

Relationship을 이용한 OSI 망관리 Naming Method

正會員 宋旺喲*, 康昌彦*

A Naming Method for OSI Network Management Using Relationship

Wang Cheol Song*, Chang Eon Kang* Regular Members

要 約

전역적으로 분포되어 있는 망들과 서비스들을 관리할 때, 대리자의 위치를 투명하게 알고 관리 대상 객체(Managed Object : MO)에 전역적으로 유일한 이름을 붙이는 것이 필요하다. 그리고 모든 객체는 관리자에 의해 식별되어야 한다. 이 논문에서 우리는 OSI 관리가 X.500 디렉토리 서비스를 명명 서비스(naming service)로 사용되어야 한다고 생각하며, 그것이 어떻게 사용되는지를 관련성(relationship)의 개념에 의해 설명한다. 포함 관계(name-binding)보다는 관련성의 개념을 사용하는 것을 전역적인 명명 방법으로 제안하였다.

ABSTRACT

In the management of globally distributed networks and services, it is needed to locate the agent transparently and identify managed object(MO) with globally unique name. All objects must be identified by the manager. This paper argues that OSI management should use the X.500 Directory service as a naming service, and describe how it is used, based on the concept of relationship. A solution for global naming has been suggested which involves the utilization of the relationship concept rather than containment.

*연세대학교 전자공학과
Dept. Electric Eng. Yonsei University
論文番號 : 95099-0311
接受日字 : 1995年 3月 11日

1. 서 론

최근에 전역적으로 분포되어 있는 망들과 서비스들의 관리는 망 통신 시스템에 있어서 중요한 요소가 되었다. 회사들뿐만 아니라 여러 표준화 기구들이나 연구 계획 프로젝트들은 (ISO, CCITT, ETSI, RACE, ESPRIT, BERKOM 등) 분산된 시스템들의 관리 방법을 찾기 위해 많은 노력을 기울이고 있다. 분산 투명성은 분산 처리 시스템에서 중요한 개념들 중의 하나이고, 응용 프로그래머, 사용자, 시스템 관리자들로부터의 분산 체계를 인식하지 않아도 되도록 하는 것을 목적으로 한다. 투명한 정보 모델은 전역적인 식별, 위치 확인, 주소 부여들의 문제들을 해결해야만 하며, 그것들 중 위치 투명성에 대해서 [1]에서는 토폴로지의 세부 사항을 감추는 것으로 [2]에서는 클라이언트(client)가 상호작용 하는 서버(server)들의 위치를 감추는 것으로 정의하였다. 일반적인 관리 플랫폼(platform)에서 위치 투명성은 관리 응용에 어떤 대리자가 그 대리자의 위치뿐만 아니라 특정 자원을 나타내는 관리 대상 객체(Managed Object : MO)를 포함하고 있는가를 알려준다. 그러므로 투명성은 대리자에게 그것이 어떤 사건을 알리고자 하는 관리 응용의 위치를 식별하는 방법을 제공해 주어야 한다. 이러한 분산 망에서의 투명성은 먼저 각 MO를 전역적으로 유일하게 식별가능해야 한다는 전제를 기초로 하고 있으므로 분산 망에 있는 각 MO에 대해서는 전역적 명명 방식이 필요하게 된다.

본 논문에서는 OSI 망 관리 표준에 기초하여 MO에 전역적 이름을 부여하므로써 망 관리자가 원격 대리자가 관리 하고 있는 어떤 MIB(Management Information Base)에 있는 MO를 전역적으로 유일하게 식별할 수 있도록 하고자 한다. 분산 망에서 망 관리를 수행하기 위해서는 각 국부 망을 관리하는 여러 관리자가 존재할 것이고 또한 여러 망 관리 대리자 및 MIB들이 망에 분산되어 위치해 있을 것이다. 이러한 환경에서 어떤 관리자가 특정 대리자가 관리하는 MIB 내의 MO에 대해 관리 동작을 수행하려 한다면 그 MO를 전역적으로 유일하게 식별하므로써 자신이 관리하는 MIB 내의 MO와 같은 국부적 이름을 갖는 MO들과 구별할 수 있어야 한다. 이러한 관점에서 명명 체계와 디렉토리 서비스는 분산 시스템 관리에 밀접하게 연결되어 있다. 디렉토리 서비스는 많은 종류의 관리 정보의 적절

한 저장소이며 명명 체계(naming scheme)를 지원하기 위한 기본 요소로서 이 X.500 디렉토리 서비스는 이름을 주소로 변화시키는데 사용될 수 있다.

관리 시스템과 관리되는 시스템이 상호 운용하기 위해서는 그들 간에 X.500 디렉토리 서비스가 명명 서비스를 지원하는 서비스로서 사용할 것인지에 대한 합의가 이루어져야 한다. 이럴 경우에는 그것이 어떻게 사용되며 디렉토리에 어떤 정보가 저장되어 있는지 정확히 합의되어야 한다. 그러나 일반적으로 OSI 표준들 특히 OSI 시스템 관리 표준들은 국부적 명명 방식을 규정하고 있으며 각 관리자는 다른 영역에 있는 대리자에 대해 관리 동작을 수행하고자 할 때에 대해서는 이것이 어떻게 이루어져야 할 지에 대해 제시하지 않고 있다.

그러므로, 이 논문에서는 X.500 디렉토리 서비스가 OSI 관리를 위한 명명 서비스를 지원한다는 가정하에, 각 MO가 관련성(relationship)의 개념에 의해 어떻게 전역적으로 명명되어 유일하게 식별될 수 있는가를 설명한다. 이미 MO의 명명 방식에 대해, 몇몇 책 [3][4][5]에서 X.500 디렉토리 서비스와 OSI 시스템들의 응용의 가능성에 대해 다루었다. 그리고 어떤 논문 [6][7][8]들에서는 관리를 위한 정적(static)이거나 준동적(semi-dynamic)데이터를 처리할 수 있는 X.500 디렉토리 서비스를 이용하는 체계를 제안하여 MO에 대한 정보를 디렉토리 서비스를 이용하여 조사가능하도록 하였다. 한편 [9][10]들은 MIB로의 정보 검색을 위해 DIT로부터 MIB의 관리대상 시스템을 가리키기 위해 NAME BINDING 템플릿을 사용하는 전역적 명명 방식에 대해 제안하였다.

다음 절에서는 DN(Distinguished Name)과 X.500 디렉토리 서비스를 살펴보고 3절에서는 본 논문에서 사용된 관련성의 개념과 이에 관련된 표준들을 설명하였다. 그리고 4절에서는 관련성을 이용한 X.500 디렉토리 정보(Directory Information Base: DIB)와 관리 정보(Management Information Base: MIB)를 결합하는 구조가 제안되었다. 5절에서는 구현 시에 고려할 점에 대해서 언급하였고, 6절에서는 다른 명명 방식들과의 비교를 하였다.

2. DN과 X.500

DN의 개념은 X.500 디렉토리 표준[11]의 구조와

정보 프레임(frame)과 같이 병행하여 개발되어 왔다. 우선 이런 종류의 이름들의 문법(syntax)과 특징들을 살펴보고 X.500 디렉토리 서비스를 지원하는 것에 대해 논의한다.

2.1 DN

DN은 상대적인 DN(Relative Distinguished Name: RDN)의 순차적인 배열로 구성되어 있다. 여기서 RDN은 하나 또는 그 이상의 속성 값 표현방식(Attribute-Value Assertion:AVA)의 비순차적인 집합이다. 각각은 속성의 종류와 개요화된 속성 값으로 이루어져 있다. 속성의 종류들은 속성 값들이 취하는 특정한 문법 체계를 가리킨다. 이름 요소에 속성의 종류들을 집어넣는 것은 다른 명명 체계에서 볼 수 없는 상당한 유연성을 주게된다.

전 세계에 있는 DN들은 전역적인 명명 트리(naming tree)에서 계층적으로 구성되어 있다. 트리는 X.500 디렉토리 서비스에 의해 지원된다. 각각의 RDN은 특정한 명명 권한자(authority)에 의해 할당되어진다. 명명 권한자는 계층적으로 낮은 다른 권한자에 자신의 명명 책임을 위임시킬 수 있다. 어느 한 단계에 있는 권한자는 그것이 할당하는 RDN들이 모호하지 않도록 해야 한다. 계층적인 명명 체계는 전역 망을 통해 이름들의 유일성을 보장함과 동시에 그 다양함에 제한을 두지 않는다. 그 예로 계층적으로 관련된 객체들을 가정하여 표 1에 나타내었다.

표 1. DN과 RDN의 예
Table 1. An example of DN and RDN

| RDN | DN | Object Class |
|--------------------|--|--------------------|
| CountryName=KR | CountryName=KR | Country |
| OrgName=YonSei | CountryName=KR OrgName=YonSei | Organization |
| OrgUnitName=ITRI | CountryName=KR OrgName=YonSei OrgUnitName=ITRI | Organization Unit |
| CommonName=LeeHyun | CountryName=KR OrgName=YonSei OrgUnitName=ITRI CommonName=LeeHyun | Residential person |

디렉토리 엔트리 중에서 alias 엔트리는 디렉토리 정보 트리(Directory Information Tree:DIT)내의 어떤 다른 노드를 가리키는 포인터를 갖고 있다. 이러한 alias 엔트리는 자체적으로 실체를 가지고 있지는 않고 다른 엔트리를 가리키기만 하지만 다른 엔트리와 같이 DN을 갖고 있다. 그러므로, alias 엔트리가 가리키는 어떤 엔트리의 하부에 있는 엔트리들은 복수의 DN을 갖게 된다.

2.2. 관리 대상 객체의 DN

OSI 관리 대상 객체들은 자신의 시스템을 근간으로 하는 국부적인 DN을 통해 명명되어 진다. CMIP 프로토콜이 다른 이름 형식(예를 들어 옥테트 스트링에 근간을 둔 이름)을 지원할 수는 있으나 그러한 이름을 사용하는 것은 권장되지 않고 있으며 SMI(structure of management information)은 다른 형식의 이름을 정의하는 수단을 제공하고 있지 않다.

MO의 구조는 MIT(management information tree)라고 하며 그 트리 내에서 하위(subordinate) 객체는 상위(superior) 객체에 포함되는 관계를 가진다. 이 MIT의 구조는 이름 묶음(name binding)에 의해 정의된다. GDMO(12)에서는 NAME BINDING 템플릿을 이용해서 MO의 인스턴스(instance)의 이름에 대한 형식을 정의해 놓고 있다. 각각의 객체는 이름을 갖고 있지만 이름 묶음들은 MO 클래스의 형태로 정의된다. 이름 묶음들은 MO 클래스의 이름을 정의한 것이 아니라 한 클래스의 인스턴스에 대한 이름을 정의해 놓은 것이다.

이름 묶음은 다음의 3가지 사항과 관련되어 있다.

1. 하위 관리 대상 객체 클래스
2. 상위 객체 클래스 - 이는 관리 대상 객체 클래스일 수도 있고 디렉토리 객체 클래스일 수도 있다.
3. 하위 관리 대상 객체 클래스의 고유(distinguished) 속성에 대한 속성의 형태

그림 1은 일반적인 명명 구조를 보이기 위한 예로서 OSI 트랜스포트 계층에 대한 명명 구조를 나타낸 것이다(13). 이름 묶음 템플릿의 이러한 구조는 관리 정보 트리(DIT)에서 인스턴스화된 형태로 나타나게 된다. 그림에서 사각형은 클래스를 표시하고 모서리가 둥근 사각

형은 인스턴스를 나타낸다.

2.3. X.500 디렉토리 서비스

X.500 표준[11][14]들은 개체들에 대한 정보의 제공과 관리를 위해 디렉토리 서비스를 규정하고 있다. X.500 디렉토리 서비스는 정보 저장과 사용자로부터의 재생 요구를 다루는 정보 저장소와 서비스 대리자를 제공한다. 디렉토리에 의해 유지되는 정보는 DIB(Directory Information Base)라고 불린다. DIB는 개체들에 대한 정보와 함께 엔트리(혹은 객체)들을 포함하고 있다. 각 엔트리는 일련의 속성의 집합으로 이루어지며, 각각은 하나 또는 그 이상의 값으로 구성된다. 각 엔트리에 있는 여러 종류의 속성들은 엔트리에 의해 표현되는 개체의 클래스에 종속적이다. DIB의 개체들은 DIT라고 하는 전역적이고 계층적인 이름 체계에 있는 엔트리에 의해 표현된다. 엔트리들은 그것들이 나타내는 개체들간의 조직화된 관계에 따라 DIT에 놓이게 된다. X.500에 의해 정의되는 전역적인 명명체계는 각 엔트리의 RDN을 연결시키는 것이다. DIT를 포함한 디렉토리 서비스는 디렉토리 시스템 대리자(Directory System Agent:DSA)라고 불리는 물리적으로 떨어져 있는 개체들에 분산되어 있다. 이 분포는 DSA간의 디렉토리 시스템 프로토콜(DSP)을 이용하므로써 사용자에게 투명하다. 각 사용자나 사용자 프로세스는 디렉토리 사용자 대리자(DUA)를 통해 디렉토리와 상호 작용하게 된다. 그래서 사용자들은 국부적이거나 인접한 DSA들을 통해 쉽고 효과적으로 디렉토리 정보에 접근

할 수 있다. 디렉토리 서비스의 기능적 모델을 그림 2에 나타내었다.

X.500 표준안을 또한 디렉토리 정보를 복구, 탐색, 수정하기 위한 사용자 기능들을 제공하는 추상적인 포트(port)와 동작을 정의하고 있다. 이들 동작들은 디렉토리 접근 프로토콜(DAP)이라는 간단한 인터페이스 프로토콜을 통해 이뤄진다[14].

제안된 디렉토리 체계가 그림 3에 나타나 있는데, 이것은 CCITT X.521 표준안(1988년 판)의 224페이지에서 취해진 것이다.

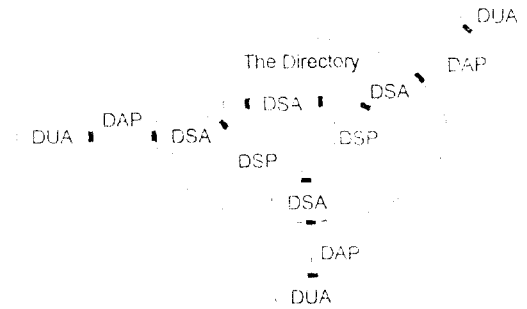


그림 2 디렉토리 서비스의 기능적 모델
Figure 2. Functional Model of the Directory Service

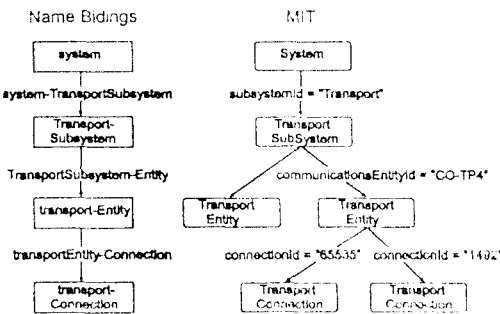


그림 1. MO 클래스 사이의 이름 묶음과 MIT
Figure 1. Name Bindings between MO classes and the Management Information Tree(MIT)

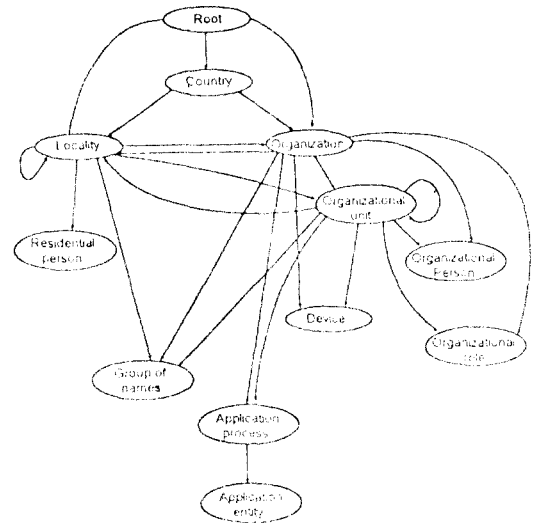


그림 3. X.521에 제안된 DIT 구조
Figure 3 Suggested DIT structure from X.521

3. 관련성의 개념과 관련 표준안

하나의 관리대상 객체 클래스의 기능이 다른 관리되는 객체 클래스에 의존하거나 영향을 미칠 때 관리대상 객체 클래스들 간에 관련성이 존재한다고 한다. 이러한 영향을 주는 성질은 관련성을 규정함으로써 묘사될 수 있다. 그러한 관계에 대한 지식은 여러 종류의 관련성들이 존재하는 말을 묘사하는 정보 모델의 경우에 있어서 특히 중요하다.

현재의 표준안들에서 관련성은 그 관련성을 나타내는 속성을 사용하여 모델링 된다. 각각의 객체 클래스는, 그 클래스들이 참여하고 있는 관련성들을 임의의 방식으로 나타내는 속성을 가지고 있다고 가정한다[15]. 이 속성들은 참여하는 관리대상 객체 클래스들의 어느 하나에만 존재할 수 있거나(one-way relationship), 또는 양쪽에 존재할 수 있다(two-way relationship). 간단한 경우에서, 어떤 관리대상 객체 인스턴스에 관련성을 나타내는 속성을 구현하는 것은 다른 MO 인스턴스에 향하여지는 객체 포인터를 구현하는 것과 같다.

이후의 표준에서 일반적인 관련성 모델(General Relationship Model: GRM)[16][17]은 더욱 일반적이고 유연성 있는 관련성 모델링 과정을 제공한다. 그것은 객체들간에 일반적인 n차 관련성을 모델링 할 수 있게 하며, n차 관계는 n개의 다른 역할을 가질 수 있고 각각의 참가하는 객체는 그 역할들 중의 최소한 하나는 수행해야 한다. 관련성들은 RELATIONSHIP CLASS 템플릿(template)을 이용하여 정의된다. GRM은 관련성에 연관되어 있음을 나타내기 위해 참가하는 관리대상 객체에 속성으로서 상호 포인터를 반드시 가져야 할 필요가 없다. 즉, 그것은 관련성 객체들을 사용하거나 다른 구현 방법을 이용해 관련성을 구현하는 다른 방법을 가능하도록 한다. 비슷한 모든 관련성들의 집합은 관련성 클래스로 정의된다. 하나의 관련성 클래스는 관련성 객체들을 발생시키고, 각각은 그것의 관련된 관리대상 객체들을 포함하고 있다. 관련성 클래스는 다음과 같은 RELATIONSHIP CLASS 템플릿를 사용하여 정의된다.

```
<relationship-class-label> RELATIONSHIP CLASS
[DERIVED FROM <relationship-class-label>
[, <relationship-class-label>]*];
```

```
[BEHAVIOR <behaviour-definition-label>
[, <behaviour-definition-label>]*];
[ROLE <role-label> roleproperties
[REGISTERED AS object identifier]*]
REGISTERED AS object identifier;
```

각각의 관련성 클래스는 이 템플릿를 사용해 등록된다. 필요한 많은 역할들이 이 관련성 클래스를 위해 정의될 수 있고, 각각은 또한 그 자신의 OBJECT IDENTIFIER를 사용하여 등록된다. 각 역할의 성질은 다음과 같이 지원되는 "roleproperties" 부분에 의해 규정된다.

```
roleproperties ->
[CHARACTERIZED BY <package-label>
[, <package-label>]*];
[CONDITIONAL PACKAGE <package-label>
PRESENT IF condition-definition
[, <package-label>
PRESENT IF condition-definition]*]
[PERMITTED ROLE CARDINALITY SubtypeSpec]
[REQUIRED ROLE CARDINALITY SubtypeSpec]
[DYNAMIC ENTRY [CREATE [<parameter-label>]*]]
[DYNAMIC DEPARTURE
[DELETE [<parameter-label>]*]
[DELETE-ALL-IN-ROLES <role-label> [, <role-label>]*]
[RELEASE-ALL-IN-ROLES <role-label> [, <role-label>]*]
[ONLY-IF-NONE-IN-ROLES <role-label> [, <role-label>]*]]
```

한 역할은 패키지(package)를 가지고 있을 수 있다. 그리고 이것은 일상적인 PACKAGE 템플릿으로서 표현된다. 그래서 한 역할은 그 자신의 속성과 동작 그리고 통보(notification)에 의해 특정 지워진다. 각각의 역할 객체의 개수(cardinality)는 SubtypeSpec으로서 표시된다. 이것은 음이 아닌 정수의 집합인 INTEGER(0 .. MAX)의 부분 집합으로 나타내는 ASN.1 형태의 참조 이름이다.

상속 계층(inheritance hierarchy)에서 관련성 클래스들은 관리대상 객체 클래스들처럼 정렬된 클래스이다. 이것이 비록 개념적으로 완전히 다른, 순수한 관련성 계층만을 포함하는 일반화된 계층 구조이지만 이는

관리대상 객체 클래스 계층 구조와 묶여 있다. 그래서 관련성 클래스들은 또한 관리대상 객체 클래스이다. 이 계층 구조의 root는 relationshipTOP이며, 차례로 이것은 관리대상 객체 클래스들을 위한 정규적인 top으로부터 DERIVED FROM이라는 관계를 갖는다. 관련성에 참여하는 관리대상 객체 클래스들 간의 설정은 RELATIONSHIP CLASS에 의해 이루어지는 것이 아니라 RELATIONSHIP BINDING 템플릿에 의해 독립적으로 이루어진다. 하나의 관련성에 참여하는 관리대상 객체 클래스의 특정한 그룹은 관련성 묶음에 의해 함께 묶여진다. 하나의 관련성 클래스는 하나 이상의 관련성 묶음을 만들어 낼 수 있다. 이것은 NAME BINDING과 유사한데, 이 NAME BINDING은 포함 관계물에 있는 관리대상 객체 클래스들의 다른 쌍들을 함께 묶는데 쓰이고 있다. RELATIONSHIP BINDING 템플릿의 형식은 다음과 같다.

```
<relationship-binding-label> RELATIONSHIP BINDING
  RELATIONSHIP CLASS <relationship-class-label>
  [BEHAVIOUR <behaviour-definition-label>
  [, <behaviour-definition-label>]*;
  [RELATIONSHIP OBJECT <class-label> [AND SUBCLASSES];]
  ROLE <role-label> role-specifier;
  [ROLE <role-label> role-specifier]*
REGISTERED AS object-identifier;

role-specifier ->
  RELATED CLASS <class-label> class-specifier
  [RELATED CLASS <class-label> class-specifier]*
  roleproperties

class-specifier ->
  [AND SUBCLASSES]
  [RESTRICTED ATTRIBUTES
  <attribute-label> [value-constraint]
  [, <attribute-label> [value-constraint]]*]
  [RESTRICTED ACTIONS
  <action-label> [argument-constraint]
  [, <action-label> [argument-constraint]]*]
```

RELATIONSHIP BINDING 템플릿은 RELATIONSHIP CLASS

항에서 자신이 어느 관련성 클래스에 속하는 지를 나타낸다. 만일 RELATIONSHIP BINDING에 있는 모든 관련성들이, 참가하고 있는 관리대상 객체 클래스들에서의 상호 포인터 속성으로 구현된다면 RELATIONSHIP OBJECT 항은 필요하지 않다. 그러나 RELATIONSHIP BINDING에서의 어떤 관련성들이 개별적인 관련성 객체들로서 실현된다면, RELATIONSHIP OBJECT 항은 관리대상 객체 클래스를 가리키고, 그 관리대상 객체 클래스의 인스턴스는 이 관련성 묶음의 개별적인 객체 클래스를 가리키게 되는데, 그 관리대상 객체 클래스는 relationshipTOP이나 그 하위 클래스들 중의 하나가 되어야 한다. 그것은 주어진 관련성 묶음으로부터의 관련성들이 관련성 묶음 자체의 관련성 클래스보다는 관련성 클래스의 인스턴스로서 구현되어질 수 있도록 한다. 각각의 역할에 있어서 ROLE 항은 관련성에서 그 역할을 가지는 관리대상 객체 클래스를 표시한다. 그 역할에 참여하는 각각의 관리대상 객체 클래스는 RELATED CLASS 항을 사용해 규정된다. 만약에 [AND SUBCLASSES]가 참여한 클래스를 위해 존재한다면 그것의 subclass도 또한 참가할 수 있다. 각 참여 속성 값의 변화는 RESTRICTED ATTRIBUTED항에 의해 나타내어진다. 관련성에 참여하고 있지만 제한되어 있거나 금지되어 있는 동작(action)들은 RESTRICTED ACTION 항에 규정할 수 있다.

관련성 클래스가 관리대상 객체 클래스인 것처럼 역할도 또한 관리대상 객체 클래스에 의해 구현될 수 있다. 이것은 roleObject 관리대상 객체 클래스라고 하고, 그것의 subclass는 관련성 클래스들의 개별적인 역할이다. 역할 객체는 포함 관계에 의해 관련성 클래스에 포함된다.

4. 관련성을 이용한 명명

기존의 망 관리 관련 표준안에서는 MIB내의 MO들에 대한 명명 방식으로서 NAME BINDING 템플릿을 이용하고 있으며 이러한 명명 방식은 국부 시스템 내의 MO에 한정시켜 이름의 시작을 system이라는 MO로부터 하고 있다.

본산망에서 망 관리자가 다른 영역에 있는 MO에 대해 관리 동작을 수행할 필요가 있을 때 관리자는 그

MO를 관리하는 대리자에게 객체에 대한 이름을 전역적으로 유일하게 식별 가능하도록 전달해야 할 것이다. 이러한 필요성에 따라 이미 몇몇 논문[9][10]들은 DIT로부터 MIB의 관리대상 시스템을 가리키기 위해 NAME BINDING 템플릿을 사용하는 체계를 제안하고 있다. 그러나 이들은 기존의 MO 명명 방식에 쓰이는 NAME BINDING 템플릿의 확장을 필요로 하고 있으며, X.500 디렉토리 서비스의 "alias" 서비스를 해결하지 못하므로써 디렉토리 서비스에서는 가능한 복수의 이름 체계가 MO의 이름에는 적용이 불가능하게 되어 있다. 이는, NAME BINDING 템플릿에 의해 이러한 서비스들을 지원하기 위해서는 복수의 이름 묶음(name binding) 관계가 관리대상 시스템과 관리 담당 시스템인 대리자 시스템 사이에 존재해야 하기 때문인데, 복수의 이름 묶음 관계는 논리적으로 포함 개념에 상반되는 것이다. 왜냐하면 포함관계는 한 객체가 포함관계에 있는 상위의 다른 객체의 내부 시스템인 관계를 의미하기 때문이다.

반면 하나의 대리자는 하나의 MIB를 관리하기는 하지만 MIB는 대리자와 다른 시스템에 위치할 수도 있고 여러 다른 대리자가 특정 MIB의 부분을 공유할 수도 있다. 그러므로 n개의 관계를 설정하는 것이 가능하기 위해서는 관리 대상 시스템과 관리자간에 포함 관계보다는 다른 관계 설정이 고려되어야만 전역적으로 유일하게 식별 가능한 복수의 이름 체계가 가능할 것이다.

먼저 생각해야 할 것은 관리 시스템과 관리대상 시스템간의 관계는 포함 관계이기보다는 각각 supervise하고 supervised-by되는 관계라는 것이다. 이런 종류의 관계가 고려된다면 DIB의 대리자 엔트리와 MIB의 관리대상 시스템간의 n차 관련성을 설정하는 것이 가능하게 된다.

그림 3에 보이는 바와 같이 망 관리 대리자에 해당하는 DIT 구조에 있는 응용 프로세스의 전역적 이름에 기존의 MO에 대한 국부적 이름을 묶어 줌으로써 기존 디렉토리 서비스의 전역적 명명 방식을 이용하여 MO에 전역적 이름을 부여하고자 하는 것이다. 이 때 DIT상의 망 관리 응용 프로세스 엔트리와 MIT 상의 system MO 클래스를 묶는 방식으로 n차의 관련성 개념을 이용하므로써 복수의 전역적으로 식별 가능한 MO의 이름을 제공할 수 있다.

4. 1. 관련성 템플릿의 이용

이 절에서는 RELATIONSHIP 템플릿을 이용한 명명 체계에 대해 설명한다. 우선 전 절에서 설명한 템플릿을 이용한 관련성 클래스가 다음과 같이 정의된다.

```
supervision RELATIONSHIP CLASS
    DERIVED FROM relationshipTop
    ROLE supervise
    PERMITTED ROLE CARDINALITY [1..N]
    DYNAMIC DEPARTURE
        ONLY-IF-NONE-IN-ROLES supervise
    REGISTERED AS {supervise-role};
    ROLE supervised-by
    PERMITTED ROLE CARDINALITY [1..1]
    DYNAMIC DEPARTURE
        ONLY-IF-NONE-IN-ROLES supervised-by
    REGISTERED AS {supervised-by-role};
    REGISTERED AS {supervise-supervised-by}
```

여기서 N은 복수의 관련성을 나타내기 위한 임의의 상수이다. 이러한 관련성을 사용하여 관련성 묶음 클래스는 다음과 같이 정의된다.

```
dib-mib RELATIONSHIP BINDING
    RELATIONSHIP CLASS supervision
    ROLE supervise
    RELATED CLASS smap
    CHARACTERIZED BY
        dibContainmentPackage PACKAGE
    ATTRIBUTE
        cn;
    ROLE supervised-by
    RELATED CLASS system
    CHARACTERIZED BY
        mibContainmentPackage PACKAGE
    ATTRIBUTE
        systemId;
    REGISTERED AS {dib-mib-binding}
```

이것은 NAME BINDING과 비슷하다. NAME BINDING은 포함 관계에 있는 각각의 관리대상 객체

클래스 쌍들을 연결한다. RELATIONSHIP 템플릿에서의 각 역할들로부터 "roleproperties"를 확장하는 방식으로 RELIATIONSHIP BINDING 템플릿에 관련성을 맺은 두 클래스의 신원 속성을 "cn"과 "systemId"로 설정 하였다. 이렇게 설정된 각각의 속성 종류는 DIB와 MIB 각각에서 그 자신의 OBJECT IDENTIFIER를 통해 속성으로 정의될 것이다. 이러한 ATTRIBUTE에 의해 이러한 관리대상 클래스들은 NAME BINDING 템플릿을 이용해서 결정된 것과 동일한 계층 구조를 가질 수 있으며 동시에 명명 체계에 있어서 복수의 관련성 설정이 가능해진다. 그러므로써 X.500 서비스의 alias에 의해 서비스되는 관리대상 객체는 복수의 DN을 가질 수 있으며, 이들은 또한 모호하지 않은 전역적인 이름이 될 것이다.

4.2. 전역 명명 방식의 예

그림 4에는 RELATIONSHIP BINDING 템플릿을 이용하여 MIB와 DIB를 연결하므로써 MIB내의 한 MO에 유일한 DN을 부여하는 구조를 보이고 있다. 이 그림에서 볼수 있듯이, "cn=SMA_catseye"라는 RDN을 갖는 망 관리 시스템의 대리자는 다음과 같은 DN을 갖는다.

```
c=KR@o=Yonsei University@ou=ITRI@cn=SMA_catseye
```

관리자가 관리하고자 하는 시스템(bluesky) 내에 있는 로그 MO(OSISEC_Log)는 다음과 같은 이름(DN)을 갖고 있을 수 있다.

```
c=KR@o=Yonsei University@ou=ITRI@cn=SMA_catseye
@systemId=bluesky@logId=OSISEC_Log
```

이 OSISEC_Log MO에 관리 동작을 수행하고자 하는 관리자는 먼저 이 MO의 DN을 디렉토리에 전달한다. 디렉토리 서비스는 이 DN에서 다음과 같은 대리자의 이름을 구분해낸 후 그에 대한 주소를 찾아서 관리자에게로 서비스 하게 된다.

```
c=KR@o=Yonsei University@ou=ITRI@cn=SMA_catseye
```

관리자는 디렉토리 서비스로부터 반환된 대리자의 주

소를 이용해 대리자와 CMIP 프로토콜을 통한 연결(association)을 이루고 나서 다음과 같은 DN을 갖는 MO에 관리자가 원하는 관리 동작을 취하게 된다.

```
c=KR@o=Yonsei University@ou=ITRI@cn=SMA_catseye
@systemId=bluesky@logId=OSISEC_Log
```

또한 디렉토리 서비스에서 제공되는 alias 서비스를 해결하기 위해서, 앞에서 제시한 dib-mib RELATIONSHIP BINDING을 인스턴스화 하는 과정에서 그림에서 보는 바와 같이 logId=OSISEC_Log라는 RDN을 갖는 MO에 다음과 같은 또 하나의 전역적 이름을 부여할 수 있다.

```
c=KR@o=Yonsei University@ou=EE@cn=SMA_dragon
@systemId=bluesky@logId=OSISEC_Log
```

이 이름 역시 앞에서 보인 것과 동일하게 디렉토리 서비스는 cn=SMA_dragon이라는 DN을 갖는 대리자에 대한 주소를 관리자에게 넘겨주고 관리자는 MO의 전역적 이름에 대한 변화 없이 대리자와 접속(association)을 맺어 관리 동작을 할 수 있게 된다.

이러한 동작은 그림 5에서 보는 바와 같은 구조를 기반으로 이뤄질 것이다. 여기서 대리자는 자기의 이름과 관련 정보, 특히 자기의 주소에 대한 정보를 디렉토리 서비스로 개정통보 해주게 된다.

5. 전역적 명명의 구현 방식 고찰

분산된 망 관리 시스템에 있어서 관리자가 망에 대한 관리 운용을 하고자 할 때 관리자는 그 망 요소의 위치에 관계없이 운용관리를 수행할 수 있어야 한다. 이러한 위치 투명성은 X.500 권고안에서 제시된 디렉토리 서비스를 이용함으로써 해결될 수 있을 것이다. 망 관리자는 먼저 디렉토리 서비스를 이용하여 수행하고자 하는 MO의 이름만으로도 그 MO에 대한 망 관리 대리자의 주소를 서비스 받고 이를 이용하여 그 대리자에게 접속하므로써 목적하는 MO에 대한 관리 운용을 할 수 있을 것이다.

5.1. 관련성의 구현

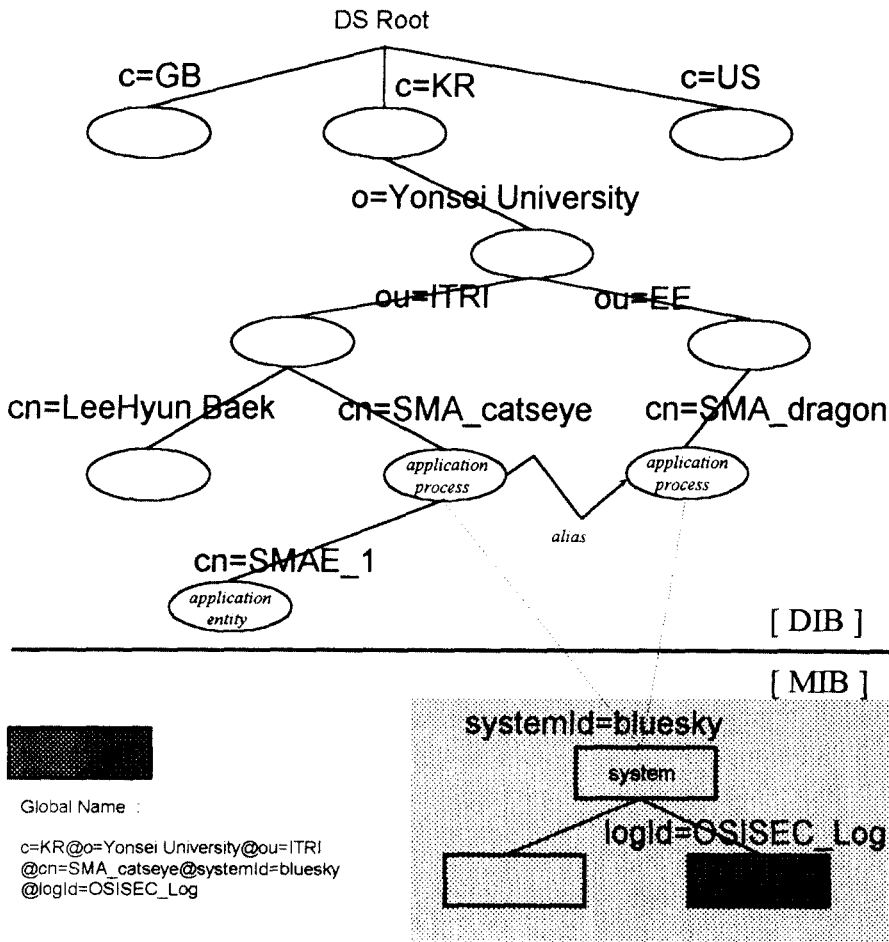


그림 4. 유일한 RDN을 이용한 전역적 명명 방식의 예
Figure 4. An Example of the Global Naming Method with unique RDNs

본 논문에서 제안하는 전역적 명명 방식은 관련성 템플릿에 근간하고 있으며 망 관리 대리자와 MIB의 system MO간에 supervision 관련성을 설정하고 있다.

이러한 관련성은 클래스들 간의 관계를 설정하는 것이며 이러한 관련성 클래스 역시 관리 대상 객체 클래스로서 정의된다. 그러나 관련성 클래스를 인스턴스화 했을 때, 관련성 클래스의 인스턴스는 클래스 사이의 관계를 나타내는 것이므로 일반적인 인스턴스와는 같지 않다. 이러한 클래스는 클래스의 클래스 개념인 메타(meta) 클래스이다. 이러한 메타 클래스는 그 인스턴스 자체가

클래스로서 존재하는 클래스를 말한다[18]. 이러한 메타 클래스의 개념은 Smalltalk와 CLOS 언어 등을 이용할 때는 언어 자체에서 제공되고 있으나 C++ 언어를 이용한다는 가정을 하면 이 언어에서는 제공되고 있지 않으므로 관련성이라는 메타 클래스에 대한 설정이 필요하다.

C++ 언어로 구현되어 있는 메타 클래스의 예를 살펴 보면, UCL에서 구현한 망관리 플랫폼인 OSIMIS 플랫폼에서는 메타 클래스로서 MOClassInfo라는 클래스를 제공하고 있다. 이 MOClassInfo 클래스는 관리 대상

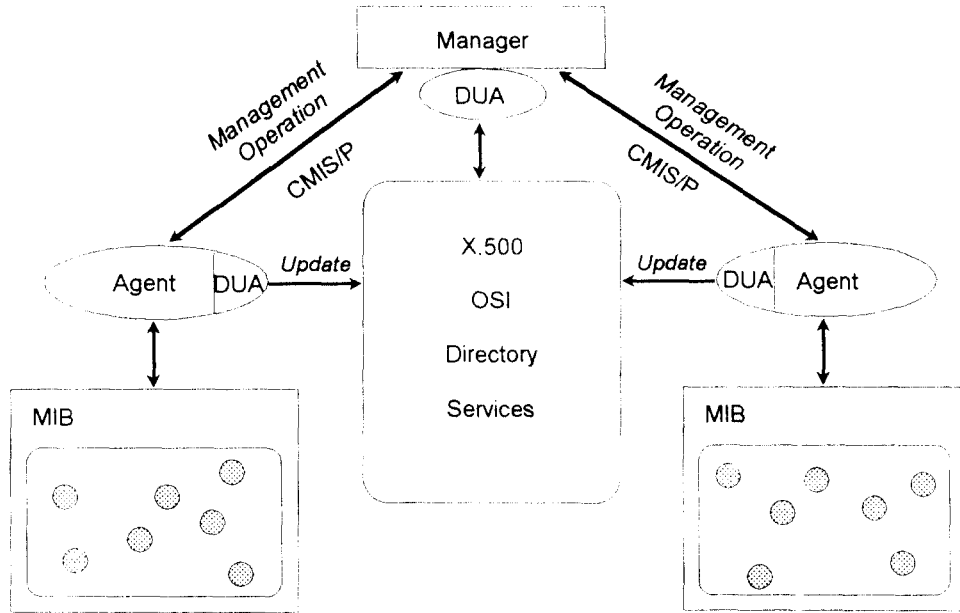


그림 5. 망 관리자 및 대리자와 디렉토리 서비스와의 관계
 Figure 5. A Scheme for the management operations using directory services

객체 클래스의 메타 클래스이며 모든 관리 대상 객체 클래스 각각에 대해 항상 MOClassInfo 클래스의 객체가 존재한다. 이것은 해당 클래스의 모든 객체에 대한 공통적인 정보 즉, 그 객체의 속성, 사건과 동작(action) 식별자, 이름 묶음 및 그 클래스의 처음 발생한 인스턴스 등에 대한 정보들을 포함하고 있다. 모든 관리 대상 객체 클래스는 이 클래스의 정적인 인스턴스를 가지고 있어서 모든 멤버들에 의해 공유되도록 하고 있다.

C++로서 망관리 시스템을 구현하는 경우를 가정했을 때, 이러한 MOClassInfo 클래스와 같은 방식으로 메타 클래스인 관련성 클래스를 정의할 수가 있다. 관련성 클래스는 그 템플릿 내에 ROLE들을 설정하고 있으므로 각 ROLE이 어떤 역할로 대응되는 지에 대한 설정을 하는 함수와 ROLE에 대한 정보를 읽어오는 함수를 구현자가 그 필요에 따라 적절하게 설정할 수 있을 것이다. 그러므로 관련성 클래스를 권고안에서 언급된 바와 같이 RelationTop에서 상속받게 하여 역할 클래스에 대한 메타 클래스로 구현할 수 있다. 이러한 메타 클레

스는 역할에 대한 특성들인 역할의 개수, 역할의 이름, 역할 내에 포함된 패키지의 개수, 허용 역할 개수, 필요 역할 개수 및 생성 모드, 삭제 모드 등에 대한 정보들을 가지게 된다.

그러면, 이에 대한 관련성 묶음은 각 역할에 대해 정의된 관련성으로부터 상속을 받아 각 역할에 대한 클래스를 대응시키게 된다. 그러므로 관련성 묶음에서는 각 역할에 대해 관련성 클래스에서 각 Role 클래스에 대해 설정된 역할 개수(cardinality)의 제한 등의 여러 특성을 Role 클래스를 통해 그대로 가져가면서 각각 클래스를 대응시키게 된다. 그러므로 관련성 묶음은 역할과 관리대상 객체 클래스간의 묶음을 설정하는 메타 클래스로서 정의할 수 있다.

이러한 방식으로 메타 클래스에서 지정된 역할들은 각각 객체 클래스로서 선언되어 자기 자신에 대한 제한된 속성과 제한된 동작을 일반 MO 클래스와 같이 갖게 된다. 이러한 관련성과 관련성 묶음, 역할 클래스들에 대한 연관성을 그림 6에 나타내었다.

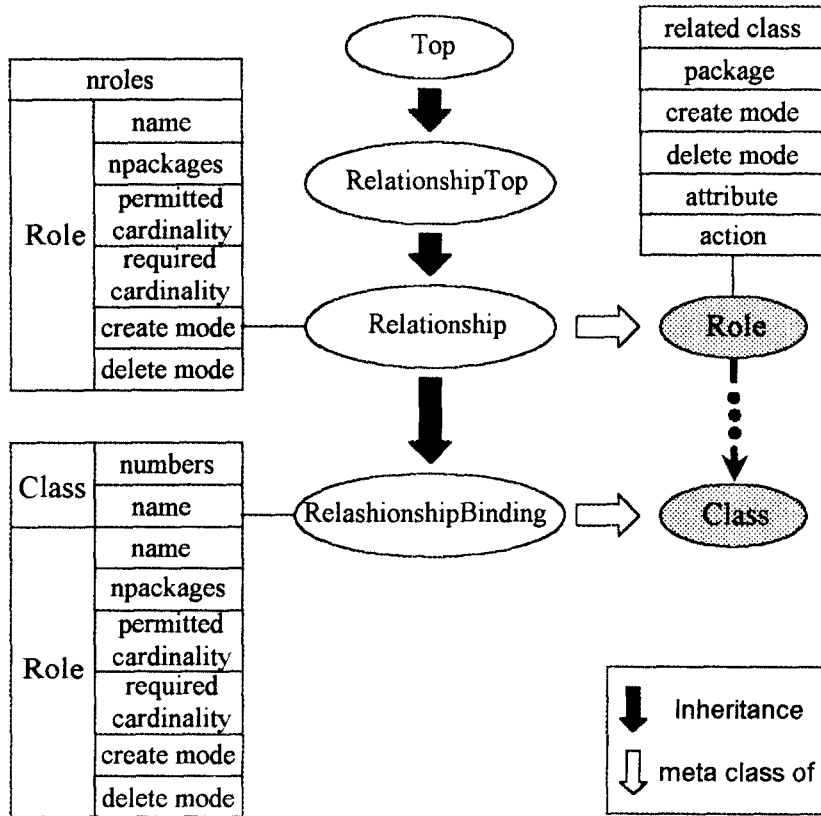


그림 6. 메타 클래스로서의 관련성 및 관련성 묶음
Figure 6. Relationship and Relationship Binding as meta classes

지금까지 정의된 관련성 및 관련성 묶음, 역할 클래스들은 망관리 플랫폼 전체를 관장하는 메타 클래스에 등록되어야 한다. 그러므로써 망관리 시스템이 관리 동작을 취할 때 MIB에 있는 각 MO에서 관련성을 인식하고 전체적인 망 관리 동작이 이루어질 수가 있다.

망 관리 대리자가 MIB를 초기화하는 과정에서 MIB 전체를 관장하는 메타 클래스를 초기화하게 되고, 이러한 초기화 과정에서 Relationship 클래스는 다시 메타 클래스로서 Role 클래스의 역할들을 초기화하게 되며 RelationshipBinding 클래스 역시 역할들과 관리 객체 클래스들 사이의 관계를 초기화하여 그 관련성 및 관련성 묶음들이 MIB에 설정되게 된다.

그러면, 명명 절차에 대해서 고찰해 보자. OSI 망 관리 시스템에서 사용하고 있는 MO에 대한 AVA 명명

방식은 디렉토리 서비스에서도 동일하게 쓰이고 있다. 즉, 디렉토리 시스템과 OSI 망 관리 시스템에서 사용하고 있는 DN은 다음과 같은 ASN.1 형식을 사용하고 있다.

```
DistinguishedName ::= SEQUENCE OF RelativeDistinguishedName
RelativeDistinguishedName ::= SET OF AttributeValueAssertion
AttributeValueAssertion ::= SEQUENCE (AttributeType, AttributeValue)
AttributeType ::= OBJECT IDENTIFIER
AttributeValue ::= ANY DEFINED BY AttributeType
```

RDN은 속성 형태와 속성 값의 집합으로 이뤄져 있으며 이러한 AVA 형태는 포인터 형식을 이용하여 연결되어 있으며, RDN은 시퀀스로 연결되어 고유 이

름(DN)을 형성한다.

GDMO에 정의된 각 MO는 앞서 정의된 바와 같이 Relationship과 RelationshipBinding 메타 클래스 정보를 초기화 과정에서 설정하므로써 각 관련성 및 관련성 묶음의 각 역할과 해당 클래스를 연결시킨다. 또한 이 과정에서 supervision 관련성의 supervise 역할에 설정된 대리자와 supervised-by 역할에 설정된 시스템 간의 속성 값을 이용하여 두 개의 DN을 붙이고 MIB 전체에 대한 메타 클래스에 등록시킨다. 이로써 MIB 내의 각 MO들에 대한 전역적 이름이 형성되며 망 관리 대리자가 이들 전역적 이름을 OSIMIS 망관리 플랫폼의 MOClassInfo 메타 클래스와 같은 MIB 전체에 대한 메타 클래스를 통해 인식할 수 있다.

6. 다른 MO 명명 방식과의 비교

분산 망에서 MO에 대한 대리자를 찾아내어 망 관리를 하는 방법을 위해 몇 가지 명명 방식들이 제안된 바 있다.

6.1. 명명 방식이 관리자에게 국부적인 경우

분산 망에서 각 관리자는 자체적으로 국부적 이름을 주소로 변환시켜주는 서비스를 제공하며 국부적 데이터 베이스를 이용한다. 이 서비스가 분산 명명 방식 보다 더 나은 성능을 보이기는 하지만 각 데이터는 국부적으로만 이용 가능하며 MO에 어떤 사건이 발생하여 통보 및 관리 동작이 필요한 경우 다른 관리 시스템에서는 그 MO에 대해 전역적으로 유일한 식별을 할 수 없다.

6.2. 모든 MO가 디렉토리에 엔트리로서 명명되는 경우 {6}{7}{8}

각 MO는 디렉토리 엔트리를 가지게 된다. 그러나 이러한 방식에서의 근본적인 문제점은 디렉토리 서비스에 저장되는 정보와 MO 사이의 특성 차이에 있다. 디렉토리 서비스에 저장되는 정보는 분산 망에서 전역적으로 연관되어 있으므로 정적인 데이터나 준동적인 데이터에 한해서만 디렉토리에 저장되어 있을 수 있으나 MO는 대부분이 동적인 성질을 가지고 있다. 그러므로 디렉토리 서비스를 MO 자체에 대한 정보 저장소로서 이용한다는 것은 매우 제한적이다.

6.3. 디렉토리 엔트리의 이름을 AVA에서의

하나의 값으로서 처리하는 경우

이 명명 방식에서는 MIB에 접목되는 디렉토리 엔트리에 대한 이름을 하나의 관리 정보 RDN으로 캡슐화하여 처리한다. 이 경우 그 DN의 시퀀스에 속한 어느 한 엔트리에 변화가 생겼을 경우 캡슐화된 특성 때문에 각 MO들에 대한 이름에 일관성을 주지 못하며 이를 바로잡기 위해서 새로이 이름 부여 과정이 필요하다.

6.4. NAME BINDING의 확장을 통한 명명 방식{9}{10}

기존의 MO 명명 방식에 쓰이는 NAME BINDING을 변형한 방식으로서 DIT와 MIT를 묶어 준다. OSI 망 관리 환경 및 TMN에서 쓰일 수 있는 합리적인 방법으로 여겨지나, 이를 위해서는 기존의 명명 방식에 쓰이는 기존의 템플릿을 확장해야 한다. 또한 NAME BINDING 자체가 포함관계를 바탕으로 하고 있으므로 이 방식에서는 디렉토리 서비스에서 제공하는 alias 서비스를 수용하지 못한다. 즉, 복수의 전역적으로 유일하게 식별 가능한 이름이 부여될 수 없다.

6.5. 관련성을 이용한 전역적 명명 방식

이 방식은 본 논문에서 제안한 것으로, NAME BINDING 자체가 포함 관련성을 갖는 관련성의 특별화된 한 종류이며, GRM의 RELATIONSHIP BINDING이 자체의 관련성을 이용하여 NAME BINDING에서처럼 실질적으로 MO 클래스들을 묶어 주는 역할을 할 수 있다는 관점에서 기존의 국부적 명명 방식을 새로운 표준안의 확장 없이 전역적 명명 방식으로 응용할 수 있다. 또한 이 방식은 n차의 관련성을 지원하기 때문에 디렉토리에서 쓰이는 alias 서비스를 수용할 수 있다.

이 방식은 OSF(Open Software Foundation)의 DCE(Distributed Computing Environment){19}에서의 "junction object"와 비슷한 구조로 MIB를 DCE의 디렉토리 시스템에 있는 CDS(Cell Directory Service)와 같이 디렉토리 서비스와 연계하여 이용할 수 있는데, 각 CDS의 root는 전역적 이름속에 포함되어 있는 대리자(agent)에 대응될 수 있다.

이 방식은 MIB가 초기화되어 메타 클래스로서 저장되어 있을 때 요구되는 기억 용량이 NAME BIND-

ING의 확장을 통한 명명 방식에 비해 좀더 요구되는 부분이 있겠으나 n차 관련성 설정이 가능하다는 점에서 보면 좀더 유연한 특성을 가질 수 있다. 일 예로서 DCE의 보안에 관련된 CDS에서는 그 root에 보안 관련 정보를 저장하고 있는데, 이것을 관련성을 이용하여 포인팅 해준다면 본 명명 방식에 보안 관련 정보 및 체계를 도입하는 응용 예도 가능하리라고 생각된다. 또한, TMN 간의 상호 작용을 위해서는 다른 TMN 영역에 있는 NE에 대한 식별이 중요한 점을 생각할 때, 이를 TMN으로의 응용도 가능하다[20].

7. 결 론

망 관리자가 분산 망에서 원격 대리자를 통해 망 운용 관리를 수행하기 위해서는 원격 대리자가 관리하는 MIB 내의 MO에 대한 전역적으로 유일하게 식별할 수 있는 이름이 필요하다.

OSI 망 관리 시스템은 지금까지 국부적 명명 방식에 대해서만 규정해 놓고 있다. 즉 국부 시스템에서의 system MO를 포함 트리 상에서 루트로 설정하여 그 MIB 내의 모든 MO들은 그 system MO를 기점으로 하는 이름만을 가지고 있다. 그러므로 본 논문에서는 관련성(relationship) 개념을 이러한 명명 방식으로 확장하여 메타 클래스로서 관련성을 구현함으로써 전역적 루트로부터 망 관리 대리자까지의 DN(Distinguished Name)과 국부 시스템의 MO의 DN을 연결하여 MO에 전역적 이름을 부여하였다. 또한, MO의 이름에 대리자의 RDN(Relative Distinguished Name)을 삽입함으로써 망 관리자는 MO에 대한 전역적 이름만으로도 X.500 디렉토리 서비스를 이용하여 망 관리에 있어 위치 투명성을 얻을 수 있다.

참고문헌

1. Draft N7053, ISO/IEC JTC/SC2/WG7, *Basic Reference Model of Open Distributed Processing - Part 1: Overview and Guide to Use*, May. 1992. Ottawa meeting.
2. ANSA: *Assumption, Principles and Structure*: ISA Project, 1991.
3. A. Lloyd, contribution to ISO/JTC1/SC21:

Modeling OSI Systems Management, WG4 N1029, 12-1-90.

4. J. Hall et al. "Management in a Heterogeneous Broadband Environment," in *Proc. IFIP 2nd Int. Symp. Integrated Network Management*, Washington, Apr. 1991. pp.613-624.
5. K. Jakobs et al. "Using Directory Services in Mobile Communication," *Proc. of IFIP COM-NET'90*, Budapest, May. 1990, pp.219-228
6. X. Zhang, D. Seret, J. L. Tesson, and C. Remy, "X.500 directory and OSI management," in *Proc. Globecom'92*, Orlando, FL, Dec. 1992, pp.1106-1110.
7. X.Zhang, and D. Seret, "Supporting Network Management Through Distributed Directory Service," in *IEEE J. Select. Areas Commun.*, vol. 12, Aug. 1994, pp.1000-1010.
8. J. W. Hong, M. A. Bauer, "Integration of the Directory Service in Distributed Systems Management", in *Proc. IFIP 3rd Int. Symp. Integrated Network Management*, San Fransisco, CA, Apr. 1993, pp.149-160.
9. U. Meyer, M.Tshichholz, and S. Waberroth, "A service for administering management information," in *Proc. IFIP 3rd Int. Symp. Integrated Network Management*, San Fransisco, CA, Apr. 1993, pp.137-148.
10. M. Saylor, "Junction objects: A solution to a problem in naming and locating OSI managed objects," in *Proc. IFIP 3rd Int. Symp. Integrated Network Management*, San Fransisco, CA, Apr. 1993, pp.161-173.
11. CCITT, *The Directory - Overview of Concepts, Models and Services*, CCITT X.500 Series Recommendation, Dec. 1988.
12. ISO/IEC DIS 10164-41 CCITT Rec. X.722, *OSI Management Information Services, Structure of Management Information - Part 4: Guideline for the Definition of Managed Objects(GDMO)*.
13. ISO/IEC DIS 10737, *Transport Layer*

- Management*.
14. CCITT, *The Directory - Overview of Concepts, Models and Services, Draft CCITT X.500 Series Recommendation*, Dec. 1991.
 15. ISO 10164-3, *Systems Management - Part 3: Attributes for Representing Relationships*, 1992. X.732.
 16. ISO 10165-7, *Management Information Services - Structure of Management Information - Part 7: General Relationship Model*, 1994. X.725.
 17. Subodh Bapat, *Object-Oriented Networks, Models for Architecture, Operations, and Management*, Prentice-Hall, Englewood Cliffs, New Jersey, 1994.
 18. Grady Booch, *Object Oriented Design with applications*, The Benjamin/Cummings, 1991.
 19. Ward Rosenberry, David Kenney, and Gerry Fisher, *Understanding DCE*, O' Reilly & Associates, Inc. 1992.
 20. Wang Cheol Song, Chang Eon Kang, "Globally Unique Names for Network Management between TMNs," JTC-CSCC' 95, Japan, July, 1995.



宋 旺 暉(Wang Cheol Song) 정회원

1986년 2월 : 연세대학교 식품공학과(공학사)
 1989년 2월 : 연세대학교 전자공학과(공학사)
 1991년 2월 : 연세대학교 대학원 전자공학과(공학석사)

1995년 8월 : 연세대학교 대학원 전자공학과(공학박사)

※주관심 분야 : 디지털통신 및 망관리 기술 등

康 昌 彦(Chang Eon Kang)

정회원

1982년~현재 : 연세대학교 전자공학과 교수
 한국통신학회 논문지 제20권 제1호 참조