3] in order to store management knowledge as the attribute values of managed objects. For example, "moClass repertoire objects" are defined for the managed object (MO) class and naming schema, and "discovery objects" are defined for the representation of information related to MO instances. As this approach tends to require many MO class definitions for the necessary SMK information, it may increase the size of Management Information Base (MIB) for the storage of MOs. What is needed is an efficient mechanism for the representation, storage and manipulation of SMK information. The implementation of SMK system is at the early stage of research and development throughout the world.

Presently, the architecture of management systems tends to move from the manger-agent structure of OSI management model [4] to distributed object-based architecture of OMG CORBA [5] and ISO ODP [6]. The distributed processing infrastructure of TMN, though it is currently based on that of OSI management model, is expected to adopt the distributed object-based structure for the efficient dynamic establishment of manager-agent relationships [7]. OMG CORBA is also used as a distributed processing infrastructure in the OMNIPoint management model [8].

Zhang and Seret [9] have used a distributed directory service system for the access of network management information. While the architecture presented in [9] supports the distributed processing, it is neither suitable for the storage of dynamic management information, nor based on object-oriented concept as it is in CORBA. The ODP computational model has been utilized in the RACE Project [10] for the sharing of management information such as SMK. However, SMK is mainly defined between service layers, and the realization of the system is at the very early stage. Recently, Pratten, et al., [11] presented an architecture of resource information management system which utilizes the ODP Trader concepts to facilitate the management and use of distributed system resources. A recent joint work by X/Open and NMF on inter-domain management (JIDM) has been focusing on providing a solution for automatically translating GDMO/ASN.1 to CORBA IDL [12], which attempts to accommodate the interoperability between OSI-based and CORBA-based management systems.

In this paper, we present the design of a CORBA-based SMK system including the procedures of obtaining the SMK information from MIB and of SMK context negotiations. This system is based on the architecture where the client objects can access the SMK information through the CORBA IDL interface, independent of the implementation of server objects which provide the SMK access service. The communication mechanism of the CORBA Object Request Broker (ORB) enables the location transparency between distributed objects to be supported, and this provides the dynamic, efficient distributed processing environment. The OSIMIS management platform [13] developed by University College London and ORBeline [14] of PostModern Computing Technologies Inc. are used for the underlying distributed platform environment for the implementation of the SMK system. A preliminary work of this paper has been presented in [15].

The organization of this paper is as follows. Section 2 presents an overview of TMN SMK. Section 3 presents the modeling of SMK using CORBA IDL. The design of a CORBA-based SMK system is presented in Section 4. Section 5 describes a prototype implementation of our design. Section 6 presents a performance evaluation of our prototype. Section 7 summarizes our current work and suggests some future work.

2. OVERVIEW OF TMN SMK

In this section, we present the basic concepts of SMK, context negotiations in SMK and the types of SMK information.

2.1. SMK Concepts

The exchange of management information between two function blocks in TMN requires the common view about the protocol knowledge, management functions, managed object classes and their instances, and authorized capabilities. Such information is collectively referred to as the shared management knowledge (SMK) and is defined in ITU-T M.3010 [2] and in ITU-T X.750 [3]. SMK is required to perform management functions and to obtain the knowledge on the specific role as a manager or an agent in addition to the specific options for each function. The method of trial and error can be used for the acknowledgement of this knowledge, but more appropriate mechanism is desired.

Instances of MO classes constitute the most important knowledge for the management communication interface. The CMIS scoping can be used for the retrieval of this knowledge. As for the MO instances, the managed object classes are also important knowledge for the interface. In particular, it is necessary to find out MO classes for each management communication interface pair. The CMIS scoping is not a suitable technique for the identification of MO classes, and thus a more generic identification mechanism is necessary.

Figure 1 illustrates the SMK between two management systems. As SMK is defined between a pair of manager and agent, two SMKs actually exist (i.e., from system A to system B, and from system B to system C). Although SMK does not exist between system A and system C, they could be indirectly interworked using system B as an intermediate system.

2.2. Context Negotiations in SMK

The systems involved in the exchange of management information should establish and modify the contents of SMK between them. The context negotiation is required before SMK is exchanged and understood between a pair of management interfaces. Since the management interface is dependent on the management application and the requirement of management policy, the interface may request the different type of context negotiation. There exist two types of context negotiations: static and dynamic. In static context negotiation, SMK is exchanged only for a definite time and is fixed for some contractual time. In dynamic context negotiation, SMK is exchanged throughout the association by multiple interactions. Static and dynamic context negotiations are not mutually exclusive, thus a management interface may select and utilize both types of negotiation together.

2.3. SMK Information

The management knowledge which is shared between a manager and an agent can be categorized into the following types: instance information, repertoire, definition information, and other information. Each type of information is listed below in detail.

Instance information: information currently in the MIB
- managed object instance knowledge
- managed object instance relationship knowledge

Repertoire: information on what the managed system is capable of performing
- managed object class knowledge
- system management functions
- name bindings

Definition information: class templates
- management information definition knowledge
- managed object relationship knowledge

Other information
- management information services knowledge
- domains and polices knowledge

It is important to note that the frequency of changes to the above mentioned SMK information is different depending on the type. For example, definition information hardly changes once they are defined. Repertoire information is also fairly static and it will possibly change on the order of weeks to months. However, instance information is very dynamic, and changes may occur in the second to minute time scale. This has implications for appropriate access mechanisms. In Section 4, we will discuss our update mechanism and discuss how it accommodates this difference.

3. MODELING OF SMK USING CORBA IDL

Up to recently, the standards for the network and system management have been pursued by the several standardization bodies, according to the corresponding objects which they want to manage respectively. For example, ITU-T is in charge of TMN standardization, and ISO/IEC of OSI management, and IETF of Internet management (i.e., SNMP). The integration of these different standards is very difficult, since the interoperability of different management systems is dependent on the chosen options of the communication interface specification and managed object definition specified in each standard.

OMG CORBA has been proposed for interoperability in the computing domain, whereas the OSI management architecture of TMN model in the communication domain. In order to provide the system which could support both computing and communication domains, the information link between the external system interface and internal system interface should exist. For example, the managed objects defined in GDMO should be mapped into the form which could be understood by CORBA IDL.

Two of the most important aspects of designing a CORBA-based SMK system are: 1) SMK information modeling, and 2) SMK information access interface. Below, we present the definitions of the following information using CORBA IDL: managed systems, managed object classes, and managed object instances. Our SMK information modeling is based on the earlier work of JIDM on GDMO/ASN.1 to CORBA IDL translation [12]. When we carried out the modeling, there were no automatic GDMO to CORBA IDL mapping tools available and thus our mapping was carried out manually. However, automatic mapping tools are becoming available (such as GDIDL from SMILE Inc. [16]) and these would simplify and expediate the translation work. Finally, we also define the SMK information access interface using the CORBA IDL interface.

3.1. Definition of Data Types

The basic data types which are used to describe other data types are defined as follows.

```
typedef string Name;
typedef sequence <unsigned short> OID;
typedef string RDN;
typedef string DefinedType;

struct TemplateLabel {
    Name name;
    OID oid;
};
typedef sequence <TemplateLabel> TemplateList;
```

"Name" is a type for application entity (AE) title and the name of class and attribute. "OID" is defined for object identifier. Object identifier is represented by a sequence of integers. "RDN" is the type for the name of object instances. "TemplateLabel" represents the label for GDMO templates, and "TemplateList" is used to contain a list of templates.

3.2. Definition for Management Systems

The structure types for management systems are given in Figure 2. The "LayerEntity" structure reflects the layer management aspect and includes entities, several service access points (SAPs) and communication protocols. The "ApplicationEntity" structure represents the OSI-relevant aspect of an application process, and the "Domain" structure represents information on domains. "domainMembers" is an attribute which describes objects in the domain.

3.3. Definition for Managed Object Classes

A managed object class preserves the information on parent class, class name, attributes, actions, events, and packages. The name and identifier of an object are common information that all objects have. The <sequence> type which is defined in IDL is used to represent attributes, actions, events, and packages. IDL structures which represent managed object class and inheritance relationship are shown in Figure 3. Inheritance relationship is represented as a tree structure. Note that some features of GDMO have been intentionally omitted for the sake of simplicity. A complete list of IDL definitions can be found in [17].

3.4. Definition for Managed Object Instances

Managed object instances that MIB is composed of preserve the information on class, and also form containment relationship. Object name and name binding are important information for example. Each instance contains the reference for its superior object. Containment tree, like inheritance tree, is recursively defined. Figure 4 describes the CORBA IDL definition for managed object instances.

3.5. Definition for SMK Interface

An SMK interface is one that a manager invokes to acquire SMK. A management agent implements SMK server and provides SMK information. Manager accesses SMK information through the SMK interface which is defined in IDL and corresponds to a CORBA object.

A CORBA IDL interface has the following structure.

```
interface <identifier> [<inheritance_spec>]
{
 <interface_definition;>*
};
```

<interface_definition;> includes the declaration of type, constant, exception, operation, and attribute. In the declaration of operations, the operation names and parameter types are specified in detail. Figure 5 describes SMK interface in IDL.

4. DESIGN OF A TMN SMK SYSTEM

In this section, we present the design of a TMN SMK system using the CORBA technology. We first describe how CORBA has be integrated into Operation Systems. We then present the shared management knowledge access protocol and finally the design system architecture.

4.1. Integration of CORBA in Operation Systems

In TMN, the management functions such as configuration management and fault management are performed by the specific function blocks which have distributed processing capabilities. These management functions are realized by manipulating the managed objects with the services provided by CMIS. Each function block can be regarded as a unit for the provision of TMN management interface. The distributed processing capabilities of a function block can be realized by utilizing the distributed processing architecture of CORBA, where Object Request Broker (ORB) object could be assigned to each function block. This method can be treated as making ORB objects to exist as constituent element of a function block. ORB provides the distributed processing capability inside a function block as well as those between function blocks. An Operation System (OS) function block can use the Q3 interface for the communication with other function blocks, and use the communication function of ORB object for the mechanism of sharing of management knowledge. Figure 6 illustrates the employment of ORB objects in the operation systems, consisting of a manager and agent.

In this architecture, the manager plays the role of client requesting management information and the agent playing the role of server providing requested information. The communication mechanism for the acquisition of SMK is realized by that provided by ORB, not by CMIP/S. The client objects of a manager is associated with relevant implementation objects by locating the agents which are physically and functionally distributed.

Figure 7 shows the protocol architecture for management systems supporting the SMK access functionality. In a CORBA-based TMN SMK system, two sets of protocol stacks should coexist to support both the communication for TMN management interactions (via CMIP) and one for SMK interactions (via ORB). Management applications (managers and agents) interact with CMIP and its underlying protocol stack to perform network management functions. In addition, they exchange and use SMK information using ORB (e.g., over TCP/IP).

The rationale behind having two protocol stacks is as follows. Although extra overhead is required to keep both protocols in the management system, we believe that using a separate communication channel for the SMK interactions via ORB outweighs the extra overhead involved. The separate channel provides an "out-of-band" signalling for management interactions between the manager and agent. This advantage is enriched with the higher performance that can be provided by using ORB instead of CMIP as we will show in Section 6. Also, ORB supports interoperability of managers and agents in a heterogeneous distributed environment very well. Furthermore, the trend of replacing CMIP/S with CORBA ORB for manager/agent interactions in TMN environment is rapidly gaining acceptance [7]. We believe that our work can be thought as a transitional interim solution from CMIP/S only to ORB only environment.

4.2. Shared Management Knowledge Access Protocol

The context negotiation for SMK can be accomplished either prior to the establishment of association (static negotiation), or during the establishment (dynamic negotiation). In this section, we mainly focus on the static negotiation mechanism. Figure 8 illustrates the procedure for the exchange of management information between managing and managed systems. Below, we describe in detail the steps of the SMK access protocol.

Step 1) Management application sends a request message to a managed system to discover the name of the system which contains MOs. The SMK client object can use either broadcasting or multicasting technique in order to bind with an SMK server object. The manager can adjust the parameters for binding function in accordance with the degree of its knowledge on the managed system. In other words, if it has scarce knowledge on the managed system, it can use broadcasting technique for binding, otherwise it can use multicasting with some knowledge on the managed system. If the manager already possesses enough information about the managed system, this step can be omitted. The functions and their parameters for binding are described below.

```
SMKInterface *_bind (
        CORBA::Environment &_env,
        const char *object_name = NULL,
        const chat *host_name = NULL,
        const CORBA::BindOptions *_opt = NULL
);

SMKInterface *_bind(
        const char *object_name = NULL,
        const chat *host_name = NULL,
        const CORBA::BindOptions *_opt = NULL
);
```

Step 2) The management application can receive the object reference of the managed system as a response to the request. The binding procedure is described below.

```
SMKInterface *client_object = SMKInterface::_bind(parameter_list);
```

The <client_object> is the reference to an SMK server object.

Step 3) The management application requests the information on the managed system, using the object reference obtained in Step 2.

Step 4) The requested SMK information is transferred to the manager.

Step 5) The management application determines whether the desired management functions are supported or not by inspecting the acquired information. If the SMK information is not the desired, the request message is sent to the other managed system using the object reference and the control goes back to Step 3. Otherwise, go to Step 6.

Step 6) The management application establishes the management association with the managed system using the acquired SMK information.

Step 7) After the association establishment, the management application performs management activities on the appropriate MO using the supported management functions.

4.3. CORBA-based TMN SMK System Architecture

The SMK client object at the managing system is associated with the SMK server object at the managed system through the communication facility provided by the underlying ORB. The SMK information can be accessed by using the interface which is defined in IDL. This interface is independent of the object implementation. The manager system has relatively simple architecture which consists of CMIS interface and IDL interface. On the contrary, the managed system might have additional functions for the provision of SMK information. Figure 9 illustrates the architecture of SMK system based on CORBA.

The SMK server performs the functionality of providing the information on the managed system to the SMK client. SMK information Repository (SIR) exists in the managed system, and it is the place for the storage of SMK information. The SMK server can, with direct access to MIB, obtain the management information without accessing the SIR. In cases where the contents of MIB changes dynamically (e.g., instance information), and the size of MIB is relatively small, it may be more efficient to directly access the MIB to get the SMK. However, if the contents of MIB is fairly static (e.g., definition information such as classes, attributes and repertoire information), and the size of MIB is big, it may be more efficient to use SIR. This can be achieved by making the SMK server access the SMK information and store them in SIR, when the agent process is activated. When the SMK client object requests the SMK information, the SIR is accessed for the provision of required SMK information. In this case, the MIB is only accessed once each time the agent process is activated. The management of SMK information in a sense is similar to the view maintenance problem in distributed database management system. The suitable SMK management strategy may depend on the update frequency and the size of MIB, the complexity of MIB access functionality, and the semantic correctness of SMK update operations.

As the contents of MIB is changed, the update procedure, which is described below, is required.

Step 1) The changes in managed objects in MIB triggers a notification to the CMIS agent of the updated information.

Step 2) The CMIS agent transfers the notification from the managed object to the manager, and sets the change-flag in SIR.

Step 3) The SMK server inspects the change flag. If it is set, it reflects the updated MIB information to SIR, and resets the flag. Otherwise, it provides immediately service in response to the request from the client.

The update strategy presented here is fairly generic in the sense that it could handle very static information such as definition and repertoire information as well as very dynamic information such as instance information in a uniform way.

5. IMPLEMENTATION

This section describes our implementation effort based on the design presented in the previous section. We first describe the design of the SMK server process and then describe the prototype implementation.

5.1. Design of Server Process

There exist two processes at a managed system: an agent process which performs management functions for the manager, and an SMK server process which provides SMK information access service. SMK server process consists of several SMK server objects, and each SMK server object has an internal reference point for the access of the MIB in an agent.

Figure 10 shows the code for the implementation of SMK server object. The "SMKInterface_impl" class is generated by the IDL compiler. The "SMKObject" class inherits properties from the "SMKInterface_impl" class, and new member functions, SMKObject and GetMO are defined, whose declarations observe the SMK interface specification described in IDL. The variable "mo", being internally defined in the class, is a reference point for the access of MIB information at the agent, and its value can be retrieved by the invocation of the function SMKObject::GetMO. The variable "flag" indicates the occurrence of any change in the contents of SMK information.

As the SMK information is changed, the updated information is notified to the SMK server object. Since the SMK server object only receives the updated information, the system load can be reduced. The communication between SMK server object and an agent is established by using shared memory in which the type of SMK information is recorded. As the flag in SIR is set to ON, and the content recorded in the shared memory is identical to that of SMK information serviced by itself, each member function of SMK server object may update SIR by accessing MIB in the agent.

The SMK server process can be generated by forking the agent process. An agent generates a child process after initializing the MIB when running the program. The child process creates the SMK server object, and brings in the reference point to the MIB. The code segment for the creation of SMK server object process at an agent is given below.

```
//  ISODECoordinator coordinator;
//  CMISAgent           agent;

        // initialize syntaxes
        initialiseSyntaxes(syntaxes);
        // initialize the agent
        agent.initialise(environment_variable, service);
        // initialize MIB
        MO::initialiseMIB(MIB_INIT_FILE);

        // create SMK server process
        if (fork() == 0) {
                // create SMK server object
                SMKObject *smkObj = new SMKObject(object_name);
                // get reference pointer to MIB
                MO **root       = MO::getRoot();
                smkObj->GetMO(root->getWholeSubtree());
                CORBA::BOA::impl_is_ready();
        }
        coordinator.listen();
```

5.2 Prototype Implementation

Figure 11 illustrates the prototype implementation of a CORBA-based TMN SMK system. We used the OSIMIS 4.0 platform [13] for implementing OSI manager-agent interactions. CORBA IDL and ORB implementation embedded in ORBeline from PostModern Computing Technologies [14] was used as a CORBA implementation.

When an agent starts up, the SMK server ORB object within it retrieves all the pertinent information in the MIB. When a manager wishes to obtain SMK information from an agent, its SMK client ORB object contacts ORBeline's Smart Agent to discover the binding information of the SMK server ORB object. The manager then contacts the agent to request for the appropriate SMK information.

6. PERFORMANCE EVALUATION

In this section, we present some performance results of our prototype implementation. Figure 12 illustrates the system configuration for the performance evaluation. The first host (Host A) is configured with a manager and an agent and the second host (Host B) is configured with only an agent. The manager in turn consists of a manager process and an SMK client process. Both agents consist of an agent process and an SMK server process. For the simplicity of our evaluation, we have used an identical MIB, namely the UNIX-MIB which represents the number of users logged on each system.

Since there does not exist any SMK system implementation (which we are aware of), it is difficult to compare our results with others. Thus, we simply present our results and describe the performance characteristics that we believe are important.

Table I shows the performance results of binding times between an SMK client object to an SMK server object. Binding time represents the average time (from 100 trials) for an SMK client object to discover the desired SMK server object and connecting to it, and thus this is a very important performance metric. The times are measured in milliseconds (msec). The columns represent the number of SMK server objects. We have used a broadcasting technique for locating and binding a desired SMK server object. In order to evaluate the scalibility of our solution, we have measured the binding times by doubling the number of SMK server objects repeatedly. The results shown in Table I indicate the increase of binding times as the number of SMK server objects increases. However, the increase is not directly proportional to the rate of increase.

Our next evaluation involved measuring the times for SMK operations using the CORBA SMK interface and for the OSI management operations using the CMIS M-primitives. Figure 13 illustrates the interactions between SMK object entities and between management entities. We have measured the following times as indicated in the figure.

¨ ç: SMK client object requests a service to SMK server object
¨ è: SMK returns the requested information
¨ é: Manager requests a service to an agent
¨ ê: Agent returns the results of the request

In the measurement, we have not counted the times taken to perform the requested operations since they mainly depend on the internal implementation of SMK server object and manager/agent processes as well as the platforms these processes run on. The experiment we performed were from the same two hosts on the same network for both cases. The service requests used in the experiment were Get_MOClassList and Get_MOInstanceList for the ORB SMK interface and M_Get and M_Action for the CMIS management operations. We have essentially measured the communication cost of SMK operations via ORB versus CMIP/S.

Table II illustrates the times measured for the SMK service requests and replies using ORB and Table III illustrates the times measured for the CMIS requests and replies. The performance results as shown in these two tables indicate that the CMIS operations take several times longer than the ORB SMK operations. The results indicate that the overhead of using the SMK client and server objects compared to the manager-agent operations is not significant. The approach suggested in the NMF OMNIPoint uses the SMK MOs explicitly, which requires the use of manager-agent CMIS operations for SMK operations [8]. The OMNIPoint's approach would result in the performance degradation considerably since CMIS operations take much longer. Although there is a disadvantage of having two separate protocol stacks in the management system, it shows that our approach is more efficient way of exchanging SMK information in addition to the added "out-of-band" signaling and increased interoperability advantages as mentioned earlier.

7. CONCLUDING REMARKS

Shared Management Knowledge (SMK) is critical to the operation of management systems in TMN. Currently, there lacks close investigation on the detailed design or the implementation effort in realizing SMK systems.

In this paper, we have examined important issues in designing an SMK system for supporting management activities in TMN. We have provided a detailed design of TMN SMK system based on OMG CORBA. We modeled various SMK information using CORBA IDL data types and the SMK access interface using CORBA IDL interface. We have also presented a TMN SMK system architecture based on CORBA concepts. An SMK Access Protocol between a manager and an agent has been designed and presented in detail. Prototype implementation of our design has been completed using the OSIMIS platform and ORBeline. The preliminary performance results of our prototype implementation show that ORB-based SMK information access outperforms CMIS-based SMK information access and that our approach is an efficient way of exchanging SMK information.

For future work, we plan to implement our SMK system on other CORBA platforms and OSI platforms and perform more extensive performance comparisons. We also plan to investigate extending the SMK Access Protocol that can accommodate dynamic context negotiation. We are currently working on the design and implementation of SMK system using other distributed processing supporting entities such as the X.500 Directory Service and ODP Trader [18].

ACKNOWLEDGMENTS

Parts of this paper were presented at NOMS96 and DSOM96. The authors would like to thank Mr. Joong-Gu Song and Mr. Sung-Bum Kim of Korea Telecom Research Laboratories for their valuable comments. This work was supported in part by POSTECH Research Grant No. P96001.

REFERENCES

1. J. K. Shrewsbury, An Introduction to TMN, *Journal of Network and Systems Management*, Vol. 3, No. 1, March 1995, pp. 13-38.
2. ITU-T Draft Recommendations M.3010, Principles of Telecommunication Management Network, 1995.

3. ITU-T Recommendation X.750, Management Knowledge Management Function, 1993.
4. ISO, Information Processing Systems - Open Systems Interconnection - Systems Management Overview, International Standard 10040, 1991.
5. Open Management Group, *The Common Object Request Broker: Architecture and Specification*, Dec. 1993.
6. ITU-TS, Basic Reference Model of Open Distributed Processing Part 1: Overview and Guide to the Use of the Reference Model, ITU-TS Rec. X.901, ISO/IEC 10746-1, July 1992.
7. OMG Telecom Special Interest Group, CORBA-Based Telecommunication Network Management System, OMG White Paper Draft 2, January 1996.
8. Network Management Forum, A Technical Strategy: Implementing TMN Using OMNIPoint, pp. 65-77, 1994.
9. Xinxin Zhang and Dominique Seret, Supporting network management through distributed directory service, *IEEE Journal on Selected Areas in Communications*, Vol. 12, No. 6, August 1994, pp. 1000 - 1010.
10. RIC Association International, RACE Common Functional Specifications: Document 13 - Evolution 2, December 1993.
11. A. W. Pratten, J. W. Hong, M. A. Bauer, A Resource Management System Based on the ODP Trader Concepts and X.500, *Proc. of IEEE/IFIP International Symposium on Integrated Network Management*, Santa Barbara, May 1995, pp. 118-130.
12. X/Open Company Ltd., Inter-Domain Management Specifications: Specification Translation, X/Open Preliminary Specification, June 1996.
13. G. Pavlou, The OSIMIS TMN platform: support for multiple-technology integrated management system, *RACE Conference on Intelligence in Broadband Services and Network*, November 1993.
14. PostModern Computing Technologies, *ORBeline User's Guide*, 1994.
15. J. T. Park, S. H. Ha, J. W. Hong and J. G. Song, Design and Implementation of a CORBA-Based TMN SMK System, *Proc. of the IEEE/IFIP 1996 Network Operations and Management Symposium*, Kyoto, Japan, April 1996, pp. 64-74.
16. SMILE Inc., GDIDL: a GDMO/ASN.1 to IDL Translation, Paris, France, 1996.
17. Su-Ho Ha, CORBA ORB-based TMN SMK System, *MSc Thesis*, School of Electronic and Electrical Engineering, Kyungpook National University, Korea, January 1996.
18. J. W. Hong and J. T. Park, Supporting Distributed Management in Telecommunications Management Network (TMN) Environment, *Proc. of the Second International Workshop on Systems Management*, Toronto, Canada, June 1996, pp. 148-157.

Jong-Tae Park received the B.S.E.E. degree from Kyungpook National University, Korea and the M.S.E.E. degree from Seoul National University, Korea, respectively. He received the Ph.D. degree in Computer Science and Engineering from the University of Michigan, U.S.A, in 1987. From 1987 to 1988, he was at AT&T Bell Laboratories, New Jersey, working on network management and service provisioning. Since 1989, he has been working at the School of Electronic and Electrical Engineering at Kyungpook National University, Korea, where he is now an associate professor. He is a chairman of the Committee of Network Operations and Management of Korean Information Communication Society. His research interests include Telecommunication Management Network (TMN), Distributed Database Systems, Network and Resource Management in ATM Networks, Multimedia Communication, and Personal Communication Services.

Su-Ho Ha received the B.S.E.E and the M.S.E.E degrees in Electronic Engineering from Kyungpook National University, Korea, in 1994 and 1996, respectively. He is currently working in the Production Engineering Laboratory of Wire Equipment Engineering Group Team at LG Information & Communications, Ltd. He is now developing the management and control software for Optical Transmission Equipment.

James Won-Ki Hong is an assistant professor in the Dept. of Computer Science and Engineering, Pohang University of Science and Technology (POSTECH), Pohang, Korea. He has been with POSTECH since May 1995. Prior to joining POSTECH, he was a research professor in the Dept. of Computer Science, University of Western Ontario, London, Canada, where he worked on the CORDS project and MANDAS project. Dr. Hong received the BSc and MSc degrees from the University of Western Ontario in 1983 and 1985, respectively, and the PhD degree from the University of Waterloo, Waterloo, Canada in 1991. His research interests include network and systems management, distributed computing and multimedia systems. He is a member of IEEE Communications Society and ACM.

List of Figures

List of Tables

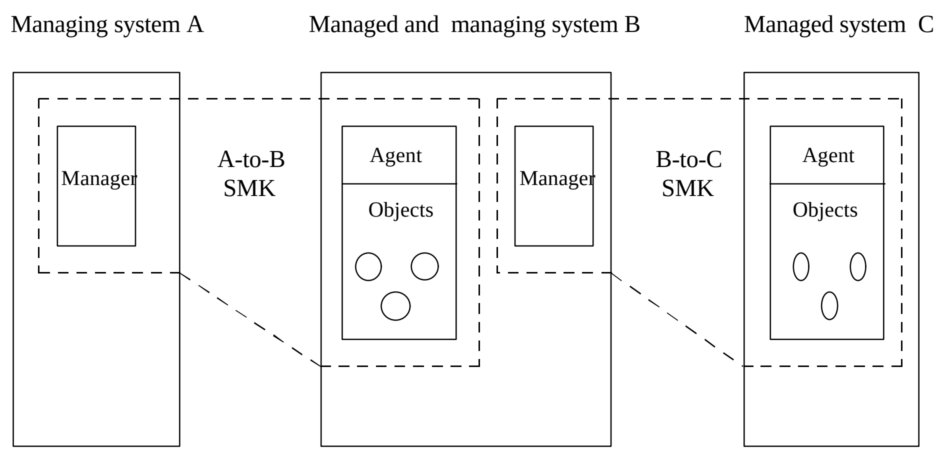


Figure 1. Shared Management Knowledge between Management Systems

```
typedef any Protocol;
typedef any Address;
struct LayerEntity {
    Name layerEntityName;
    Protocol layerProtocol;
    sequence <Service> layerServices;
};

typedef sequence <LayerEntity> LayerEntities;

struct ApplicationEntity {
    Name commonName;
    Address presentationAddress;
    Name localityName;
    Name organizationName;
    Name organizationalUnitName;
    Context supportedApplicationContext;
    string description;
};

struct Domain {
    Name domainName;
    any domainType;
    sequence <any> domainMembers;
    any domainManager;
    string description;
};
```

Figure 2. Management System Definition

```
struct ConditionalPackage {
    TemplateLabel label;
    string condition;
};

typedef sequence <ConditionalPackage> ConditionalPackageList;

struct MOClass {
    TemplateLabel derivedFrom;
    OID classOid;
    Name className;
    TemplateList characterizedBy;
    ConditionalPackageList conditioalPackages;
};

typedef sequence <MOClass> MOClassList;

struct InheritanceTree {
    MOClass base;
    sequence <InheritanceTree> subclasses;
};
```

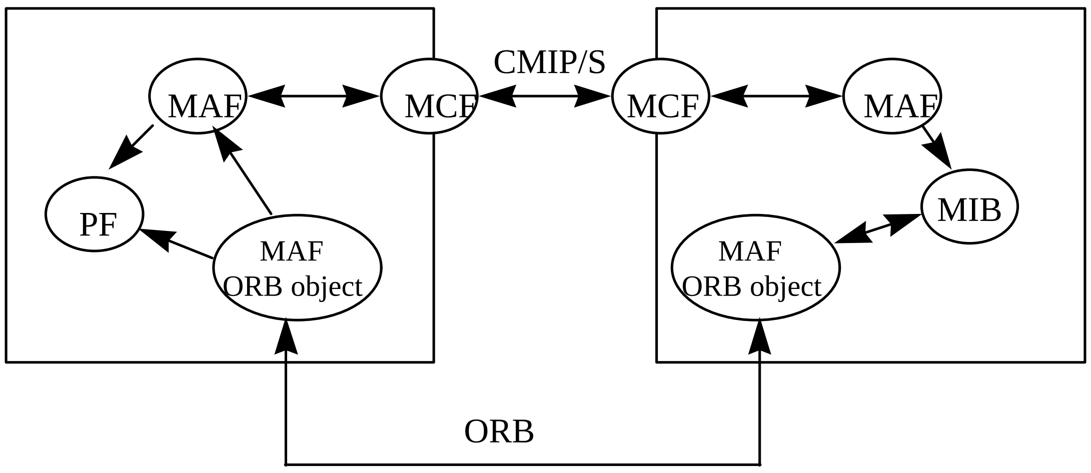
Figure 3. Managed Object Class Definition in CORBA IDL

```
struct MOInstance {
    OID nameBinding;
    RDN rdn;
    Name superior;
    MOClass  classInfo;
};

typedef sequence <MOInstance> MOInstanceList;

struct ContainmentTree {
    MOInstance base;
    sequence <ContainmentTree> subordinates;
};
```

Figure 4. Managed Object Instance Definition in CORBA IDL

```
interface SMKInterface {
    boolean Get_ApplicationEntity(out ApplicationEntity applEntity);
    boolean Get_LayerEntity(out LayerEntities layerEntities);
    boolean Get_Domain(out Domain domain);
    boolean Get_MOClassList(out MOClassList moClassList);
    boolean Get_InheritanceTree(out InheritanceTree inheritTree);
    boolean Get_MOInstanceList(out MOInstanceList moInstList);
    boolean Get_ContainmentTree(out ContainmentTree containTree);
    boolean Get_PackageInfo(in TemplateLabel packageLabel,
                           out PackageTemplate packageInfo);
    boolean Get_ParameterInfo(in TemplateLabel parameterLabel,
                             out ParameterTemplate parameterInfo);
    boolean Get_AttributeInfo(in TemplaeLabel attributeLabel,
                             out AttributeTemplate attributeInfo);
    boolean Get_AttributeGroupInfo(in TemplateLabel
                                    attributeGroupLabel,
                                  out attributeGroupTemplate
                                     attrGroupInfo);
    boolean Get_ActionInfo(in TemplateLabel actionLabel,
                          out ActionTemplate actionInfo);
    boolean Get_NotificationInfo(in TemplateLabel
                                  notificationLabel,
                               out NotificationTemplate notifInfo);
};
```

Figure 5. SMK Interface Defintion



MAF : Management Application Function
MCF : Message Communication Function

Figure 6. CORBA-based TMN Operation Systems Functional Architecture

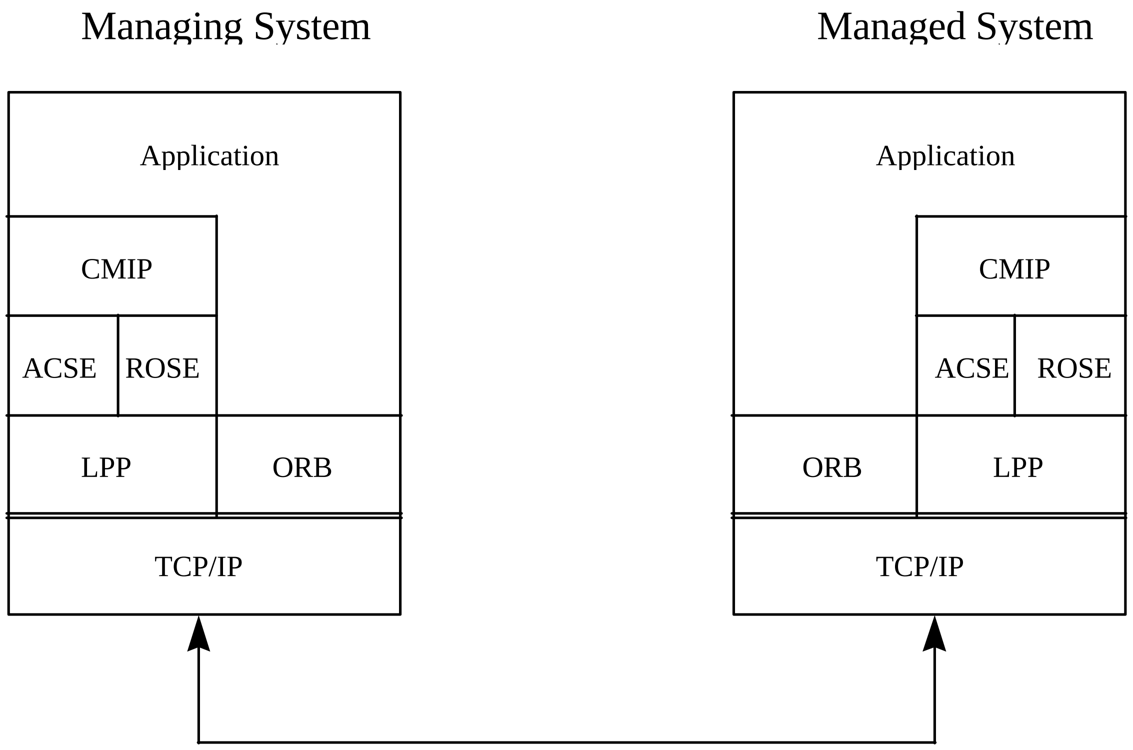


Figure 7. Protocol Architecture of TMN SMK System

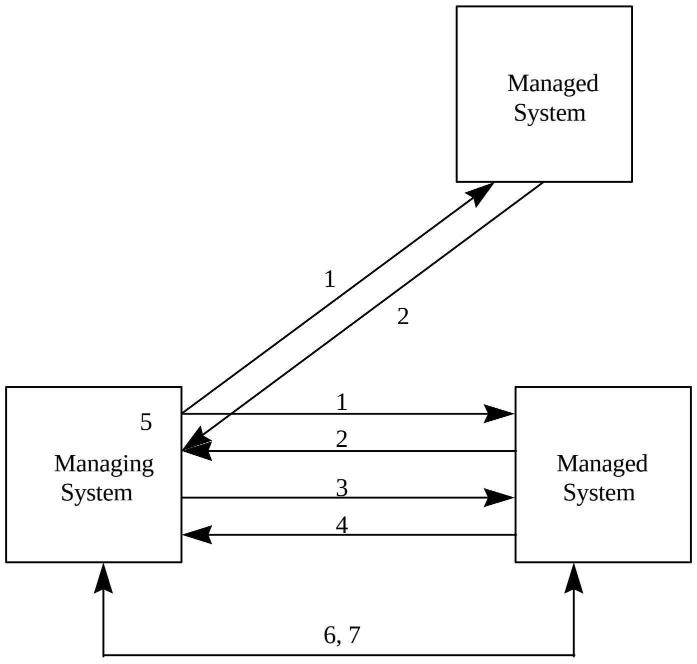


Figure 8. Management Information Exchange Procedure

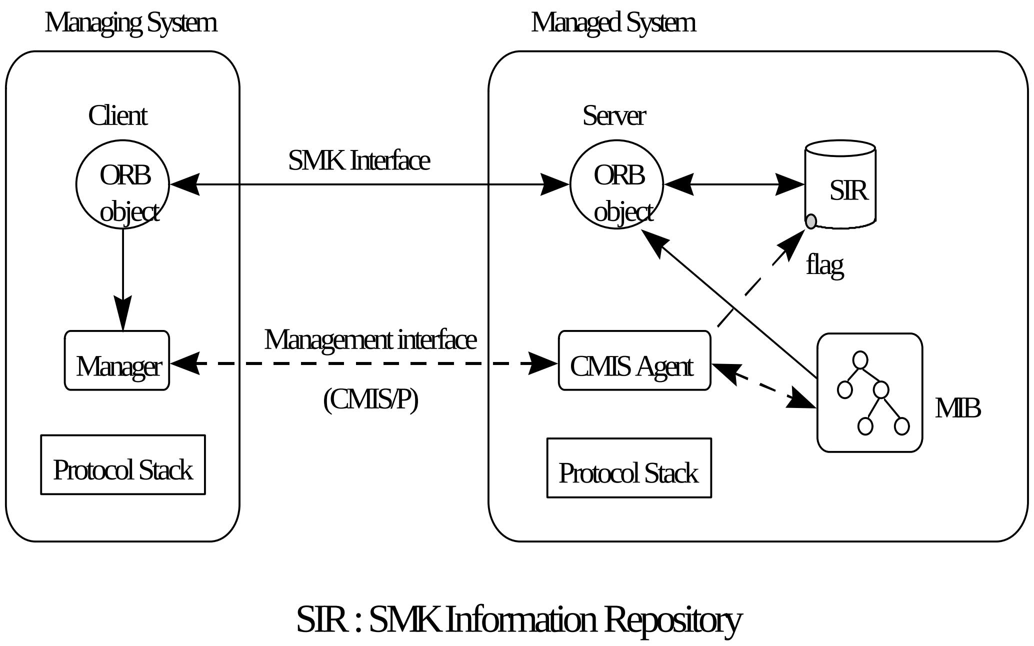


Figure 9. CORBA-based SMK System Architecture

```
class MO; // Declaration of MO  Class

class SMKObject : public SMKInterface_impl

      MO     **mo;    // reference pointer MO
      FILE   *SIB;    // SMK information base
      boolean       flag;
      int     shm_id;
public:
      SMKObject(const char *);
      static void GetMO(MO **);
      // SMK Interface declaration
;

MO** SMKObject::mo = (MO **) 0;

SMKObject::SMKObject(const char *object_name)
        : SMKInterface_impl(object_name)

      // SIB    = fopen(SIB_file_name);
      // flag   = OFF;
      // shm_id = shmget(key, size, permflags);


void  SMKObject::GetMO(MO **scopedMO)

      // mo = scopedMO;
```

Figure 10. Defintion of SMK Object Class

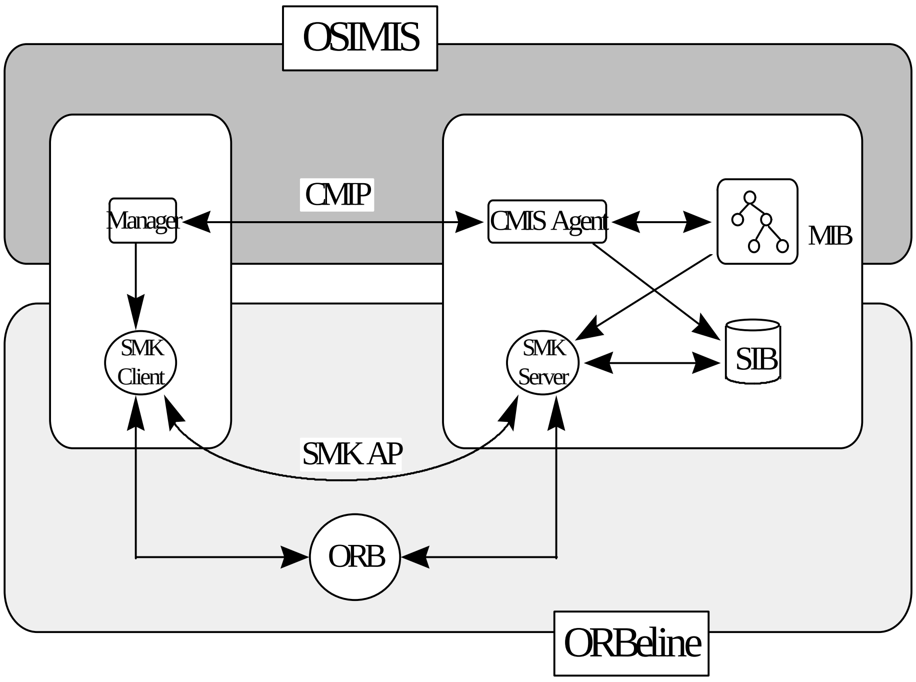


Figure 11. Prototype Implementation Architecture Using OSIMIS and ORBeline

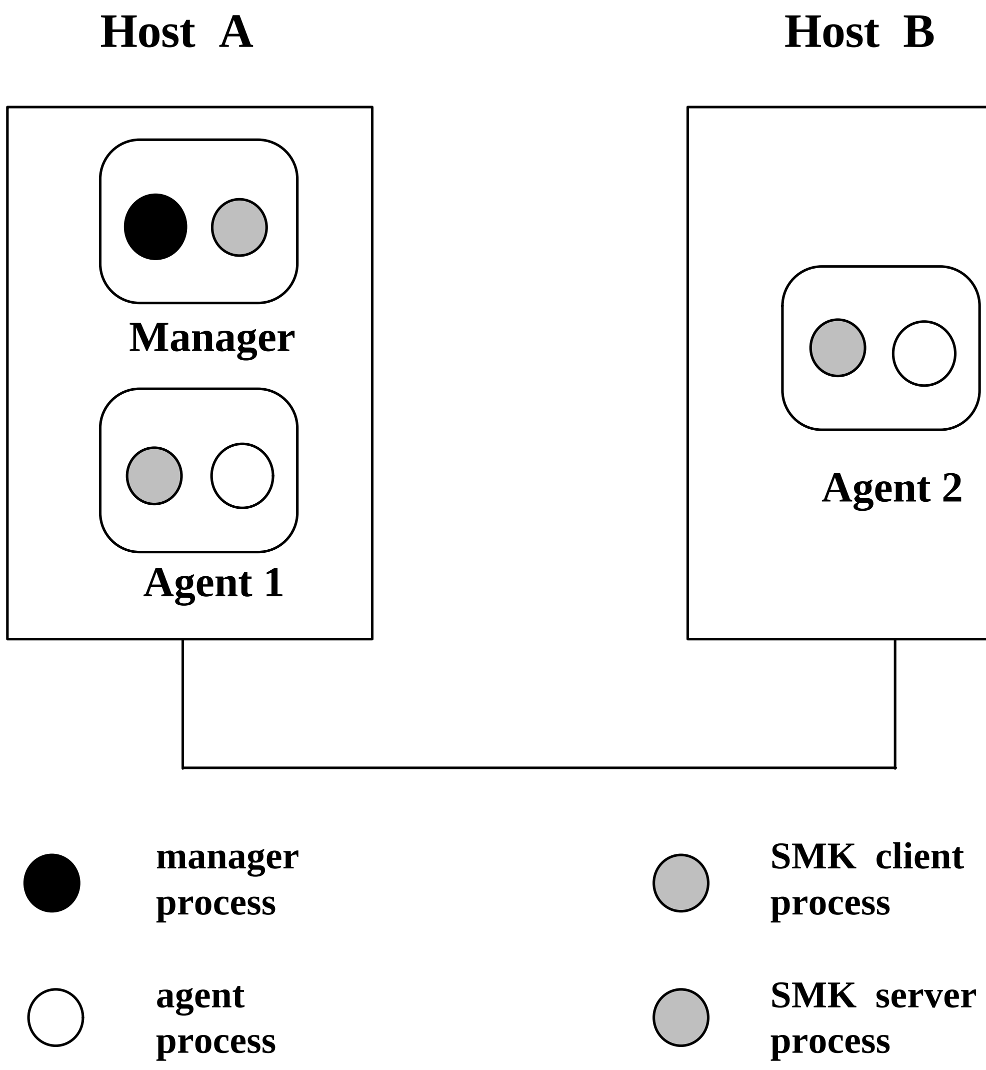


Figure 12. Configuration of Processes for Performance Evaluation

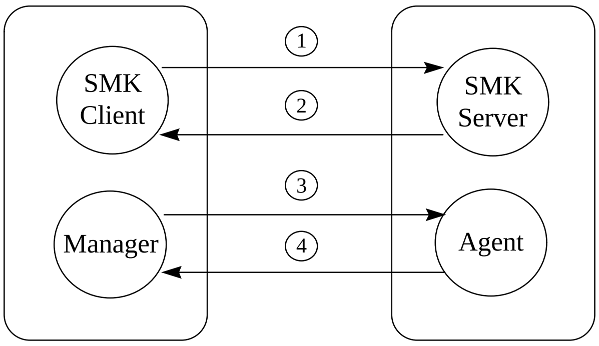


Figure 13. Measurement of request and response times of operations

Table I. Binding Times between SMK Objects

# of Server Objects	1	2	4	8	16	32
Avg. Time (units: msec)	65.29	73.08	87.89	119.97	180.25	302.14

Table II. Request and response times of service through ORB

Operations	Get_MOClassList			Get_MOInstanceList		
	Request	Response	Total	Request	Response	Total
Avg. Time (units: msec)	11.72	12.28	24.00	12.04	9.22	21.36

Table III. Request and response times of service through CMIP/S

Operations	M_GET			M_ACTION		
	Request	Response	Total	Request	Response	Total
Avg. Time (units: msec)	23.16	32.12	55.28	24.57	30.75	55.32