碩士學位論文

Object Web을 기반으로 한 분산 시스템 관리

濟州大學校 大學院
情報工學科

金　　康　　石

1998年　12月

Object Web을 기반으로 한 분산 시스템 관리

指導敎授 宋 旺 暾

金 康 石

이 論文을 工學 碩士學位 論文으로 提出함

1998年 12月

金康石의 工學 碩士學位 論文을 認准함

審査委員長 안 기 중 印

委 員 이 상 준 印

委 員 송 왕 철 印

濟州大學校 大學院
1998年 12月

The Object Web based Management of Distributed Systems

Gang-Suk Kim

(Supervised by professor Wang Cheol Song)

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF INFORMATION ENGINEERING
GRADUATE SCHOOL
CHEJU NATIONAL UNIVERSITY

1998. 12.

목 차

SUMMARY

So far, the relationship between the TMN physical components has been defined and modeled by being applied to a manager-agent structure used in the OSI management model. It is, however, expected that, in order to effectively manage a network under a widely-distributed computing environment, an architecture based on distributed objects, which is used in OMG/CORBA and ISO/ODP, will be chosen as the TMN management architecture. In addition, the Object Web translates the existing CORBA IDL into the Java language and uses IIOP for interaction between a client and a server under the existing Web environment. The client and the server through the Object Web, do not use HTTP, a Web protocol, but IIOP that is an inter-object communication protocol provided by the CORBA.

This paper proposes an architecture which consists of the gateway, the CORBA manager and the OSI agent. The CORBA manager is based on Object Web and can manage MOs in the OSI domain. The gateway has a role to translate CORBA IDLs into CMIS service and vice versa. Hence, an OSI agent can be abstracted into a CORBA agent. In this architecture, if a client has the Web browser, it can manage any OSI MOs in distributed environment. The client can invoke the management operations through CORBA server after it download the Java applet from HTTP server.

Ⅰ. 서론

현재까지 여러 종류의 통신망 관리의 국제 표준은 상호 연관성을 가지면서도 주요 관리 대상에 따라 각각 독립적으로 연구되어 왔다. ITU-T의 TMN은 통신망의 전송 시스템 및 교환 시스템을 관리할 목적으로 개발되었고(조 등,1997), ISO의 OSI 관리는 OSI 자원을 관리할 목적으로(Tang과 Scoggins,1994), 또한 IETF의 SNMP는 TCP/IP등의 인터넷을 관리할 목적으로 개발되었다(Stallings,1993).

통신망을 구성하는 다양한 망 요소들을 관리하기 위하여 사용되는 표준 망관리 프로토콜로는 인터넷에서 널리 사용되고 있는 SNMP(Simple Network Management Protocol)와 OSI(Open System Interconnection) 프로토콜을 사용하는 통신망을 관리하기 위한 CMIP(Common Management Information Protocol)이 있다(ISO/IEC 9596,1990),(RFC 1157,1990).

한편, 통신망의 거대화와 네트웍기술의 발전으로 인하여 대형 서버중심의 클라이언트/서버 컴퓨팅 환경에서 벗어나 개방형 통신망에 접속된 다양한 플랫폼을 이용한 분산 컴퓨팅 환경으로 바뀌어가고 있다. 분산 시스템 분야에서는 최근 소프트웨어와 하드웨어의 이질성을 해결하기 위하여 미들웨어에 대한 연구가 OMG(Object Management Group)를 중심으로 진행되고 있으며, 객체 지향 기술을 바탕으로 분산 객체 시스템의 표준으로 CORBA(Common Object Request Broker Architecture)를 제안하고 있다(Object management Group, December 1997).

대규모 분산 컴퓨팅 환경에서의 효율적인 망관리를 위해 OMG/CORBA, ISO/ODP에서 사용하고 있는 분산 객체 기반 구조가 채택될 것으로 예상됨에 따라, NMF(Network Management Forum)의 OMNIPoint에서는 기존의 ITU-T/TMN, ISO/OSI, OMG/CORBA, OSF/DME 등에서 제시된 다양한 관리 구조의 통합에 대한 연구를 진행하고 있고(NMF,1994), OMG에서도 CORBA를 이용한 망 관리 구조를 연구하고 있으며(OMG Telecom

Special Interest Group,January 1996), X/Open과 NMF의 JIDM(Joint Inter Domain Management) 태스크포스는 OMG/CORBA, ISO/OSI, IETF/SNMP간의 관리를 위해 정보 모델과 프로토콜의 변환에 대한 연구를 진행중이다(JIDM,January 1996).

CORBA는 어플리케이션들이 네트워크, 언어, 컴포넌트 경계, 운영체제간에 걸쳐 그들의 범위를 확장할 수 있게 하는 분산 객체 기반 구조를 제공한다. 자바는 모든 주요 운영체제 위에서 작업을 수행할 수 있는 이식 가능한 객체를 제공할 수 있다. 웹과 CORBA, 그리고 CORBA의 자바 매핑은 자바에서 제공하는 모빌코드 기능과 CORBA에서 제공하는 객체 컴포넌트 기술을 적용하여 대규모 정보 시스템을 웹 위에서 구축할 수 있게 해준다. 이러한 개발환경을 Object Web이라고 정의하고 있다(Orfali와 Harkey,1997). 또한, 웹 브라우저는 거의 모든 플랫폼 상에서 동작하며 일관된 사용자 인터페이스를 가진다. 따라서 망관리 시스템의 사용자 인터페이스로서 웹 브라우저를 사용한다면 다양한 장점을 얻을 수 있으며, 웹 브라우저에서 자바와 IIOP를 이용함으로써 운영체제에 독립적이며, 모빌 코드 속성을 제공함으로써 어플리케이션 개발에 많은 유연성을 제공해줄 뿐만 아니라 IIOP(Internet Inter-ORB Protocol)를 통하여 직접 CORBA 객체와 통신할 수 있다 (정 등,1998).

본 논문에서는 효율적인 망관리를 위해 분산 객체 기술인 CORBA를 기반으로 하여 Object Web을 이용한 CORBA 관리자와 CMIS-레벨 게이트웨이 모델을 적용한 CORBA/CMIS 게이트웨이 구조를 설계하였으며, 그 구조를 기반으로 실제 시스템을 구현하였다. CORBA/CMIS 게이트웨이는 기존의 대리자를 수정하지 않은 채 CORBA 대리자로 보여지도록 설계하였는데, 이는 TMN의 관리구조로 CORBA 기반의 구조가 사용될 때 게이트웨이를 이용하여 기존의 자원을 CORBA 대리자로 동작하도록 함으로써 현재 사용중인 자원 또한 CORBA를 이용하여 관리가 가능하다는 장점이 있다.

본 논문의 구성을 보면 2장에서는 망관리와 분산 객체 기술에 대해 간략히 기술하였다. 3장에서는 이러한 연구를 기반으로 하여 Object Web을 이용한 분산 시스템 관리에 대해 설계 및 구현하였으며, 구현된 시스템을 시험하였다. 마지막으로 4장에서 결론을 맺는다.

Ⅱ. 망관리 시스템과 분산객체기술

1. OSI 망관리 시스템

망관리의 목적은 통신망을 효율적으로 운용하고 망이 제대로 동작하고 있는가를 보장해 주는데 있다. 이를 위해서 망관리 시스템은 망을 구성하는 다양한 장비들로부터 망의 사용에 관한 정보를 수집하여 운용자에게 보고한다. 망을 관리할 때의 대상은 사설교환기(PBX), 컴퓨터, 모뎀, 동축케이블, 광섬유 등과 같은 물리적인 망의 자원뿐만 아니라 대역폭(bandwidth), 서비스 품질(Quality of Service), 통신 프로토콜, 응용 소프트웨어 등의 논리적인 자원도 포함한다(손,1996).

OSI 시스템 관리에서 기본적인 기능이 관리자와 대리자간의 정보를 교환하는 것이다. 이 시스템 관리를 목적으로 정보와 명령을 교환하기 위하여 이용하는 기능이 CMISE(Common Management Information Service Element)이다. CMISE는 관리응용과의 인터페이스 서비스를 담당하는 CMIS(Common Management Information Service)와 PDU 및 관련절차를 규정한 CMIP(Common Management Information Protocol)으로 이루어진다(Stallings,1993).

1) CMISE

CMIS서비스는 기본적으로 요청받은 서비스에 대하여 실패, 성공이나 요청을 받았다는 응답을 확인해주는 확인 서비스들과 응답을 하지 않는 비확인 서비스로 나눌수 있으며, 다음 두 개의 ISO 어플리케이션 프로토콜을 사용한다.

① ACSE(Association Control Service Element)

어플리케이션간의 어소시에이션을 설정하고 해제한다.

② ROSE(Remote Operations Service Element)

어플리케이션간의 상호작용을 요구하거나 응답한다.

2) CMIS

CMIS는 네트워크 관리 어플리케이션이 수행시킬 수 있는 오퍼레이션이다. 여기에는 시스템간 연결 및 해제를 위한 서비스인 A-Associate, A-Release, A-Abort가 있고 대리인에서 관리자에게 관리 객체에 발생한 특정 상황을 통지하는 M-EVENT-REPORT서비스가 있다. 그리고 실제로 관리객체들에 대한 정보를 주고받는 서비스인 관리운용서비스에는 여섯 가지 서비스가 있다. 이들 서비스는 다음 세 가지 범주로 나눌 수 있다.

① Association Service

통신을 하기 위해서 해당 시스템의 어플리케이션간의 연결 및 해제를 하기 위해서 필요하며 ACSE를 이용한다.

② Management-notification Service

관리객체들에 대한 특정사건을 보고하는데 이용하는 것으로 필요한 속성, 행동 및 작동절차 등에 대한 것은 각 관리객체들에 정의되어 있다. 이를 위한 서비스로 M-EVENT-REPORT가 있다.

③ Management-operation Service

시스템 관리를 위해서 정보를 주고 받는데 이용되는 M-GET, M-SET, M-CREATE, M-DELETE, M-ACTION, M-CANCEL-GET과 같은 여섯 가지 서비스가 있으며 관리객체에 대한 사항은 관리객체들에 정의된 것에 의존한다.

M-EVENT-REPORT를 포함한 일곱 가지 서비스 중에서 M-GET, M-CREATE, M-DELETE, M-CANCEL-GET은 반드시 응답을 받아야하는 확인 서비스이고 나머지는 서비스를 요청할 때에 확인/비확인을 위한 파라미터를 주어야 한다. M-CANCEL-GET은 이미 요청된 M-GET을 중지시킬 때 사용되는 서비스로써 이를 사 용하는 이유는 MIB이 수정 도중에 문제가 발생하면 MIB의 일관성을 보증하기가 어렵기 때문이다.

CMIS 서비스들과 관련된 각 서비스들의 파라미터들의 리스트를 다음의 Table 1 ~ Table 7에서 나타내었다(Stallings,1993).

Table 1. M-GET parameters

Parameter Name	Request/ Indication	Response /Confirm
Invoke identifier	M	M
Linked identifier	-	C
Base-object class	M	-
Base-object instance	M	-
Scope	U	-
Filter	U	-
Access control	U	-
Synchronization	U	-
Attribute-identifier list	U	-
Managed-object class	-	C
Managed-object instance	-	C
Current time	-	U
Attribute list	-	C
Errors	-	C

Table 2. M-SET parameters

Parameter Name	Request/ Indication	Response /Confirm
Invoke identifier	M	M
Linked identifier	-	C
Mode	M	-
Base-object class	M	-
Base-object instance	M	-
Scope	U	-
Filter	U	-
Access control	U	-
Synchronization	U	-
Modification list	M	-
Managed-object class	-	C
Managed-object instance	-	C
Current time	-	U
Attribute list	-	U
Errors	-	C

Table 3. M-ACTION parameters

Parameter Name	Request/ Indication	Response/ Confirm
Invoke identifier	M	M
Linked identifier	-	C
Mode	M	-
Base-object class	M	-
Base-object instance	M	-
Scope	U	-
Filter	U	-
Managed-object class	-	C
Managed-object instance	-	C
Access control	U	-
Synchronization	U	-
Action type	M	C(=)
Action information	U	-
Current time	-	U
Action reply	-	C
Errors	-	C

Table 4. M-DELETE parameters

Parameter Name	Request/ Indication	Response/ Confirm
Invoke identifier	M	M(=)
Linked identifier	-	C
Base-object class	M	-
Base-object instance	M	-
Scope	U	-
Filter	U	-
Access control	U	-
Synchronization	U	-
Managed-object class	-	C
Managed-object instance	-	C
Current time	-	U
Errors	-	C

Table 5. M-CREATE parameters

Parameter Name	Request/ Indication	Response/ Confirm
Invoke identifier	M	M(=)
Managed-object class	U	C
Managed-object instance	U	C
Superior-object instance	U	-
Access control	U	-
Reference-object instance	U	-
Attribute list	U	C
Current time	-	U
Errors	-	C

Table 6. M-EVENT-REPORT parameters

Parameter Name	Request/ Indication	Response/ Confirm
Invoke identifier	M	M
Mode	M	-
Managed-object class	M	-
Managed-object instance	M	-
Event type	M	-
Event time	-	C
Event information	-	C
Current time	-	U
Event reply	-	U
Errors	-	C

Table 7. M-CANCEL-GET parameters

Parameter Name	Request/ Indication	Response/ Confirm
Invoke identifier	M	M
Get invoke identifier	M	-
Errors	-	C

M : Mandatory

(=) : The value of the parameter is equal to the parameter in the column to the left

U : The use of the parameter is a service user option

- : not present

C : conditional

3) CMIP

CMIP은 관리정보를 전송하는 절차 즉, CMISE사이에 CMIS서비스를 완성시키기 위해서 교환하는 CMIP PDU를 만들고 전송하는 것에 대해 정의해 놓은 것이다. 양단의 CMISE 사용자들이 정보교환을 위해서 시스템을 연결하는 데 ACSE를 이용하며, 이때는 CMIP이 이용되지 않는다. 관리 서비스를 위해서 CMISE는 PDUs를 교환하기 위해 CMIP를 채용한다. 그리고 CMIP는 CMIP PDU전송을 위해서 ROSE를 이용한다.

4) 관리자와 대리자

Fig. 1은 OSI 시스템 관리에서의 관리자와 대리자사이의 상호관계를 보여주고 있다. 관리자와 대리자간에는 CMIP 프로토콜을 사용하여 망관리 정보를 주고받는다. 관리객체는 망에서 감시의 대상이 되는 자원들을 객체로 모델링한 것인데, 관리객체의 속성이 자원의 상태를 반영하고 있고 특정한 사건이 있을 경우 사건에 대한 통지를 대리자에게 보낸다. 대리자는 관리자로부터의 요청에 대하여 관리객체의 값을 읽어 반환하고, 관리객체에서 발생한 사건을 수신하여 관리자로 전달해 준다. 관리자를 사용하여 운용자는 망을 운용하고, 관리객체에서 발생한 사건을 수신하면 이를 운용자에게 알려 적절한 조치를 취할 수 있도록 한다.

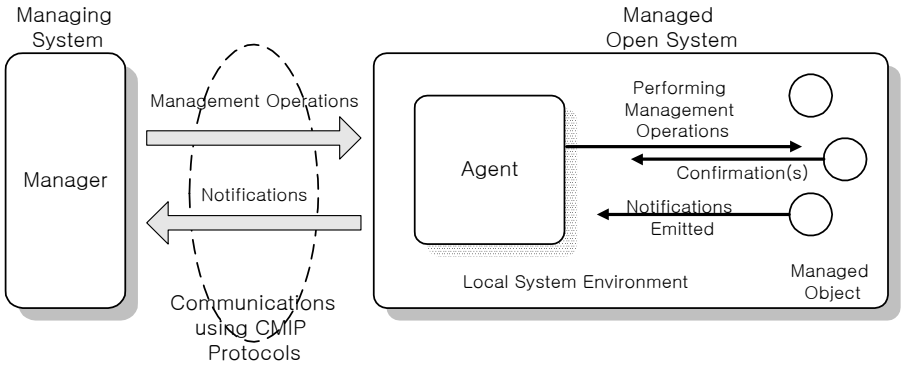

Fig. 1 Manager and agent for OSI system management

통신망을 구성하는 자원이 하나의 벤더에 의해 모두 생산된다면, 또 공급자가 각 자원에 대해 단일한 망관리 방법을 제공하여 준다면 도메인간의 망관리는 필요 없을 것이다. 그러나 다양한 제공자가 다양한 방법으로 망관리 방법을 제공해 주기 때문에 이러한 자원을 통합하여 관리하기 위해서는 도메인간의 망관리가 필수적이다. 따라서, 관리된다는 측면에서의 망자원으로 추상화시킬 수 있다면 복잡한 망을 보다 쉽게 관리할 수 있는 방법을 제공해 줄 수 있다.

2. 분산객체기술과 망관리

분산객체는 네트웍을 통해서 자유롭게 접근할 수 있는 소프트웨어 객체를 말한다. 이것은 다른 객체와 마찬가지로 데이터와 그 데이터를 조작할 수 있는 메소드들로 이루어져 있으며, 이러한 객체들은 프로퍼티와 메소드들을 사용해서 네트워크 상에서 사용 가능하다. 이러한 객체들이 다양한 플랫폼과 프로그래밍 언어를 복합적으로 사용할 수 있도록 해주는데 이것을 분산객체라고 한다. 이러한 특성은 두 가지 큰 장점이 있다. 우선 플랫

폼에 대해 독립성을 가진다는 것이다. 플랫폼 독립성은 다양한 서버들로 이루어진 네트웍 환경에서 많은 이점을 가지고 있으며, 또한 프로그램 언어에 독립적인 면은 기존 코드를 재 사용할 수 있고 개발자들에게 기술 습득을 용이하게 하며, 시스템을 개발할 때에 데이터가 반드시 한 서버에 존재해야 하는 부담감을 없애준다.

1) CORBA

1989년 4월, 현재 존재하는 객체 기술을 바탕으로 응용 프로그램들을 결합하기 위한 객체지향 표준을 제정하기 위해 OMG라는 비영리 단체가 탄생하였다. 이 단체에는 마이크로소프트를 포함하여 600개의 컴퓨터 관련 단체들이 참가하여 객체지향 기술을 기반으로 이종의 분산된 환경 하에서 응용 프로그램들이 서로 통합할 수 있는 표준기술을 탄생시켰다. 바로 이 표준이 OMA(Object Management Architecture)이다. OMA는 응용 프로그램간의 결합뿐만 아니라 객체의 생성,소멸에서부터 저장, 트랜젝션 기능에 이르기까지 분산 객체 환경에서 필요한 모든 서비스를 총칭하는 것이다. 이들 기능 중 CORBA는 컴퓨터 내부의 버스처럼 서로 다른 프로그램들 사이의 버스 역할을 하는 모듈로서 OMA 구조에서 가장 중요한 요소가 된다. 또한 CORBA에서 가장 중요한 요소는 ORB이다.

CORBA는 OMA의 한 부분이고 ORB는 CORBA의 핵심기술을 말하는 것이다. 현재 OMG는 1990년 OMA를 발표한 이래 지금까지 CORBA2.0 스펙을 발표하였다. OMA는 이종의 분산환경에서 필요한 모든 기능을 정의하기 때문에 상당히 광범위하다. 먼저 OMA를 이해하는 데 있어 가장 기초사항이 객체지향 개념의 이해이다. 왜냐하면 OMA가 제공하는 모든 서비스는 객체개념을 기반으로 하기 때문이다. OMA는 크게 이종의 분산환경에서 통신을 담당하는 CORBA와 객체를 조작하는 데 필요한 각종 기본 기능들을 정의하고 있는 COSS (Common Object Service Specification), 그리고 추가적으로 제공되는 기능 서비스가 존재하고 있다. 그리고 마지

막으로 이들 기능들을 이용하여 사용자들이 작성한 각종 응용 객체들이 존재한다. ORB의 기능을 요약하면, 클라이언트가 발행하는 요구를 목적지의 객체로 보내 그 응답을 반환하는 것이다. 이것을 어플리케이션으로부터 이용가능 하게 하기 위해서 CORBA에서는 다음의 사양을 설정하고 있다.

① 객체인 인터페이스를 공통으로 기술하기 위한 인터페이스 정의 언어
② 프로그램 언어로 어플리케이션을 기술하기 위한 언어 매핑
③ 객체의 생성, 요구의 발송 등 각종 서비스를 제공하는 컴포넌트 기능과 인터페이스
④ 객체간의 통신을 위한 프로토콜

Fig. 2는 ORB를 통한 CORBA 클라이언트와 구현객체의 구조를 보여주고 있다. CORBA객체는 CORBA IDL로 기술된 인터페이스를 통해서 상호작용 한다. CORBA IDL로 기술된 객체의 인터페이스는 IDL 컴파일러를 통해서 객체 통신을 위한 클라이언트 스터브(stub) 코드와 구현 스켈러톤(skeleton) 코드가 생성된다. 이 코드들은 통신을 위해 필요한 요소들을 포함하며, 사용자는 생성된 코드들을 이용해서 응용을 구현할 수 있다. CORBA의 클라이언트는 IDL 스터브를 이용한 정적 호출 기능과 프로그램 수행시 원하는 메소드를 호출하는 동적 호출 기능을 통해 구현 객체를 호출할 수 있다. 이와같이, 설계와 구현을 분리함으로써 사용자는 네트워크와 구현 언어에 대해서 고려하지 않아도 된다.

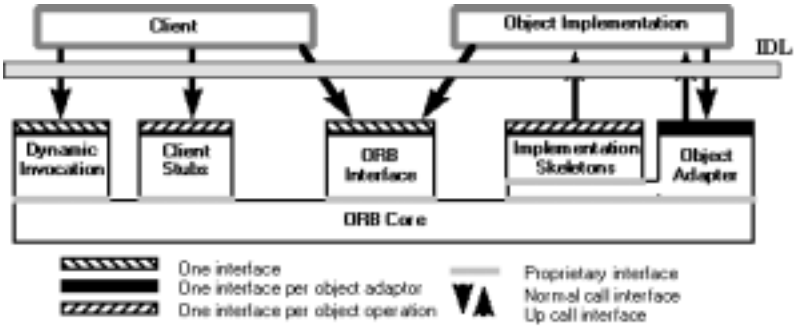

Fig. 2 Architecture of CORBA client and implementation object on the ORB

2) TMN과 CORBA의 통합방법

OSI 시스템 관리는 TMN 같은 환경에서 망이나 망 요소를 관리하도록 채용되었지만, 그 자체로는 분산되고 이질적인 환경에서 서비스관리를 하는데는 문제가 있다. 반면, CORBA는 분산 성질, 객체 지향, 그리고, 복잡한 객체들을 제어할 수 있는 능력 때문에 전기통신망 서비스 관리에 적합한 차세대 기술로 여겨진다.

현재까지는 TMN의 관리구조를 OSI 관리모델을 기반으로 모델링 하였으나 통신망이 커질수록 분산 객체 기반 구조를 적용하도록 하는 것이 필요하다(Genilloud,1996). CORBA는 현재 거의 업계표준으로 인정되는 추세이며, NMF와 X/Open은 공동으로 JIDM 태스크포스를 조직하여 CMIP/SNMP/CORBA 도메인간의 관리에 대한 연구를 진행중이다.

CORBA와 TMN의 통합을 위한 방법 중 대표적인 것으로 GDMO-to-IDL, Value-Added, Abstract Object mapping 등이 있다(Kong과 Chen,1996),(Jong-Tae Park,1998).

3) Object Web

인터넷을 대중화하는데 있어 중심적인 역할을 한 것이 웹이며, 이는 사용자에게 매우 쉽고, 일관된 인터페이스를 제공한다. 정보의 분산 여부에 무관하게 웹은 모든 종류의 정보의 다목적 저장소의 역할을 한다. 또한 웹 브라우저는 이미 거의 모든 플랫폼에서 작동한다. 이는 거의 모든 플랫폼에서 망관리 시스템에 접근하는 것이 가능하다는 것을 의미하며, 망자원을 공통적이며 일관된 방법으로 관리하는 것을 가능하게 한다.

최근 들어 인터넷 기술분야에서 급속도로 확산되고 있는 분야 중 하나가 인트라넷이란 분야이다. 이 작은 정보 고속도로는 외부와의 연결에도 가장 안전한 연결방법을 제공하며 기존의 인터넷에서 개발된 정보 시스템을 그대로 이용할 수 있게 해준다. 예를 들어, 특정 기관에서 TCP/IP를 응용한 인트라넷을 구성한다면 기존에 인터넷에서 사용했던 전자우편이나 유즈넷 뉴스, 웹 기술 등을 재사용 할 수 있다. 이를 통해 가장 저렴하고 효율적인 정보 네트워크를 구성할 수 있게 된다. 이 때 해당 인트라넷은 기존의 조직이나 기관에서 사용했던 응용 프로그램이나 시스템을 새로 구축하는 시스템과 통합해야만 한다. 이 문제를 해결하기 위한 방법으로 가장 적합한 것이 바로 CORBA를 이용하는 것이다.

Object Web이란 웹에서 CORBA와 자바에서 제공하는 다양한 분산 서비스들과 객체지향 패러다임을 이용하기 위해 시스템 변형 없이, 웹에 CORBA를 통합한 환경이다. Object Web은 하나의 독자적인 시스템이 아니라 현재 분산 개발 환경을 선도하고 있는 기술들을 통합하여 상승효과를 얻는 방법이다(왕과 이,1998). Object Web은 현재 인터넷을 중심으로 한 기술 중 가장 중요한 핵심 기술인 웹과 CORBA, 자바의 장점들을 연동 하여 보다 유연한 시스템을 개발하게 해준다. 따라서 Object Web 사용자는 애플릿을 통해 여러 호스트상의 객체들과 서비스를 주고받을 수 있으며, 엔터프라이즈 규모의 시스템을 디자인하고 개발하게 해준다. 또한 Object Web은 웹 기반의 시스템을 구축하는 데 있어 기존의 CGI에서 발생한 문

제를 해결할 뿐만 아니라 CORBA에서 제공하는 분산 객체 기술을 통해 인터넷 어느 곳에서든지 수행 가능한 시스템을 구축하고 이를 규모 있게 조정할 수 있다(Orfali와 Harkey,1997). 또한 기존의 자바 애플릿을 통해 CORBA로 작성된 다른 언어의 객체들을 이용할 수 있게 때문에 컴포넌트 소프트웨어 구축과 재사용이 가능하다. 이러한 이유들로 인해 Object Web 기술은 웹을 기반으로 한 표준 시스템 구축 방법으로 확산될 것이며 기본의 CGI나 기타 연동 기술은 Object Web 기술로 대체될 것이다. 특히, 웹을 기반으로 한 정보시스템중 규모와 종류가 다양한 자료와 환경 위에 기존의 시스템을 바탕으로 새로운 표준 시스템을 구축하려 할 때 Object Web은 그 진가를 발휘할 것이다.

3. Object Web을 이용한 OSI 망관리 구조

망의 관리자가 망의 세세한 부분에 신경을 쓰지 않으면서도 망 전체를 관리하기 위해서는 관리자에게 망자원이 투명하게 보여질 필요가 있다. 즉 망관리 시스템은 망자원의 종류나 관리 방법 등에 무관하게 망자원을 관리자에게 “관리 가능하도록” 보여주어야 한다는 것이다. 따라서 어떤 자원이 어떤 프로토콜을 이용하여 관리되는지에 무관하게 어떤 “자원”으로 볼 수 있어야 하며 그 수단이 되는 프로토콜은 관리자에게 숨겨져야 한다. 따라서, 망관리 시스템 또는 관리자가 망자원을 프로토콜과 무관하게 볼 수 있도록 즉, 관리된다는 측면에서의 망자원을 추상화시킬 수 있다면 복잡한 망을 보다 쉽게 관리할 수 있는 방법을 제공해 줄 수 있다(손,1996).

OSI 시스템 관리는 TMN과 같은 환경에서 망이나 망 요소를 관리하기에 적합하지만, 그 자체로는 분산되고 이질적인 환경에서 서비스관리를 하는데는 문제가 있다. 반면, CORBA는 분산 객체 환경을 제공해 주는 기능

때문에 전기통신망 서비스 관리에 적합한 차세대 기술로 여겨지고 있다(채 등,1998).

Fig. 3은 Object Web을 이용한 OSI 망관리 시스템의 구조를 보여주고 있다.

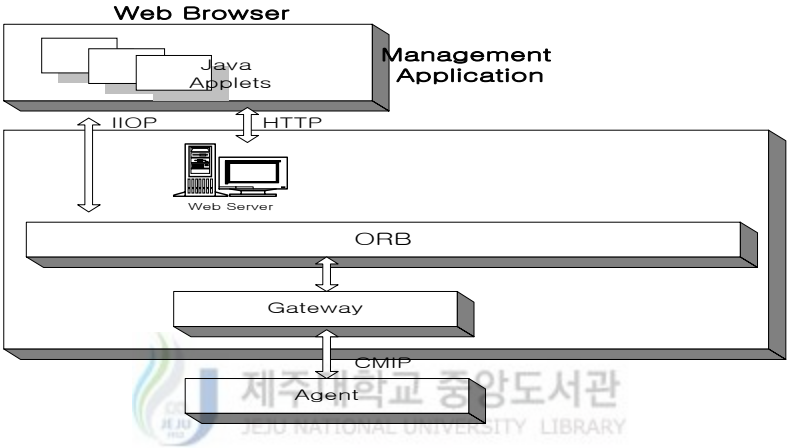

Fig. 3 Architecture of the OSI network management using object web

이 시스템은 관리응용과 웹 기반 관리 서버, 대리자로 구성되어 있다. 관리응용은 자바 인터프리터와 ORB를 탑재한 웹 브라우저에서 실행되어진다. 관리응용은 자바로 구현된 CORBA 클라이언트로, 웹 서버로부터 다운로드되어 실행되며 망관리를 위해 게이트웨이에 바인딩 되어진다. 웹 기반 관리 서버는 웹 서버, 게이트웨이로 구성되어진다. 웹 서버는 HTML 문서와 관리자 기능을 갖는 자바 애플릿 등을 웹 브라우저에게 제공한다. 게이트웨이는 SNMP나 CMIP과 같은 망관리 프로토콜과 CORBA 기술을 이용하는 관리자 사이의 상호운용에 사용되는 메소드를 제공한다. 또한 프로토콜에 의존적인 관리정보 구조와 상호작용을 위한 기능들을 CORBA 인터페이스로의 변환·역변환 역할도 수행하게 된다. 망관리 대리자는 MIB를

처리하여 망을 관리하며, CORBA와의 변환을 위해 게이트웨이가 필요하다.

Object Web을 이용하여 망관리 시스템을 구성할 경우, 다음과 같은 이점들이 있다.

- 사용자 인터페이스를 독립
- 플랫폼 독립성 제공
- 프로토콜 독립성 제공
- 자바를 이용한 이동 관리자와 이동 대리자 기능 제공

Ⅲ. 관리자와 게이트웨이의 설계 및 구현

이 장에서는 CORBA를 이용하여 망자원을 추상화시켜 관리할 수 있는 망관리 구조를 설계 및 구현하였다. 먼저 분산 시스템 관리를 충족하기 위해 설계된 구조 및 이의 특징을 기술한 후, 각 구성 요소간의 인터페이스에서 망관리 정보를 주고받기 위한 방법을 자세히 기술하였다.

1. 설계 구조

본 논문에서 설계한 망관리 구조는 Fig. 4와 같으며 크게 관리 시스템 도메인, 피 관리 시스템 도메인으로 구분된다. 사용자 인터페이스로는 웹 브라우저를 이용하며, 다수의 웹 브라우저를 통하여 하나의 피 관리 시스템에 접근하는 것이 가능하다. 이는 각 웹 브라우저에게 각각 다른 형태의 뷰(view)를 제공할 수 있다. 피 관리 시스템 도메인은 CORBA/CMIS 게이트웨이, 이벤트 처리기, 그리고 OSI 대리자와 실제 관리되는 물리적·논리적 자원들로 구성된다. CORBA 관리자인 관리 응용은 CORBA 클라이언트 역할을 수행한다. CORBA/CMIS 게이트웨이는 CORBA 서버의 역할을 수행하며, 대리자와는 CMIP 프로토콜을 사용하여 통신한다. 본 논문에서의 CORBA/CMIS 게이트웨이는 OSI CMIS 서비스들을 CORBA IDL 객체로 매핑하는 방식을 사용하였다(Jong-Tae Park,1998). CORBA 클라이언트는 웹 브라우저로 다운로드 되면 IIOP를 통하여 관리자 기능을 수행하게 된다. 이벤트 처리기는 OSI 대리자로부터 발생하는 이벤트를 감지하여 CORBA 클라이언트로 이벤트를 통보한다. OSI 대리자는 기존에 사용되고 있던 시스템 또는 기존의 OSI 관리자를 사용할 수도 있다.

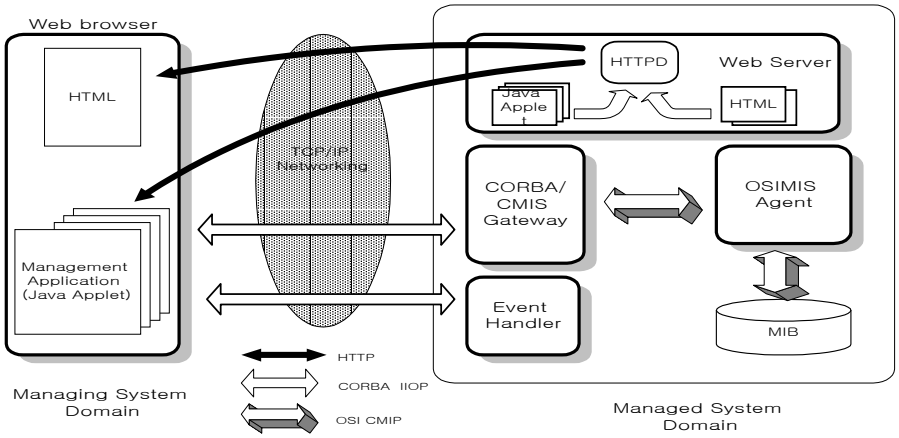

Fig. 4 Architecture of OSI network management using object web

본 논문에서 설계한 구조에 따른 관리 시스템과 피 관리 시스템간의 통신 절차는 Fig. 5와 같으며 다음과 같이 동작한다.

① 망 관리자는 웹 브라우저를 이용하여 웹 서버로 접속을 시도한다. 이때 웹 브라우저로부터의 메시지는 HTTP를 통하여 URL의 형태로 웹 서버로 전달된다.

② 웹 서버는 웹 브라우저로부터 수신한 URL을 분석하여 망 관리자가 요청한 HTML 문서를 웹 브라우저로 보낸다. 이 경우, HTML페이지는 삽입된 자바 애플릿에 대한 참조를 포함한다.

③ 웹 브라우저는 HTML 문서내의 애플릿 태그를 참조하여 웹 서버로부터 자바 애플릿을 검색한다. 웹 서버에 의해 검색된 애플릿은 바이트 코드 형태로 스터브와 함께 웹 브라우저로 다운로드 된다.

④ 웹 브라우저로 다운로드된 애플릿은 자바의 런타임 보안 점검 모듈을 통해 실행된 후 메모리로 로드되어 CORBA 서버 객체를 호출한다. 이때, 대리자와 관리자간의 변환을 담당하는 게이트웨이와 바인딩 한다.

⑤ 망 관리자는 바인딩된 게이트웨이에게 망관리 요청을 전달한다. 망관

리 요청을 받은 게이트웨이는 대리자를 호출하며, 대리자로부터 수신한 응답은 관리자로 반환한다.

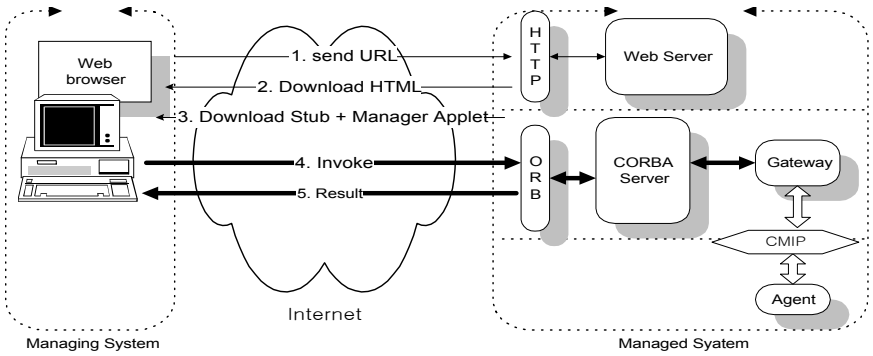

Fig. 5 Communication steps of OSI network management system using object web

2. 대리자와 게이트웨이 간의 인터페이스

본 절에서는 OSI의 망관리 요정 종류와 CORBA/CMIS 게이트웨이를 설계하는데 필요한 CMIS 서비스 파라미터와 그 구조에 따른 CORBA IDL 파라미터 구조 및 이들간의 매핑구조를 자세히 설명하였다.

1) OSI 망관리 요청의 종류

망관리 요청은 관리자가 대리자에게 요청하는 것과 대리자가 관리자에게 요청하는 것으로 크게 구분할 수 있다.

관리자가 대리자에게 요청하는 경우

① 대리자 내에 존재하는 관리객체의 값을 읽어올 때

② 대리자 내에 존재하는 관리객체의 값을 변경할 때

③ 대리자 내에 존재하는 관리객체에 특정한 행동을 취할 때

④ 대리자 내에 관리 객체를 새로 생성할 때

⑤ 대리자 내에 존재하는 관리객체를 삭제할 때 등이 있다.

대리자가 관리자에게 요청하는 경우

⑥ 대리자에게서 발생한 사건을 관리자에게 알리고자 할 때이다.

현재 망관리 프로토콜에서 표준으로 사용하고 있는 것은 CMIP과 SNMP가 있으며 Table 8에서 CMIP 프로토콜이 위의 여섯 가지 기능을 제공하기 위한 망관리 메시지를 비교하였다.

Table 8. The kind of CMIP messages

	CMIP
①	M-GET
②	M-SET
③	M-ACTION
④	M-CREATE
⑤	M-DELETE
⑥	M-EVENT-REPORT

2) CMIS 서비스 파라미터와 IDL 파라미터 구조

(1) CMIS 서비스 파라미터

Table 9의 CMIS 서비스 인수는 앞 절에서 기술한 OSI 망관리 요청 기능을 수행하는 CMIS 서비스들 중에서 Table 1의 CMIS GET 서비스 파라미터와 이 서비스 파라미터에 대응되는 CORBA IDL 타입을 기술하였다. Table 9의 첫 번째 부분은 서비스 파라미터이며 두 번째 부분은 서비스 요청 파라미터들의 조건과 이들 요청 파라미터들이 지시 프리미티브에 어떠한 조건으로 매칭 되는지를 나타낸다. 세 번째 부분은 요청 서비

스에 대한 응답 파라미터들의 종류와 조건들을 나타낸다. 본 논문에서는 이러한 서비스 파라미터들의 특성들을 기반으로 하여 서로 관련이 깊은 파라미터들을 모아서 하나의 구조체로 사용하였다. Table 9에서 보는바와 같이 CMIS_GETREQ라는 CORBA IDL 구조체 안에 Base-object class, Base-object instance, Scope, Filter, .. 등과 같이 대리자에게로 전달되어야 할 파라미터들을 각각을 구조체 필드로 정의하였다.

Table 9. CMIS GET service parameters and the corresponding IDL type name

Parameter Name	Request/ Indication	Response/ Confirm	IDL type name
Invoke identifier	M	M	RequestID
Linked identifier	-	C	CMIS_GET_LR_LIST
Base-object class	M	-	CMIS_GETREQ
Base-object instance	M	-	
Scope	U	-	
Filter	U	-	
Access control	U	-	
Synchronization	U	-	
Attribute-identifier list	U	-	
Managed-object class	-	C	CMIS_GETRES
Managed-object instance	-	C	
Current time	-	U	
Attribute list	-	C	
Errors	-	C	CMIS_ERROR
			CMIS_ERRORInfo

(2) CMIS 서비스 파라미터에 따른 게이트웨이의 IDL 파라미터 구조

본 논문에서 게이트웨이의 모든 망관리 서비스 메소드는 Table 1 ~ Table 7의 CMIS 서비스 종류와 파리미터에 따라 정의하였다. CORBA IDL로의 매핑은 Table 9에서 본 바와 같이 CMIS 서비스 파라미터 구조에 따라 적절한 CORBA IDL로 정의하였고, 정의된 CORBA IDL의 구조체 일부분을 아래에 기술하였다. (부록 A 참조)

```
  ..................
// GET Request Argument
struct    CMIS_GETREQ {
          CMIS_MID                    GET_req_class;
          CMIS_MN                     GET_req_instance;
          CMIS_EXTER                  GET_req_accessControl;
          CMIS_SYNC                   GET_req_sync;
          CMIS_SCOPE                  GET_req_scope;
          CMIS_FILTER                 GET_req_filter;
          ASN1_Integer                GET_req_nattrs;
          CMIS_MID_LIST               GET_req_attrs;
};
    ..................
```

위의 구조체는 GET 메소드에 사용되는 CORBA IDL 파라미터 구조의 일부분을 보여주고 있다.

3) 게이트웨이 인터페이스의 정의

게이트웨이 인터페이스는 대리자와 게이트웨이 간에 존재하는 인터페이스이며 앞 절에서 기술한 망 관리 요청을 수행하는 메소드와 관련 파라미터를 정의하여야 한다. 아래에 대리자와 게이트웨이간의 인터페이스 일부분을 기술하였다. (부록 B 참조)

```
interface CMISService {
    short CMISGet (in RequestID                GTrequestID,
                   in CMIS_GETREQ              GTgetReq,
                   out CMIS_GET_LR_LIST        GTlinkedList,
                   out CMIS_GETRES             GTgetRes,
                   out CMIS_ERROR              GTerrors,
                   out CMIS_ERRORInfo          GTerrorInfo);
```

.....

}

각 메소드의 매개변수는 서비스 메소드마다 파라미터들이 다르며, 입출력 방향은 관리자에서 게이트웨이로 파라미터를 보낼 때에는 in으로, 게이트웨이에서 관리자로 보내는 것은 out으로 되어있다.

4) 사건통지를 위한 인터페이스

(1) 일반적인 사건 통지

사건 통지를 위해 OMG의 공통 객체 서비스(Common Object Service)중의 하나인 사건서비스(Event Service)를 이용하였다(OMG TC Document 93.7.3,1993). 사건 서비스 규격에서는 일반적인 사건의 통신을 위해 푸쉬 모델(push model)과 풀 모델(pull model)의 두 가지를 이용하며 사건이 발생하는 쪽을 공급자(supplier), 사건을 수신하는 쪽을 소비자(consumer)라 한다. 푸쉬 모델은 공급자가 소비자에게 사건 데이터를 보내는 것이고, 풀 모델은 소비자가 공급자로부터 사건 데이터를 가져오는 것이다.

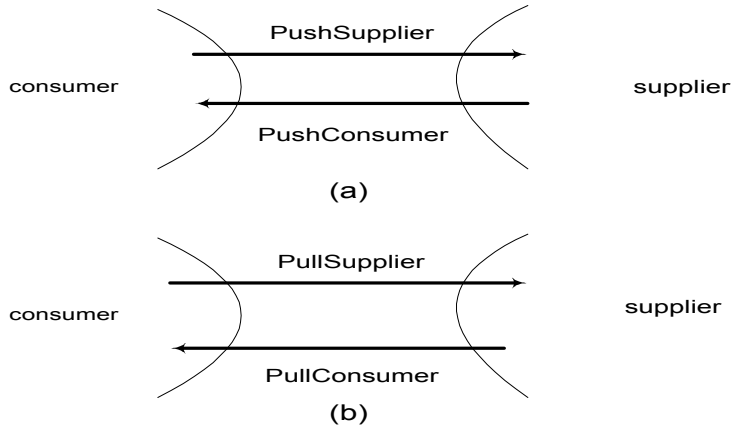

Fig. 6 (a) Push-style and (b) pull-style communications between the supplier and the consumer

Fig. 6에서 보여진 통신 형태는 다음의 네 가지 사건 서비스 인터페이스에 의해 지원된다.


```
module CosEventComm {
  exception Disconnected {};
  interface PushConsumer {
    void push(in any data) raises(Disconnected);
    void disconnect_push_consumer();
  };
  interface PushSupplier {
    void disconnect_push_supplier();
  };
  interface PullSupplier {
    any pull() raises(Disconnected);
    any try_pull(out boolean has_event) raises(Disconnected);
    void disconnect_pull_supplier();
```

```
    };
    interface PullConsumer {
      void disconnect_pull_consumer();
    };
};
```

(2) 게이트웨이에서 관리자로의 사건통지

게이트웨이는 대리자로부터 사건통지를 수신할 수 있다. 즉 대리자에서 M-EVENT-REPORT CMIS를 사용하여 사건을 통지한다는 것이다. CORBA와 전용의 망 관리 프로토콜과의 게이트웨이에서는 각 프로토콜에 알맞은 방법으로 사건통지 구조를 설계하여 구현할 수 있다.

게이트웨이에서 수신한 사건은 사건채널을 이용하여 관리자로 전달되며 이러한 구조에서는 게이트웨이가 사건 데이터의 공급자가 되고 관리자가 소비자가 된다.

3. 관리자와 게이트웨이의 구현

1) 구현 환경

본 논문은 SOLARIS 2.5.1 운영체제하에서 구현하였으며, OSI 대리자는 UCL(University College of London)에서 개발된 OSI 망관리 플랫폼인 OSIMIS(OSI Management Information Service)를 이용하여 구현하였으며 하부 통신은 ISODE(ISO Development Environment)를 이용하였다. CORBA 플랫폼으로는 Inprise사의 Visibroker C++ and Java 3.3을 이용하였으며 자바 애플릿을 구현하기 위해 JDK 1.1.6을 사용하였고 웹 서버는

Aparche 1.3을 사용하였다.

2) 관리자의 구현

본 논문에서 구현한 관리자는 웹 브라우저에서 실행이 되어져야하기 때문에 게이트웨이 인터페이스 정의를 Visibroker의 idl2java 컴파일러로 컴파일 하여 클라이언트 스터브 코드를 생성하였으며 이 소스코드를 이용하여 관리자 애플릿을 작성하였다. 관리자 애플릿은 자바로 작성되었으며 게이트웨이와의 통신을 위해 CORBA 2.0 스펙에 따라 관리코드를 작성하였다. 따라서 구현된 관리자 애플릿은 CORBA 스펙과 자바를 사용하여 웹 브라우저에서 실행되는 Object Web방식으로 구현함으로써 관리자에게 웹 브라우저 위에서 관리자 애플릿이 실행하여 관리정보를 주고받도록 하였다. 관리자 애플릿은 관리자에게 대리자를 관리할 수 있는 인터페이스를 제공하며, 구현된 게이트웨이에 의해 지원되는 인터페이스를 통하여 대리자로 관리정보를 보낼 수 있으며 또한 관리자로부터 자료를 입력받을 수도 있다. 이때 사용되는 메소드는 Get, Set, Action, Create, Delete 중에 하나이며, 웹 브라우저를 이용하여 OSI 객체 및 그 속성들에 대하여 관리할 수 있게 하였다.

아래에 스터브(stub)를 기초로 구현된 관리자 애플릿의 일부를 기술하였다.

```
.................
// Variable creation and instance
    int GTrequestID=0;
    CMIS_GETREQ GTgetReq = new CMIS_GETREQ();
.................
// Assignment to Variable
    GTrequestID = GTrequestID + 1;
```

```
// Request
   Getstatus = CMISGET(GTrequest, GTgetReq, GTlinkedList,
                                    GTgetres, Gterrors, GTerrorInfo);
.....................
```

3) 게이트웨이 구현

CORBA에서 CMIP/CMIS를 통하여 OSI시스템과 상호작용하기 위해서는 관련 객체 모델들 사이의 매핑과 도메인간에 프로토콜 변환을 처리하는 메커니즘이 제공되어야 한다. 이러한 상호작용변환을 담당하는 개체를 게이트웨이라고 하며, 사용되는 망관리 프로토콜에 따라 다수의 게이트웨이가 존재할 수 있다. 본 논문에서는 OSI 대리자와 관련 있는 게이트웨이를 CORBA/CMIS게이트웨이라고 정의하였고 CMIS-레벨 인터페이스를 기반으로 하는 게이트웨이 모델을 사용하여 OSI기반의 대리자와 CORBA기반의 관리자 사이의 변환을 담당하도록 구현하였으며, 망관리 프로토콜에 의존적인 대리자를 관리자 측면에서 CORBA 대리자로 볼 수 있도록 추상화하였다. 구현된 CORBA/CMIS게이트웨이는 게이트웨이 인터페이스를 정의한 IDL을 컴파일 했을 때 생성되는 스켈레톤(skeleton)을 기초로 구현되었으며 구현 결과 중의 일부를 아래에 기술하였다.

```
Gateway (const char *object_name = NULL):_sk_CMISService
(object_name)
{   MInitOption *miop[2]; /* Connect options */
    if (initialiseSyntaxes(NULLCP, attrSyntaxes) == NOTOK)
        error(NULLCP, "CMIPGateway: cannot initialise syntaxes");
    miop[0] = miop[1] = (MInitOption *) 0;
    if ((msd = m_initialise(service, host, miop)) < 0)
```

```
        error(NULLCP, "cannot establish association to %s at %s",
                service, host);          }

        ....................
 // Convert the string Argument to CMIS parameter
 // Object Class
     if(GTgetReq.GET_req_class.MID_mid_type = GlobalForm) (
       Obj_class.Mid_type = MID_GLOBAL;
       strObjClass =
           CORBA::String_dup(GTgetReq.GET_req_class.mid_mid_global);
       Obj_class.mid_global = name2oid(strObjClass);
     }
     else
     ....................
// Request Identification
  requestID = GTrequestID;
// Perform the CMIS M-GET request
    mi = &mis;
    status = M-GET(msd, requestID, &Obj_class, objInst, scope, filter,
                          NULLMACCESS, Sync, nattrs, mi);
       request_Invoke = mi->mi_invoke;
    ....................
```


CORBA/CMIS 게이트웨이에는 Get, Set, Action, Create, Delete 등 모든 CMIS 메소드가 구현되어져 있다. Get 메소드의 구현에서는 CMIS의 M-GET서비스를 호출하며 이러한 서비스를 호출할 때 그 매개변수가 되는 것들은 ISODE와 OSIMIS에서 제공되는 정보모델 변환 함수를 사용하여 CORBA IDL 타입을 적절한 정보모델로 변환·역변환 하였다.

본 논문에서 구현한 CORBA/CMIS 게이트웨이는 CORBA 관리자에 대한 관리 대리자로 동작한다. 즉, Object Web을 지원하는 관리 응용 애플릿

으로부터 관리 요청을 받아 적절한 CMIS API를 통하여 OSI 대리자에게 관리 요청을 전달한다. OSI 대리자는 CORBA/CMIS 게이트웨이를 통해 전달된 관리자의 관리 요청을 수행 한 다음, 다시 CMIS API를 통해 그 결과를 CORBA/CMIS 게이트웨이에게 반환한다. CORBA/CMIS 게이트웨이는 관리자에게 응답을 할 때는 클라이언트 역할을 하게되며, 관리자 상에서 적절한 동작을 호출하여 그 응답을 관리자에게 IIOP를 통해 전달한다.

4. 시험 및 검토

구현된 시스템에서 OSI 대리자는 피 관리 시스템에서 실행되며, 이들은 OSIMIS에서 제공해주는 대리자를 이용하였다. 이 대리자에 의해 유지되는 MIB은 Unix-MIB이며 시험의 편의를 위해 더 이상의 MIB확장은 하지 않았다.

구현된 시스템은 CORBA 기반 관리자 애플릿과 CORBA/CMIS 게이트웨이이며, 게이트웨이는 피 관리 시스템에서 실행된다. 망의 운용자는 어디에서든지 웹 브라우저를 이용하여 게이트웨이에 접근함으로써 대리자들의 정보를 읽어들이거나 변경할 수 있다.

1) 기능 시험

Fig. 7은 관리자가 PC 또는 터미널에서 웹 브라우저를 이용하여 웹 서버로부터 관리자 애플릿을 가져온 결과를 보여주고 있다. 다운로드된 관리자 애플릿은 웹 브라우저로부터 보안 검증을 받아 자바 인터프리터 또는 자바 가상 머신에 의해 실행이 되어진다. Fig. 7은 관리자 애플릿이 CORBA/CMIS 게이트웨이에 바인딩 한 후, CMISGet 메소드를 실행하여

게이트웨이에서 가져올 수 있는 정보를 IIOP를 통해 Object Web방식으로 가져온 결과를 보여 주고 있다. 왼쪽 창의 자료는 OSI 대리자에서 인스턴스화된 객체를 보여주고 있으며, uxObj1은 OSIMIS에서 제공하는 객체이다. 오른쪽 창은 왼쪽 창에서 선택된 객체의 속성을 보여주고 있으며 이들은 시스템 관리 서비스가 가져올 수 있는 정보를 IIOP를 통하여 Object Web방식의 관리자에게 전달하게된 것이다.

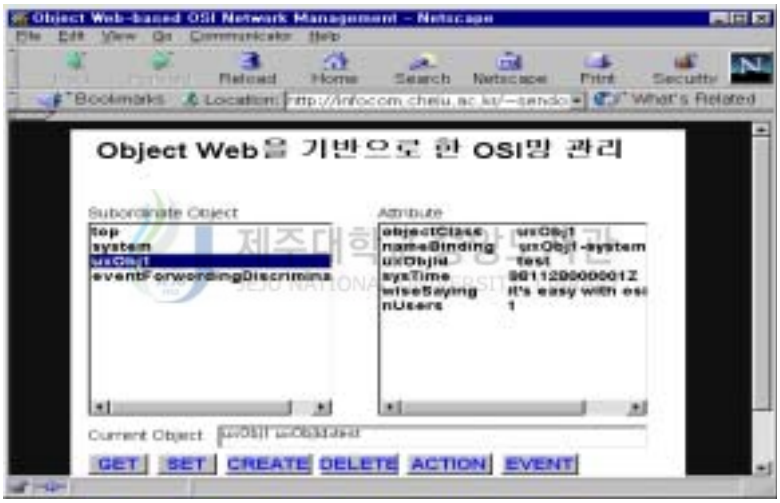

Fig. 7 Result of get requesting the value of managed object in OSI agent

Ⅳ. 결론

대규모 망관리에서는 이질적인 통신망 자원, 다양한 통신망 하부구조, 그리고 각 시스템의 언어에 관계없이 효율적으로 망을 운용·관리할 수 있는 기술이 필요하다.

Object Web을 기반으로 한 망 관리는 CORBA에서 제공되는 다양한 서비스 외에도 기존의 웹에서 제공하고 있는 서비스들을 함께 지원한다. 특히 자바는 운영체제에 독립적이며, 모빌코드 속성을 제공함으로써 어플리케이션 개발에 많은 유연성을 제공해 준다.

따라서 본 논문에서는 효율적인 망 관리를 위해 Object Web을 도입하여 웹 방식의 관리자를 구현하였다. 또한 CORBA를 이용하여 CMIP 프로토콜을 사용하는 대리자를 관리할 수 있는 CORBA/CMIS 게이트웨이를 구현하였고, 각각의 요소들이 정상적으로 동작하는지 확인하였다. 게이트웨이 구현에 CORBA를 사용하여 객체들 사이에 투명하게 서비스를 요청하고 응답할 수 있도록 하였다. 또한 OSI 대리자를 추상화하여 CORBA 대리자로 보여지도록 함으로써 현재 사용중인 자원 또한 CORBA를 이용하여 관리가 가능하도록 하였다.

본 논문에서는 OSI 관리 환경만을 고려하였지만 다양한 망관리 환경을 지원하기 위한 더 많은 고찰이 필요하다.

향후 연구과제로 망에 존재하는 여러 게이트웨이들 간의 통합 연동을 위한 CORBA 연합 트레이더 연구가 진행될 필요가 있다.

참고문헌

A. Tang, S. Scoggins, 1994, "OPEN NETWORKING WITH OSI", PTR Prentice Hall, 710 pp.

채석헌, 하은주, 정문상, 박종태, 1998, "TMN과 CORBA의 통합", 한국정보과학회지 Vol. 25. No. 1.

Guy H. Genilloud, 1996, "Towards Distributed Architecture for System Management", for of Doctor Sciences, Swiss Federal Institute of Technology in Lausanne.

ISO/IEC 9596, 1990, Common Management Information Protol(CMIP).

정재문, 고준수, 김승언, 임희경, 안선후, 1998, "ObjectWeb을 이용한 분산 어플리케이션 통합에 관한 연구", 한국정보과학회지 경남·제주지부 학술발표논문집 vol. 10. No. 1.

JIDM taskforce Draft Document, January 1996, "Inter-Working between OSI Management, SNMP and CORBA,".

조영현, 김영명, 석승학, 1997, "TMN을 향한 첫걸음", 하이테크정보, pp.11-28.

Jong-Tae Park, Jong-Wook Back, Seok-heon Chae, 1998, "Integration of CORBA Technology in TMN Framework", School of Electronic and Electrical Engineering Kyungpook National University.

NMF, August 1994, "OMNIPoint Integration Architectute - Delivering a Management System Framework to Enable Service Management Solutions", Technical report.

Object Management Group, December 1997, "The Common Object Request Broker: Architecture and Specification".

OMG TC Document 93.7.3, 1993 Revised July 2, "Joint Object Services Submission: Event Service Specification,".

OMG Telecom Special Interest Group, January 1996, "CORBA-Based Telecommunication Network Management Syatem, OMG White Paper Draft 2".

Qinzheng Kong, Graham Chen, 1996, "Integrating CORBA and TMN Environments", IEEE, Citr Pty Ltd.

RFC 1157, May 1990, "A Simple Network Management Protocol(SNMP)".

Robert Orfali, Dan Harkey, 1997, "Client/Service Programming with Java And CORBA", John Wiley & Sons, inc., Canada.

손경찬, 1996, "CORBA와 WWW을 이용한 통합 망 관리 시스템의 설계 및 구현", 경북대학교 대학원 전자공학과 석사학위논문.

왕창종, 이세훈, 1998, "CORBA 프로그래밍", 도서출판 대림, pp.121-142.

W. Stallings, 1993, "SNMP, SNMPv2, and CMIP: The Practical Guide to Network-Management Standards", ADDISON-WESLEY, 625 pp.

부록 A. CORBA/CMIS 게이트웨이에서의 파라미터 구조

일부 생략

```
// ---------------------------------------------
// GET Request Argument
struct      CMIS_GETREQ      {
            CMIS_MID                        GET_req_class;
            CMIS_MN                 GET_req_instance;
            CMIS_EXTER                      GET_req_accessControl;
            CMIS_SYNC                       GET_req_sync;
            CMIS_SCOPE                      GET_req_scope;
            CMIS_FILTER                     GET_req_filter;
            ASN1_Integer                    GET_req_nattrs;
            CMIS_MID_LIST           GET_req_attrs;
};


// GET Response  Argument
struct      CMIS_GETRES {
            CMIS_MID                        GET_res_class;
            CMIS_MN                 GET_res_instance;
            ASN1_GeneralizedTime            GET_res_time;
            ASN1_Integer                    GET_res_nattrs;
            CMIS_GETATTR            GET_res_attrs;
};

// GET Linked Reply Response
struct      CMIS_GETLINKEDREPLY {
            ASN1_Integer                    GET_LR_linkedid;
            ASN1_Short                      GET_LR_type;
            CMIS_GETRES                     GET_LR_res;
            CMIS_ProcessingFail     GET_LR_procfail;
};
typedef     sequence <CMIS_GETLINKEDREPLY>          CMIS_GET_LR_LIST;


// SET Request Argument
struct      CMIS_SETREQ      {
            CMIS_MID                        SET_req_class;
            CMIS_MN                 SET_req_instance;
```

```
        CMIS_EXTER                      SET_req_accessControl;
        CMIS_SYNC                       SET_req_sync;
        CMIS_SCOPE                      SET_req_scope;
        CMIS_FILTER                     SET_req_filter;
        ASN1_Integer                    SET_req_nattrs;
        CMIS_SETATTR_LIST               SET_req_attrs;
};

// SET Response Argument
struct  CMIS_SETRES     {
        CMIS_MID                        SET_res_class;
        CMIS_MN                 SET_res_instance;
        ASN1_GeneralizedTime            SET_res_time;
        ASN1_Integer                    SET_res_nattrs;
        CMIS_SETATTR            SET_res_attrs;
};

// SET Linked Reply
struct  CMIS_SETLINKEDREPLY {
        ASN1_Integer                    SET_LR_linkedid;
        ASN1_Short                      SET_LR_type;
        CMIS_SETRES                     SET_LR_setres;
        CMIS_ProcessingFail     SET_LR_procfail;
};
typedef sequence <CMIS_SETLINKEDREPLY>
                                CMIS_SET_LR_LIST;

// CREATE Request Argument
struct  CMIS_CREATEREQ {
        CMIS_MID                        CREATE_req_class;
        CMIS_MN                 CREATE_req_instance;
        ASN1_Short                      CREATE_req_inst_type;
        CMIS_MN                 CREATE_req_reference_inst;
        CMIS_EXTER                      CREATE_req_accessControl;
        ASN1_Integer                    CREATE_req_nattrs;
        CMIS_PARAM                      CREATE_req_attrs;
};
```

```
// CREATE Response Argument
struct      CMIS_CREATERES {
            CMIS_MID                        CREATE_res_class;
            CMIS_MN                   CREATE_res_instance;
            ASN1_GeneralizedTime            CREATE_res_time;
            ASN1_Integer                    CREATE_res_nattrs;
            CMIS_PARAM                      CREATE_res_attrs;
};

// CREATE Linked Reply Response
struct      CMIS_CREATELINKEDREPLY {
            ASN1_Integer                    CREATE_LR_linkedid;
            ASN1_Short                      CREATE_LR_type;
            CMIS_CREATERES            CREATE_LR_res;
            CMIS_ProcessingFail       CREATE_LR_procfail;
};
typedef     sequence <CMIS_CREATELINKEDREPLY>
                                      CMIS_CREATE_LR_LIST;

// DELETE Request Argument
struct      CMIS_DELETEREQ {
            CMIS_MID                        DELETE_req_class;
            CMIS_MN                   DELETE_req_instance;
            CMIS_EXTER                      DELETE_req_accessControl;
            CMIS_SYNC                       DELETE_req_sync;
            CMIS_SCOPE                      DELETE_req_scope;
            CMIS_FILTER                     DELETE_req_filter;
};

// DELETE Response Argument
struct      CMIS_DELETERES {
            CMIS_MID                        DELETE_res_class;
            CMIS_MN                   DELETE_res_instance;
            ASN1_GeneralizedTime            DELETE_res_time;
};

// DELETE Error Response Argument
struct      CMIS_DELETEERROR_TYPE  {
            CMIS_MID                        DELETE_err_class;
            CMIS_MN                   DELETE_err_instance;
            ASN1_GeneralizedTime            DELETE_err_time;
            CMIS_ERROR                      DELETE_err_error;
```

```
};
typedef     CMIS_DELETEERROR_TYPE
                                        CMIS_DELETEERROR;

// DELETE LinkedReply Argument
struct      CMIS_DELETELINKEDREPLY {
            ASN1_Integer                    DELETE_LR_linkedid;
            ASN1_Short                      DELETE_LR_type;
            CMIS_DELETERES          DELETE_LR_res;
            CMIS_DELETEERROR                DELETE_LR_error;
            CMIS_ProcessingFail     DELETE_LR_procFail;
};
typedef     sequence<CMIS_DELETELINKEDREPLY>
                                        CMIS_DELETE_LR_LIST;

// CANCELGET Request Argument
struct      CMIS_CANCELGETREQ_TYPE          {
            ASN1_Integer                    CANCELGET_req_invokeid;
};
typedef     CMIS_CANCELGETREQ_TYPE
CMIS_CANCELGETREQ;

// ACTION Request Argument
struct      CMIS_ACTIONREQ {
            CMIS_MID                        ACTION_req_class;
            CMIS_MN                 ACTION_req_instance;
            CMIS_EXTER                      ACTION_req_ActionControl;
            CMIS_SYNC                       ACTION_req_sync;
            CMIS_SCOPE                      ACTION_req_scope;
            CMIS_FILTER                     ACTION_req_filter;
            CMIS_PARAM                      ACTION_req_info;
};

// ACTION Response Argument
struct      CMIS_ACTIONRES {
            CMIS_MID                        ACTION_res_class;
            CMIS_MN                 ACTION_res_instance;
            ASN1_GeneralizedTime            ACTION_res_time;
            CMIS_PARAM                      ACTION_res_reply;
};
```

```
// ACTION LinkedReply Error Argument
struct      CMIS_ACTIONERROR_TYPE  {
            CMIS_MID                        ACTION_err_class;
            CMIS_MN                   ACTION_err_instance;
            ASN1_GeneralizedTime            ACTION_err_time;
            CMIS_ERROR                      ACTION_err_error;
            CMIS_MID                        ACTION_err_accessDenied;
            CMIS_MID                        ACTION_err_noSuchAction;
            CMIS_MID                  ACTION_err_noSuchArgument;
            CMIS_PARAM                ACTION_err_invalidArgumentValue;
};
typedef     CMIS_ACTIONERROR_TYPE
                                  CMIS_ACTIONERROR;

// ACTION LinkedReply Argument
struct      CMIS_ACTIONLINKEDREPLY {
            ASN1_Integer                    ACTION_LR_linkedid;
            ASN1_Short                      ACTION_LR_type;
            CMIS_ACTIONRES            ACTION_LR_res;
            CMIS_ACTIONERROR                ACTION_LR_error;
            CMIS_ProcessingFail       ACTION_LR_procfail;
};
typedef     sequence<CMIS_ACTIONLINKEDREPLY>
                                  CMIS_ACTION_LR_LIST;

// EVENT REPORT Request Argument
struct      CMIS_EVENTREPORTREQ     {
            CMIS_MID                        EVENTREPORT_req_class;
            CMIS_MN                   EVENTREPORT_req_instance;
            ASN1_GeneralizedTime            EVENTREPORT_req_time;
            CMIS_MID                        EVENTREPORT_req_type;
            CMIS_PE                   EVENTREPORT_req_info;
};

// EVENT REPORT Response Argument
struct      CMIS_EVENTREPORTRES     {
            CMIS_MID                        EVENTREPORT_res_class;
            CMIS_MN                   EVENTREPORT_res_instance;
            ASN1_GeneralizedTime            EVENTREPORT_req_time;
            CMIS_PARAM                      EVENTREPORT_req_reply;
};
```

```
// EVENT REPORT LinkedReply Argument
struct      CMIS_EVENTREPORTLINKEDREPLY    {
            ASN1_Integer                   EVENTREPORT_LR_linkedid;
            ASN1_Short                     EVENTREPORT_LR_type;
            CMIS_EVENTREPORTRES    EVENTREPORT_LR_res;
            CMIS_ProcessingFail    EVENTREPORT_LR_procfail;
};
typedef     sequence<CMIS_EVENTREPORTLINKEDREPLY>
CMIS_ER_LR_LIST;
```

부록 B. CORBA/CMIS 게이트웨이에서의 서비스 인터페이스

```
// ------------------------------------------------------
interface CMISService {
    short   Get(in RequestID                    GTrequestID,
                in CMIS_GETREQ                  GTgetReq,
                out CMIS_GET_LR_LIST            GTlinkedList,
                out CMIS_GETRES                 GTgetRes,
                out CMIS_ERROR                  GTerrors,
                out CMIS_ERRORInfo              GTerrorInfo);

    short   Set(in RequestID                    STrequestID,
                in CONF_MODE                        STmode,
                in CMIS_SETREQ                  STsetReq,
                out CMIS_SET_LR_LIST            STlinkedList,
                out CMIS_SETRES                 STsetRes,
                out CMIS_ERROR                  STerrors,
                out CMIS_ERRORInfo              STerrorInfo);

    short   Create(in RequestID                     CRrequestID,
                   in CMIS_CREATEREQ            CRcreateReq,
                   out CMIS_CREATE_LR_LIST      CRlinkedList,
                   out CMIS_CREATERES           CRcreateRes,
                out CMIS_ERROR                  CRerrors,
                out CMIS_ERRORInfo              CRerrorInfo);


    short   Delete(in RequestID                     DLrequestID,
                   in CMIS_DELETEREQ            DLdeleteReq,
                   out CMIS_DELETE_LR_LIST      DLlinkedList,
                   out CMIS_DELETERES           DLdeleteRes,
                out CMIS_ERROR                  DLerrors,
                out CMIS_ERRORInfo              DLerrorInfo);

    short   CancelGet(in CMIS_CANCELGETREQ CGcancelGetReq,
                   out InvokeID                 CGgetInvokeID,
                   out CMIS_ERROR                CGerrors,
                   out CMIS_ERRORInfo            CGErrorInfo);


    short   Action(in RequestID                     ACrequestID,
                in CONF_MODE                    ACmode,
```

```
                    in CMIS_ACTIONREQ           ACactionReq,
                    out CMIS_ACTION_LR_LIST     AClinkedList,
                    out CMIS_ACTIONRES          ACactionRes,
                    out CMIS_ERROR              ACerrors,
                    out CMIS_ERRORInfo          ACerrorInfo);

    short       EventReport(in RequestID            ERrequestID,
                            in CONF_MODE            ERmode,
                            in      CMIS_EVENTREPORTREQ
                                        EReventReportReq,
                            out CMIS_ER_LR_LIST     ERlinkedList,
                            out     CMIS_EVENTREPORTRES
                                        EReventReportRes,
                            out CMIS_MID            EReventType,
                            out CMIS_ERROR          ERerrors,
                            out CMIS_ERRORInfo      ERerrorInfo);
};
```