울산대학교 공학연구논문집 제30권 2호 pp. 409~423, 울산대학교.
Journal of Engineering Research Vol. 30, No. 2, pp. 409~423, 1999
University of Ulsan.

CORBA 기반 망관리 시스템 통합에 관한 연구

윤채운 · 김명균 · 허정석
컴퓨터 · 정보통신공학부

<요 약>

통신망이 커지고 복잡해짐에 따라 통신망 관리는 더욱 중요해지고 있다. 이러한 통신망을 효율적으로 관리하기 위해 국제 표준화 그룹인 ITU와 ISO는 CMIS/CMIP 기반의 OSI 망관리 모델 [1]을 표준화하고 IAB에서는 SNMP [7] 기반 망관리 모델을 표준화하였다. 이들 망관리 모델은 통신망 요소들의 관리에는 효율적이지만 복잡한 망들의 관리나 서비스 관리에는 많은 문제들이 있다. 이러한 문제들을 해결하기 위해 본 논문에서는 분산 객체 지향 기술인 CORBA [3]를 이용하여 기존의 망관리 시스템들을 통합하기 위한 방법에 대해 기술한다. CORBA 기반 통합 망관리 구조를 제시하고 CORBA와 OSI 망관리 모델사이의 관리정보 및 관리기능 변환 기능을 위한 CORBA/CMIP 게이트웨이 구조를 제시한다. 또한 COBRA 관리자에서 OSI객체들을 접근하기 위한 기법과 OSI 객체에서 발생하는 이벤트를 CORBA 관리자에 전달하기 위한 기법들에 대해 기술한다.

A Study on the Integration of Network Management Systems Based on CORBA

Chae-Woon Yoon · Myung-Kyun Kim · Jeong-Seok Heo
School of Information and Communication Technology

<Abstract>

Network management systems become more important as the communications networks evolve larger and more complicated. For the efficient management of the communications networks, the international standard organizations, ITU and ISO, have

standardized OSI network management model [1] based on CMIS/CMIP, and IAB has proposed SNMP [7] as the Internet management model. The traditional management models are efficient in managing simple communications networks, but have many problems in managing complicated networks and services. To solve those problems, this paper proposes an integration method of network management systems based on CORBA [3] which is a distributed and object-oriented technology. We describe an integrated network management architecture based on CORBA/CMIP gateway. The CORBA/CMIP gateway converts the management informations and operations between the CORBA-based and CMIP-based management systems. We also describe how the CORBA managers access the OSI managed objects and how the events occurred in the OSI managed objects are delivered to the CORBA managers.

1 서 론

통신망이 조직의 정보 인트라 구축에 있어 필수 불가결한 요소가 되고 규모 또한 커지고 복잡해짐에 따라 통신망 관리는 더욱 중요해지고 있다. 이러한 통신망은 많은 제작자들로부터 제작된 서로 다른 장비들을 갖는 크고 복잡한 망으로 점차적으로 진화해 왔다. 서로 다른 망들을 효율적으로 관리하기 위해서는 각 통신망내의 관리 정보가 통합되고 서로 다른 통신망 요소들 사이에 수행되는 관리 기능과 그들 사이의 상호 운용 방식 등이 서로 일치해야 한다. 이에 따라 국제표준화 그룹인 ITU (International Telecommuincation Union)과 ISO (International Standards Organization)는 OSI 망관리 모델 [1]을 제정하고 IAB(Internet Architecture Board)에서는 Internet 관리를 위해 SNMP [7, 8, 9] 기반 망관리 모델을 제정하였다. 이 OSI 망관리 표준에 기초한 TMN (Telecommunications Management Network) [2]은 통신망을 관리하기 위한 독립적인 관리망으로, 통신망의 각 요소 (피관리 망요소)들은 TMN에서 정의된 인터페이스에 따라 TMN과 통신을 한다. TMN은 전기통신망 관리 기능을 여러 레벨로 나누어 정의하고 있고, OSI 망관리 프로토콜인 CMIS/CMIP (Common Management Information Service / Common Management Information Protocol)을 사용하고 관리 정보 모델은 GDMO/ASN.1 (Guidelines for the Definition of Managed Objects / Abstract Syntax Notation 1)을 사용한다. 그러나 OSI 망관리의 제한으로 인하여 TMN 기반 통신망 관리 시스템은 복잡한 통신망 관리 및 서비스 관리에 한계가 존재하고, 망관리 시스템 개발자들은 이를 극복하기 위한 새로운 기술을 찾기 위한 많은 노력을 해왔다. 개방형 분산 객체 기술인 OMG(Object Management Group)의 CORBA(Common Object Request Broker Architecture) [3]는 응용 서비스 레벨에서 통일된 API(Application programming Interface)를 제공하고, 일관성 있는 통신망 관리 방법을 제공함으로써 위와 같은 문제들을 해결할 수 있는 대안으로 여겨진다. 이러한 CORBA기반 통합 망관리 시스템은 기존의 망관리 시스템들을 효과적으로 수용할 수 있는 기법이 필요하다. 이를 위해 OMG Telecom Working 그룹[4]과 X/Open과 NMF(Network Management Forum)[5] 그리고 TINA-C[6] 등과 같은 표준화 그룹들과 많은 연구자들에

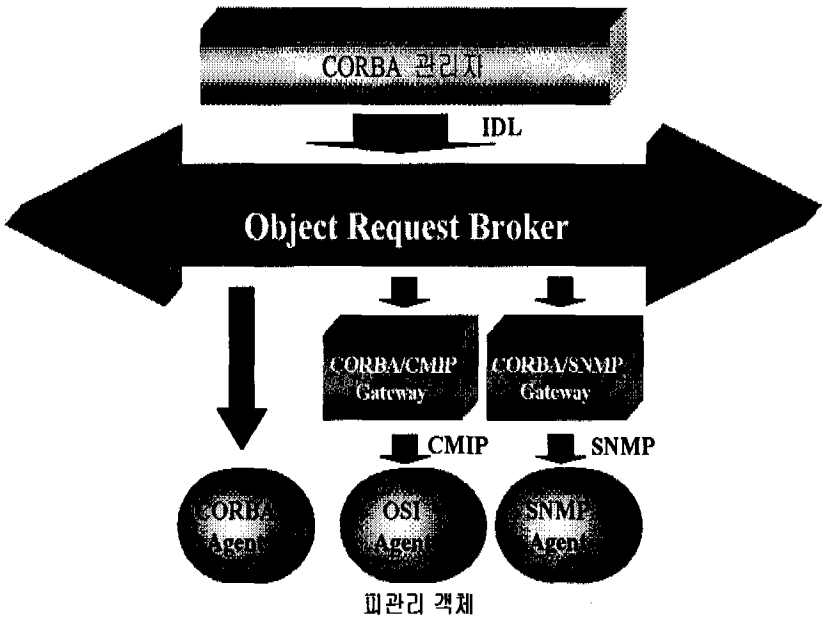
의해 연구가 진행되고 있다 [10, 11, 12].

본 논문에서는 기존의 망관리 객체들을 적은 비용으로 효과적으로 수용하며 통합하기 위해 CORBA/CMIP 게이트웨이를 이용한 통합 방법을 제시한다. CORBA/CMIP 게이트웨이는 CORBA 관리 모델과 OSI 관리 모델 사이의 서로 다른 관리 정보와 관리 연산 사이의 변환 기능을 수행한다. 변환 게이트웨이를 이용한 망관리 시스템 통합에 관한 연구는 현재 OMG의 Telecom task force [4]와 NMF의 JIDM (Joint Inter-Domain Management) 연구 [5]에서 표준화를 진행하고 있고, Kong 등[10]과 같은 연구자에 의한 연구들이 있었다. 그러나 이러한 연구들이 초보단계에 있어 각각 게이트웨이를 이용한 변환 시나리오를 제시하는 수준에 머물러 있다. 본 논문에서는 CORBA/CMIP 게이트웨이를 이용한 전체적인 통합 망관리 시스템 구조를 제시하고, CORBA 관리자에서 망관리 객체들을 접근하기 위한 기법과 망관리 객체들로부터 발생한 이벤트를 해당 관리자로 전달하기 위한 기법들에 대해 기술한다. 본 논문에서 제시한 CORBA 관리자가 OSI 객체를 접근하는 방법은 기존의 [4]와 [12]의 연구와 유사하지만, 이벤트 처리 방법은 다르다. 본 논문에서는 OSI 객체로부터 발생한 이벤트를 CORBA 관리자들에게 전달하기 위해 EventPort 객체를 이용한 방법과 ProxyEFD 객체를 이용한 방법을 제시하고 있는데, [4]와 [12]의 이벤트 처리 방법은 EventPort 객체를 이용한 방법과 유사하다. 본 논문에서는 구현이 용이하고 필터조건 처리 등에 있어 유연성이 좋은 ProxyEFD 객체를 이용한 방법을 제시한다.

다음절에서는 CORBA기반 통합 망관리 전체 구조에 대해 기술하고 3절에서는 CORBA 관리자로부터 OSI 객체를 접근하기 위한 기법에 대해 기술한다. 4절에서는 OSI 객체에서 발생하는 이벤트를 효과적으로 CORBA 관리자에 전달하기 위한 기법에 대해 기술하고 마지막으로 5절에서는 결론에 대해 기술한다.

2 CORBA 기반의 통합 망관리 구조

CORBA 기반 통합 망관리 시스템 구조는 [그림 1] 과 같다. CORBA 관리자 객체는 ORB(Object Request Broker)를 통해 피관리 객체들을 접근하는데, 피관리 객체들은 CORBA 객체이거나 OSI 객체 또는 SNMP 객체일 수 있다. CORBA 피관리 객체에 대한 접근은 ORB의 정적 또는 동적 호출 인터페이스를 이용하여 수행하고, CORBA 객체가 아닌 OSI 또는 SNMP 피관리자의 경우에는 CORBA 호출을 해당 피관리자가 속한 망관리 모델의 관리기능으로 변환이 필요하다.

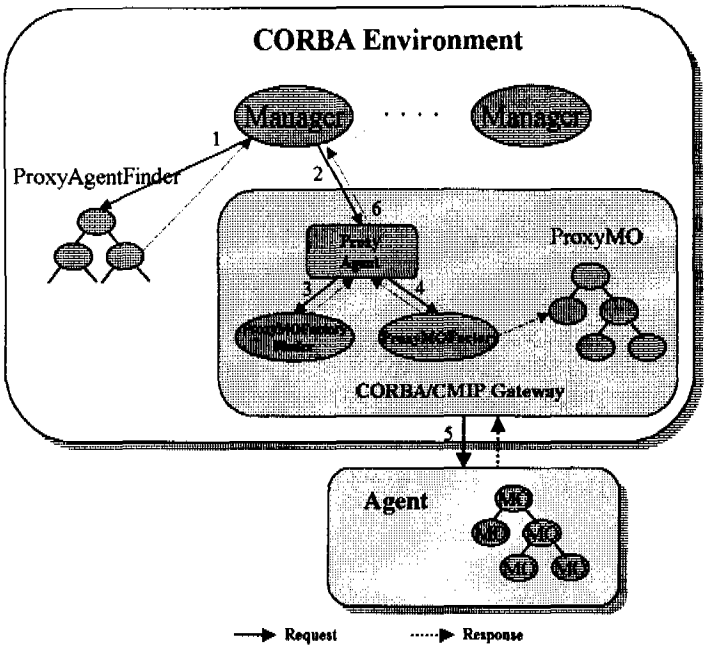


[그림 1] CORBA 기반 망관리 시스템 구조

[그림 1] 에서 CORBA/CMIP 게이트웨이는 CORBA와 CMIP 간의 관리기능 변환을 수행하며 CORBA/SNMP 게이트웨이는 CORBA와 SNMP간의 관리기능 변환을 수행하는데, 본 논문에서는 CORBA/CMIP 게이트웨이 기능에 대해서만 기술한다. CORBA 관리자는 피관리 객체가 CORBA 객체이거나 또는 OSI 객체이거나 SNMP 객체이거나에 상관없이 모든 객체를 CORBA 객체로 인식하고 피관리 객체의 IDL을 통해 접근한다. 따라서 CORBA/CMIP 게이트웨이에서는 CORBA와 OSI 망관리 모델 사이의 변환을 위해 다음과 같은 기능을 가져야 한다. 첫째는 CORBA 관리자의 객체에 대한 호출을 OSI 망관리 시스템의 피관리 객체에 대한 호출로 변환하여야 한다. 이를 위해 CORBA의 객체참조자와 OSI 피관리 객체의 이름 사이의 변환과, CORBA의 IDL로 표현된 객체와 OSI 망관리 모델의 GDMO/ASN.1을 이용한 피관리 객체 사이의 사양 변환이 이루어져야 한다. 둘째로 OSI 피관리 객체에서 발생한 이벤트가 CORBA 관리자로 효율적으로 전달하기 위한 기법과 OSI 망관리 모델에서 불필요한 이벤트 메시지를 줄이기 위한 이벤트 필터링(Filtering) 기능 등을 가져야 할 것이다.

3 OSI 피관리 객체 생성 및 접근 절차

CORBA 관리자가 OSI 피관리자 객체를 생성하고 이름을 할당하기 위해 필요한 절차는 [그림 2]와 같다.

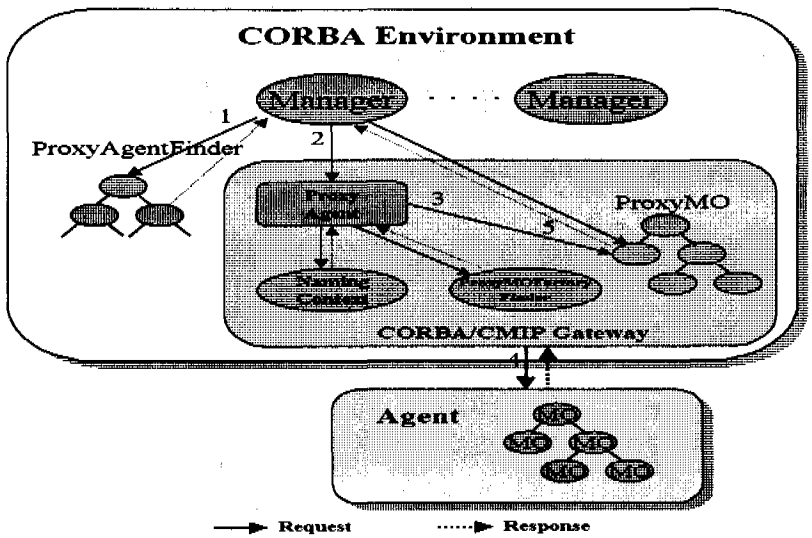


[그림 2] OSI 피관리 객체 생성 절차

1. CORBA 관리자가 하나의 OSI 도메인에 접근하려면 해당 도메인의 ProxyAgent에 대한 객체 참조자를 얻어야 하는데, 이를 위해 CORBA 관리자는 지역 ProxyAgentFinder를 통해 해당 ProxyAgent의 참조자를 얻는다.
2. CORBA 관리자는 생성할 OSI 피관리객체에 대한 생성자를 찾기 위해 해당 Proxy-Agent의 get_ProxyMO_factory_finder 연산을 통해 그 도메인의 ProxyMOFactory-Finder 객체의 참조자를 얻는다.
3. ProxyMOFactoryFinder 객체는 해당 ProxyMOFactory를 찾아 CORBA 관리자에게 돌려준다.
4. CORBA 관리자는 돌려받은 객체참조자를 이용하여 피관리객체 생성을 위한 연산을 호출한다.
5. CORBA 관리자로부터 호출된 ProxyMO에 대한 피관리객체 생성 연산은 CORBA/CMIP 게이트웨이 프로세스에 의해 OSI의 피관리객체 생성 연산으로 변환된 후, 해당 OSI 에이전트에 전달되어 피관리객체 생성 연산을 수행한다.
6. OSI 에이전트로부터 생성연산의 결과가 도착하면 CORBA/CMIP 게이트웨이 프로세스는 생성된 피관리객체의 ProxyMO에 대한 객체참조자를 CORBA 관리자에 돌려준다.

CORBA 관리자가 이미 존재하는 OSI 피관리자 객체에 접근하기 위해 필요한 절차는

[그림 3]과 같다.

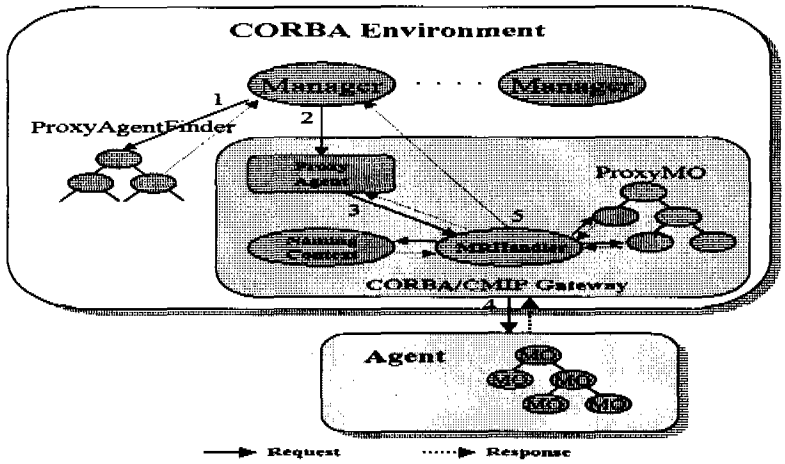


[그림 3] OSI 피관리 객체 접근 절차

1. CORBA 관리자는 하나의 OSI 도메인에 접근하기 위해서는 해당 ProxyAgent에 대한 객체 참조자를 얻어야 하는데, 이를 위해 CORBA 관리자는 지역 ProxyAgentFinder를 통해 해당 ProxyAgent의 참조자를 얻는다.
2. CORBA 관리자는 호출된 피관리객체의 이름으로부터 해당 ProxyMO의 객체참조자를 얻기위해 그 ProxyAgent의 get_domain_naming_context 연산을 통해 해당 도메인의 NamingContext를 얻고, NamingContext의 resolve 연산을 통해 해당 ProxyMO의 객체 참조자를 얻는다. 만약 해당 OSI 피관리객체에 대한 ProxyMO가 존재하지 않으면 NamingContext는 ProxyMOFacroty를 통해 ProxyMO를 생성한다.
3. CORBA 관리자는 돌려받은 ProxyMO 객체참조자를 이용하여 OSI 피관리객체에 대한 연산을 호출한다.
4. CORBA 관리자로부터 호출된 ProxyMO 객체에 대한 연산은 CORBA/CMIP 게이트웨이 프로세스에 의해 OSI의 피관리 객체에 대한 연산으로 변환된 후, 해당 OSI 에이전트에 전달되어 피관리 객체에 대한 연산을 수행한다.
5. OSI 에이전트로부터 연산에 대한 결과가 도착하면 CORBA/CMIP 게이트웨이 프로세스는 그 결과를 CORBA 관리자에 돌려준다.

OSI 관리는 한번의 호출을 통해 여러 피관리자들로부터 연산을 요청할 수 있다. 이를 위해 OSI 관리는 Scoping 기능을 통해 피관리 객체 그룹을 지정하고 이들에게 하나의 연산을 호출하고, 그 그룹내의 각각의 피관리 객체들로부터 결과를 받을 수 있는 다중응답(Multiple replies) 기능을 가지고 있다. 이러한 기능을 통해 OSI는 한번의 호출을 통해 여러 피관리 객체들에 대한 연산을 효율적으로 처리할 수 있다. 그러나 CORBA에서는 이러

한 Scoping 기능과 다중응답기능을 가지고 있지 않다. 따라서 이러한 OSI의 Scoping 기능과 다중응답기능을 처리하기 위해 CORBA/CMIP 게이트웨이에 MRHandler(Multiple Reply Handler)를 두고 이 MRHandler 객체가 이를 처리하는데 그 과정은 [그림 4]와 같다.



[그림 4] OSI Scoped 연산 처리 절차

1. CORBA 관리자는 하나의 OSI 도메인에 접근하기 위해서는 해당 ProxyAgent에 대한 객체 참조자를 얻어야 하는데, 이를 위해 CORBA 관리자는 지역 ProxyAgentFinder를 통해 해당 ProxyAgent의 참조자를 얻는다.
2. CORBA 관리자는 Scoped 연산을 처리하기 위해 ProxyAgent의 create_MRHandler 연산을 통해 MRHandler를 생성한다.
3. CORBA 관리자는 리턴받은 MRHhadler 객체 참조자를 이용하여 OSI 피관리객체들에 대한 Scoped 연산을 호출한다.
4. MRHandler는 scoped_resolve 연산을 통해 해당되는 ProxyMO 객체들의 참조자들을 얻고, 호출된 연산은 OSI의 피관리객체들에 대한 연산으로 변환된 후, 해당 ProxyMO 객체들을 통해 OSI 에이전트에 전달되어 피관리객체들에 대한 연산을 수행한다.
5. OSI 에이전트에서 수행된 각각의 피관리객체들에 대한 연산들의 결과가 도착하면 해당 ProxyMO는 그 결과를 MRHandler에 돌려주고, MRHandler는 해당 ProxyMO들로부터 받은 결과가 모두 도착하면 그 결과를 CORBA 관리자에게 돌려준다.

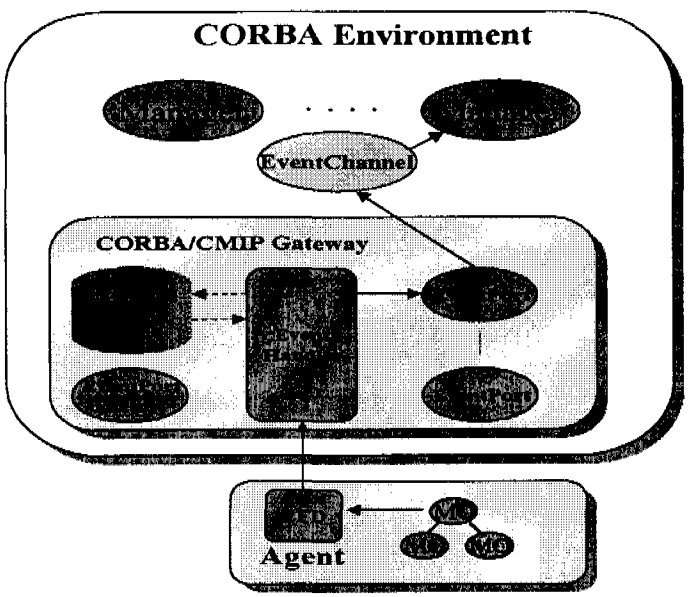
4 CORBA/CMIP 게이트웨이를 이용한 이벤트 처리

OSI 피관리 객체에서 발생한 이벤트는 해당 이벤트를 요청한 CORBA 관리자에게 효과

적으로 전달될 수 있어야 한다. CORBA에서는 이러한 이벤트를 이벤트 채널을 이용하여 전달하고, 이벤트 생성자(Producer)들과 소비자(Consumer)들 사이에 푸시(Push) 또는 풀(Pull) 형태의 인터페이스를 정의하고 있다. 이러한 이벤트 처리를 위해 본 논문에서는 각 CORBA 관리자당 하나의 이벤트 포트를 생성하고 이를 이용하여 이벤트를 처리하는 방법과, 각 이벤트당 하나의 ProxyEFD(Proxy Event Forwarding Discriminator)를 생성하고 이를 이용하여 이벤트를 처리하는 두 가지 방법에 대해 소개하고 그들 사이의 장단점에 대해 기술한다.

4.1 이벤트포트를 이용한 방법

이벤트포트를 이용한 방법은 각 CORBA 관리자당 하나의 EventPort 객체를 두고 해당 관리자로 오는 모든 이벤트는 이 EventPort를 통해 전달되도록 하는 방법이다. 이를 위해 필요한 CORBA/CMIP 게이트웨이 구조는 [그림 5]와 같다.



[그림 5] 이벤트처리를 위한 CORBA/CMIP 게이트웨이 구조
(EventPort 이용)

각 객체들의 기능은 다음과 같다.

● EventHandler 객체는 크게 두 가지 기능을 수행한다. 하나는 CORBA 관리자로부터 이벤트포트 생성요구를 받아 EventPort 객체를 생성하여 그 정보를 이벤트정보저장소(Event Information Repository: EIR)에 저장하고 CORBA 관리자에게 돌려주는 기능을 수행하고, 또 하나는 OSI 피관리자에서 발생하는 이벤트를 받아 그 이벤트를 요청한 관리자들을 EIR에서 찾아 해당 EventPort들로 전달해 주는 역할을 수행한다.

● EventPort 객체는 이벤트 서비스를 원하는 CORBA 관리자당 하나씩 생성되며, 해당 CORBA 관리자로부터 새로운 이벤트의 추가 및 기존 이벤트의 삭제 요구에 대해 해당 정보를 EIR에 저장하고 새로운 이벤트채널의 생성 및 삭제를 수행하고 해당 정보를 EIR에 저장한다. 또한 새로운 이벤트가 EventHandler를 통해 도착하면 해당 이벤트채널을 통해 CORBA 관리자에게 전달한다. CORBA 관리자와 EventPort 사이에는 해당 CORBA 관리자 가 요청한 이벤트 종류만큼의 이벤트채널과 각 이벤트에 대한 필터가 존재하며, 각 이벤트는 필터 조건을 만족할 때 해당 이벤트채널을 통해 전달된다. 그들 사이에 이벤트가 전달되는 방법은 [그림 5]와 같은 CORBA의 이벤트 전달 모델을 사용하며 각각의 전달 모델은 CORBA 관리자가 해당 이벤트를 추가할 때에 이벤트채널의 관리인터페이스(CosEventChannelAdmin)를 이용하여 지정한다.

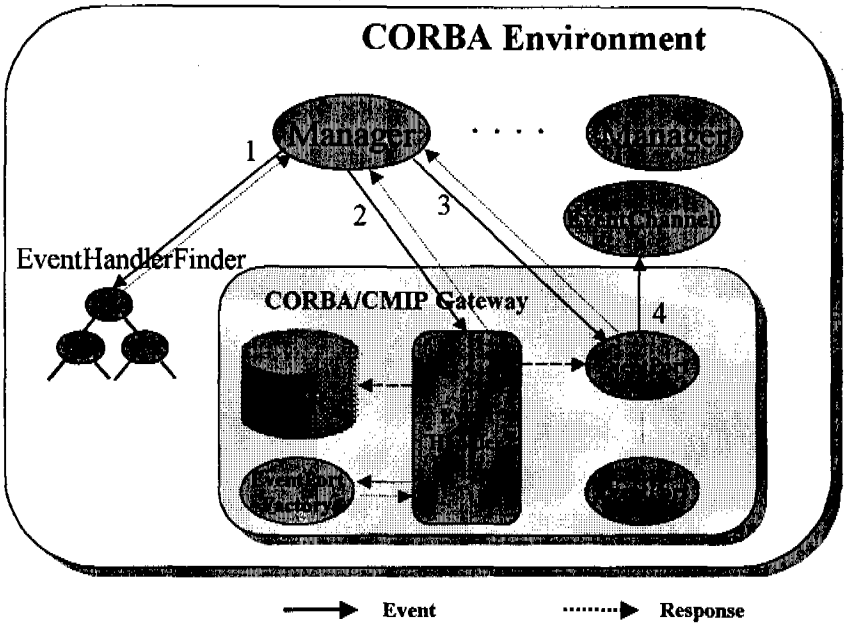
● EventPortFactory 객체는 EventHandler에 의해 해당 CORBA 관리자에 대한 EventPort를 생성하기 위한 객체이다.

● EIR(Event Information Repository)은 각 EventPort에 대한 정보를 저장하고 있는데 해당 EventPort가 할당된 CORBA 관리자, 각 EventPort에 대해 요청된 이벤트들의 종류들에 대한 정보들을 가지고 있다.

4.1.1 이벤트등록 절차

CORBA 관리자는 이벤트 리포트를 얻기 위해서는 자신의 EventPort를 생성하고 그 EventPort를 통해 요청할 이벤트 정보를 등록해야 한다. CORBA 관리자는 자신이 초기화될 때 또는 처음 이벤트가 필요할 때 EventPort를 생성할 수 있으며, 이를 위한 절차는 [그림 6]과 같다.

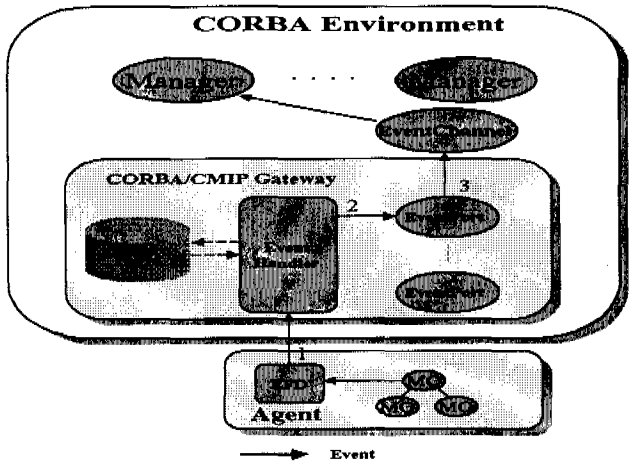
1. CORBA 관리자는 지역 EventHandlerFinder 객체를 통해 EventHandler를 찾는다.
2. CORBA 관리자는 EventHandler의 create_EventPort 연산을 통해 EventPort를 생성하고, 요청을 받은 EventHandler는 새로운 EventPort를 생성한 뒤, 요청한 CORBA 관리자와 EventPort 정보를 EIR에 저장하고, 그 EventPort의 객체참조자를 CORBA 관리자에게 돌려준다.
3. CORBA 관리자는 이후 새로운 이벤트가 필요할 때마다 할당된 EventPort의 add_Event 연산을 통해 새로운 이벤트를 등록하고 이벤트를 등록을 해제하고 싶을 때에는 drop_Event 연산을 통해 이벤트를 해제한다.
4. EventPort 객체는 add_Event 연산을 통해 이벤트 추가 요청을 받으면 새로운 EventChannel을 생성하고 해당 이벤트에 대한 필터 정보의 일부는 자신에게 일부는 해당 OSI 에이전트의 EFD에 설정하고 이 정보를 EIR에 저장한 뒤, 새로 생성된 EventChannel의 객체참조자를 CORBA 관리자에 돌려준다. 그리고 drop_Event 연산을 통해 이벤트 삭제 요청을 받으면 EventChannel의 삭제를 수행하고 이 정보를 EIR에 갱신한다.



[그림 6] EventPort를 이용한 이벤트 등록 절차

4.1.2 이벤트처리 절차

이벤트포트를 이용하여 OSI 피관리객체에서 발생한 이벤트를 CORBA 관리자에 전달하는 절차는 [그림 7]과 같다.

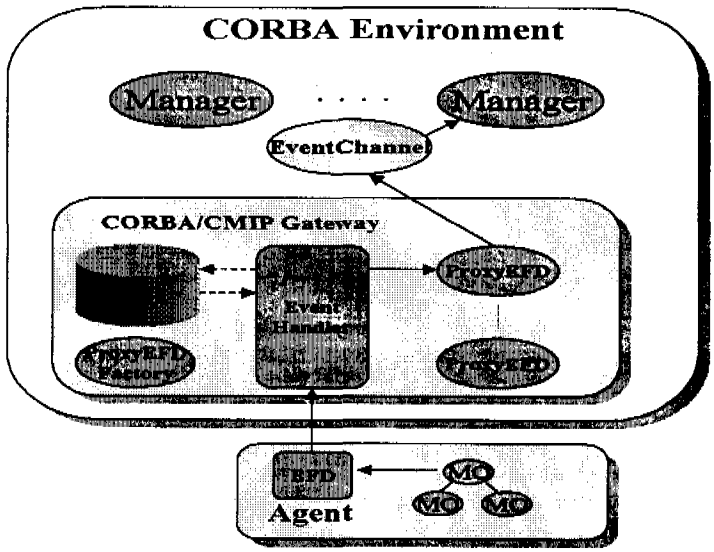


[그림 7] EventPort를 이용한 이벤트 처리 절차

1. OSI 피관리객체에서 발생한 이벤트는 OSI 에이전트의 EFD 필터 조건을 만족하면 CORBA/CMIP 게이트웨이의 EventHandler로 전달된다.
2. EventHandler는 EIR에 저장된 이벤트들에 대한 정보를 이용하여 해당 이벤트를 요청한 CORBA 관리자의 EventPort들의 객체들을 얻고 그 이벤트 정보를 해당 EventPort 객체들에 전달한다.
3. 이벤트를 전달받은 EventPort 객체는 자신에 저장된 필터 조건을 만족하면 그 이벤트 정보를 해당 EventChannel을 통해 CORBA 관리자에게 전달한다.

4.2 ProxyEFD 객체를 이용한 방법

ProxyEFD(Proxy Event Forwarding Discriminator) 객체를 이용한 방법은 각 이벤트당 하나의 ProxyEFD 객체를 생성하고 해당 이벤트를 요청한 CORBA 관리자들에 대한 정보를 EIR에 유지한다. 이후 OSI 피관리객체로부터 이벤트가 발생하면 EIR로부터 해당 이벤트를 요청한 CORBA 관리자들을 찾아 각 관리자들에게 전달하여 주는 방법이다. 이를 위해 필요한 CORBA/CMIP 게이트웨이 구조는 [그림 8]과 같다.



[그림 8] 이벤트처리를 위한 CORBA/CMIP 게이트웨이 구조
(ProxyEFD 이용)

각 객체들의 기능은 다음과 같다.

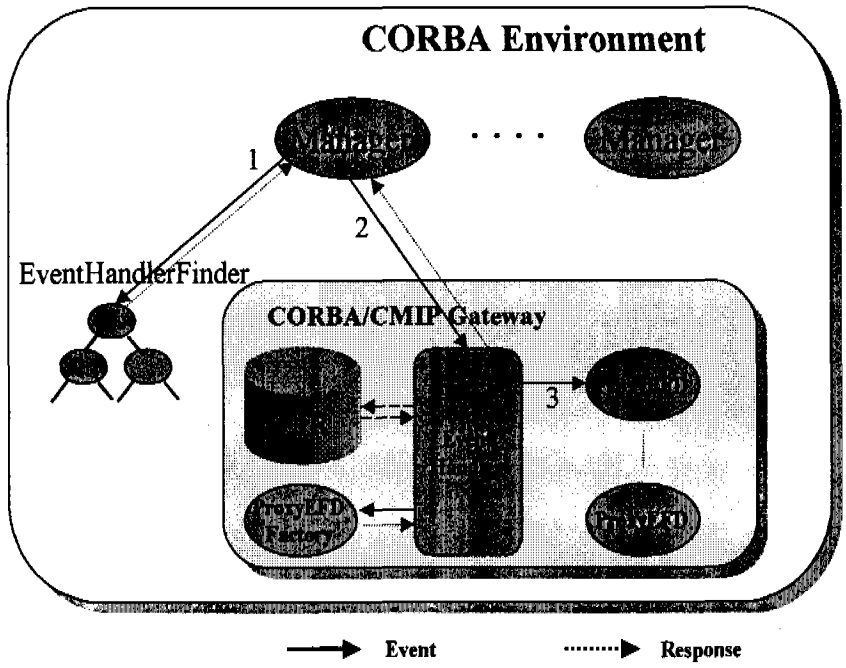
● EventHandler 객체는 크게 두 가지 기능을 수행한다. 하나는 CORBA 관리자로부터 새로운 이벤트 리포트 요구를 받아 해당 이벤트에 대한 ProxyEFD 객체를 EIR로부터 탐색하여 존재하지 않으면 새로운 ProxyEFD를 생성하고 그 ProxyEFD와 이벤트에 대한 종류와 필터 정보를 EIR에 저장하고, 그 ProxyEFD의 객체참조자를 CORBA 관

리자에게 돌려준다. 만약 요청한 이벤트가 이미 존재하면 EIR로부터 찾은 ProxyEFD의 이벤트 요청 관리자 리스트에 추가하고, 그 ProxyEFD의 객체참조자를 CORBA 관리자에게 돌려준다. 또 하나는 OSI 피관리객체로부터 이벤트가 발생하였을 경우, 그 이벤트에 해당하는 ProxyEFD를 EIR에서 찾아 그 이벤트 정보를 해당 ProxyEFD에 전달해 주는 역할을 수행한다.

- ProxyEFD 객체는 이벤트 종류당 하나씩 생성되며 자신의 이벤트를 요청한 CORBA 관리자들의 리스트를 유지하고 있어, EventHandler로부터 발생한 이벤트 정보를 전달받아 직접 자신의 이벤트 정보를 요청한 CORBA 관리자들에게 전달하거나 CORBA의 EventChannel을 통해 전달하는 기능을 수행한다.
- ProxyEFDFactory 객체는 새로운 이벤트를 요청할 경우 해당 이벤트에 대한 ProxyEFD를 생성하기 위한 생성자이다.
- EIR은 각 ProxyEFD에 대해 이벤트의 종류 및 필터정보들을 저장하고 있는 이벤트정보 저장소로, CORBA 관리자가 새로운 이벤트를 요청할 경우 또는 OSI 피관리객체로부터 새로운 이벤트가 도착하였을 때 해당 이벤트에 대한 ProxyEFD를 찾기 위해 사용된다.

4.2.1 이벤트 등록 절차

하나의 이벤트는 하나의 ProxyEFD를 통해 전달되며 CORBA 관리자가 새로운 이벤트를 요청할 경우, CORBA/CMIP 게이트웨이는 해당 이벤트에 대한 ProxyEFD 객체를 생성하고 그 정보를 EIR에 저장하는데, 이를 위한 절차는 [그림 9]와 같다.

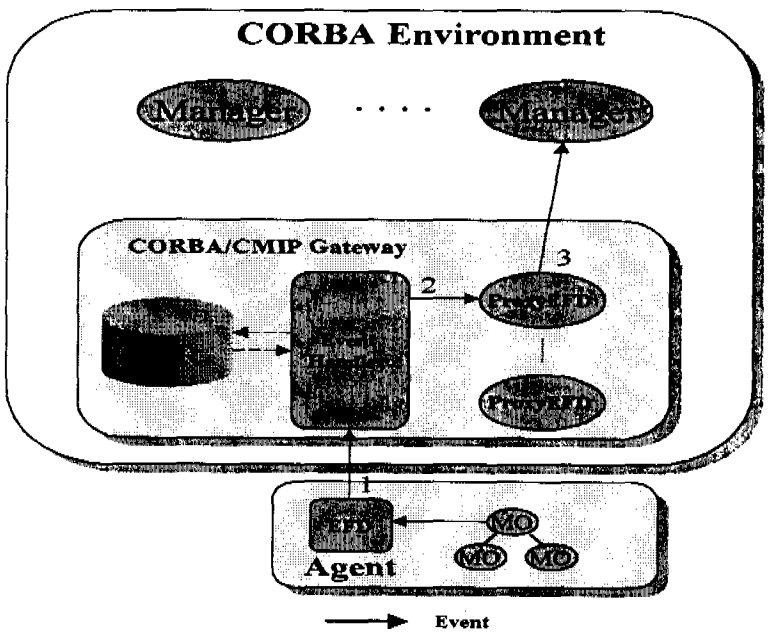


[그림 9] ProxyEFD를 이용한 이벤트 등록 절차

1. CORBA 관리자는 지역 EventHandlerFinder 객체를 통해 EventHandler를 찾는다.
2. CORBA 관리자는 EventHandler의 get_ProxyEFD 연산을 통해 새로운 이벤트에 대한 ProxyEFD에 대한 참조자를 요청한다.
3. EventHandler 객체는 CORBA 관리자의 get_ProxyEFD 요청에 대해 EIR로부터 해당 ProxyEFD를 탐색하고 해당 ProxyEFD가 존재하면 그 ProxyEFD의 이벤트 요청 관리자 리스트에 정보에 새로운 CORBA 관리자를 추가하고, 그 ProxyEFD의 객체 참조자를 CORBA 관리자에 돌려준다. 만약 EIR에 해당 ProxyEFD가 존재하지 않으면 ProxyEFDFactory를 통해 새로운 ProxyEFD를 생성하고, 그 정보를 EIR에 등록한 뒤 그 ProxyEFD의 객체참조자를 CORBA 관리자에 돌려준다. 새로운 이벤트에 ProxyEFD가 생성되는 경우 EventHandler는 그 이벤트에 대한 EFD가 해당 OSI 에이전트에서 생성(또는 활성화)되도록하고 그 EFD에 해당 이벤트에 대한 필터를 설정한다.

4.2.2 이벤트처리 절차

ProxyEFD를 이용하여 OSI 피관리객체에서 발생한 이벤트를 CORBA 관리자에 전달하는 절차는 [그림 10]과 같다.



[그림 10] ProxyEFD를 이용한 이벤트 처리 절차

1. OSI 피관리객체에서 발생한 이벤트는 OSI 에이전트의 EFD 필터 조건을 만족하면 CORBA/CMIP 게이트웨이의 EventHandler로 전달된다.
2. EventHandler는 EIR에 저장된 이벤트들에 대한 정보를 이용하여 해당 ProxyEFD 객체참조자를 얻고 그 이벤트 정보를 해당 ProxyEFD 객체에 전달한다.
3. 이벤트를 전달받은 ProxyEFD 객체는 그 이벤트 정보를 자신에 등록되어 있는 CORBA 관리자들에게 전달한다.

4.3 이벤트처리 방법들의 비교

EventPort를 이용한 이벤트처리 방법은 각 CORBA 관리자마다 하나씩 EventPort 객체를 이용하여 이벤트들을 전달하는 방식으로, 일반적으로 관리자는 에이전트에 비해 그 수가 적으므로 비교적 적은 수의 EventPort를 필요로 한다. 그러나 서로 다른 CORBA 관리자들이 종류와 필터 조건이 같은 이벤트를 원할 경우 그 이벤트가 각각의 이벤트 포트에 복사되어야 하고, 또한 하나의 이벤트 포트가 여러 개의 서로 다른 이벤트를 전송하여야 하므로 이벤트 포트의 구현이 어렵다. 그리고 필터링을 각각 해당 EventPort에서 수행하므로 OSI 에이전트와 CORBA/CMIP 게이트웨이 사이의 이벤트 메시지의 양이 많아진다.

ProxyEFD를 이용한 방법은 각 이벤트당 하나씩 ProxyEFD를 생성하여 이벤트가 발생하면 해당 ProxyEFD에 전달되고, ProxyEFD는 요청한 CORBA 관리자들에게 전달하게 된다. 이 방법은 이벤트 종류가 많아지면 필요한 ProxyEFD 객체가 많아지지만, 서로 다른 CORBA 관리자들이 종류와 필터 조건이 같은 이벤트를 원할 경우 하나의 ProxyEFD에 의해 각 CORBA 관리자들에게 전달되므로 효율적으로 이벤트를 전송할 수 있다. 또한 같은 종류의 이벤트에 대해 필터조건을 일부는 OSI 에이전트의 EFD에 또 일부는 ProxyEFD에 유지함으로써, 필터조건 설정이 유연하고 OSI 에이전트와 CORBA/CMIP 게이트웨이 사이의 이벤트 메시지 양을 줄일 수 있다.

EventPort를 이용한 방법과 ProxyEFD를 이용한 방법은 각각 장단점이 있지만, ProxyEFD를 이용한 방법이 구현에 있어 용이하고 또한 필터조건 설정이 용이하여, 향후 CORBA/CMIP 게이트웨이의 구현은 ProxyEFD를 이용한 방법을 사용하려고 한다.

5 결 론

통신망의 새로운 통신 기술과 서비스들의 도입으로 망과 서비스 관리 기술이 더욱 중요해지고 있다. 지금까지 망과 서비스 관리 기술은 적용 분야에 따라 각기 서로 다른 표준화 그룹에 의해 표준화가 되어 왔다. 즉, SNMP 기반 망관리 모델은 일반적인 컴퓨터 통신망의 관리를 위한 사실상의 표준으로 정착되었고, OSI의 CMIS/CMIP 기반의 TMN은 전기통신망 관리를 위한 표준으로 ISO와 ITU에 의해 제정되었다. 이와는 별개로 응용 프로그램들 사이의 상호운용성을 제공하기 위한 기술로 분산 객체지향 기술이 OMG를 비롯한 많은 시스템 개발자들에 의해 표준화가 되고 있다. 통신망에서의 망과 서비스 관리도 역시 하나의 분산 응용 프로그램이라는 관점에서 이러한 분산 객체지향 모델은 다양한 이질적

인 기술을 포함하는 망과 서비스 관리를 일관되고 통일된 방법으로 제공할 수 있는 모델로 여겨진다. 이에 따라 OMG를 비롯한 X/Open, NMF, TINA-C 등 많은 표준화 그룹과 기업들에 의해 OSI 기반 TMN과 CORBA를 통합하기 위한 연구가 이루어지고 있다.

본 논문에서는 이러한 연구의 하나로 CORBA/CMIP 게이트웨이를 이용한 CORBA 관리 모델과 OSI 관리 모델의 통합 방법에 대해 기술하였다. 서로 다른 모델을 사용하는 망관리 시스템들 사이의 통합을 위해서는 각기 다르게 표현되는 망관리 정보와 관리기능의 변환이 필요한데, 이와 같은 변환 기능이 CORBA/CMIP 게이트웨이에 의해 수행이 된다. 본 논문에서는 이러한 게이트웨이를 이용한 통합 망관리 구조를 제시하고, CORBA 관리자가 OSI 피관리 객체를 투명하게 접근하기 위한 절차와 OSI 피관리 객체에서 발생한 이벤트를 CORBA 관리자들에게 효율적으로 전달하기 위한 방법에 대해 기술하였다.

향후 할 일은 CORBA/CMIP 게이트웨이에서 정의된 각 객체들을 IDL로 표현하고, 이 게이트웨이에 대한 프로토타입을 구현하여 본 논문에서 제시한 게이트웨이를 이용한 망관리 모델의 타당성을 보이는 일과, SNMP 기반의 Internet 관리 모델을 수용하기 위한 CORBA/SNMP 게이트웨이에 대한 연구이다.

[참고문헌]

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

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

[3] Object Management Group, The Common Object Request Broker : Architecture and Specification, Revision 2.0, 1995.

[4] OMG Telecom Task force, CORBA-based Telecommunication Network Management, OMG White Paper, May 1996.

[5] NM Forum, Joint Inter-Domain Management: Specification translation, 1997.

[6] TINA-C, Overall Concepts and Principles of TINA, 1995.

[7] J. Case, M. Fedor, M. Schoffstall and J. Davin, A Simple Network Management Protocol (SNMP), RFC 1157, 1990

[8] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser, Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2), RFC 1907, January 1996.

[9] D. Levi, P. Meyer, and B. Stewart, SNMPv3 Applications, RFC 2273, January 1998.

[10] Bella Ban, IBM Zurich Research Laboratory, A Generic Management Model for CORBA, CMIP, and SNMP, http://www.zurich.ibm.com/bba/outline/outline.html.

[11] Marnix Harssema, University of Twente, Integrating TMN and CORBA, Master Thesis, Oct. 1996.

[12] QinZheng Kong and Graham Chen, Integrating CORBA and TMN Environments, CiTR Technical Journal, 1996.