망 관리 프로토콜 연동을 위한 확장된 GDMO

임 미 경† · 김 태 수†† · 이 광 휘††

요 약

본 논문에서는 서로 다른 망 관리 프로토콜을 사용하는 통신망(SNMP망과 CMIP망) 사이의 상호연동을 위한 시스템을 제안한다. 이를 위해 MOVI(Managed Object View Interface) 개념을 이용하여 두 종류의 접속(interface)을 가지는 새로운 관리 대상 객체를 정의 하였으며, GDMO 컴파일러를 수정하여 확장된 GDMO(EGDMO) 컴파일러를 구현하였다. 또한 새롭게 정의된 관리 대상 객체는 두개 이상의 인터페이스도 제공할 수 있으므로 이를 위한 일관성 제어(consistency control)와 원자적 수행(atomic action) 등을 수행하기 위한 접근 방법을 정의하였다.

관리 대상 객체가 다양한 접속을 제공할 수 있도록 새롭게 정의하는 방법을 사용함으로써 앞으로 다른 종류의 망 관리 프로토콜이 구현되어도 새로운 상위 계층의 게이트웨이를 구현할 필요 없이, 간단한 인터페이스의 변환만을 통하여 계속적으로 사용할 수 있도록 하였다. 물론 기존의 관리 대상 객체를 변형하여야 하지만, 기존의 관리 대상 객체를 모두 수용하며, 차후에 정의되는 관리 대상 객체에 대해서만 새롭게 정의하면 되도록 하였다.

이러한 연구를 통하여 복잡한 망 관리 프로토콜의 상호 연동도 가능하며, SNMP및 CMIP의 효용성을 증대 시킬 수 있고, 기존의 망 관리 프로토콜 자체의 연동뿐 아니라 앞으로 새로운 망 관리 프로토콜의 연동에도 그대로 적용할 수 있다는 장점을 가지고 있다.

An Extended GDMO for Interworking Different Network Management Protocols

Mi-Kyoung Lim† · Tae-Soo Kim†† · Kwang-Hui Lee††

ABSTRACT

This paper proposes an integrated system for interworking different network management protocols. The interworking system between SNMP(Simple Network Management Protocol) and CMIP(Common Management Information Protocol) has been designed.

We defined new managed object architecture using MOVI(Managed Object View Interface) concept and implemented an extended GDMO(EGDMO) compiler. An access mechanism for consistency control and atomic action on the sharing resources has been defined. Thus, even if a new management protocol would be introduced, new gateway system need not to be redesigned. It can be achieved with minimum modification on the view interface of managed objects. As a managed object can support various view interfaces, several network management protocols can directly access it.

This approach, therefore, provides some benefits. For instance, manager can access managed objects without additional functions and tools. It, also, could be achieved interworking between SNMP and CMIP with minimum modification on the view interface of the managed objects defined previously. It is able to interwork more complicated network management protocols and to increase usefulness of SNMP and CMIP.

※ 이 연구는 한국과학재단 핵심전문 연구비(과제번호: 961-0905-032-2) 지원에 의한 결과의 일부임
† 준 회 원 : 창원대학교 대학원 컴퓨터공학과
†† 정 회 원 : 창원대학교 컴퓨터공학과 교수
논문접수 : 1999년 9월 18일, 심사완료 : 1999년 11월 29일

1. 서 론

컴퓨터 통신망이 보다 복잡하게 되고 사용자가 원하는 서비스도 다양화됨으로 컴퓨터 통신망을 효율적으로 관리하는 기능이 필요하게 되었다. 현재 컴퓨터 통신망으로 널리 사용되는 인터넷에서는 망 관리 프로토콜로써 SNMP(Simple Network Management Protocol)를 제안하여 이용하고 있고 ISO에서는 OSI(Open System Interconnection) 프로토콜을 사용하는 통신망 관리를 위하여 CMIP(Common Management Information Protocol)을 제안하였으며, CMIP은 국내의 TMN(Telecommunication Management Network)에서 통신망 관리 프로토콜로 채택하고 있다[1, 2].

물론 CMIP과 SNMP는 대상이 되는 통신망이 서로 다르기 때문에 특성, 구조 및 기능에서 상당한 차이점을 보이지만 망이 통합되어가는 현 추세에 비추어 볼 때 두 통신망의 완전한 연동이 필요하다고 할 수 있다. 따라서 현재 가장 널리 사용되고 있으며 추후 계속적으로 사용될 두 통신망 관리 프로토콜의 서로 다른 특성들을 연동할 수 있는 시스템의 구현은 필수적이다.

본 연구의 목표는 망 관리 프로토콜로서 CMIP을 사용하는 시스템과 SNMP를 사용하는 시스템간의 상호 연동이다. 즉, TCP/IP를 기반으로 하는 인터넷과 OSI 프로토콜을 기반으로 하는 통신망을 관리적인 측면에서 연동함으로써 각각의 응용 범위를 확대할 수가 있다.

이러한 망 관리 프로토콜의 연동을 위해 본 연구에서는 새로운 개념과 기법을 사용하고자 한다. 즉, 관리 대상 객체(managed object)가 다양한 접속(interface)을 제공할 수 있도록 새롭게 정의함으로써 앞으로 다른 종류의 망 관리 프로토콜이 출현하여도 새로운 상위 계층의 게이트웨이를 구현할 필요 없이 간단한 인터페이스의 변환만을 통하여 계속적으로 사용할 수 있도록 한다. 이때 본 연구에서는 기존의 관리 객체를 모두 수용하며 차후에 정의되는 관리 객체에 대해서만 새롭게 정의하면 되도록 한다.

2. 망간 연동을 위한 MIB 구축 방안

2.1 제안한 방식

망 관리 프로토콜을 연동하는 방식에는 몇 가지가 알려져 있는데, 이들 중 대표적인 방식으로는 응용 게이트웨이 방식과 관리 객체 변환 방식이 있다[3]. 응용 게이트웨이 방식은 CMIP 프로토콜과 SNMP 프로토콜을 상호 변환하는 시스템을 구축하는 것으로 다중 스택 방식이라고도 하며 CMIP 및 SNMP 관리 영역이 상호 연동할 수 있도록 한다. 이 방법은 다른 계층에서 이미 구현한 게이트웨이 관련 기술을 이용할 수 있지만 관리 시스템사이의 프로토콜 매핑이 적당하지 않은 단점이 있다. 즉, 적은 수의 PDU(Protocol Data Unit)만이 직접적인 대응이 가능하며 프로토콜 대응이 PDU의 내용(content)과 의미(semantics)를 포함하지는 못한다. 또한 게이트웨이 시스템의 과부하와 병목 현상을 배제하기 어려운 점이 있다.

관리 객체 변환 방식은 OSI 관리 프로토콜과 관리 정보 구조를 관리자 시스템이 사용하고 대리인 시스템은 인터넷 관리 프로토콜과 관리 정보 구조를 사용하는 경우, 통합적으로 관리할 수 있도록 하는 데 초점을 맞춘 연구이다. IIMC(ISO/CCITT and Internet Management Coexistence)에서는 인터넷에서 망 관리 프로토콜인 SNMP와 OSI의 망 관리 프로토콜인 CMIP 사이의 상호 통합 운용을 위해 SNMP-CMIP 망 관리 표준을 통합 관리하는 방안에 대한 연구를 진행하고 있다.

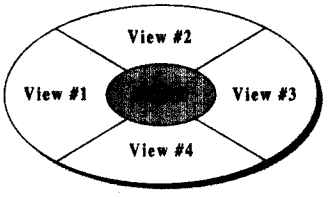
본 연구에서 제안하는 연동 방식은 관리 대상 객체를 통한 연동방식 이다. 이 방법은 기존의 응용 게이트웨이를 이용하는 방식과는 다르게 관리 대상 객체가 SNMP와 CMIP 모두를 포함하도록 새롭게 정의하는 것이다. 즉, 하나의 관리 대상 객체가 여러 종류의 접근점을 제공하므로 이를 통해 SNMP 및 CMIP 그리고 필요한 경우 다른 망 관리 프로토콜도 관리 대상 객체에 접근할 수 있는 확장성이 있다. 이러한 방식은 관리 대상 객체가 다양한 인터페이스를 제공하므로 여러 망 관리 프로토콜이 하나의 관리 객체에 직접 접근할 수 있도록 하여 관리자는 추가적인 기능 없이 관리 대상 객체에 접근할 수 있을 뿐 아니라 SNMP 및 CMIP의 효과적인 망간 연동이 이루어지며, 간단한 인터페이스의 변화만으로도 확장이 가능하다. 이것은 장기적인 측면에서 볼 때 더욱 효율적일 수 있다[4].

2.2 망간 연동을 위한 MIB 구축 방안

서로 다른 관리 망을 연동하는 데는 많은 문제점들이 있다. 예를 들어 하나의 통신 장비가 서로 다른 여러 관리자로부터 통제나 감시되어질 수 있다. 관리 프로토콜로서 SNMP를 사용하는 망과 CMIP을 사용하는

두 망이 있을 때, 하나의 통신 장비가 동시에 두 망 관리자로부터 접근된다면, 각각의 관리 도메인은 중첩된 관리 대상 객체(통신장비)에 대한 서로 다른 관리 뷰(view)를 가지게 될 것이다. 이러한 경우 관리 대상 객체의 구조는 둘 이상의 도메인으로부터 공유되도록 정의되어야 한다. 즉, 관리 대상 객체는 SNMP와 CMIP을 위한 두 가지의 관리 인터페이스 뷰를 제공해야 한다. 중첩된 도메인을 다루기 위한 방법으로는 관리 시스템이 중첩된 도메인 관리를 위한 도구를 갖추고 있거나 관리 대상 객체 자체에 이러한 메커니즘을 포함시키는 방법이 있다.

본 연구에서는 (그림 1)에서 보는 바와 같이 관리 대상 객체를 Core 부분과 View 부분으로 나누어 구성한다. 즉, 본 논문에 앞서 정의한 MOVI 개념을 도입해 모든 관리 대상 객체가 SNMP 및 CMIP을 지원할 수 있는 새로운 관리 대상 객체를 정의한다[4].



(그림 1) MOVI(Managed Object View Interface)

Core부분은 공통적인 특성을 갖는 부분으로 여러 도메인 관리자로부터 접근이 가능하며, 실제로 공유 자원에 대한 일관성이 필요한 부분이다. 반면 View부분은 각 관리자나 관리 도메인에게 보여지는 특수한 부분이다.

2.3 확장된 관리 대상 객체 모델링

CMIP에는 모두 9개의 템플릿이 존재하고 SNMP에는 5개의 매크로가 있다[5-11]. 본 연구에서는 CMIP과 SNMP를 통합하기 위해 SNMP에서 5개의 매크로와 CMIP의 5개의 템플릿을 대응시키며 CMIP에서 나머지 4개의 템플릿은 원래의 템플릿을 그대로 재사용한다. CMIP 템플릿과 SNMP 매크로의 대응을 나타내면 <표 1>과 같다.

CMIP과 SNMP 모두를 지원할 수 있는 새로운 관리 대상 객체는 GDMO를 확장시킨 형태(EGDMO : Extended GDMO)를 지닌다. 이러한 EGDMO는 SNMP와 CMIP의 공통적인 특성을 갖는 "공통 부분"과 SNMP 및 CMIP 각각을 지원할 수 있는 2개의 View 인터페이스인 "SNMP 독립 부분", "CMIP 독립 부분"으로 구성된다[12].

〈표 1〉 CMIP과 SNMP의 대응

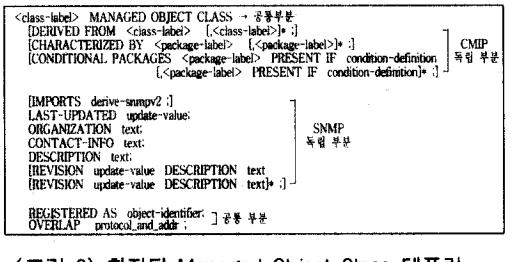
No. 프로토콜	CMIP (템플릿)	SNMP (매크로)
1	Managed-Object-Class	Module Identity
2	Attribute	Object Type
3	Package	Object Identity
4	Notification	Notification Type
5	Behaviour	Textual Convention
	Parameter, Attribute-Group, Action, Name Binding	

2.3.1 MOC(Managed Object Class) 템플릿

SNMP의 MODULE-IDENTITY 매크로와 CMIP의 Managed Object Class 템플릿을 통합하여 확장된 MOC 템플릿을 구성하였다(그림 2). MODULE-IDENTITY 매크로는 각 정보 모듈에 대한 연결(contact) 및 개정 이력을 제공하기 위해 사용되며, 모든 정보 모듈 내에 하나씩 있어야 한다. 반면 MOC 템플릿은 하나 이상의 상위 클래스로부터 특성을 상속 받고, behaviour, attributes, attribute groups, actions, notification을 포함하는 package들로 구성된다.

따라서 이들 매크로와 템플릿의 특성을 고려할 때, 객체를 정의하는 모듈의 이름 부분과 논리적인 저장 장소의 위치 그리고 OVERLAP 구문에서 공통점을 갖는다. 특히 "REGISTERED AS" 절에는 SNMP, CMIP을 위한 각각의 저장 위치를 명시하는데 이는 또 다른 관리 프로토콜이 추가되어도 간단한 인터페이스 변화로 확장이 용이하게 할 뿐만 아니라 다른 관리자의 접근을 허용하거나 제한하기 위해서 사용한다.

(그림 2)를 보면 관리 객체가 중첩되었는지 아닌지



(그림 2) 확장된 Managed Object Class 템플릿

를 나타내기 위해 OVERLAP절을 추가하였는데 관리 대상 객체가 중첩된 경우에는 중첩된 자원을 관리하는 manager address와 CMIP 도메인과 SNMP 도메인을 구분하기 위해 'C'나 'S'를 함께 명시한다. 이는 관리 대상 객체 클래스에 대한 참조와 갱신을 위해 사용된다. 이에 대하여는 4장에서 자세히 설명한다.

2.3.2 Attribute 템플릿

이 템플릿은 SNMP의 OBJECT-TYPE 매크로와 CMIP의 Attribute 템플릿을 합성하여 구성되었으며, 확장된 MOC 템플릿처럼 두 가지의 공통 부분과 각각의 독립 부분으로 이루어져 있다.

여기서 ATTRIBUTE name, SYNTAX, BEHAVIOUR 절은 속성값에 대한 구문과 연산을 나타내므로 공통 부분에 속하며, OBJECT-TYPE 매크로는 상속성과 파라미터를 내포하지 않기 때문에 DERIVED FROM 절, PARAMETER 절, MATCHES FOR 절은 CMIP 독립 부분에 해당한다. 또 STATUS, REFERENCE, INDEX 절 등은 SNMP 고유인 스칼라 형태의 2차원 배열을 표현하기 위한 수단이므로 SNMP 독립 부분에 해당한다.

2.3.3 Notification 템플릿

CMIP과 SNMP 각각에서 통지의 구문과 의미를 나타내는 Notification 템플릿과 NOTIFICATION-TYPE 매크로를 하나의 템플릿으로 확장할 수 있다.

2.3.4 Behaviour 템플릿

이 템플릿은 관리 대상 객체에 대한 상세한 설명을 위한 부분이므로 CMIP과 SNMP의 공통 부분이 된다.

2.3.5 Package 템플릿

Package 템플릿은 관리 대상 객체의 많은 특성들을 그룹화하기 위해 사용한다. 특히 ATTRIBUTE GROUP 절과 ACTION 절은 CMIP 프로토콜의 특성을 가지므로 CMIP 독립적인 부분으로 분리된다. 그리고 ATTRIBUTE 절은 attribute의 속성과 값을 지정하므로 공통 부분이 된다.

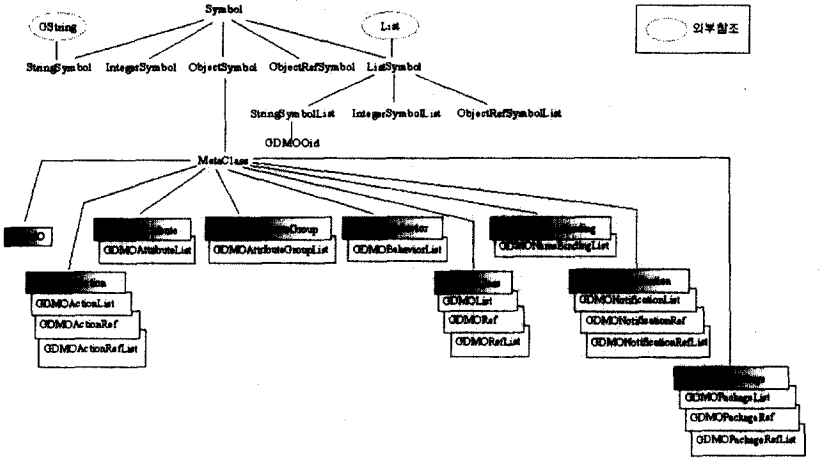
2.3.6 기타 템플릿들

나머지 확장된 템플릿들은 Parameter 템플릿, Attribute Group 템플릿, Action 템플릿, Name Binding 템플릿이며 이들은 SNMP에서는 사용되지 않고 CMIP 프로토콜 특성적인 GDMO 템플릿들이므로 CMIP 독립적인 부분이 된다. 즉, 원래의 GDMO 템플릿을 그대로 확장된 GDMO 템플릿으로 재사용한다.

3. 확장된 GDMO 컴파일러의 구현

3.1 OSIMIS의 분석

OSIMIS(OSI Management Information Service)는 영국의 UCL(Univ. College of London)에서 개발한 OSI 관리 시스템으로서 ISO 표준안에 근거를 둔 Manager/



(그림 3) symbol.h 내용을 중심으로 한 각 클래스간의 관계

Agent 구조를 따르고 있다[13-16].

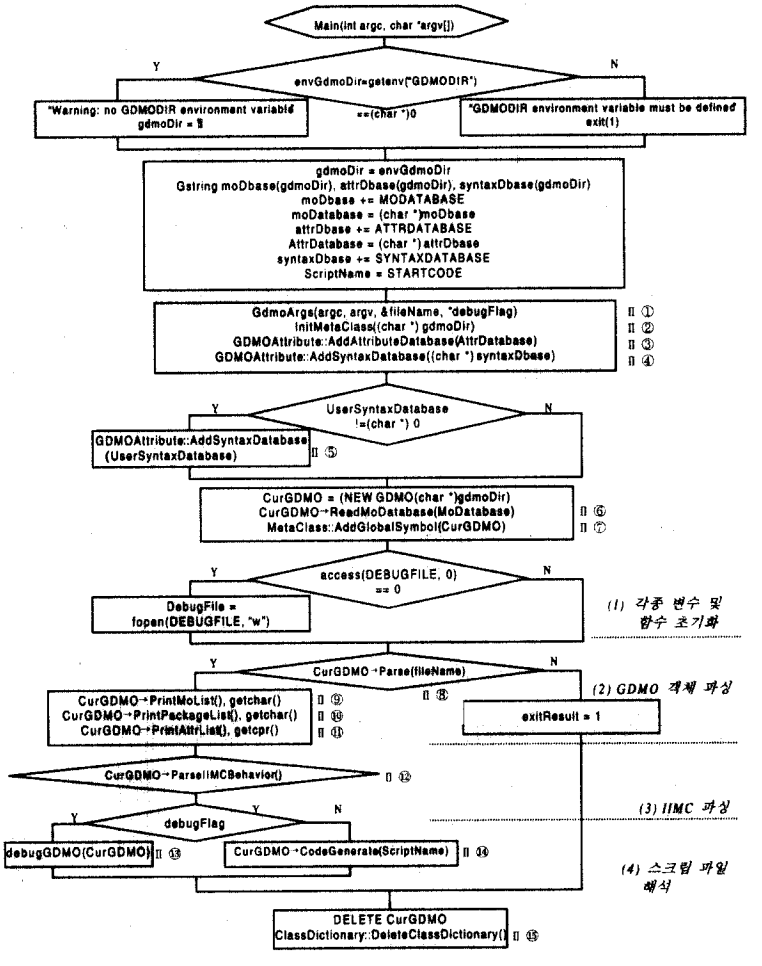
본 연구에서 수정한 부분은 'compilers' 디렉토리의 내용이며 그 중 compilers/mocompiler/gdmo 디렉토리 중심으로 분석을 하였으며, 디렉토리내의 파일들과 그 안의 클래스들의 관계를 나타내면 (그림 3)과 같다.

다음으로 각각의 클래스에서 파생된 세가지 형태의 클래스가 있는데 이들의 역할은 다음과 같다.

- ~List : 다수의 GDMO 템플릿을 정의하기 위해 사용
- ~Ref : 각각의 GDMO 템플릿을 포인트로 접근하기 위해 사용
- ~RefList : 포인트로 접근하는 GDMO 템플릿을 여러 개 정의하기 위해 사용

OSIMIS GDMO 컴파일러 소스에서 핵심이 되는 디렉토리는 /osimis/compilers/mocompiler/gdmo이며, 이 디렉토리에서 주가 되는 GDMOMain.cc 파일 중 main 클래스의 순서도를 (그림 4)에 나타내었다.

(그림 4)의 전체적인 내용을 살펴보면, (1)번 과정까지는 변수나 함수를 초기화하는 부분이고 (2), (3), (4)번 과정이 본 연구에서 핵심이 되는 부분이다. 즉, (2)번 과정은 GDMO 객체를 파싱하는 부분으로서 GDMO 템플릿들의 문법을 파싱하게 되며, Parse()라는 함수는 GDMOParse()를 호출하고 이 함수를 동작시키는 실제적인 소스는 GDMOLex.l과 GDMOYacc.y가 된다. 그리고 (3)번 과정은 IIMC 변환에 대한 파싱 과정으로서 ParseIIMCBehavior() 함수는 ParseIIMC() 함수를 호출



(그림 4) GDMOMain.cc 파일 중 main 클래스의 순서도

하고 다시 이 함수에서는 IIMC 객체를 이용하는데 이들 함수를 동작시키는 소스는 IIMCLex.l과 IIMCYacc.y이다. 마지막으로 (4)번 과정은 스크립 언어로 된 파일을 한 줄씩 읽어서 해석하는 부분으로서 InterPreter 객체를 이용하게 되는데 실제 소스는 ExecLex.l과 ExecYacc.y가 된다.

그러나 본 연구에서는 IIMC 내용에 따르는 것이 아니라 새로운 통합 방법을 제시하기 때문에 IIMC와 연관된 IIMCLex.l과 IIMCYacc.y은 생략한다. 그리고 실제 GDMO 문법 자체를 수정하기 때문에 GDMOLex.l과 GDMOYacc.y 파일을 고쳐 나간다.

3.2 확장된 GDMO 컴파일러 구현

SNMP및 CMIP을 통합한 망 관리 시스템은 SUN Sparc 10과 유닉스 환경에서 C++ 컴파일러, LEX 및 YACC를 사용하여 실험하였다.

앞에서도 살펴보았듯이 새로운 관리 대상 객체를 정의하는 EGDMO 형태는 기존의 CMIP에서 9개 템플릿과 SNMP의 5개 매크로를 통합한 것이다. 이 때 SNMP에서 5개 매크로는 CMIP의 9개 템플릿 중 5개와 대응되며 나머지는 CMIP 독립적으로 쓰이게 된다 (<표 1> 참고). 따라서 수정이 이루어져야 할 템플릿은 모두 5개 이나 본 논문에서는 공간상 3개의 템플릿 즉, Managed Object Class 템플릿, Notification 템플릿, Attribute 템플릿만을 보인다.

GDMOLex.l에 새롭게 정의되는 토큰을 나열했으며, GDMOYacc.y에서는 새로운 문법을 추가했다. 그리고 GDMO.*, GDMOClass.*, GDMOAttr.*, GDMONot.*에는 member function 및 variable을 추가하였으며 자세한 내용은 참고문헌 [12], [17]에 나타나있다.

3.2.1 문법적 변경사항

- Managed-Object-Class Template에 아래의 절(Clause)이 추가
  LAST-UPDATED, ORGANIZATION, CONTACT-INFO, DESCRIPTION, REVISIONDESCRIPTION절 등
- Notification Template에 아래의 절(Clause)이 추가
  OBJECT, STATUS, REFERENCE 절 등
- Attribute Template에 아래의 절(Clause)이 추가
  UNITS, MAX-ACCESS, STATUS, REFERENCE, INDEX, ARGUMENTS 절 등

3.2.2 소스 변경사항

1) GDMOYacc.y

이 파일에서는 SNMP와 CMIP이 통합된 새로운 문법을 제시하고 있으며 세 가지 템플릿의 변경된 내용은 (그림 5), (그림 6), (그림 7)에서 진한 글씨체로 표시하였고, 이들의 구체적인 내용은 참고문헌[17]에 나타내었다.

```
moc_body : opt_libcomment
           opt_libauthor
           opt_deriv_from
           opt_char_by
           opt_cond_pack
           last_updated
           organization
           cont_info
           description
           opt_revs_part
           reg_as
              {
                AddMoRegistration();
              }
```

(그림 5) MOC 템플릿의 변경된 내용

```
notif_body : opt_libcomment
             opt_libauthor
             opt_behaviour
             mode_def
                {
                  UpdateNotification("Mode", $4, Assign);
                }
             opt_param
             opt_syn_a_atts
             opt_notif_reply_syntax
             opt_objects
             status_notif
             opt_ref_notif
             reg_as
                {
                  AddNotificationRegistration();
                }
```

(그림 6) Notification 템플릿의 변경된 내용

```
attribute_body : opt_libcomment
                 opt_libauthor
                 deriv_or_syntax
                 opt_matches_for
                 opt_behaviour
                 opt_param
                 opt_units
                 max_access
                 status_attr
                 opt_ref_attr
                 opt_index
                 opt_arguments
                 opt_reg_as
                    {
                      AddAttributeRegistration();
                      _behavRefList.Copy(_curAttributeFindSymbol
                      "BehaviourReferenceList"));
                    }
```

(그림 7) Attribute 템플릿의 변경된 내용

2) GDMOLex.l

이 파일은 새로운 변수를 인식시키기 위해 토큰을

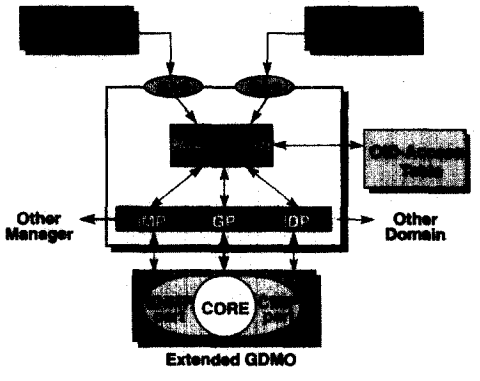
추가한 부분이다.

```
"LAST-UPDATED"      {return (LAST_UPDATED); }
"ORGANIZATION"      {return (ORGANIZATION); }
"CONTACT-INFO"      {return (CONTACT_INFO); }
"DESCRIPTION"       {return (DESCRIPTION); }
"REVISION"          {return (REVISION); }
"UNITS"             {return (UNITS); }
"MAX-ACCESSV"       {return (MAX_ACCESS); }
"not-accessible"    {return (NOT_ACCESSIBLE); }
"read-only"         {return (READ_ONLY); }
"read-write"        {return (READ_WRITE); }
"read-create"       {return (READ_CREATE); }
"STATUS"            {return (STATUS); }
"current"           {return (CURRENT); }
"deprecated"        {return (DEPRECATED); }
"obsolete"          {return (OBSOLETE); }
"REFERENCE"         {return (REFERENCE); }
"INDEX"             {return (INDEX); }
"ARGUMENTS"         {return (ARGUMENTS); }
"OBJECTS"           {return (OBJECTS); }
```

(그림 8) 새로운 토큰 추가

4. 통합 망 관리 시스템의 구축 방안

본 연구에서 새롭게 정의된 관리 대상 객체 클래스는 중첩된 실제 자원을 관리하는 시스템의 대리자 시스템 내에 두게 된다. 이것은 중첩되지 않은 다른 MIB도 관리해야 하기 때문이다. 확장된 관리 대상 객체 클래스로의 접근 방법은 크게 두 가지로 나뉜다. 하나는 참조(reference)를 위한 접근이며, 다른 하나는 갱신(update)을 위한 접근이다. 참조를 위한 접근인 경우(즉, 읽기 연산 : *M-GET, Get-Request, Get-Next* 등)에는 일관성 문제에 영향을 미치지 않으므로 특별한 제어가 필요하지 않다. 그러나 갱신을 위한 접근의 경우(즉, 쓰기 연산 : *M-CREATE, M-DELETE, Set-Request* 등)에는 공유자원에 대한 동시 제어(concurrency control)와 원자적 수행(atomic transaction)등을 위한 제어 메카니즘이 필요하다. (그림 9)에서 보는 바



(그림 9) 확장된 관리 대상 객체로의 접근

와 같이 서로 다른 프로토콜을 사용하는 관리자의 접근을 위해 두개의 SAP(Service Access Point)를 두었다.

4.1 확장된 관리 대상 객체로의 접근 방안

MIB로 접근하기 위해 AP는 요구한 관리자의 주소, OID(Object Identifier), Operation (PDU Type)을 가지고 Access Manager에게 접근 허가를 요청한다. 그리고 동시 제어(concurrency control)와 원자적 수행(atomic transaction)을 위해 GP, IMP, IDP모듈을 두었다. 이들의 자세한 내용은 아래에 설명되어 있다.

4.1.1 Access Manager

MIB로 접근하기 위해, Agent Process는 먼저 요구된 관리 대상 객체 클래스가 중첩되었는지 아닌지를 알아야 한다. 따라서 Access Manager는 OID-Access Table에서 해당 OID를 검색한다. OID-Access Table은 MIB가 컴파일될 때 생성되며 간단한 예가 (그림 10)에 나타나있다.

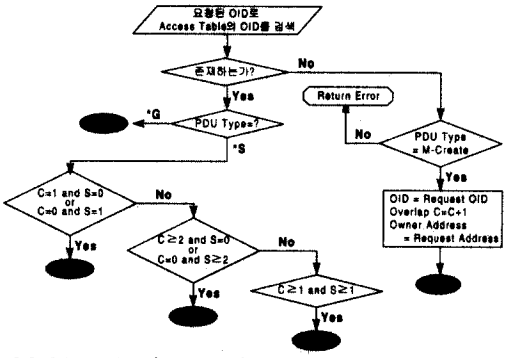
	CMIP OID	SNMP OID	Overlap	Owner Address	Other M_Address1	.....
●	Y.9.2.8	NULL	C=2,S=0	X.X.X.1	X.X.X.2	
●	Y.9.3.11	X.3.4.2	C=1,S=1	X.X.X.3	X.X.X.4	
●	NULL	X.5.7.3	C=0,S=2	X.X.X.5	X.X.X.6	
●	Y.9.5.7	NULL	C=1,S=0	X.X.X.7	NULL	
●	NULL	X.5.9.1	C=0,S=1	NULL	X.X.X.8	

(그림 10) OID-Access Table의 예

●, ●, ●은 중첩된 자원일 경우를 나타낸 것인데, ●은 Overlap 필드의 값 C = 2, S = 0으로써 CMIP 도메인 내의 두 GM(General Procedure)으로부터 중첩된 관리를 받는 관리 대상 객체이다. ●는 Overlap 필드의 값 C = 1, S = 1로써 SNMP도메인과 CMIP도메인간의 중첩을 나타낸다. ●은 ●과 마찬가지로 C = 0, S = 2로써 SNMP도메인내의 두 GM간의 중첩된 관리를 받는 관리 대상 객체의 예이다. 그리고 ●와 ●는 중첩되지 않은 일반적인 관리 대상 객체를 나타내는데, 각각 CMIP도메인과 SNMP도메인에 속하는 관리 대상 객체의 예들이다.

이 Table은 자주 접근되는 부분이므로 두개의 뷰(view) 형태로 메모리에 상주 되어 사용될 것이다. 즉, 뷰는 SNMP OID를 primary key로 갖는 뷰와 CMIP OID를 primary key로 갖는 뷰이다. Access Manager

의 처리 알고리즘을 (그림 11)에 나타내었다.

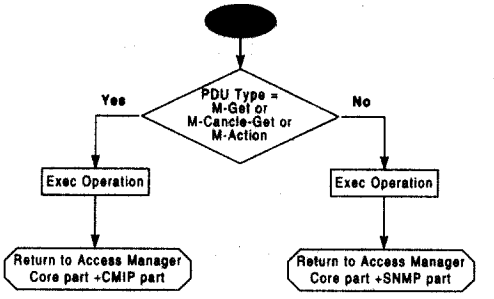


(그림 11) Access Manager의 처리 알고리즘

OID-Access Table내에 요청된 관리 대상 객체 클래스의 OID가 존재하지 않고 연산이 M-Create가 아닌 경우에는 잘못된 연산을 요청한 경우이므로 Error를 Return한다. 읽기 연산(*G)이 요청된 경우와 OID가 존재하지 않더라도 요청된 연산이 M-Create일 경우와 OID가 존재하고 갱신연산(*S)일 경우라 하더라도 Overlap 필드의 C와 S가 "C = 1, S = 0"이거나 "C = 0, S = 1"일 경우에는 관리 대상 객체가 중첩되지 않았으므로 동시성 제어와 같은 특별한 메카니즘이 필요하지 않다. 그래서 Access Manager는 GP를 호출한다. M-Create가 수행되면 Access Manager는 OID-Access Table을 갱신하고 난 후 GP를 호출한다. OID-Access Table내에 해당 OID가 존재하고 요청된 연산이 쓰기 연산일 때, Overlap필드가 "C≥2, S = 0"이거나 "C = 0, S≥2"이면 하나의 도메인내의 여러 관리자에 의한 중첩이므로 Access Manager는 IMP(intermanager procedure)를 호출한다. 이때 Overlap필드가 "C≥1, S≥1"이면, 서로 다른 도메인에 의한 중첩이므로 Access Manager는 IDP를 한다.

4.1.2 General Procedure(GP)

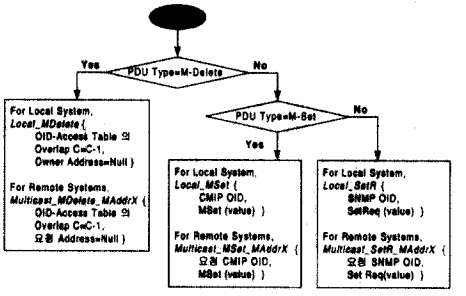
이 프로시저는 일반적인 참조연산을 수행하며, 중첩되지 않은 MO class들에 대한 연산을 수행한다. CMIP 연산과 SNMP연산을 분리 인식하여 공통부분과 각각의 독립부분을 Return한다. GP의 처리 과정을 나타내면 (그림 12)과 같다.



(그림 12) GP(General Procedure)의 처리 알고리즘

4.1.3 InterManager Procedure(IMP)

IMP는 하나의 동일한 도메인 내에서 중첩된 관리 대상 객체를 처리하는 프로시저이다. IMP의 처리 알고리즘을 (그림 13)에 나타내었다.

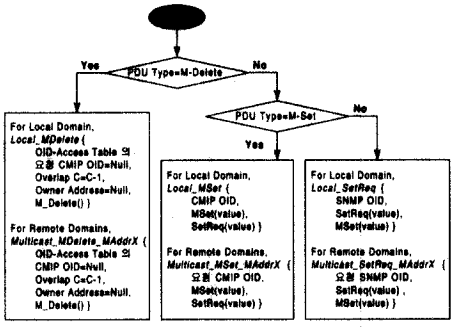


(그림 13) IMP(InterManager Procedure)의 처리 알고리즘

특히, M-Delete인 경우 IMP는 해당하는 MO class 부분을 삭제하는 대신에 OID-Access Table의 해당 OID를 찾아서, 요청한 Owner Address를 지우고 Overlap 필드의 CMIP을 나타내는 C의 값을 1만큼 감소 시킨다. 이렇게 함으로써 다른 관리자들에게 관리 대상 객체로의 접근을 계속적으로 보장할 수 있다. 그런 다음 동시제어와 원자적 수행을 위해 IMP는 다른 시스템의 OID-Access Table을 같은 방법으로 갱신하도록 한다. 이때 본 연구에서는 멀티캐스트를 이용하여 동시성제어와 원자적 수행을 하도록 제안한다. CMIP PDU인 M-Set이나 SNMP PDU인 Set-Request의 경우에는 요청된 OID의 해당 속성값들을 갱신하므로 각각 자신이 관리하는 MIB와 다른 중첩 관리자가 관리하는 MIB의 일치성을 유지하기 위해 Local Function과 Remote Function을 수행한다.

4.1.4 InterDomain Procedure(IDP)

IDP는 서로 다른 도메인 간의 중첩된 관리 대상 객체를 다루는 처리 프로시저로써 (그림 14)에 처리 알고리즘을 나타내었다.



(그림 14) IDP(InterDomain Procedure)의 처리 알고리즘

M-Delete 연산인 경우, 이는 CMIP 프로토콜만이 가지는 PDU이므로 IDP는 MO class의 CMIP 부분만을 삭제하고 공통부분은 그대로 둔다. 그리고 OID-Access Table의 CMIP OID, Owner Address를 삭제하고 Overlap 필드의 C를 1 감소시킨다. 그런 다음 IDP는 다른 도메인의 관련된 OID-Access Table에 대해 갱신 작업을 수행한다. 특히, M-Set연산이나 Set-Request연산인 경우에는 CMIP Domain에서 관리하는 관리 대상 객체의 값과 SNMP Domain에서 관리하는 관리 대상 객체의 값이 일치해야 하므로 Local MIB와 Remote MIB들에 대해서 동시에 값을 갱신하여야 한다. 그래서 IDP는 MSet(value)와 SetReq(value)를 수행한다.

5. 결 론

본 연구에서는 망이 통합되어가는 추세에 따라 현재 가장 널리 사용되는 SNMP와 CMIP이라는 통신망 관리 프로토콜을 연동하기 위한 방안을 제시하였으며 이의 방법으로 CMIP을 정의하는 GDMO(Guidelines for the Definition of Managed Objects) 문법에 SNMP 문법을 추가하여 확장된 GDMO(EGDMO-Extended GDMO)를 구성하였다. 결과적으로는 28개의 함수가 증가하였다.

이렇게 구성된 문법적인 변경사항을 실제 소스에서 변경하였고 컴파일 결과 현재까지 수정한 파일들이 오류 없이 처리됨을 확인하였다. 그리고 이렇게 정의된 관리 대상 객체에 대한 접근 방법을 제시하였다. 본 연구는 기존의 SNMP 및 CMIP 망 관리 프로토콜이 상호 연동할 수 있도록 해주고, 더 나아가 새로운 망 관리 프로토콜이 나타나도 인터페이스만 변화 시켜 적용할 수가 있다.

향후 계획으로는 실제로 통합된 망 관리 프로토콜이 정확하게 동작하는 과정을 보이며 다른 통합 망 관리 프로토콜과 성능을 비교하는 것이다. 그리고 현재 사용자 접속부분에 대한 구현을 과제로 수행하고 있다.

참 고 문 헌

[1] Divakara K.Udupa, "Network Management System Essentials," 1996.

[2] Heinz-Gerd Hegering, Sebastian Abeck, "Integrated Network and System Management," Addison-Wesley, pp227-236, 1995.

[3] 한국전산원, "통합 망 관리 표준화 연구 보고서", 1995. 12.

[4] K.-H. Lee, "MOVI : Management Object View Interface for Hierarchical Distributed Network Management Systems," pp.12-16, IEEE ICC Volume 1, 1996.

[5] William Stallings, "SNMP, SNMPv2 and CMIP," Addison-Wesley, 1993.

[6] RFC 1155, "Structure and Identification of Management Information for TCP/IP-based Internets".

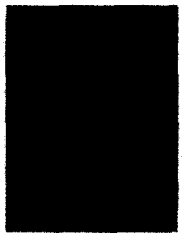
[7] RFC 1213, "Concise MIB Definitions".

[8] RFC 1902, "Structure of Management Information for Version 2 of the Simple Network Management Protocol(SNMPv2)".

[9] ISO/IEC 10165-2 | CCITT X.721 : Information technology Open Systems Interconnection Structure of Management Information : Definition of Management Information.

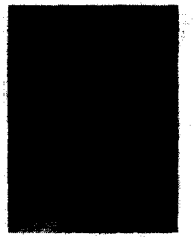
[10] ISO/IEC 10165-4 | CCITT X.722 : Information technology Open Systems Interconnection Structure of Management Information : Guidelines of the Definition of Managed Objects.

[11] Tony Jeffre, "Guidelines for the Definition of Managed Objects" In Morris Sloman(Ed), "Network and Distributed Systems Management," pp.131~164, Addison-Wesley, 1994.
[12] 김태수, "통합 통신망 관리 시스템 구축 방안 연구," 창원대학교 컴퓨터공학과 석사학위 논문, 1997, 12.
[13] 포항공과대, "OSI 네트워크 관리 플랫폼," http : //amazon.postech.ac.kr/aiit/osimis/osimis_intro.htm.
[14] OSIMIS README file.
[15] SIMIS GDMO compiler user manual.
[16] G. avlou, G. night, "The OSI Management Information Service User's Manual, Version 3.0," January 1993.
[17] 임미경, "망관리 연동을 위한 확장된 GDMO 구현", 창원대학교 컴퓨터공학과 석사학위 논문, 1998, 8.



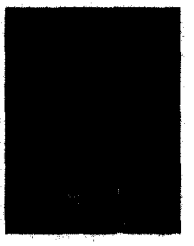
임 미 경

e-mail : mklim@sarim.changwon.ac.kr
1997년 창원대학교 자연과학대학 전자계산학과(학사)
1999년 창원대학교 대학원 컴퓨터공학과(이학석사)
1999년~현재 창원대학교 컴퓨터공학과 시간강사
관심분야 : 통신망관리, 멀티캐스트 프로토콜, 이동통신망



김 태 수

e-mail : jupi@sarim.changwon.ac.kr
1995년 경상대학교 농과대학 임산공학과(학사)
1998년 창원대학교 대학원 컴퓨터공학과(이학석사)
1999년~현재 창원대학교 대학원 컴퓨터공학과 박사과정
관심분야 : 통신망관리, 멀티캐스트 프로토콜, 이동통신망, 정보보안



이 광 휘

e-mail : khlee@sarim.changwon.ac.kr
1983년 고려대학교 공과대학 전자공학과 졸업(학사)
1985년 고려대학교 대학원 전자공학과 통신전공(공학석사)
1989년 고려대학교 대학원 전자공학과 컴퓨터전공(공학박사)
1988년~현재 창원대학교 컴퓨터공학과 교수
1991년~1993년 영국 Walse대학(Swansea) 및 Reading 대학교 연구원
1995년~1996년 영국 런던대학(UCL) 연구원
1997년~1998년 영국 Walse대학교(Swansea) 및 New Bridge Networks사 연구원
관심분야 : 망관리시스템, 멀티캐스트 프로토콜, 이동통신망, 분산시스템, ATM