CORBA/CMIP Gateway Service Scheme for CORBA/TMN Integration

Moon-Sang Jeong, Kyu-Hyung Kim, Jeong-Hwan Kim, Joon-Heup Kwon and Jong-Tae Park
School of Electronic and Electrical Engineering, Kyungpook National University
{msjeong, khkim, chkim, jhkwon}@ain.kyungpook.ac.kr, park@ee.kyungpook.ac.kr

Abstract

TMN is an infrastructure which provides interfaces for interconnection between various types of operations systems and/or telecommunications equipment to manage telecommunication network and service, and management information is exchanged through these interfaces. Up to now, a lot of efforts have been made for the management of telecommunication networks and equipments, but less effort has been made for the realization of higher-layer service and business management. CORBA provides the infrastructure for interoperability of various object-oriented management applications in a distributed environment, and being widely used to develop distributed systems in many areas of information processing technologies. There are recently worldwide growing interests for applying CORBA technology to the realization of higher layer TMN management functions for service and business management. In this paper, we describe issues for integration of CORBA technology within TMN framework. And then, we design and implement GDMO/ASN.1 to IDL translator for specification translation. We suggest an architecture for CORBA/CMIP gateway, and show an interaction scenario for CORBA-based management applications to interact with TMN-based management objects

1. Introduction

ITU-T proposes the Telecommunication Management Network (TMN)[1] Recommendations M.3000 Series to manage telecommunication networks, services, and equipment. Currently, TMN technology is being widely applied to the management of SDH networks, and others such as ATM networks, and wireless/mobile networks. However, less effort has been made for the realization of higher-layer TMN management functions for service and business management.

On the other hand, Common Object Request Broker Architecture(CORBA)[2] has been widely adopted for developing distributed systems in many areas of information technology. CORBA provides the infrastructure for the interoperability of various object-oriented management applications in a distributed environment. With CORBA, users can transparently access to management information, independent of software or hardware platform. It provides a good portability of applications that are developed across multiple network management platforms.

The efficient integration of TMN and CORBA enables system developers to design and implement management solutions using either Interface Definition Language (IDL) or Guidelines for the Definition of Managed Object (GDMO) in a protocol-independent way, resulting in a productive, reliable integrated management solutions in multi-domain and multi-vendor environments. However, there are problems to be solved to achieve the benefits mentioned above. The integration of CORBA within TMN environment raises some particular problems related to the interoperability with legacy telecommunication management applications and protocols, as well as the QoS requirements and reliability.

It is known that the implementation of GDMO-based managed object class is not easy, and most commercial TMN platforms are very expensive. CORBA technology may enable us to design and implement the managed object class in an easier and less expensive ways.

A management information is defined using GDMO[3] and ASN.1[4,5] in TMN, but IDL is used in CORBA. The integration of CORBA with TMN requires the proper mapping between these different management information models.

One of the advantages that CORBA technology provides is the provision of location and distribution transparency of distributed objects. Object Request Broker (ORB) resolves the name of the objects using naming service and automatically finds out the address of the location of the objects. In TMN, a managed object is uniquely identified by Distinguished Name (DN) and its location is identified non-transparently through several steps of operations. The OSI manager first finds out the location of the agent, next, establishes an association with the agent, and finally sends an appropriate CMIS request. The integration of CORBA with TMN may require the resolving of this discrepancy in the naming and addressing methods.

OSI management provides efficient methods for accessing several managed objects in a request using scoping and filtering mechanism. On the other hand, CORBA basically relies on request/response access protocol, so that only one object can be accessed on a request from the client. These discrepancies in access methods and protocols should be resolved without losing any semantics in management information.

There could be a problem for transferring the event report generated by a GDMO-based managed object to the management application based on a CORBA IDL. In OSI management, the Event Forwarding Discriminator (EFD) can be used to determine the destination of the event report, and log control to backup the necessary management information for the purpose of storage and protection from information loss. On the other hand, the CORBA event service does not provide the end-to-end confirmed services, and does not store the management information, which might cause the loss of important management information. The integration of CORBA with TMN may have to take into account this discrepancy in event notification and log control.

2. Comparison of TMN and CORBA

In TMN, a managed object is defined by GDMO and data types used in objects are defined by ASN.1. A managed object represents resources in the management network and is an instance of a managed object class. Each managed object class has properties of attribute, action, notification and behaviour. When managers want to access to a managed object, it requests an agent to actually access the object. Each object is related to one another by containment tree which is built by name binding template. An object can be uniquely identified by Distinguished Name.

In a CORBA-based system, an information is encapsulated as objects using IDL. Each object is characterized by attributes and operations defined in an interface. Using ORB, a client object can transparently invoke a method on a server object. The ORB intercepts the call and is responsible for finding an object that can handle the request, pass it the parameters, invoke its method, and return the results. There dose not exist containment relationship between CORBA objects, and these objects have directory structure by naming service.

It is necessary to map between TMN managed objects and CORBA objects for integrating TMN and CORBA. This issue is resolved by GDMO/ASN.1 to IDL Translator. When GDMO-based managed objects are mapped to CORBA objects, it is important to maintain containment relationship of managed objects.

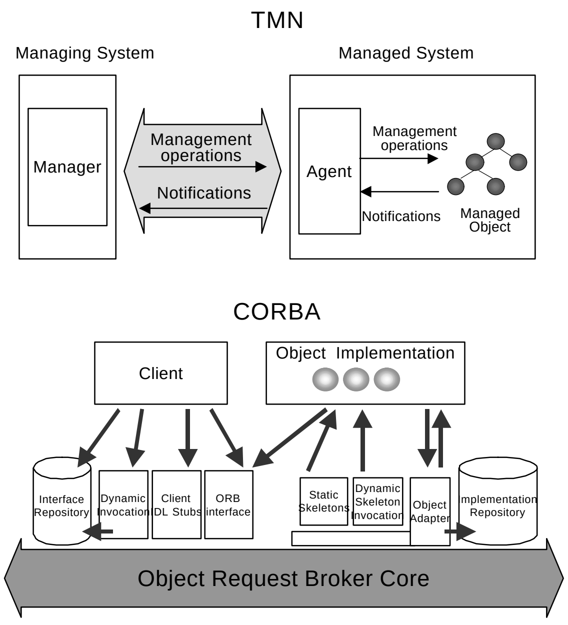


Figure 1: Interaction model

Figure 1 shows the interaction model of TMN and CORBA systems. The interaction model of TMN systems is based on the OSI manager/agent model. The manager component submits requests for management operations to agent in order to perform a specific interaction for managed object and receives notifications from agents. On the other hand, the agent component executes operation requests from the manager, and sends to manager notifications received from managed objects.

In CORBA, the object implementation component defines operations that implement a CORBA IDL interface [2]. Object implementations can be written in a variety of popular programming languages including C, C++, Java, Smalltalk and Ada. The client component is the program entity that invokes an operation on an object implementation. ORB core provides a mechanism for transparently communicating client requests to target object implementations. IDL stubs and skeleton are generated by an IDL compiler and are the static interfaces between a client and server. The dynamic invocation interface (DII) allows a client to directly access the underlying request mechanisms provided by an ORB and the dynamic skeleton interface (DSI) is the server side's equivalent to the client side's DII. The object adapter component provides the run-time environment for instantiating server objects, passing requests to them and assigning them object references. The ORB interface component provides various helper functions such as converting object references to strings and vice versa, and creating argument lists for requests made through the dynamic invocation interface. General Inter-ORB Protocol (GIOP) specifies a request format and transmission protocol that enables ORB-to-ORB interoperability. Internet Inter-ORB Protocol (IIOP) specifies a standardized interoperability protocol for the Internet. These differences of interaction model are resolved by using CORBA/CMIP gateway.

Table 1 shows the comparison of CMIP and CORBA. CMIP provides powerful information retrieval capability using scoping, filtering and linked replies, and also a powerful information modeling capability using GDMO [6]. On the other hand, the CORBA IDL interface is relatively easy to define and implement, and can be mapped separately to other programming languages. CORBA relies on request/response access protocol and client can access only one object at a time.

CMIP is relatively slow in performance and not easy to install and use. The network management platform based on CMIP is usually expensive and large in size. The benefits of CORBA are that it is a well-established and widely adopted open object-oriented middle standard which has become a critical part of software development. In addition, CORBA supports location transparency, and the integration ability of management information and services.

CMIP is only applied to telecommunication management, but CORBA can be applied to various areas, since it is independent of domain. In addition, CORBA may support JAVA programming language, and can be extended to Web-based management.

CMIP is applied to telecommunication management network and provides more diverse event notification services using EFD. CORBA handles event using event services which provide two models such as push and pull.

3. Implementation of GDMO/ASN.1 to CORBA IDL Translator

Figure 2 shows a implementation architecture of ASN.1 to CORBA IDL translator. A GDMO/ASN.1 to IDL Translator is based on specification translation of Joint Inter-Domain Management (JIDM) task force [7]. It translates GDMO/ASN.1 specifications into IDL and thus CORBA manager can deal with GDMO based managed objects as CORBA objects. The management information of TMN is composed of GDMO and ASN.1. A GDMO is the definition language of managed objects and an ASN.1 is definition of data type used in managed object.

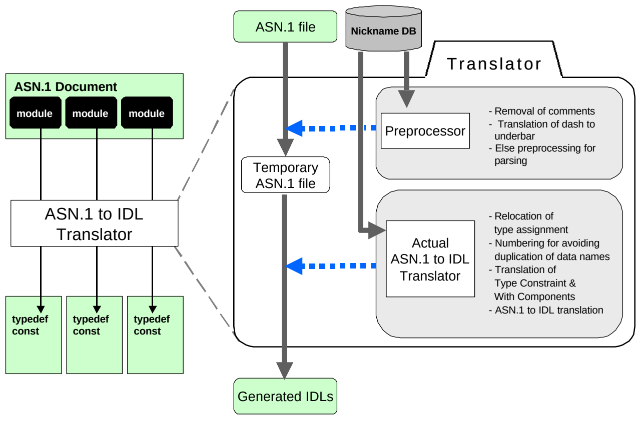

ASN.1 file
Nickname DB
Translator
ASN.1 Document
module
module
module
Preprocessor
- Removal of comments
- Translation of dash to underbar
- Else preprocessing for parsing
Temporary ASN.1 file
ASN.1 to IDL Translator
Actual ASN.1 to IDL Translator
- Relocation of type assignment
- Numbering for avoiding duplication of data names
- Translation of Type Constraint & With Components
- ASN.1 to IDL translation
typedef const
typedef const
typedef const
Generated IDLs

Figure 2 : ASN.1 to CORBA IDL Translator

ASN.1 to IDL Translator translates ASN.1 modules to CORBA IDL modules. ASN.1 modules are translated into type definitions and constant values for supporting other translated IDL interfaces. The translator is composed of preprocessor and actual translator. Preprocessor performs the following functions: removal of comments, translation of dash to underbar and other preprocessing for parsing. Actual translator uses temporary ASN.1 file which is generated by preprocessor. Actual translator performs the following functions: relocation of type assignment, numbering for avoiding duplication of data names, translation of Type Constraint and With Components, and ASN.1 to IDL translation. The translator uses nickname database for translated IDL module name. Nickname database has ASN.1 module name, its nickname and ASN.1 filename.

The referencing mechanism of ASN.1 and CORBA IDL is different from each other. When a translation is processed, type definitions are first translated, and then constant values are translated. Before actual translator translates type definitions, it reorders the type references in order to obtain a valid OMG IDL code. For example, when a data type is declared within a sequence, this data type must be predefined. Finally, the generated IDL is composed of three parts: type definition part, constant value declaration part and ConstValues Interface part.

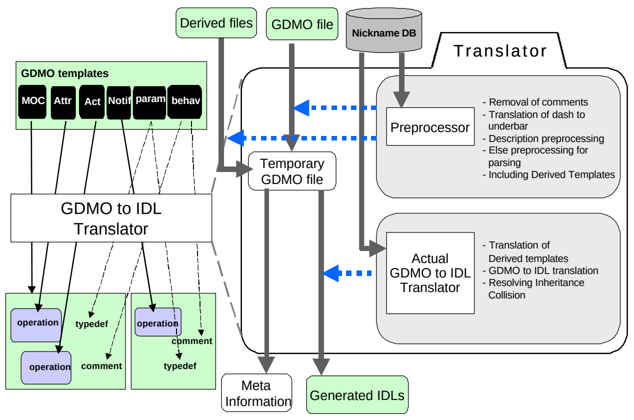

Derived files
GDMO file
Nickname DB
Translator
GDMO templates
MOC
Attr
Act
Notif
param
behav
Preprocessor
- Removal of comments
- Translation of dash to underbar
- Description preprocessing
- Else preprocessing for parsing
- Including Derived Templates
Temporary GDMO file
GDMO to IDL Translator
Actual GDMO to IDL Translator
- Translation of Derived templates
- GDMO to IDL translation
- Resolving Inheritance Collision
operation
typedef
operation
comment
operation
comment
typedef
Meta Information
Generated IDLs

Figure 3 : GDMO to CORBA IDL Translator

Figure 3 shows a implementation architecture of GDMO to CORBA IDL translator. GDMO to IDL Translator translates GDMO templates to several CORBA IDL interfaces (Primary interface, and Push and Pull Notification interfaces) and meta information. GDMO template has nine generic template structures: MOC, package, attribute, attribute group, action, notification, parameter, behaviour and name binding. MOC templates are translated into primary interfaces and attribute templates are translated into operations of the primary interface. Action templates are translated into operations of the primary interface and support multiple replies. Notifications are translated into push/pull interfaces for the support for notifications. Parameter and behaviour templates are translated into type definitions and comments of each interface. Meta information provides information that is used for the implementation of gateway and management application, and information for code generator in the platform. Management application can obtain an information for managed objects using meta information.

The GDMO to IDL translator is also composed of preprocessor and actual translator. GDMO to IDL preprocessor has some complex search function, and includes derived templates for accurate translation. Temporary GDMO files have original GDMO templates and information of derived templates. Actual translator includes the following functions: translation of derived templates, GDMO to IDL translation, and the resolution of inheritance collision. Translator uses nickname database for translated IDL module name and filename. Using nickname database, the translator can translate GDMO templates and ASN.1 type that is derived from other files.

In order to solve the inheritance collision problem, it checks again for the possible duplicated attribute and/or operation names across all ancestors in the inheritance tree. If an attribute is derived from another attribute, this is correctly translated to an operation using attribute naming database that is generated in each file.

Table 2 shows a translation example of top MO interface that is generated from top MOC in X.721 GDMO document, and Table 3 shows a translation example of ThresholdLevelInd type assignment in X.721 Attribute-ASN1Module by GDMO/ASN.1 to CORBA IDL translator.

In Table 2, top MOC template are mapped to IDL interface with same name, and attribute and behaviour templates in top MOC template are mapped to operations and comments, respectively. At this moment, Attribute-ASN1Module is mapped to X721Att IDL module using nickname database. And in Table 3, up and down SEQUENCE type assignments in ThresholdLevelInd CHOICE type assignment are mapped to ThresholdLevelIndUpType structure and ThresholdLevelIndDownType structure, respectively.

ThresholdLevelInd type assignment is mapped to ThresholdLevelIndTypeChoice enumerated type and ThresholdLevelIndType choice type.

IDL modules that are generated by GDMO/ASN.1 to CORBA IDL translator are used for developing CORBA-based network management application.

4. CORBA/CMIP Gateway and CORBA-based Management Application

The main role of the CORBA/CMIP gateway is to mediate the communication between the CORBA-based management applications and the TMN-based CMIP agents. The requirements for the system design are described below:

- CORBA-based management applications should be able to access OSI management objects in CMIP agents.
- The gateway must provide the CORBA-based management application with the IDL mapping information on CMIS operations such as M-GET,

Table 2 : Example of X.721 GDMO to CORBA IDL translation

GDMO X.721 - top MOC

```
 top MANAGED OBJECT CLASS
    CHARACTERIZED BY
        topPackage PACKAGE
          BEHAVIOUR topBehaviour;
          ATTRIBUTES objectClass GET,
                     nameBinding GET;;;
    CONDITIONAL PACKAGES
        packagesPackage PACKAGE
          ATTRIBUTES packages GET;
        REGISTERED AS {smi2Package 16};
        PRESENT IF ”...",
        allomorphicPackage PACKAGE
          ATTRIBUTES allomorphs GET;
        REGISTERED AS {smi2Package 17};
        PRESENT IF ”...";
REGISTERED AS   { smi2MObjectClass 14 };

allomorphs ATTRIBUTE
    WITH ATTRIBUTE SYNTAX
       Attribute-ASN1Module.Allomorphs;
    MATCHES FOR EQUALITY, SET-COMPARISON,
                SET-INTERSECTION;
REGISTERED AS {smi2AttributeID 50};

nameBinding ATTRIBUTE
    WITH ATTRIBUTE SYNTAX
       Attribute-ASN1Module.NameBinding;
    MATCHES FOR EQUALITY;
REGISTERED AS {smi2AttributeID 63};
...
```

X721.idl - top MO interface

```
// ---------------------------------------------
//     INTERFACE    top
// ---------------------------------------------

interface top :  OSIMgmt::ManagedObject
{

    // ----- Mandatory Package : topPackage
    //  BEHAVIOUR : topBehaviour
    //  ”..."

    //  ATTRIBUTE : objectClass
    X721Att::ObjectClassType objectClassGet()
            raises(ATTRIBUTE_ERRORS);

    //  ATTRIBUTE : nameBinding
    X721Att::NameBindingType nameBindingGet()
            raises(ATTRIBUTE_ERRORS);


    // ----- Conditional Package : packagesPackage
    //  PRESENT IF :
    //  ”..."

    //  ATTRIBUTE : packages
    X721Att::PackagesType packagesGet()
            raises(ATTRIBUTE_ERRORS);


    // ----- Conditional Package : allomorphicPackage
    //  PRESENT IF :
    //  ”..."

    //  ATTRIBUTE : allomorphs
    X721Att::AllomorphsType allomorphsGet()
            raises(ATTRIBUTE_ERRORS);

};
```

Table 3 : Example of X.721 ASN.1 to CORBA IDL translation

ASN.1 X.721 - Attribute-ASN1Module

```
  ...

  ThresholdLevelInd ::= CHOICE {
          up [1] SEQUENCE {
              high ObservedValue ,
              low ObservedValue  OPTIONAL
          } ,
          down [2] SEQUENCE {
              high ObservedValue ,
              low ObservedValue
          }
   }
...
```

X721Att.idl

```
// Mapping of the ASN.1 ThresholdLevelInd Type
union ThresholdLevelIndUpLowTypeOpt switch (boolean)  {
       case TRUE : ObservedValueType value;
};
struct ThresholdLevelIndUpType  {
       ObservedValueType   high;
       ThresholdLevelIndUpLowTypeOpt  low;
};
struct ThresholdLevelIndDownType  {
       ObservedValueType   high;
       ObservedValueType   low;
};
enum ThresholdLevelIndTypeChoice  {
       upChoice ,
       downChoice
};
union ThresholdLevelIndType
   switch (ThresholdLevelIndTypeChoice)  {
       case upChoice  :  ThresholdLevelIndUpType up;
       case downChoice  :  ThresholdLevelIndDownType down;
};
```

M-SET, M-CREATE, M-DELETE, M-ACTION.

- The gateway must support the scoping and filtering functions of CMIP.
- The gateway should be able to receive events from CMIP agents and send them to the management applications.
- The gateway should provide location and distribution transparencies.

Figure 4 shows CORBA/CMIP gateway and CORBA-based management application architecture.

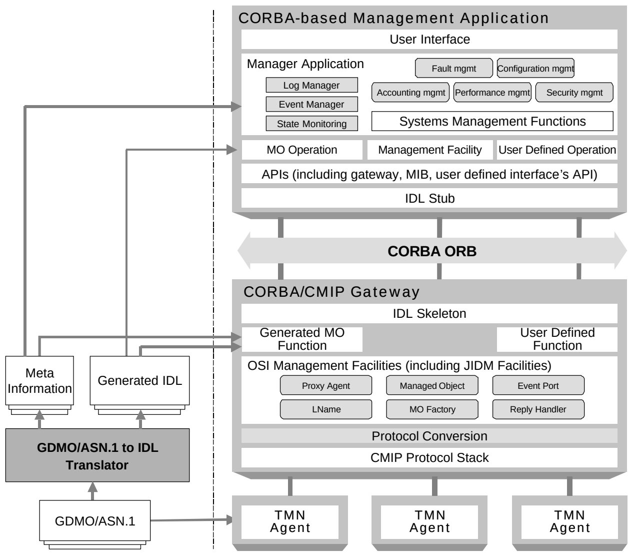


Figure 4 : CORBA/CMIP Gateway and Management Application Architectures

The gateway functions may be implemented using the CORBA IDL interfaces provided by the following components: Proxy Agent, Managed Object Factory, Proxy MO, and Event Port. Proxy Agent allows the CORBA-based management application access to proxy MOs which reflect the Managed Objects (MOs) in the CMIP agent. Also, the response from the CMIP agent may be transmitted to the CORBA-based management application through the proxy agent. Managed Object Factory creates the instance of a proxy MO which is corresponding to TMN managed object. Each of Proxy MOs is corresponding to the GDMO object class. The instances of these can invoke methods to access the OSI management object instance. Event Port receives events from CMIP agents and sends events to CORBA-based management applications using standard Common Object Service Specification event services.

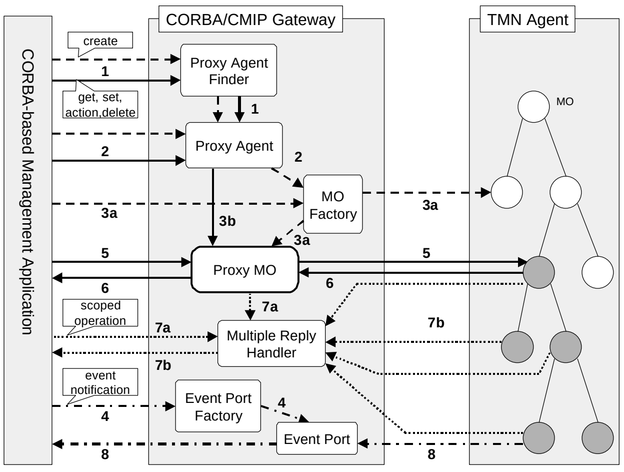


Figure 5: Interaction through the CORBA/CMIP Gateway

Figure 5 shows the interaction process between CORBA-based Management Application and TMN agent through the CORBA/CMIP gateway, which is based on JIDM Interaction Translation specification [8]. Create operation has somewhat different procedure from other operations such as get, set, action, and delete operations. Create operation is performed in accordance with the following order: 1, 2, and 3a, where the operation associated with the order number is explained in detail below. Operations related to get, set, action and delete are performed in the following order: 1, 2, 3b, 5, and 6. And scoped operations are performed in the following order: 1, 2, 3b, 5, and 7. An event can be emitted through the event port which is created by event port factory. Each operation is explained below.

1. Send a request to the Proxy Agent Finder to find or create a Proxy Agent.
2. Send a request to the Proxy Agent to find or create a Managed Object Factory.

3a. Send a request to the Managed Object Factory to create both a TMN Managed Object and an associated Proxy Managed Object.

3b. Find a CORBA object reference to an already existing managed object and create an associated Proxy Managed Object.

4. Send a request to the Event Port Factory to create an Event Port, then connect an event consumer with the

Event Port to receive the event.

5. Send CMIS request (get, set, action, and delete) from the CORBA-based management application to the TMN agent through the Proxy Managed Object.
6. Receive responses from the TMN Managed Object synchronously or asynchronously.
7. Receive multiple responses from TMN agent, and send it back to management application.
8. Receive an event from the TMN Managed Object by either using push or pull model.

5. Practical Application

Figure 6 shows the architecture of Customer Network Management (CNM) system for providing ATM CNM service [9] as an example of TMN/CORBA integration.

CNM service that is included in the TMN service management layer provides customer with management operations for the management of the sharing network resources. Customers can manage through the CNM MIB which contains the service provider's management information.

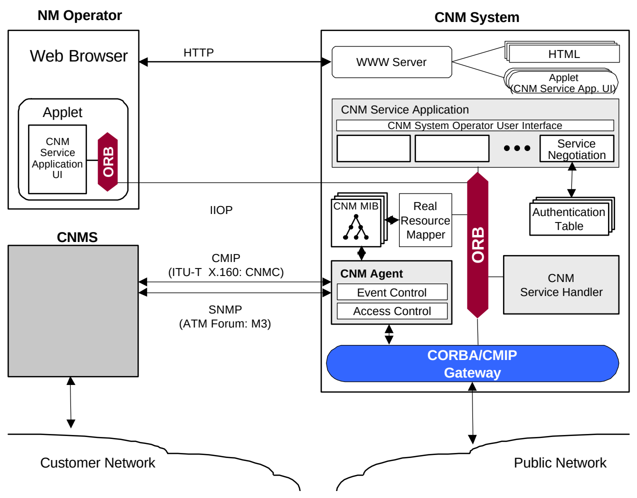


Figure 6 : Example of CORBA/TMN Integration - ATM CNM System

The proposed system consists of a Web browser which provides an interface to CNM service users and CNM system which is located at the side of the service provider. CNM service application within the CNM system is based on CORBA. On the other hand the public network is based on CMIP. Therefore, CORBA/CMIP gateway is used to communicate with public network which is in a different domain. In this case, CORBA-based CNM system can be implemented regardless of legacy CMIP-based public network.

CNM service application is implemented with comparative ease by using CORBA. In addition to, CNM system is able to communicate with Web browser since CORBA provides a flexibility to interact with WWW.

6. Conclusion

In this paper, we have investigated the technological issues related to the integration of TMN and CORBA, and compared the feature of CMIP and CORBA. And then, we have designed and implemented the GDMO/ASN.1 to IDL translator for specification translation. We have proposed CORBA/CMIP gateway and described interaction scenario through the gateway. Finally, we have applied CORBA/CMIP gateway to CNM system.

We will extend the capabilities of GDMO/ASN.1 to IDL translator. For the interaction translation, we intend to implement CORBA/CMIP gateway based on JIDM interaction translation specification. For the efficient realization of CORBA-based Management Application and CORBA/CMIP gateway, we are going to develop TMN/CORBA integration platform. The GDMO/ASN.1 to IDL Translator is now available at http://ain.kyungpook.ac.kr on your request.

References

[1] ITU-T Recommendation M.3010, *Principles for a Telecommunication Management Network*, May 1996.

[2] OMG, *CORBA 2.0/IIOP Specification*, Technical Document formal/97-02-25, February 1997.

[3] ITU-T Recommendation X.722, *Information technology - Open Systems Interconnection - Structure of management information: Guidelines for the definition of managed objects*, January 1992.

[4] ITU-T Recommendation X.208, *Specification of Abstract Syntax Notation One (ASN.1)*, 1988.

[5] ITU-T Recommendation X.680, *Information Technology Abstract Syntax Notation One(ASN.1): Specification of Basic Notation*, July 1994.

[6] ITU-T Recommendation X.711, *Information technology - Open Systems Interconnection - Common management information protocol: Specification*, October 1997.

[7] X/Open and NMF, *Inter-domain Management: Specification Translation*, Open Group Preliminary Specification P509, March 1997.

[8] OMG CORBA/TMN Interworking RFP, *JIDM Interaction Translation*, Relevant Document telecom/98-05-02, May 1998.

[9] J. W. Baek, T. J. Ha, J. T. Park, J. W. Hong, and S. B. Kim, "ATM Customer Network Management Using WWW and CORBA Technologies," *Proceeding of the IEEE/IFIP Network Operations and Management Symposium*, New Orleans, Louisiana, February 1998, pp.120-129.

[10] N. Soukouti and U. Hollberg, " Joint Inter Domain Management: CORBA, CMIP and SNMP, " *Proceeding of the 5th IFIP/IEEE International Symposium on Integrated Network Management*, May 1997, pp.153-164.

[11] Subrata Mazumdar, " Mapping of Common Management Information Services to CORBA Object Services Specification, " *Bell Laboratories TM #BL0112540-96.09.30-02,* September 1996.

[12] Graham Chen and Qinzheng Kong, " Integrated TMN Service Provisioning and Management Environment, " *Proceeding of the 5th IFIP/IEEE International Symposium on Integrated Network Management,* May 1997, pp.99-112.

Á ¤¹ ®» ó

1998.2 ° æ Ï ę Đ ³À À Ú Ọ ĐÇ Đ ç

1998 – Ç À ç° æ Ï ę Đ ³À À Ú Ọ Đ ¼ ® ç Á ¤

° ¼ É Đ ß ³ Æ ® Å ©° ü ®TMN, À Â Í Ý ü ®CORBA

± è± ÔÇ ü

1998.2 ° æ Ï ę Đ ³À À Ú Ọ ĐÇ Đ ç

1998 – Ç À ç° æ Ï ę Đ ³À À Ú Ọ Đ ¼ ® ç Á ¤

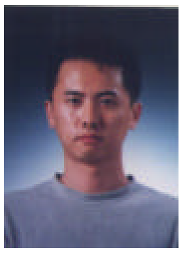
° ¼ É Đ ß ³ Æ ® Å ©° ü ®À Â Í Ý ° ü ®CORBA

± èÁ ¤È ¯

1999.2 ° æ Ï ę Đ ³À À Ú Ọ ĐÇ Đ ç

Ç À ç° æ Ï ę Đ ³À À Ú Ọ Đ¼ ® ç ú Á ¤

° ¼ É Đ ß ³ Æ ® Å ©° ü ®CORBA, TMN



± ÇÁ ØÇ ù

1999.2 ° æ Ï ę Đ ³À À Ú Ọ ĐÇ Đ ç

Ç À ç° æ Ï ę Đ ³À À Ú Ọ Đ¼ ® ç ° Á ¤

° ¼ É Đ ß ³ Æ ® Å ©° ü ®CORBA, TMN

¹ ÚÁ ¾Å Â

1978 ° æ Ï ę Đ ³À À Ú Ọ ĐÇ Đ ç

1981 ¼ ¿ Î ę Đ ³À À Ú Ọ Đ¼ ® ç

1981-1987 Univ. of Michigan , À ü â ¹ ×À ü ę Đ¹ ü ç

1987 - 1988 ¹ Ì ¹AT&T Bell ¿ ± ¸ ¼ Ò¿ ± ¿ ø

Ç À ç° æ Ï ę Đ ³À À Ă ü â Ọ Đ Î° Î ¾ ö

Ç Ñ Å ü Ą Đ «Å ü Ą Ó ¿ ë ü ® ± È ¸À § À å Á ª Ą ü Å Î ° Ọ Đ ã Ö Î¹ ×Ç Ñ Å ü ÅÀ ü ® À § ø° æ Ï ę Đ ³Á ª Ą ü Ą Đ úÇ Đ À å° æ Ï ´ ę Đ ³Â ¾ ¼ ëÁ ª Ą ü Å¿ ± ¼ Ò¼ À å

° ¼ É Đ ßTMN, À Â Í Ý ü ®¸ Ó ¾ ü ã îÅ ü Å Ã ° Å ÛÀ ü Ó ü Å ü ®