대규모 망의 안전한 관리를 위한 관리 정보베이스의 접근 제어 모형화

서 재 현† 이 창 진†† 노 봉 남†††

요 약

망 관리 시스템의 핵심적인 구성 요소 중의 하나인 관리 정보베이스는 망 관리에 필요한 모든 정보를 저장하고 있는 개념적인 정보 저장소이다. 망이 안전하게 운용되기 위해서는 관리 정보 베이스에 저장된 관리 객체들에 대한 접근을 효율적으로 통제할 수 있어야 한다. 본 논문에서는 관리자들을 3단계 계층 구조로 분할하여 거대한 망의 관리를 보다 효율적으로 수행할 수 있도록 하였다. 그리고, 강제적 접근 제어와 역할기반 접근 제어 정책을 이용하여 관리 객체에 대한 접근 제어 및 관리 객체에서 발생하는 통지들에 대한 접근 제어를 수행할 수 있는 방법을 제시하였다. 또한, 이러한 접근 제어 모델을 능동 객체지향 데이타 모델을 사용하여 모형화하므로써 능동 객체지향 데이타베이스로의 사상을 용이하게 하였다.

An MIB Access Control Modeling for the Secure Management of Large Networks

Jae Hyeon Seo† Chang Jin Lee†† and Bong Nam Noh†††

ABSTRACT

An MIB is the heart of a network management system and it stores all information that is necessary for network management. To operate networks safely, it is essential to control accesses to managed objects. This paper provides three-level architecture of managers so as to perform network management more efficiently in large networks. Moreover, mandatory access control(MAC) policy and role-based access control policy are adopted to ensure the secure access to the MIB. These policies are modeled by using the active object-oriented data model, which makes easy to map these access control models into the active object-oriented database.

1. 서 론

컴퓨터를 사용하는 사용자의 수가 급증하고 상호 독립적으로 존재하던 컴퓨터들이 통신망을 통하여 상호 연동됨에 따라 전체적인 통신 망의 규모가 점점 커지고 복잡해지게 되었다. 또한 망을 이용하는 사용자들의 요구사항이 다양해져 이를 효율적으로 관리해 줄 수 있는 망 관리 시스템이 통신 망의 운용에 필수적인 요소가 되었다.

망 관리 시스템의 가장 핵심적인 부분은 망 관리에 필요한 모든 정보를 저장하고 있는 망 관리 정보베이스(Management Information Base:MIB)이다. 즉, 망 관리 정보베이스는 망 관리에 필요한 모든 정보를 저장하고 있는 개념적인 정보 저장소로서, 많은 관리 객체들(managed objects)로 구성되어 있다[3]. 이러한 망 관리 정보 베이스가 안전하게 운용되기 위해서는 관리 객체에 대한 접근을 효율적으로 통제할 수 있어야 하며 또한 관리 객체에서 발생하는 사건의 통지에 대해서도 효율적으로 통제할 수 있어야 한다.

관리자는 CMIS/CMIP을 이용하여 대행자와 관리 정보를 교환하며, 마찬가지로 대행자는 CMIS/CMIP을 이용하여 관리 정보베이스에 저장된 관리 객체에 접근하여 관리 정보를 얻어내거나 변경을 가한다. CMIS는 M-EVENT-REPORT, M-GET, M-SET, M-ACTION, M-CREATE, M-DELETE, 그리고 M-CANCEL-GET

† 정 회 원 : 송원전문대학 전자계산학과 전임강사
†† 정 회 원 : 전남대학교 대학원 전산통계학과
††† 정 회 원 : 전남대학교 전산학과 교수
논문접수 : 1995년 5월 16일, 심사완료 : 1995년 6월 29일

등의 서비스 프리미티브로 구성되어 있는데, 이들은 실제 CMIP의 프로토콜 데이타 단위의 형태로 교환된다.

본 논문에서는 망 관리 구조를 통합 관리자, 영역 관리자, 그리고 대행자로 계층화하므로써 망 관리에 대한 부담이 하나의 관리자에게 집중되는 것을 방지하였고, 전체적인 망을 보다 효율적으로 관리할 수 있게 하였다. 또한 여러가지 종류의 보안 정책이 공존하는 거대한 망에서 관리 정보베이스에 저장된 관리 객체들에 대한 접근을 효과적으로 제어할 수 있는 방법을 제시하였으며, 이를 위한 모델을 능동 객체지향 데이타 모델[15]을 이용하여 효과적으로 설계하였다. 역할기반 접근 제어 정책에서는 관리자들을 역할로 그룹화하므로써 관리자들의 객체에 대한 접근 권한을 관리하기 쉽게 해주는 장점을 가지고 있다.

본 논문에서는 망 관리 정보에 대한 접근제어 모델 및 관련된 관리 객체들을 객체지향 방법으로 모형화하고, 이러한 정보를 지속적으로 유지할 수 있도록 망 관리 정보베이스에 포함된 관리 객체들과 이들의 능동적인 특성을 객체지향 데이타베이스 구조로 모형화하는 방법을 제안하였다. 제안된 방법은 망 관리 정보에 대한 접근을 제어하는데 필요한 관리 객체들이 갖는 능동적 특성들을 능동 데이타베이스의 능동 규칙을 이용하여 모형화하므로써 능동 객체지향 데이타베이스로의 사상을 용이하게 하였다.

2장에서는 관리 정보베이스의 모형화와 접근 제어 정책들에 관하여 수행된 기존의 연구들을 살펴보고, 3장에서는 OSI 망관리 표준안에서 제시하는 망 관리 모델과 관련 개념들을 간략히 설명하고, 효율적인 망 관리를 위하여 망 관리 구조를 3단계의 계층으로 분할한 구조를 제시하고 있다. 4장에서는 관리 정보베이스의 접근 제어 모델들에 대하여 살펴보고, 5장에서는 4장에서 정의한 접근 제어 정책들을 관리 정보베이스에 적용하는 방법과 이러한 접근 제어 모델들을 능동 객체지향 모델을 이용하여 모형화하는 방법 및 결과에 대하여 설명하고 있다. 마지막으로 6장에서는 결론과 앞으로의 추후 연구방향을 언급한다.

2. 관련 연구

망 관리에 필요한 모든 정보들을 보관하고 있는 관리 정보베이스를 효과적으로 구축하기 위한 연구가 활발하게 진행되어 왔다. 그러나 지금까지의 연구는 대부분이 관리 정보베이스의 구현 단계에 초점을 맞추어 진행되어 왔고 개념적인 수준에서의 연구는 거의 전무한 실정이다. 그러나 관리 정보베이스를 잘 구축하기 위해서는 개념적인 설계 단계에서부터 필요한 기능 및 관리 객체들이 정확하게 정의되고 모형화되어야 한다.

따라서 복잡한 망 관리 시스템의 의미를 잘 표현할 수 있고, 개념 모델의 의미를 그대로 구현 모델로 사상할 수 있는 객체지향 데이타 모델이 관리 객체의 모형화에 적합하다. 또한 OSI 망 관리 모델이 객체지향 개념에 잘 부합되기 때문에 비교적 최근에 들어와서 객체지향 데이타베이스를 기반으로 망 관리 정보베이스를 구축하려는 시도가 활발히 진행되고 있다.

Oliver[12]는 GDMO를 확장하여 일반 언어로 기술된 관리 객체의 행위 부분을 사건-조건-행위(ECA) 규칙을 기반으로 정형화하고, 이를 기존의 GDMO 템플릿에 통합시켰다. 그러나 이 논문에서는 관리 객체와 그들 사이의 관계를 모형화하기 보다는 단지 관리 객체의 일부분인 행위 부분을 정형화하였을 뿐이다.

Nader[11]는 GDMO로 표현된 관리 객체 클래스를 C++ 클래스로 자동으로 변환하는 알고리즘과 예를 보여주고 있다. 이 연구에서는 관리 객체 클래스에 포함된 모든 속성을 C++ 속성으로, 속성에 대한 연산, 통지, 행위 등은 메소드로 변환하였다. 이러한 관리 객체에 관한 정보 모델인 GDMO 형태를 구현 사양으로 변환하는 연구는 상당히 많이 진행되었다. 그러나 개념적 설계 단계에서 관리객체를 모형화하는 연구는 상대적으로 매우 미비한 편이다.

Robert[13]는 미국의 항공기의 운항 통제에 사용되어지는 서로 다른 여러 종류의 통신 망을 통합 관리하기 위한 시스템을 개발하는 연구를 수행하였다. [13]에서는 각각의 통신망을 관리하는 요소 관리자가 있고, 이들을 통합 관리하는 통합 관리자가 존재한다. 그리고 각각의 요소 관리자

가 관리하는 관리 정보를 표준화된 OSI 망 관리 정보의 형태로 변환하여 요소 관리자와 통합 관리자 사이의 관리 정보의 교환이 표준화된 형태로 이루어 질 수 있는 기능을 제공해 주는 대행자가 있다. 통합 관리자는 전체 망의 구성 관리, 경고 보고, 로깅, 그리고 사건 관리 기능 등의 역할을 수행한다. 그러나 [13]에서는 망 관리에 중요한 정보를 모두 저장하고 있는 관리 정보베이스의 보안에 관한 연구는 수행되지 않았다.

한편, 망 관리 정보베이스는 망 관리에 중요한 정보들을 모두 포함하고 있기 때문에 이 정보들에 대한 접근을 적절히 통제하여 이들을 안전하게 유지하는 것이 전체적인 망의 보안 유지 및 정상적인 운용에 매우 중요하다. 기존의 데이타베이스 분야에서 보안에 대한 연구는 상당히 진행되었지만 이들을 망 관리 정보베이스에 적용한 사례는 드문 편이다. 지금까지의 수행된 연구들 중 몇가지를 살펴보자.

Rabitti[9]는 데이타베이스 시스템에서 권한 부여 모델에 대한 연구를 수행하였는데, 권한 부여의 형식을 사용자 또는 주체의 집합, 권한 부여 객체 집합 그리고 권한 부여 형태로 구성되는 하나의 튜플로 정의하였다. 또한 사용자들에 대한 권한 부여의 수를 감소시키기 위하여 역할의 개념을 도입하여 역할 격자구조를 형성하였다. 그러나 본 논문에서는 역할을 객체지향의 상속성을 이용하여 정의하였다.

Lochovsky[8]는 특정 데이타베이스 관리 시스템의 사용자들을 가능한 작은 규모로 보안 분류하는 것을 지원하기 위하여 객체지향 기법을 이용하여 역할기반 보안 스키마를 제안하였다. 이 연구에서도 Rabitti의 역할 격자구조와 유사한 역할 네트워크를 이용하여 역할들의 구조를 정의하였다.

본 논문에서는 효율적인 망 관리를 지원하고 망 관리의 모든 부담이 하나의 관리자에게 집중되는 문제점을 해결하기 위하여 관리 구조를 세 개의 계층으로 분할하여 망 관리 업무를 분산하였다. 또한 능동 객체지향 데이타 모델을 이용하여 망 관리 정보베이스에 포함된 관리 객체들에 대한 접근 제어 모델을 효과적으로 모형화하였다.

제안된 방법에서는 관리 객체에 대한 접근 제어를 수행하는데 필요한 관리 객체들을 능동 객체지향 데이타 모델을 이용하여 모형화하므로써 이들의 능동적인 특성을 잘 모형화하였을 뿐만 아니라 개념 모델의 실제 객체지향 데이타베이스로의 사상을 용이하게 하였다.

3. OSI의 표준 망 관리 모델

3.1 OSI 망 관리 모델의 개요

OSI 시스템 관리는 객체지향 개념을 기반으로 하여 관리자/대행자 구조를 이루고 있다. OSI 망 관리 시스템의 중요한 구성 요소들은 크게 관리자, 대행자, 관리 객체 등이 있다. 먼저, 관리자(manager)는 대행자를 통하여 관리 연산을 수행하고 그로부터 발생하는 통지(notification)들을 받아서 처리하는 역할을 수행하며, 대행자(agent)는 관리 객체에 대하여 관리 연산을 수행하고 관리 객체에서 발생한 통지들을 관리자에게 보내는 역할을 수행한다.

그리고 관리 객체는 OSI 망 관리 프로토콜을 사용하여 관리되는 자원들을 망 관리 관점에서 추상적으로 표현해 놓은 것으로서, 속성, 연산, 통지들을 포함하고 있다. 또한 관리 객체 클래스는 동일한 속성, 관리 연산, 통지들을 공유하는 관리 객체들의 집합으로 정의된다[2].

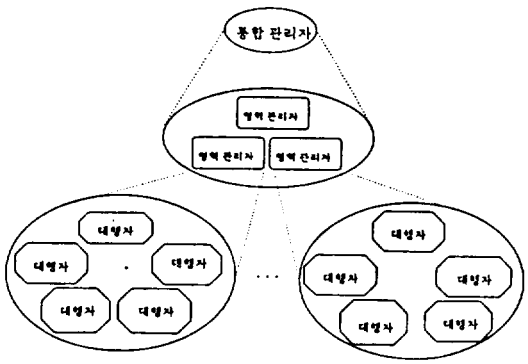
ISO 7498-4 표준안[1]에서는 망 관리를 위한 사용자들의 요구사항을 각각의 기능에 따라 다섯 개의 관리 기능 영역, 즉, 결함 관리, 계정 관리, 구성 관리, 성능 관리, 보안 관리로 구분하고 있다[1]. 이와 같은 관리 기능 영역을 지원하기 위하여, OSI에서는 지금까지 13개의 시스템 관리 기능들을 정의하고 있다. 그 중에서 특히 망의 보안 유지에 중요한 역할을 하는 것이 사건 보고 관리 기능과 접근 제어 관리 기능이다.

사건 보고 관리 기능은 망에 발생한 비정상적인 예외상황의 발생을 관리자에게 능동적으로 통보해 주는 역할을 수행한다. 그리고 접근 제어 관리 기능은 망 관리 정보의 무결성 유지에 매우 중요한 역할을 수행하며, 망 관리 정보에 대한 불법적인 접근 시도가 있을 때, 이를 효과적으로 막아주며 또한 관리자에게 이러한 사실을 알려주는 역할을 수행한다.

3.2 망 관리자들 사이의 계층 구조

대규모 망의 운용을 위해 망 관리 영역은 관리상의 편의, 지리적인 여건, 그리고 망 구성상의 문제 등과 같은 여러가지 이유 때문에 여러 개의 하위 관리 영역으로 나누어 질 수 있다. 이렇게 나뉘어진 하위 관리 영역들에게는 그 영역을 관리하는 관리자가 각각 할당된다. 영역 관리자들은 각각 자기에게 할당된 고유의 관리 작업을 수행하며, 전체적으로 자신의 상위 관리자에 의해 통합 관리된다.

본 논문에서는 망 관리 구조를 (그림 1)과 같이 3단계의 계층구조로 나누었다.



(그림 1) 망 관리 계층 구조
(Fig. 1) Network management hierarchy

계층 구조상의 최하위 수준에 있는 대행자는 망관리 시스템의 관리 자원인 실제적인 시스템 또는 서비스 구성요소들의 지역적 관리 객체를 관리하는 역할을 수행한다. 영역 관리자는 이러한 대행자들을 전체적으로 관리하여 전체 망의 일부분인 하나의 해당 영역을 효과적으로 관리하는 임무를 수행한다.

또한 통합 관리자들이 전체적인 망을 관리할 수 있도록 통합 관리자의 관리 정책을 대행자에게 전달하고, 대행자 수준에서 처리하기 힘든 망의 문제점들을 파악하여 처리하며, 만약 영역 관리자 수준에서 처리하기 힘든 일이 발생할 경우 통합 관리자에게 보고하여 적절한 조치를 취할 수 있도록 하는 역할도 수행한다.

계층 구조상의 최상위 수준에 있는 통합 관리자는 망의 전반적인 관리 및 여러 영역 관리자들의 관리, 그리고 영역 관리자들 사이의 상호작용을 관리하는 역할을 수행한다. 통합 관리자는 실질적으로 개개의 관리 객체들의 관리보다는 관리자들의 관리에 더 비중을 두고 있다.

이렇게 다단계의 계층 구조로 망을 관리하므로써 하나의 망 관리자가 매우 크고 복잡한 망을 총괄하여 관리할 때 발생하는 부담을 효과적으로 분산할 수 있고, 보다 신속하게 망의 결함들을 복구하여 결과적으로 훨씬 효율적인 망 관리를 수행할 수 있다.

4. 관리 정보베이스의 접근 제어 모델

본 논문에서는 거대한 망에서 서로 다른 형태의 접근 제어 정책이 사용되고 있는 것을 가정하고, 그중 대표적인 접근 제어 정책인 강제적 접근 제어(MAC) 정책과 역할기반 접근 제어 정책을 적용하여 관리 정보베이스에 대한 접근 제어를 수행하는 과정을 제시한다.

다음 절에서는 강제적 접근 제어 정책과 역할 기반 접근 제어 정책에 대하여 간단히 살펴본다.

4.1 강제적 접근 제어

관리 정보베이스에서 정보 흐름을 통제하는 강제적 접근 제어 정책은 관리 정보베이스에 저장된 정보의 존재와 내용의 비밀성을 노출시키지 않는 보호 기법을 제공하고 있다. MAC 기법은 흔히 다단계 보안 정책이란 말과도 의미가 일맥상통하며, 객체의 보안등급을 여러 단계로 구분하고 서로다른 인가등급을 갖는 사용자들에 의해서 공유되도록 허용하는 방법이다. 따라서 객체는 그들 자신의 보안등급을 갖으며 사용자는 각기 인가등급과 결합된다. 따라서 다단계 보안 기법이 사용자에 의한 모든 객체의 접근을 조정하게 된다.

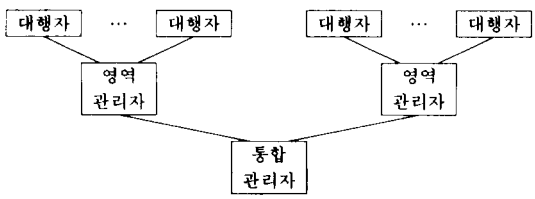
4.2 역할 기반 접근 제어

역할 기반 접근 제어 정책은 사용자들의 요구가 중첩되는 경우에 효과적으로 관리 객체에 대한 접근을 제어하는데 유용한 방법이다. 이 방법은 특정한 환경에 있어서 조직체의 구조에 맞추어 보안의 일관성을 유지하므로써 접근 제어의

복잡성을 감소시킨다.

역할 기반 접근 제어 정책에서 관리자들은 각각의 역할 클래스에 배정이 되는데 이러한 역할은 역할 이름과 그 역할이 접근할 수 있는 객체 및 그 객체에 대한 접근 형태 등의 쌍으로 이루어진 능력 리스트로 구성된다. 즉, 개개의 역할은 그 역할이 접근할 수 있는 객체 및 그 객체에 대한 접근 형태가 능력 리스트의 형태로 정의되어 있어서 실질적으로 그 역할이 수행할 수 있는 관리 범위가 능력 리스트에 의하여 제한이 된다. 각각의 관리자에게는 이렇게 정의된 역할이 적절히 배정이 되며, 이 역할에 할당된 접근 권한에 따라 관리자의 관리 객체에 대한 접근 범위가 결정된다.

역할은 접근 권한에 대한 관리를 단순화해주는 적당한 형태로 조직될 수 있다. 역할들은 함축적인 역할 상호간의 연관성에 따라 링크로 연결되어 계층구조를 형성하는데, 권한이 낮은 상위 역할과 묵시적으로 상위 역할을 상속받는 하위 역할이 링크로 연결된 구조를 갖으며, 이를 역할의 상속 계층구조로 정의할 수 있다. (그림 1)에서 제시한 관리 계층구조에서 역할들의 상속 계층구조를 (그림 2)와 같이 정의할 수 있다.



(그림 2) 역할의 상속 계층구조
(Fig. 2) Role inheritance hierarchy

역할의 상속 계층구조는 3.2절에서 제시한 (그림 1)의 관리 계층구조를 거꾸로 뒤집어 놓은 형태를 취하고 있다. (그림 1)에서 최하위 수준의 관리자인 대행자는 역할 상속 계층 구조상의 최상위 역할에 해당하며, 대행자의 역할은 묵시적으로 영역 관리자에게 모두 상속된다. 마찬가지로 영역 관리자의 역할은 통합 관리자에게 상속되어 결과적으로 통합 관리자는 대행자와 영역 관리자의 역할까지 모두 포함하게 된다.

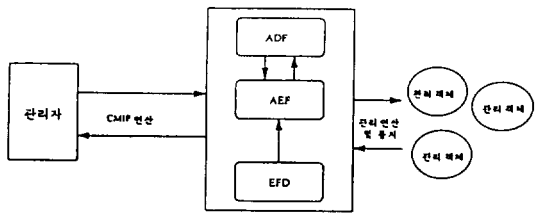
5. 관리 정보베이스의 접근 제어 모형화

ISO 10164-9/ITU-T X.741[6] 표준안에서는 관리 정보나 관리 연산에 대한 접근을 제어하기 위한 모델을 제안하고 있다. 이 표준안에서는 접근 제어 관리 정보에 표현된 접근 제어 정책에 따라 원하는 관리 개체에 대한 접근을 허가하거나 금지시키는데 사용되는 관리 객체와 속성들을 기술하고 있다. 뿐만 아니라, 관리 객체에서 발생한 통지의 결과로 생성되는 사건 보고에도 접근 제어를 적용하므로써 사건 보고가 정확한 관리자에게 전달될 수 있도록 하는 기능을 정의하고 있다.

다음 절에서는 접근 제어 관리 기능 및 필요한 관리 객체에 대하여 간단히 살펴보고, 능동 객체 지향 데이타 모델[15]을 이용하여 모형화한 역할 기반 접근 제어 모델을 제시한다.

5.1 접근 제어 관리 기능 및 관리 객체

접근 제어 관리 기능은 관리 정보에 대한 불법적인 접근을 방지하고, 망 관리 연산 및 통지 서비스들이 적법한 객체에 대하여 수행될 수 있도록 한다. (그림 3)은 접근 제어 관리 모델을 보여주고 있다[6].



(그림 3) 접근 제어 관리 모델
(Fig. 3) Access control management model

(그림 3)에서 CMIP 연산에 의한 관리 객체에 대한 접근 요청은 먼저 접근 제어 강화 기능(AEF : Access Control Enforcement Function)으로 보내어 진다. 이것은 접근 여부를 결정하기 위하여 접근 제어 결정 기능(ADF : Access Control Decision Function)에 보내어 진다. 그러면 접근 제어 결정 기능은 접근 가능 여부를 결정하여 그 결과를 접근 제어 강화 기능으로 돌려보낸다.

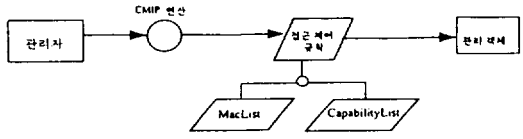
이때 이 결정이 참이 되면 해당 관리 객체에 대하여 접근할 수 있다.

그리고 관리 객체에서 발생한 통지들에 대하여도 접근 제어가 적용된다. 관리 객체에서 발생한 통지들은 EFD를 통과하면 이 사건 보고는 다시 접근 제어 강화 기능에 보내어 진다. 이것은 다시 접근 제어 결정 기능에 보내지고 다른 망 관리 연산과 마찬가지로 처리된다.

5.2 관리 연산의 접근 제어

본 논문에서 채택한 강제적 접근 제어 정책과 역할 기반 접근 제어 정책에 의한 관리 객체에 대한 접근 제어 모델을 (그림 4)와 같이 능동 객체지향 모델을 이용하여 모형화할 수 있다.

(그림 4)에서와 같이 관리자는 일반 클래스로 모형화가 가능하고, 모든 관리 연산은 사건 클래스로, 그리고 관리자의 관리 객체에 대한 접근 권한을 나타내는 능력 리스트는 규칙 클래스로 모형화할 수 있다. 즉, 관리자 클래스가 관리 객체에 접근하려고 할 때, 관리 연산이라고 하는 사건 클래스가 발생하고, 이 사건 클래스가 실제로 관리 객체에 대하여 수행되기 위해서는 관리자와 관리 객체 사이에 접근을 허용하는 규칙 클래스가 존재해야만 접근이 가능하다. (그림 5)는 (그림 4)에 나타난 클래스들의 서술 구조를 보여주고 있다.



(그림 4) 관리 연산의 접근 제어 모델
(Fig. 4) Access control model for management operations

```
CLASS  integration-manager
    Inheritance : manager
    Attributes
       manager_name : manager_id;
       clearance_level : {TS, S, C, U};
       domain_manager_list : {domain_manager_id};
    Method
       create(); delete(); modify();
ENDCLASS
```

```
CLASS  domain_manager
    Inheritance : manager
    Attributes
      manager_name : manager_id;
      domain_name : domain_id;
      clearance_level : {TS, S, C, U};
      agent_list : {agent_id};
    Method
      create(); delete(); modify();
ENDCLASS
```

(a) 관리자 클래스

```
EVENT CLASS  M-GET
    Inheritance : cmis
    Active_class : class_name;
    Attribute :
       invoke_id : object_id;
       base_object_class : class_name;
       base_object_instance : object_id;
       scope : scope_type;
       filter : object_id;
       access_control : access_control_policy;
       attribute_identifier_list : {attribute_id};
    Method :
       create(); delete(); modify();
ENDEVENT_CLASS
```

```
EVENT CLASS  M-SET
    Inheritance : cmis
    Active_class : class_name;
    Attribute' :
       invoke_id : object_id;
       mode : mode_type;
       base_object_class : class_name;
       base_object_instance : object_id;
       scope : scope_type;
       filter : object_id;
       access_control : access_control_policy;
       synchronization : sync_type;
       modification_list : {(attribute, value)};
    Method :
       create(); delete(); modify();
ENDEVENT_CLASS
```

```
EVENT CLASS  M-ACTION
    Inheritance : cmis
    Active_class : class_name;
    Attribute :
       invoke_id : object_id;
       base_object_class : class_name;
       base_object_instance : object_id;
       scope : scope_type;
       filter : object_id;
       access_control : access_control_policy;
       synchronization : sync_type;
       mode : mode_type;
       action_type : action_type;
       action_information : action_inform_type;
    Method :
       create(); delete(); modify();
ENDEVENT_CLASS
```

```
EVENT CLASS  M-DELETE
    Inheritance : cmis
    Active_class : class_name;
    Attribute :
      invoke_id : object_id;
       base_object_class : class_name;
       base_object_instance : object_id;
        scope : scope_type;
       filter : object_id;
        access_control : access_control_policy;
        synchronization : sync_type;
    Method :
      create(); delete(); modify();
ENDEVENT_CLASS
```

```
EVENT CLASS  M-CREATE
       Inheritance : cmis
       Active_class : class_name;
       Attribute :
         managed_object_instance : mo_id;
         superior_object_instance : mo_id;
         reference_object_instance : mo_id;
         attribute_list : {attribute};
         access_control : access_control_policy;
       Method :
          create(); delete(); modify();
 ENDEVENT_CLASS
```

```
EVENT CLASS  M-CANCEL-GET
    Inheritance : cmis
    Active_class : class_name;
    Attribute :
       invoke_id : object_id;
       get_invoke_identifier : get_op_id;
    Method :
      create(); delete(); modify();
ENDEVENT_CLASS
```

(b) 사건 클래스

```
RULE CLASS MacList
    Inheritance : access_control
    Attribute:
         Subject : UesrID;
         Target : OID;
         OpType : cmip_operation;
         Event : Mgr_Op_ID;
         Condition : EvaluationMAC();
         Action : GrantRight();
    Method:
         EvaluationMAC();
         GrantRight();
ENDRULE_CLASS
```

```
RULE CLASS CapabilityList
    Inheritance : access_control
    Attribute:
          CapabilitySet  : ID_List;
          DefaultAccess : ID;
          Event : Mgr_Op_ID;
          Condition : EvaluationCapList();
          Action : GrantRight();
    Method:
          EvaluationCapList();
          GrantRight();
ENDRULE_CLASS
```

(c) 접근 제어 규칙 클래스

(그림 5) 접근제어 관리객체 클래스 구조
(Fig. 5) Object class structures for access control management

(그림 5)의 (a)에는 통합 관리자, 영역 관리자, 그리고 대행자로 구성된 관리 계층구조를 보여주고 있다. 한편, (그림 5)의 (b)는 CMIS/CMIP 연산 중 M-EVENT-REPORT를 제외한 여섯개의 연산들의 클래스 구조를 보여주고 있다. 이들은 능동 객체지향 데이타 모델에서 사건 클래스로 정의할 수 있다. 그리고 (c)에서 event, condition, 그리고 action 속성이 있는데, 바로 이 부분이 능동적인 특성을 나타내는 부분이다. 즉, event 속성은 관리자가 관리 객체에 접근하려고 하는 사건이 발생하였음을 나타내고, condition은 능력 리스트를 탐색하여 해당하는 능력 리스트가 있는지를 검사하며, 마지막으로 action은 condition에서 검사한 결과를 바탕으로 관리자가 취할 행위를 기술하고 있다.

(그림 5)의 (c)에 정의된 Capability list 규칙 클래스와 MacList 클래스에서 관리자의 관리 객체에 대한 접근 허용 여부를 검사하는 접근 권한 검사 메소드를 (그림 6)과 같이 정의할 수 있다.

5.3 사건 보고의 접근 제어

ISO 10164-5/ITU-T X.734 표준안에서는 사건 보고를 통제하기 위한 사건 보고 관리 기능을 제안하고 있다. 사건 보고는 관리 객체의 상태에 변화가 생겼다거나, 또는 한계값을 넘었을 경우에 어떤 사건이 발생했다는 통지의 결과로 생성된다. 사건 보고 관리 기능[5]은 관리자가 관리 객체로부터 발생한 이러한 사건 보고를 전송하는 것을 통제할 수 있는 기능을 제공한다. 특히, 관리 객체로부터 발생하는 사건 보고를 통제하기 위하여 EFD(event forwarding discriminator)라고 하는 관리 객체를 정의하고 있다. EFD는 다른 관리 객체와 관련된 관리 연산을 수행하는 관리 객체로서, 다른 관리 객체들로부터 발생한 잠정적인 사건 보고를 검사하기 위하여 사용된다.

이와 같은 사건 보고가 적절한 관리자에게 통보되어 관리자가 그에 상응하는 관리 연산을 수행하기 위해서는 사건 보고도 다른 CMIS/CMIP

```
MacList :: EvaluationMAC(s, o, op_type) {
   switch(op_type) {
      case GET:
               if( level(s) >= level(o) )
                  return(TRUE);
               else return(ERROR_Msg);
               break;
      case SET:
      case CREATE:
      case DELETE:
              if( level(s) == level(o) )
                 return(TRUE);
               else return(ERROR_Msg);
               break;
      case ACTION:
      case CANCEL_GET:
               if( level(s) != level(o) )
                  return(ERROR_Msg);
               else return(TRUE);
               break;
   }
   return(ERROR_Msg);
}
```

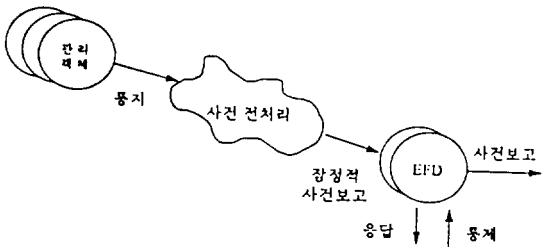
(a) 강제적 접근 제어 정책에서의 접근 권한 검사 메소드

```
CapabilityList :: GrantRight()
{
   return(PERMISSION);
};
CapabilityList :: EvaluationCapList(MgrID, AccessType)
{
   if( "select x from CapabilitySet {
        where MgrID = x.UserID" )
       if( AccessType == x.AccessType )
          return(TRUE);
       else
          return(ERROR_Msg);
   }
   return(ERROR_Msg)
}
```

(b) 역할기반 접근 제어 정책에서의 접근 권한 검사 메소드

(그림 6) 접근 권한 검사 메소드
(Fig. 6) Methods for access authorization checking

연산과 마찬가지로 접근 제어가 수행되어야 한다. (그림 7)은 사건 보고 관리 기능 및 접근 제어 모델을 나타내고 있다[6].

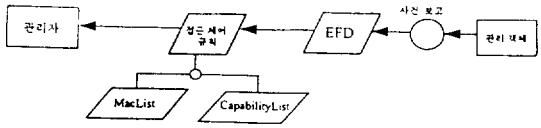


(그림 7) 사건 보고 관리 기능 및 접근 제어 모델
(Fig. 7) Event report management function and access control model

사건-전처리 기능은 관리 객체들로부터 국부적인 통지나 잠정적인 사건 보고를 받는다. 이러한 잠정적인 사건 보고들은 국부적인 개방 시스템 내에 포함된 모든 EFD에 전달되어 EFD 내에 정의된 조건을 만족한 사건 보고만이 접근 제어를 위하여 AEF로 전달된다. AEF로 전달된 사건 보고는 앞장에서 설명한 접근 제어 정책이 적용되어 해당 관리자에게로의 보고 여부가 결정된다.

5.4 사건 보고 접근 제어의 모형화

관리 객체에서 발생한 통지로부터 생성된 사건 보고는 먼저 EFD에게 보내져 조건 검사를 하여 통과한 경우에만 접근 제어를 위하여 AEF로 보내어 진다. 그리고 AEF를 거쳐 접근이 허가된 사건 보고만이 최종적으로 관리자에게 보내져 그 사건 보고를 받은 관리자가 적절한 관리 연산을 수행할 수 있게 된다. 사건 보고에 대한 접근 제어 모델을 능동 객체지향 모델을 이용하여 (그림 8)과 같이 모형화할 수 있다.



(그림 8) 사건 보고 접근 제어 모델
(Fig. 8) Access control model for envent report

관리 객체에서 발생한 통지로부터 국부적인 처리를 거쳐 잠정적인 사건 보고인 EFD 입력 객체를 생성할 수 있는데, 이것을 (그림 8)에서 처럼 사건 클래스로 모형화할 수 있다. 그리고 EFD 자체는 잠정적인 사건 보고를 검사하여 EFD 내

```
EVENT CLASS NAME M-EVENT-REPORT
   Inheritance : cmis
   Active_class : class_name;
   Attribute :
      invoke_identifier :
      mode :
      managed_object_class :
      managed_object_instance :
      event_type :
      event_time :
      event_information :
   Method :
      create(); delete();
ENDEVENT_CLASS
```

(a) EFD 입력 객체 클래스

```
RULE CLASS NAME  event_forwarding_discriminator
   Inheritance : discriminator
   Attribute :
      event : event_id;
      condition : check_discriminator_construct;
      action : event_report;
      destination : object_id;
      discriminator_construct : construct_type;
      administration_state : (lock || unlock);
      operational_state : (enable || disable);
      availability_state : (duty_on || duty_off);
   Method :
      create(); delete();
      check_discriminator_construct();
      state_change();
      attr_value_change(); event_report();
ENDRULE_CLASS
```

(b) EFD 클래스

(그림 9) 사건 보고 관리 객체 클래스 구조
(Fig. 9) Object class structures for event report management

에 정의된 조건들을 만족했을 때만 사건 보고를 AEF로 보내주기 때문에 규칙 객체로 모형화할 수 있다. (그림 8)에 있는 클래스들의 구조는 (그림 9)와 같이 기술할 수 있다.

(그림 9)의 (a)에서 정의된 속성 중에서 active_method는 관리 객체에서 통지를 발생시킨 메소드를 가리키며, object class는 통지를 발생시킨 객체가 속한 객체 클래스를 나타내며, object는 실제 인스턴스를 말한다. 또한 event type은 발생한 사건의 종류를 나타내며, event_time은 사건이 발생한 시간을 가리킨다.

한편, (그림 9)의 (b)에 정의된 event는 규칙 클래스를 호출한 사건을 나타내며, condition은 발생한 사건에 대하여 조건을 검사하는 메소드를 가리킨다. action은 condition이 만족되었을 때 취할 행동에 대한 메소드를 나타낸다. 그리고 destination은 조건을 만족한 사건 보고가 도달할 목적지를 나타내며, 나머지는 EFD의 상태를 나타내기 위하여 사용된다.

(그림 9)의 (b)에 정의된 event forwarding discriminator 규칙 클래스에서 잠정적인 사건 보고를 AEF에게 전달할 것인가의 여부를 검사하는 check discriminator construct 메소드를 (그림 10)과 같이 정의할 수 있다.

```
event_forwarding_discriminator::check_discriminator_construct
{
   IF ( event.parameter_list SATISFY discriminator_construct
        IN event_forwarding_discriminator)
   THEN
        send event_report to AEF;
        log event to event_log;
   ELSE
        discard event_report;
}
```

(그림 10) 사건 보고 검사 메소드
(Fig. 10) Methods for event report checking

위와 같은 메소드에 의한 검사를 통과한 사건 보고는 AEF로 전달되어 다른 관리 연산과 마찬가지로 강제적 접근 제어 정책이나 역할기반 접근 제어 정책의 적용을 받아서 해당 관리자에게 보고된다.

6. 결론 및 추후 연구방향

사회 각 분야에서 컴퓨터의 이용이 일반화되고 이들이 통신 망을 통하여 상호 연결되어감에 따라 전체적인 통신망의 규모가 커지게 되고 이를 관리하는 망 관리 작업이 더욱 복잡해지게 되었다. 또한 망을 이용하는 사용자들의 요구사항도 복잡해져 이들을 적절히 지원하고 망을 효과적으로 관리해 줄 수 있는 망 관리 시스템이 통신 망의 운용에 필수적인 요소가 되었다.

망 관리 시스템의 여러 가지 구성 요소들 중 가장 핵심적인 요소 중의 하나는 망 관리에 필요한 정보들인 관리 객체들의 개념적인 저장소인 관리 정보베이스이다. 관리 정보베이스에 저장된 관리 객체들은 망 관리에 필수적이며 중요한 모든 정보들을 유지하고 있기 때문에 안전하게 유지가 되어야 한다. 따라서, 본 논문에서는 서로 다른 보안 정책을 사용하고 있는 여러개의 하위 망으로 구성된 대규모 통신망에서 관리 정보베이스에 대한 접근을 제어하기 위하여, 대표적인 접근 제어 정책인 강제적 접근 제어 정책과 역할기반 접근 제어 정책을 사용하였다.

또한, 본 논문에서는 관리 계층구조를 통합 관리자, 영역 관리자, 그리고 대행자로 계층화하여 모형화하므로써 하나의 관리자가 크고 복잡한 망을 전체적으로 관리할 때 발생하는 부담을 적절히 분산하여 전체적인 망 관리를 보다 효율적으로 수행할 수 있게 하였다. 뿐만 아니라 망 관리 정보에 대한 접근제어 모델 및 관련된 관리 객체들을 능동 객체지향 데이타 모델을 이용하여 모형화하므로써 객체지향 데이타베이스로의 사상을 용이하게 하였다.

추후 연구 방향은 역할들 사이에 발생할 수 있는 권한의 위임 문제에 대하여 계속 연구해 나아갈 예정이며, 더 나아가서는 망 관리에 필요한 모든 관리 기능들을 능동 객체지향 모델을 이용하여 모형화 할 수 있는 방법에 대하여 연구를 진행하고자 한다.

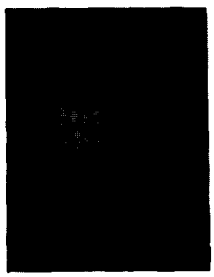
참 고 문 헌

[1] ISO/IEC 7498-4/ITU-T X.700, "Management

Framework".
[2] ISO/IEC 10040/ITU-T X.701, "System Management Overview".
[3] ISO/IEC 10165-1/ITU-T X.720 "Management Information Model".
[4] ISO/IEC 10165-2/ITU-T X.721, "Definition of Management Information".
[5] ISO/IEC 10164-5/ITU-T X.734, "Event Management Function".
[6] ISO/IEC 10164-9/ITU-T X.741, "Objects and Attributes for Access Control".
[7] Allan Leinwand, Karen Fang, Network Management A Practical Perspective, 1993.
[8] F. H. Lochovsky, C. C. Woo, "Role-Based Security in Database Management Systems," Database Security : Status and Prospects, North-Holland, 1988.
[9] F. Rabitti, et al., "A Model of Authorization for Next Generation Database Systems", ACM Trans. on Database Systems, Vol. 16, No. 1, March 1991.
[10] K. Klemba, M. Kosarchyn, "A Model for Object Relationship Management," IFIP Integrated Network Management II, pp801-812, 1991.
[11] Nader Soukouti, "Automatic Translation of OSI Managed Object Class to C++ Classes," IEEE Journal on Selected Areas in Communication Vol. 12 No. 6, pp1011-1019, 1994.
[12] Olivier Festor, Georg Z rmtlein, "Formal Descriptionof Managed Object Behavior-A Rule Based Approach," IFIP Integrated Network Management III, pp45-58, 1993.
[13] Robert H. Stratman, "Development of an Integrated Network Manager for Heterogenous Networks Using OSI Standards and Object-Oriented Techniques," IEEE Journal on Selected Areas in Communications, Vol. 12, No. 6, pp1110-1120, 1994.
[14] Thierry Desparts, Fabienne Faure, Michelle Sibilla, "Interaction Modelling for Structuring Cooperative Management Architecture," DSOM '94, 1994.
[15] 김영균, 다단계 보안 성질을 유지하는 능동 객체지향 데이타베이스의 개념모형화, 전남대학교 박사학위 논문, 1995.

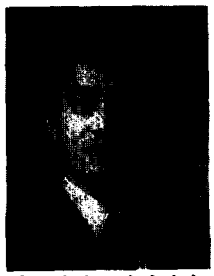
서 재 현

1985년 전남대학교 계산통계학과 이학사
1988년 중앙대학교 대학원 전자계산학과 이학석사
1993년~현재 전남대학교 대학원 전산통계학과 박사과정
1988년~현재 송원전문대학 전자계산학과 전임강사
관심분야 : 통신망 관리, 객체지향 시스템, 분산처리 시스템, 정보 보안 등.



이 창 진

1994년 전남대학교 전자계산학과 이학사
1994년~현재 전남대학교 대학원 전산통계학과 석사과정
관심분야 : 통신망 관리, 정보 보안 등.



노 봉 남

1978년 전남대학교 수학교육과 이학사
1982년 한국과학기술원 전산학과 공학석사
1994년 전북대학교 대학원 전산통계학과 이학박사
1983년~현재 전남대학교 전산학과 교수
관심분야 : 객체지향 시스템, 통신망 관리, 정보 보안, 컴퓨터와 정보사회 등.