MODE : TMN 체계의 ATM 망 관리를 위한 관리 객체 개발 환경

강 원 석† · 김 기 형†† · 김 영 탁†††

요 약

다양한 통신장비로 구성된 대규모 종합 네트워크를 운용하기 위해서는 이들을 체계적으로 관리할 수 있는 통신망 관리 기술이 필수적이다. 통신망의 체계적인 관리를 위해 ISO 및 ITU-T에서는 CMIP기반의 TMN 권고안을 제정하고 있다. TMN에서는 관리객체를 정의하기 위해 GDMO(Guidelines for the Definition of Managed Objects)를 사용하며, 따라서 GDMO 에이전트 플랫폼을 개발시 GDMO 컴파일러가 필요하게 된다. 본 논문에서는 GUI 기반의 GDMO 관리객체 개발환경인 MODE를 제시한다. MODE는 관리객체 구현을 쉽게 하기 위해 관리객체 코드를 시스템에 독립적인 코드(SIC)와 시스템에 종속적인 코드(SDC)로 나누고 각 코드의 개발을 도와 준다. 실험결과로 MODE를 이용하여 ATM스위치의 관리 객체를 구현해 보고 MODE의 효용성을 보인다.

MODE : Managed Objects(MOs) Development Environment for TMN-based ATM Network Management

Won-Seok Kang† · Ki-Hyung Kim†† · Young-Tak Kim†††

ABSTRACT

Systematic telecommunication network management is essential for operating large-scale integrated networks which consist of various network components manufactured by different vendors. ISO and ITU-T recommend the CMIP-based TMN architecture for this purpose. TMN uses GDMO for the definition of managed objects, and various GDMO compilers have been developed. However, the development of managed objects by using these compilers is still a difficult task.

In this paper, we present a GUI-based managed objects development environment, MODE. MODE divides managed object codes by system independent code(SIC) and system dependent code(SDC). By providing development environments for SIC and SDC, MODE can ease the development of managed objects. To show the efficiency of MODE, we develop the managed objects of an ATM switch in MODE.

1. 서 론

초고속 광대역의 급속한 기술 발전 추세에 있는 공중 통신망에 있어서 해결해야 할 가장 큰 과제중의 하나는 ATM/B-ISDN등의 새로운 통신망을 얼마나 효율적으로 운용, 관리할 수 있는가 하는 점이다. 특히 다양한 장비 제조업체의 통신 장비들을 하나의 대규모 종합 네트워크로 운용하기 위해서 효율적이고 체계적인 통신망 관리 기술이 필요하다. 또한, 통신망으로 유

† 정 회 원 : 영남대학교 대학원 컴퓨터공학과
†† 종신회원 : 영남대학교 컴퓨터공학과 교수
††† 정 회 원 : 영남대학교 정보통신공학과 교수
논문접수 : 1998년 9월 7일, 심사완료 : 1998년 11월 20일

입되는 트래픽 양이 저속에서 초고속에 이르기까지 다양하며 시간에 따라 급격히 변화하는 서비스도 함께 수용해야 함으로 통신망의 효율적인 자원관리가 필요하다.

통신망의 체계적인 관리를 위해 ISO 및 ITU-T에서는 CMIP기반의 TMN 권고 Series를 제정하고 있으며, 대규모 종합정보통신망을 운용하고 있는 공중통신망에서는 ISO/ITU-T의 TMN체계를 기반으로 한 통신망 관리기능을 구축하고 있다. TMN에서는 관리객체를 정의하기 위해 GDMO를 사용하며, 따라서 GDMO 에이전트 플랫폼의 개발시 GDMO 컴파일러가 필요하게 된다.

기존의 GDMO 컴파일러로는 OSIMIS의 GDMO 컴파일러[2,3,4,5], DSET사 GDMO 컴파일러[6], 그리고 그 외 연구용 GDMO 개발환경[8,9]등이 있다. 이들 컴파일러들은 GDMO 템플릿을 입력으로 받아 C++로 번역해 준다. 그러나 GDMO 템플릿에 대한 정보등록 및 초기화 코드만을 생성하기 때문에 완전한 객체를 구현하는 작업은 여전히 어렵다.

본 논문에서는 GDMO 에이전트 플랫폼에 필요한 관리 객체 개발을 위해 GUI 기반의 관리객체 개발환경인 MODE를 제시한다. MODE에서는 관리객체의 개발작업을 쉽게 하기 위해 객체 코드를 두 가지 부류 -- 시스템에 독립적인 코드(SIC)와 시스템에 종속적인 코드(SDC) -- 로 분류한다. 즉, SIC는 GDMO 템플릿에 대한 정보등록 및 초기화를 위한 코드이고, SDC는 CMIP-Request에 대한 메소드를 위한 코드이다. MODE는 SIC와 SDC를 위한 편집 환경을 GUI를 이용하여 제공함으로써 관리객체 구현 부담을 덜어준다. 실험결과로 MODE를 사용하여 ATM 스위치의 관리객체를 구현함으로써 MODE의 효용성을 보인다.

본 논문의 구성은 다음과 같다. 2장에서 GDMO의 개요를 나타내며, 3장에서는 MODE 구성 및 특성을 나타낸다. 4장은 ATM 스위치의 관리 객체 구현을 설명하며, 5장에서는 MODE의 실행 결과를, 마지막 6장에서는 본 논문의 결론을 맺는다.

2. GDMO

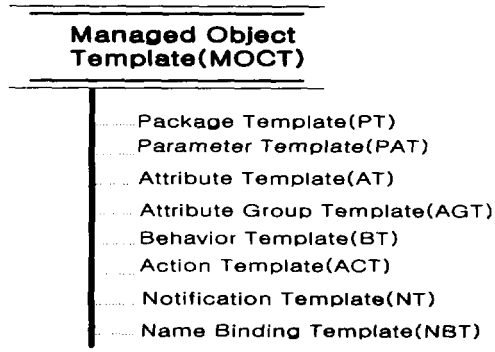
2.1 GDMO 개요

OSI 네트워크 관리는 네트워크에 연결된 장치를 표현하기 위해 객체 지향 모델을 사용하고 각 장치는 관리 객체(MO)로 불리는 객체 집합으로 표현된다. 관리 객체의 집합을 관리 정보 베이스(Management Information Base : MIB)로 부르며 GDMO는 관리 객체 정의의 정형적인 의미를 제공한다. 즉 이것은 관리 객체의 템플릿으로 실행 중에 관리 객체들로 인스턴싱될 관리 객체 클래스(Managed Object Classes : MOCs)집합을 정의하기 위해 사용하는 객체 지향 언어이다. 각 MOC에 대하여 사용자는 이것이 갖는 MIB의 상속과 포함 관계 계층 양쪽에서의 어떤 위치, 속성의 이름과 타입들, 신고, 이것에서 수행될 수 있는 행위들을 명세한다.

GDMO는 통신 관련 연구자들에게 그들이 생각하는 개념을 세부적으로 구현 할 수 있도록 도와준다.

2.2 GDMO Template

GDMO에는 (그림 1)과 같이 관리 객체 하나 당 9개의 템플릿이 사용된다.



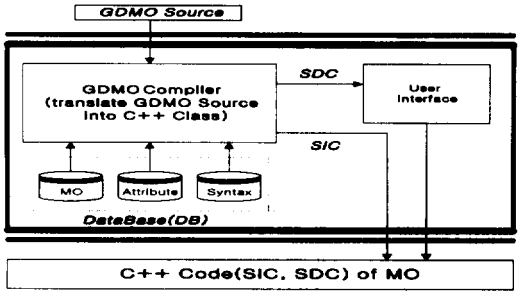
(그림 1) 관리 객체 구조
(Fig. 1) Managed Object hierarchy

MOCT는 어떤 관리 객체 클래스 정의의 "core"형태로 정의되는데, NBT를 제외한 나머지 템플릿의 정보를 가지고 있다. MOCT의 내부에는 상속 MOC와 MOCT에 속한 PT와 Conditional PT들을 포함 할 수 있다. PT는 템플릿들의 그룹화된 어떤 객체의 특성을 나타낸다. AT, NT, ACT, PAT가 여기에 속한다. PT는 MOC의 BT, AT, AGT, ACT, NT들을 포함 할 수 있다. PAT는 Syntax에 관련된 정보가 포함된다. PAT는 AT, BT와 어떤 Attribute의 Syntax Type를 포함할 수 있다. AT는 Attribute Type와 Behavior를 정의하는데 사용된다. AT는 상속 AT, BT, PAT와 At-

tribute의 Syntax Type를 포함 할 수 있다. AGT는 하나의 그룹으로서 수행되어질 Operation들을 그룹화 한다. AGT는 여러 개의 Attribute들을 포함 할 수 있다. BT는 관리 객체의 행동이나 구성요소의 행동을 정의한다. ACT는 Get이나 Replace와 같은 명령 형태의 의미로 구현할 수 없는 관리 객체에 대하여 명령을 정의한다. ACT는 BT, PAT를 포함 할 수 있으며 Information Syntax와 Reply Syntax Type을 포함 할 수 있다. NT는 관리 객체의 resource와의 관계에 있어서 event를 유도한다. NT는 BT, PAT를 정의 할 수 있으며 Information Syntax와 Reply Syntax Type을 포함 할 수 있다. NBT는 관리 객체에 대하여 논리적 제한과 제한적 가능성을 정의한다. NBT는 상속되어진 클래스와 자신의 클래스를 정의 할 수 있으며 AT와 BT, PAT들을 포함 할 수 있다.

3. MODE(Managed Objects Development Environment)

MODE는 **(그림 2)**와 같이 크게 GDMO Compiler, SIC(System Independent Code), SDC(System Dependent Code), User Interface, DataBase(DB)로 구성된다. 관리 객체를 개발하는 과정은 크게 두 단계로 이루어진다. 첫 번째 단계에는 GDMO Source를 입력으로 받아 GDMO Compiler를 이용하여 SIC와 SDC를 생성한다. 두 번째 단계에는 MODE의 User Interface를 이용하여 사용자가 원하는 관리 객체를 생성하게된다. 다음은 각각의 구성요소별로 특징 및 기능을 설명한다.



(그림 2) MODE 구성도
(Fig. 2) MODE composition diagram

3.1 DataBase(DB)

DB는 Attribute, MO, Syntax DB로 구성된다. Attribute DB는 GDMO Source에서 제공되는 Attribute 클래스들을 리스트로 텍스트파일로 저장된다. MO DB는 GDMO Source에서 정의된 MO와 상속 MO들을 정의하는 것으로 텍스트 파일로 저장된다. Syntax DB는 ASN.1 syntax들을 리스트로 저장된다.

3.2 SDC와 SIC

SIC는 GDMO Source나 특정 장비에 관계없이 생성되는 코드 부분으로 C++코드로 변환할 때 관리 객체에 대한 기본적인 Class 초기화 부분이다. **<표 1>**은 SIC에 포함되는 기본 메소드이다. SIC의 생성과정은 **3.4절**에서 설명한다.

〈표 1〉 SIC에서 생성되는 메소드
〈Table 1〉 The Methods of SIC

함 수	기 능
Constructor	관리 객체 생성자로써 이에 포함된 모든 Attribute들을 생성한다.
Destructor	관리 객체가 소멸될 때 사용된다.
InitialiseClass	관리 객체에 포함된 모든 template에 대한 초기화를 설정한다.
Create	관리 객체를 생성

SDC는 GDMO Source나 특정 장비에 따라 다른 시스템 종속적 코드로써 GDMO Source에 나타난 관리 객체에 대하여 사용자가 관리 객체 구현 시 자주 사용하는 메소드들을 생성해준다. **3.3절**에 설명할 MODE의 User Interface를 사용하여 사용자가 수정 변경하여 관리 객체를 완성한다. SDC에 포함되는 기본 메소드는 **<표 2>**와 같다. SDC생성과정은 **3.4절**에서 설명한다. **<표 2>**에 나타난 함수들의 기능은 아래와 같다.

〈표 2〉 SDC에서 제공되는 API
〈Table 2〉 The API of SDC

CMIP .Requests에 대한 함수
createRR()
deleteRR()
get()
set()
action()
buildReport()

• **createRR()**

createRR()은CMIP의M-CREATE-Request에 대한 메소드로 관리 객체 생성 시 호출된다. 이 함수의 내부에는 어떤 관리 객체에 속한 Attribute에 대한 초기화 설정 메소드, SNMP MIB와 연동을 위한 메소드, Polling을 위한 메소드들을 제공한다.

• **deleteRR()**

deleteRR()은 CMIP의 M-DELETE-Request에 대한 메소드이다. 이 함수는 어떤 관리 객체가 생성될 때 설정한 것을 해제하는 메소드를 포함한다.

• **get()**

get()은 CMIP의 M-GET-Request에 대한 메소드이다. 이 함수는 어떤 관리객 체에 속한 Attribute들을 변수화한 부분과 SNMP MIB와 연동을 위한 메소드들을 제공한다.

• **set()**

set()은 CMIP의 M-SET-Request에 대한 메소드이다. 이 함수는 어떤 관리객 체에 속한 Attribute들을 변수화한 부분과 SNMP MIB와 연동을 위한 메소드들을 제공한다.

• **action()**

action()은 CMIP의 M-ACTION-Request에 대한 메소드이다. 이 함수는 어떤 관리 객체에 속한 action들을 변수화한 부분을 포함한다.

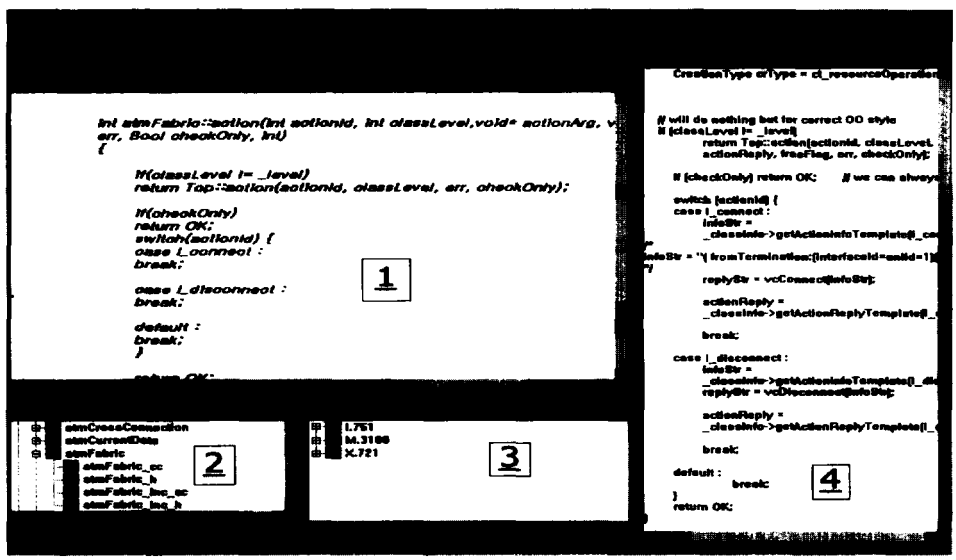
• **buildReport()**

buildReport()는 CMIP의 M-Event-Report에 대한 메소드이다. 이 함수는 어떤 관리 객체에 속한 Notification들을 변수화한 부분과 Event 전달 시 값을 설정하 기 위한 조건문을 제공한다.

3.3 MODE의 User Interface

MODE의 User Interface는 **3.4**절에서 설명할 GDMO Compiler에서 생성된 SDC를 이용하여 사용자가 망 관리에 있어서 어떤 특정 기능을 추가하기 위해 제공되는 개발툴이다.

MODE의 User Interface는 **(그림 3)**과 같다. 아래 **(그림 3)**에서와 같이 [1]은 HTML 형태의 문서를 볼 수 있는 뷰어로 사용자가 [1] 화면을 보고 SIC와 SDC 코드를 본다. [2]는 SDC를 Tree형태로 나타내는 뷰어로 GDMO Source에 나타난 관리 객체들에 대하여 MOName.cc, MOName.h, MOName.inc.cc, MOName.inc.h를 나타낸다. [2]의 관리객체를 클릭하면 [1]에 나타난



(그림 3) MODE의 User Interface
(Fig. 3) The User Interface of MODE

다. [3]은 GDMO Source에 나타난 MO들을 Tree형태로 나타낸다. [3]의 관리객체를 클릭하면 [1]에 GDMO Source를 나타낸다. [4]는 사용자가 코딩을 할 수 있는 Edit 창이다. 사용자는 [1], [2], [3]을 이용하여 SDC를 완전한 형태로 변환하여 하나의 관리 객체를 생성한다.

3.4 GDMO Compiler

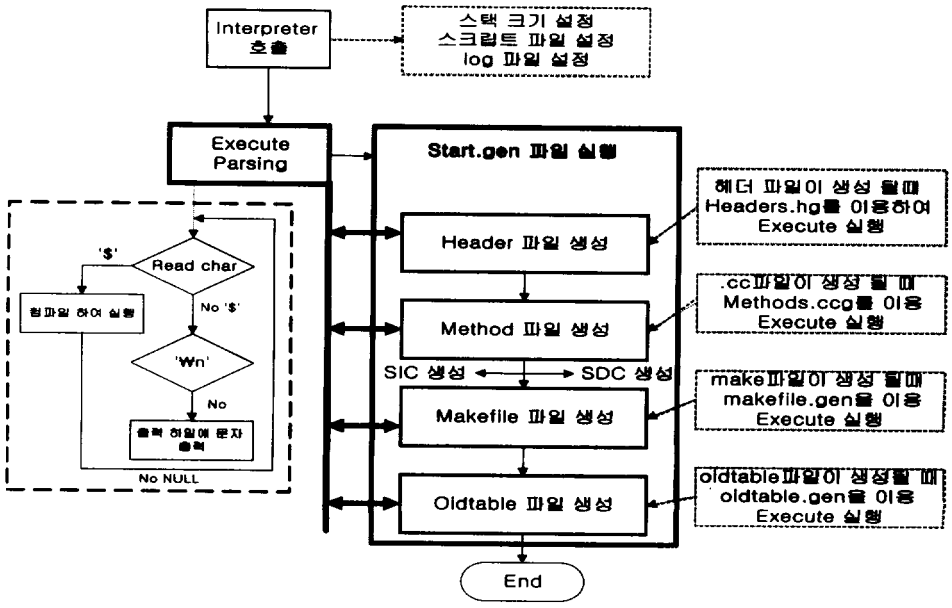
GDMO Compiler는 스캐너와 파서로 구성된다. 스캐너는 어휘 분석기로서 정형화되지 않은 입력 문자열로부터 단말 심볼(토큰)을 가려내는 compiler의 한 과정이다. GDMO Compiler에서는 스캐너로 UNIX에서 제공되는 flex를 사용한다[11]. 파서는 구문 분석기로서 BNF형태의 문법을 명세한 Yacc Specification을 입력받아, 파싱 테이블인 y.output과 드라이브 루틴인 yyparse()를 생성한다. GDMO Compiler에서는 파서로 UNIX에서 제공되는 byacc를 사용한다[11]. GDMO Compiler는 GDMO Source에서 Managed Object에 포함된 conditional package의 정보를 등록 및 다중상속을 지원한다[15]. GDMO Compiler에서는 GDMO Source에 대하여 2가지 출력 코드가 생성된다. SIC와 SDC가 출력 코드이다.

SIC 와 SDC의 코드 생성은 (그림 4)와 같이 GDMO Compiler에서 제공되는 5개의 스크립트 파일을 이용한다. 코드 생성단계는 6단계로 이루어진다. 첫 번째 단계로 스크립트들의 파싱을 위한 스택의 크기를 설정한다. 두 번째 단계로 Start.gen파일을 읽어 파싱한다. 이 스크립트 파일에 의해 SIC, SDC, Makefile, Oidtable 스크립트 파일들이 실행된다. 세 번째 단계로 C++ Class를 얻기 위해 Headers.hg라는 스크립트를 파싱하여 클래스 헤더파일(SIC&SDC)을 생성한다. 네 번째 단계로 C++ 메소드 파일을 얻기 위해 Methods.ccg라는 스크립트를 메소드 파일(SIC&SDC)을 생성한다. 다섯 번째 단계로 생성된 SIC와 SDC 컴파일하기 위해 Makefile를 얻기 위해 Makefile.gen라는 스크립트를 파싱하여 Makefile을 생성한다.

마지막, 여섯 번째 단계로 클래스에 대한 ID를 생성하기 위해 Oidtable.gen라는 스크립트를 파싱하여 Oidtable을 생성한다.

파싱 과정중에는 9개의 함수가 사용된다. 그 함수들은 다음과 같다.

- EXISTFILE(file name)
- LISTCOUNT(variable)
- EXECUTE(script file name, output file name,



(그림 4) 코드 생성 플로우
(Fig. 4) Code generation flow

MO Class)

- DISPLAY(expressions...)
- PRINT(expressions....)
- INCLUDE(file name)
- STRCMP(str1, str2)
- SPC(MO Name, output file)
- SPC2(root MO, output file)

EXISTFILE는 입력된 파일이 존재의 여부를 판별하여 True와 False를 반환한다. LISTCOUNT는 어떤 Object에 대한 리스트들이 존재 시 그 리스트에 노드들의 갯수를 계산하여 반환한다. EXECUTE는 특정의 관리 객체에 대한 Class 메소드 파일과 Header 파일을 생성하기 위한 파싱 과정을 실행한다. DISPLAY는 터미널에 어떤 문자열을 출력한다. PRINT는 출력될 파일에 입력된 값을 저장한다. INCLUDE는 출력파일에 입력 파일을 복사한다. STRCMP는 두 문자를 비교하여 같으면 0을 반환한다. SPC는 관리 객체에 다중 상속기능을 제공하기 위한 것으로 특정 관리 객체에 대한 Parent Class들의 생성자를 출력 파일에 저장한다. SPC2는 다중 상속에서 최상위 클래스의 정보를 저장한다.

4. MODE를 이용한 ATM 스위치 관리 객체의 구현

본 연구에서는 TMN과 ATM의 연동을 위하여 SNMP 기반의 ATM MIB와 GDMO를 연동하는 방법을 사용하였다[16].

4.1 관리 객체와 Real Resource의 통신 기능 관리 객체 구현

Agent process가 처음 실행될 때 초기에 자신이 가져야 할 관리 객체들을 초기화한다. 이때 초기화되는 관리 객체들은 system, managedElement등과 같이 관리 대상 시스템 모델링에 필요한 기본적인 관리 객체들과 초기에 관리될 필요가 있는 Real Resource를 모델링 한 관리 객체등이다. 이외에 다른 관리 객체들은 Agent초기화가 끝나고 관리에 필요에 따라 M-CREATE, M-ACTION요청에 의해 생성된다.

Agent 초기화 시에 managedElement 관리 객체가 SNMP agent와 통신하여 자신이 가지는 Attribute의 값을 할당하기 위해서는 createRR()함수를 이용한다. createRR()은 관리 객체와 Real Resource와의 관계를 정의하는 함수로 하나의 관리 객체 instance가 생길 때마다 호출된다.

managedElement 관리 객체를 구현하기 위해서는 managedElement 관리 객체가 가지는 Attribute와 그 기능이 일치되는 SNMP의 OID를 찾아야 한다. <표 3>은 managedElement 관리 객체가 가지는 4개의 Attribute와 FORE ATM 스위치가 가지는 SNMP MIB의 OID를 서로 매핑 한 것을 나타낸다.

〈표 3〉 managedElement MO의 Attribute와 SNMP MIB 매핑 관계
〈Table 3〉 Mapping managedElement MO's Attributes to SNMP MIB

Attributes	SNMP OID
userLabel	.1.3.6.1.2.1.1.2.0
venderName	.1.3.6.1.2.1.1.1.0
version	.1.3.6.1.4.1.326.2.2.2.1.1.1.0 .1.3.6.1.4.1.326.2.2.2.1.1.2.0
locationName	.1.3.6.1.2.1.1.6.0

(그림 5)와 같이 managedElement 관리 객체가 가지는 Attribute중의 하나인 userLabel의 값을 할당하는 과정은 먼저 SNMP-GET을 통해 userLabel에 매핑된 system.sysObjectID.0을 읽고 이 값을 통해 userLabel에 Attribute값을 설정한다. (그림 10)은 manager가 managedElementR1 관리 객체에 대하여 M-GET 요청시 결과 값을 나타낸다. managedElementR1은 managedElement로부터 유도되어진 관리 객체이다.

```
int managedElement::createRR(AVA*& err, void*, int)
{
        [illegible]
        [illegible] == NOTOK)
            [illegible] NOTOK;

// for the userLabel attribute
        char* user_label = snmp_get(S_userLabel);
        if(!strlen(user_label)) {
            err = new AVA(m_processingFailure, "Can't find userLabel
              attribute");
            return NOTOK;
        }
        else [illegible]->set(user_label);
// for the vendorName attribute
        char* vendor_name = snmp_get(S_vendorName);
        if(!strlen(vendor_name)) {
            err = new AVA(m_processingFailure, "Can't find vendorName
              attribute");
            return NOTOK;
        }
        else [illegible]->set(vendor_name);
```

```
// for the version attribute
        char ver[512], ver_hw[20], ver_sw[20];;
        char* version_hw = snmp_get(S_version_hw);
        strcpy(ver_hw, version_hw);
        char* version_sw = snmp_get(S_version_sw);
        strcpy(ver_sw, version_sw);
        if(!(strlen(version_hw) && strlen(version_sw))) {
            err = new AVA(m_processingFailure, "Can't find version
              attribute");
            return NOTOK;
        }
        else {
            sprintf(ver, "[H/W : %s, S/W : %s]",ver_hw, ver_sw);
            version()->set(ver);
        }

// for the locationName attribute
        char* location_name = snmp_get(S_locationName);
        if(!strlen(location_name)) {
            err = new AVA(m_processingFailure, "Can't find
              locationName attribute");
            return NOTOK;
        }
        else locationName()->set(location_name);

        if(!classInitialised) classInitialised = True;
        return OK;
}
```

(그림 5) managedElement MO의 SDC
(Fig. 5) The SDC of managedElement MO

4.2 PVC 설정 관리 객체 구현

ATM 통신망에 있어서 종단간 VP/VC 연결은 signaling을 통한 사용자 연결과 망 관리 기능에 의한 연결로 나눈다. 망 관리 기능을 통한 VP/VC 연결은 SVC(Switched Virtual Connection)와 PVC(Permanent Virtual Connection)로 나누는데, SVC연결은 B-ISDN signaling의 연결 제어 기능에 의해 연결이 설정 및 해제 되며, PVC는 전용 회선의 개념으로 망 관리 기능에 의해서만 설정 및 해제된다.

FORE ATM 스위치에 하나의 PVC연결을 설정하기 위해서는 열 두 번의 SNMP-SET 요청으로 PVC가 연결 설정된다. **<표 4>**은 입력 VPI값이 0, 입력 VCI값이 300, 출력 VPI값이 0, 출력 VCI값이 500, 입출력 VC에 할당된 대역폭이 1000 cells/sec, 입출력 VC의 셀 지연 변동이 250 msec인 PVC하나를 설정하기 위해 필요한 정보이다. **(그림 6)**은 EntryStatus의 Syntax이다.

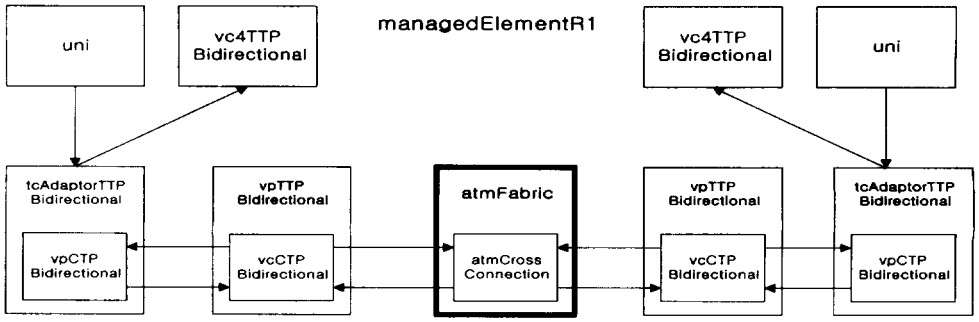
〈표 4〉 PVC 설정 정보
〈Table 4〉 The Information of PVC initialization

OID	Type	Value
.1.3.6.1.4.1.326.2.2.2.1.4.2.1.chanrStatus.1.0.300.2.0.500	I	EntryStatus : 2
.1.3.6.1.4.1.326.2.2.2.1.4.2.1.chanrStatus.2.0.500.1.0.300	I	EntryStatus : 2
.1.3.6.1.4.1.326.2.2.2.1.4.2.1.chanStatus.1.0.300	I	EntryStatus : 2
.1.3.6.1.4.1.326.2.2.2.1.4.2.1.chanStatus.2.0.500	I	EntryStatus : 2
.1.3.6.1.4.1.326.2.2.2.1.4.2.1.chanAllocBandwidth.1.0.300	I	INTEGER : 1000
.1.3.6.1.4.1.326.2.2.2.1.4.2.1.chanAllocBandwidth.1.0.500	I	INTEGER : 1000
.1.3.6.1.4.1.326.2.2.2.1.4.2.1.chanCDV.1.0.300	I	INTEGER : 250
.1.3.6.1.4.1.326.2.2.2.1.4.2.1.chanCDV.2.0.500	I	INTEGER : 250
.1.3.6.1.4.1.326.2.2.2.1.4.2.1.chanrStatus.1.0.300.2.0.500	I	EntryStatus : 1
.1.3.6.1.4.1.326.2.2.2.1.4.2.1.chanrStatus.2.0.500.1.0.300	I	EntryStatus : 1
.1.3.6.1.4.1.326.2.2.2.1.4.2.1.chanStatus1.0.300	I	EntryStatus : 1
.1.3.6.1.4.1.326.2.2.2.1.4.2.1.chanStatus.2.0.500	I	EntryStatus : 1

```
EntryStatus ::= INTEGER
{
        valid(1),
        createRequest(2),
        underCreation(3),
        invalid(4)
}
```

(그림 6) EntryStatus의 Syntax
(Fig. 6) The Syntax of EntryStatus

아래에 나타난 것은 EntryStatus의 syntax type이다. chanrStatus, chanStatus OID에 대해서 EntryStatus를 '2'에서 '1'로 변경한다. manager가 chanrStatus, chanStatus를 처음 createRequest로 설정하면 agent는



(그림 7) ATM 연결 관리에 사용되는 MOs
(Fig. 7) Managed Objects being used ATM connection management

channelEntry, channelRout-eEntry의 상태를 '3'으로 변경할 뿐 실제 하나의 PVC가 활성화되는 것은 아니다. 그러므로 SNMP-SET을 통해 이 EntryStatus를 '3'에서 '1'로 바꾸어야 PVC가 완전히 연결설정 된다.

ATM 연결에 관련된 관리 객체들은 I.751에 정의된 atmFabric, tcAdaptorTTPBidirectional, vpTTPBidirectional 등이 있다**[14]**. 이들 관리 객체들의 포함 관계는 **(그림 7)**에 나타난다. agent 초기화 시 FORE ATM 스위치에서 이용 가능한 port의 개수만큼 tcAdaptorTTPBidirectional를 생성하고 이 관리 객체가 생성될 때 uni 관리 객체가 자동으로 생성된다. 그리고 나머지 관리 객체들은 M-CREATE요청에 의해 생성되거나 하나의 VP/VC 연결을 설정하기 위해서 M-ACTION(connect)을 요청할 때 생성된다. 이 부분을 SDC에서 코딩한다.

```
{ fromTermination: {
(interfaceId=networkId=YNUATM@managedElementId=ASX200@uniId=1A1)
(vpi=0)(vci=300)(ingressCDVTolerancePCR=500)(ingressPeakCellRate=1000)}
toTermination:{(interfaceId=networkId=YNUATM@managedElementId=ASX
200@uniId=1A2)(vpi=0)(vci=500)(egressCDVTolerancePCR=400)(ingressPeak
CellRate=1000)}
```

(그림 8) M-ACTION(connect)의 파라미터
(Fig. 8) A parameter of M-ACTION(connect)

(그림 8)과 같이 M-ACTION요청시 agent는 관련된 관리 객체를 생성하고, gateway를 통해 FORE ATM 스위치에 실제 PVC를 생성한다. 그 과정은 아래와 같다.

① Manager가 action의 파라미터로 보낸 fromTermination과 toTermination의 interfaceId(uni)를 통해 tcAdaptorTTPBidirectional 관리 객체를 찾고 각각을 입력 포트와 출력 포트를 지정한다.

② P trail의 양 종단을 나타내는 vpTTPBidirectional 관리 객체를 2개 생성한다.

③ VP connection의 양 종단을 나타내는 crossConnectionObjectPointer attribute가 관련된 VP trail(vpTTPBidirectional 관리 객체)를 가리키도록 지정한다.

④ vpTTPBidirectional 관리 객체가 가지는 attribute인 upstreamConnectivityPointer와 downstreamConnectivityPointer를 vpCTPBidirectional 관리 객체를 가리 키도록 지정한다.

⑤ Manager가 보낸 파라미터 중에서 vpi, vci, traffic parameter, QoS class등을 이용해 vcCTPBidirectional 관리 객체를 두개 생성한다.

⑥ 연결 종단을 나타내는 관리 객체들이 모두 생성되었으므로 이 정보들을 이용 해 SNMP-SET을 통해 PVC 연결 요청한다.

⑦ 하나의 PVC가 정상적으로 생성되면 종단간 ATM VP/VC crossconnection을 나타내는 atmCrossConnection 관리 객체를 하나 생성한다.

⑧ Manager로 보낼 reply syntax를 만든다.

5. 실행 결과

본 논문에서는 GDMO Source로 Fore ATM GDMO Source(X.721, M.3100, I.751)**[12,13,14]**를 입력하여 GDMO Compiler를 실행하였고 각각의 관리 객체에 대하여 Window환경에서 사용자가 쉽게 코딩을 할 수 있도록 하는 MODE를 이용하여 **(그림 9)**와 같이

```
int atmFabric::action(int actionId, int classLevel, void* actionArg, void*&
actionReply, Bool& freeFlag, AVA*& err, Bool checkOnly, int)
{
    if (classLevel != _level)
        return Top::action(actionId, classLevel, actionArg,
                actionReply, freeFlag, err, checkOnly);

    if (checkOnly) return OK;
      switch (actionId) {
      case I_connect :
                infoStr =
          _classInfo->getActionInfoTemplate(I_connect)->print(actionArg);
                replyStr = vcConnect(infoStr);
                actionReply =
                _classInfo->getActionReplyTemplate(I_connect)->
                    parse(replyStr);
                break;
    default :
                break;
    }
    return OK;
}
```

(그림 9) atmFabric MO의 action SDC
(Fig. 9) The SDC for atmFabric's action

```
Get Result - id 0x000001, class managedElementR1, instance{networkId=
YNUATM@managedElementId=ASX200}, time = 19980725220047
objectClass : managedElementR1
nameBinding : managedElement-network
managedElementId : ASX200
systemTitle : FORE ATM Switch in Ynu univ.
administrativeState : unlocked
operationalState : enabled
usageState : idle
userLabel : .iso.org.dod.internet.private.enterprises.fore.systems.atmSwitch
vendorName : FORE Systems ASX-200
version : [H/W : 0, S/W : 262400]
locationName : Computing-Services-Wiring-Plant
1 replies received
```

(그림 10) managedElementR1의 M-GET결과
(Fig. 10) A Result of managedElementR1's M-GET

atmFabric 관리 객체에 대하여 M-ACTION(connect)

의 SDC를 사용자가 생성한다.

(그림 11)은 PVC가 정상적으로 생성되었을 때의 결과를 나타낸다.

```
Action Result - id 0x000001, class atmFabric instance { networkId=
YNUATM@managedElementId=ASX200@atmFabricId=ATMFabric} time
1998072504409
Result : { PVC setting ok! IPORT = 1A1 : OPORT=1A2 : IVPI=0 : IVCI
= 300 : OVPI = 0 : OVCI = 500 : ICDV = 250msec : OCDV = 250msec :
IPCR = 1000 cells/sec : OPCR = 1000 cells/sec }

1 succesful actions
```

(그림 11) M-ACTION(connect) 결과
(Fig. 11) A Result of M-ACTION(connect)

6. 결 론

본 논문에서는 ATM/B-ISDN 망관리 기술을 이용하여 TMN체계의 GDMO 플랫폼을 구축하기 위한 관리 객체생성 시 필요한 GUI 기반의 관리 객체 개발 환경(MODE)을 구현하였다. MODE를 이용하여 ATM 스위치 관리 객체들을 개발하였다. 실험결과 MODE는 전문가나 일반 사용자가 관리 객체 생성 시에 GDMO에 표현된 9개의 템플릿들에 대한 정보관리, CMIP-Request 메소드에 대한 SDC의 기본 골격 제시와 MO들의 상속관계를 코드에 반영함으로써 MODE가 유용함을 증명한다. 그러나 ATM/B-ISDN 망 관리에 대한 권고안의 방대한 양으로 SDC 지원이 제한되었다. 따라서, ATM/B-ISDN 등과 같은 특정 망 관리에 대한 권고안에 나타난 관리 객체들의 행동 특성을 데이타베이스화 하여 SDC를 생성할 수 있을 것이다. 또한, Window기반의 환경때문에 다른 OS에서 사용할 수 없으므로 Web기반의 GDMO Compiler를 지원하여 유동성있는 애플리케이션으로 구축할 것이다.

참 고 문 헌

[1] (주) 토미스, 영남대학교 정보통신 연구소, ATM/-ISDN 통신망 관리를 위한 TMN체계의 GDMO Agent Platform개발, 연차보고서, 정보통신부.

[2] The OSI Management Information Service OSIMIS) User's Manual, Version 1.0 for system version 3.0, 1993.

[3] UCL, "A guide to implementing Manged Objects using the GMS," OSIMIS technical paper, 1993.

[4] UCL, "OSIMIS GDMO Compiler user manual," OSIMIS technical paper, 1993.

[5] George Pavlou, Graham Knight and Simon Walton, "Experience of Implementing OSI Management Facilities," OSIMIS technical paper, 1993.

[6] "GDMO Compiler Programmer's Reference Manual Version 1.2.2," DSET Corporation.

[7] "TMN/C++ Application Programming Interface," X/Open, 1996.

[8] 김민석, 김성근, 최재영, 송후봉, "망 관리 원시코드 자동생성을 위한 GDMO개발환경의 설계 및 구현", 한국정보과학회 논문집, KISS 97, pp.61-64, 1997.

[9] 현창문, 김행곤, 한덕수, "에이전트 플랫폼을 위한 GDMO 컴파일러 설계 및 구현에 관한 연구", 한국정보처리학회, KIPS 97, pp.1077-1080, 1997.

[10] Soukouti, Nader Bull S.A., "Automatic translation of OSI managed object classes to C/sup ++/ classes," IEEE Journal Vol.12, No.6, IEEE Communications Society, pp.1011-1019, 1994.

[11] UNIX를 이용한 컴파일러 설계, 김상욱 저, 홍릉출판사.

[12] CCITT Recommendation X.700-736, 1992.

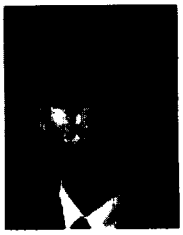
[13] ITU-T Recommendation M.3100, "Generic Network Information Model," 1992.

[14] ITU-T Recommendation I.751, "Asynchronous Transfer Mode Management of The Network Element View," 1996.

[15] 강원석, 김호철, 김기형, 김영탁, "ATM/B-ISDN 망관리를 위한 GUI기반의 관리 객체 생성기", 98 통신망운용관리 학술대회, pp.235-244, 1998.

[16] 구수용, 신해준, 김언우, 김영탁, "TMN 체계에서의 ATM/B-ISDN 통합 관리를 위한 GDMO/CMIP과 SNMP관리 기능 연동 방안", 98 통신망운용관리 학술대회, pp.174-185, 1998.

[17] (주) 토미스, 영남대학교 정보통신 연구소, ATM/B-ISDN 통신망 관리를 위한 TMN체계의 GDMO Agent Platform개발, 최종보고서, 정보통신부.



강 원 석

e-mail : wskang@white.ce.yeungnam.ac.kr

1998년 2월 영남대학교 컴퓨터공학 학사

1998년 3월~현재 영남대학교 컴퓨터공학 석사 과정

관심분야 : 망관리, 시뮬레이션, 분산병렬 처리

김 기 형

e-mail : kkim@yuncc.yeungnam.ac.kr

1990년 2월 한양대학교 전자통신공학 학사

1992년 2월 한국과학기술원 전기 및 전자공학과 석사

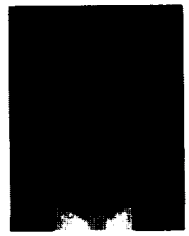
1996년 8월 한국과학기술원 전기 및 전자공학과 박사

1996년 8월~1997년 2월 한국과학기술원 위촉연구원

1997년 3월~현재 영남대학교 컴퓨터공학과 전임강사

1998년 3월~현재 한국전자통신연구소 수퍼컴퓨터센터 겸임연구원

관심분야 : 네트워크 기반 컴퓨팅, 멀티미디어 운영체제, 시스템 모델링 및 성능평가, ATM 네트워크 관리



김 영 탁

e-mail : ytkim@yuncc.yeungnam.ac.kr

1984년 2월 영남대학교 전자공학과 학사

1986년 2월 한국과학기술원 전기 및 전자공학과 석사

1990년 2월 한국과학기술원 전기 및 전자공학과 박사

1990년 3월~1994년 8월 한국통신 통신망 연구소 전송망 연구실장/선임연구원

1994년 9월~현재 영남대학교 정보통신공학과 조교수

관심분야 : ATM/B-ISDN, TMN 체계의 망운용관리, 광대역 멀티미디어 정보통신 유무선 정보통신망 통합, 정보통신공학