객체 지향 기법의 신호망 관리 방식

김 태 형[†] · 고 병 도[††] · 류 재 철[†††]

요 약

효율적인 통합 통신망 관리의 필요성이 대두됨에 따라 ITU-T 및 ISO 등 여러 국제 기구에서는 망 관리를 위한 표준안 제정에 박차를 가하고 있으며, 그 중 대표적인 것이 ITU-TS의 TMN(Telecommunication Management Network)이다. 본 고에서는 TMN 기반의 신호망 운용 관리를 위하여, 현재 개발되어 사용되고 있는 SIGNOS(SIGnalling Network Operating System) 시스템을 TMN 개념을 수용할 수 있는 진보된 신호망 운용 관리 시스템(T_SIGNOS)으로 설계하여 제안한다. 또한, 신호망에서 필요한 관리객체(Managed Objects)에 대하여 알아보고, 이를 저장하기 위한 MIB(Management Information Base)를 구현한다.

Signalling Network Management Using Object-Oriented Methods

Tae-Hyoung Kim[†] · Beung-Do Go[††] · Jae-Cheol Ryou[†††]

ABSTRACT

As the efficient management of the integrated network has been an important issue, the ITU-TS develops Telecommunication Management Network(TMN) and suggests standards to associate signalling network management with TMN. In this paper, we propose a method to manage signalling network, which is based on the standards. We suggest the TMN version SIGNOS(T_SIGNOS) structure as a new Signalling Network OS (SIGNOS) structure and define the managed object classes in signalling network, and develop the Managed Information Base(MIB) to store the managed information of the object classes.

1. 서 론

통신망의 종류가 다양해지고 통신망 구성에 사용되는 교환기 및 전송 설비의 종류가 급격히 증가함에 따라 통신망 운용 관리 시스템(Network management system)의 개발이 활발히 진행되고 있다. 통신망 운용 관리에서 중요한 문제점 중의 하나는 상이한 네트워크 시스템들을 효율적으로 통합하여 통합망 운용자에게 일관성 있는 관리 기능을 제공하여 주는 것이다. 효율적인 통합을 위해서는 망과 망 사이의 통신 프로토콜 변환 문제, 관리 정보를 일관성 있고 무결하게 저장하고 교환하는 문제, 관리 기능 및 응용상의 차이점, 비획일적인 사용자 인터페이스 문제 및 시스템 구조상의 차이점 등을 해결해야 한다.

본 고에서는 ITU-T TMN에 기반을 둔 신호망 관리 방식을 연구하고, 국내 구축 신호망 관리 시스템의 SIGNOS와 신호 중계 교환기(SMX)간의 네트워크 관리 서비스를 TMN을 근간으로 설계한다.

신호망은 지능망과 종합 정보 통신망의 근간이 되는 망으로 고 신뢰도를 가진 신호 채널을 이용하여 망 설비 사이의 신호 정보를 전달하는 역할을 담당하

* 이 논문은 1994년도 한국 학술 진흥 재단의 학술 연구 조성비 지원에 의하여 연구되었음.
† 준 회 원:충남대학교 컴퓨터과학과 박사과정
†† 준 회 원:한국전자통신연구소 광대역서비스연구실장
††† 종신회원:충남대학교 컴퓨터과학과 조교수
논문접수:1996년 1월 27일, 심사완료:1996년 3월 26일

고 있다. 즉, 전화 가입자 정보에 대하여 각각의 가입자의 호 설정 및 취소, 기타 부가가치 서비스 등과 같은 각 독립된 망에서 공통적으로 발생되는 신호 메시지를 고 신뢰도를 가지고 전송한다. 이러한 신호망은 통신망에서 중요한 신호 정보를 전달하기 때문에 고성능, 고 안정성, 고 신뢰도가 요구되며, 이를 위해 신호망의 운용 상태가 항상 감시되어야 하고 신호망 차원에서 관리가 요구된다[1~4]. 이러한 운용 관리 기능을 수행하기 위해 신호망 관리 시스템(SIGNOS)이 개발 되었다. 그러나, SIGNOS 개발 당시 ITU-TS 권고안 중 No.7 신호망의 OMAP 프로토콜 제정이 초기 정립 단계였기 때문에 이에 대한 수용이 어려웠다. 이에 본 고에서는 신호망 관리를 목적으로 개발된 신호망 관리 시스템을 TMN 접속을 통하여 진화된 신호망 관리 구조를 설계하고, 신호망 내에서 관리되어야 하는 관리 객체를 저장하기 위한 MIB를 구현한다.

이를 위해 2장에서는 TMN의 관리 기능과, 기능 구조(Functional Architecture), 정보 구조(Information Architecture) 및 물리 구조(Physical Architecture)에 대하여 고찰하고, 이어서 3장에서는 TMN이 근간으로 하는 ISO의 OSI 관리 기능 분야, 관리 구성, 공통 관리 정보 서비스(CMIS) 와 공통 관리 정보 규약(CMIP)에 대하여 고찰한다. 이를 토대로 4장에서는 TMN 요구 사항을 수용한 새로운 신호망 관리 구조와 T_SIGNOS 시스템 구조를 제안한다. 5장에서는 신호망 관리의 중요한 요소가 되는 관리 정보 모델을 제시하며 6장에서는 5장에서 제시한 관리 정보 모델을 중심으로 MIB를 구현한다.

2. ITU-TS TMN에 대한 고찰

TMN의 목적은 전송 장치의 관리를 위한 기본 구조를 제공하고 관리를 위한 모델과 표준 인터페이스를 제공함으로서 방대한 규모의 전기 통신망의 다양한 통신 요소를 효율적으로 관리하는데 있다[5~11]. 이를 위한 TMN의 관리 기능에는 성능 관리(Performance management), 장애 관리 (Fault management), 구성 관리(Configuration management), 요금 관리(Account management), 보안 관리(Security management)등의 5가지가 있다.

TMN의 일반적인 구조는 TMN을 계획하고 고안할 때 다음의 3가지 측면, 즉, 기능 구조(The TMN Functional Architecture), 정보 구조(The TMN Information Architecture) 및 물리 구조(The TMN Physical Architecture)에 대하여 고려 한다. 이들 구조에 대하여 살펴보면 다음과 같다.

TMN의 기능 구조는 다수의 기능 블록(OSF, DCF, NEF, WSF, MF, QAF)으로 구성되어 있으며 이들은 네트워크 관리를 위한 일반적인 기능을 제공한다. 이들 각 기능 블록들은 분리된 참조점(reference point: q, f, x, g, m)에 의하여 관리 정보를 교환한다.

TMN 정보 구조는 객체 지향 접근법을 사용하고 관리자/피관리자(Manager/Agent) 개념을 사용한다. 관리자와 피관리자는 망 관리를 위한 정보를 주고 받는데 이를 관리 정보라 한다. 관리 정보는 관리 정보 모델과 관리 정보 교환의 두 가지 측면에서 고려된다. 관리 정보 모델은 객체 지향적 모델링 기법을 이용하여 관리 객체의 특성을 속성, 통지 사항, 동작, 그리고 행위 형태로 요약 추출한 것을 말하고, 객체화된 정보를 관리자와 피관리자 간에 주고 받기 위해 관리 행위에 적합한 프로토콜을 사용하여 관리 정보 교환이 이루어 진다.

TMN 물리 구조는 TMN building 블록들로 구성되며, 이들을 통해 TMN 기능들이 구현된다. TMN 물리 구조에서 OS, MD, QA, DCN, NE 및 WS는 TMN 기능 블록과 관련되어 각각 OSF, MF, QAF, DCF, NEF 및 WSF 기능을 수행하는 building 블록들이고, 인터페이스 Qx, Q3, F, G, X는 TMN 참조점과 관련되어 각각 q, f, g, x 참조점에 해당한다.

TMN 권고안들은 대부분 ISO의 OSI 망 관리를 위한 표준안을 수용하고 있으므로, 다음 장에서 이를 고찰한다.

3. OSI 시스템 관리에 대한 고찰

ISO는 네트워크 관리를 위해서 ISO 7498-4 문서를 시작으로 많은 표준 문서를 제정하였다. ITU-TS는 이러한 표준 문서에 X.700 시리즈 번호를 부여하고 있으며 이들 표준 문서들은 다음과 같이 크게 5가지로 구별할 수 있다[12~15].

첫째, OSI 관리 골격 및 개관(OSI Management

Framework and Overview): ISO 7498-4에서는 기본적인 관리 개념을 소개하고, ISO 10040은 다른 표준 문서들에 대한 간단한 설명을 하고 있다. 둘째, CMIS/CMIP: OSI 관리 서비스들을 제공하는 공통 관리 정보 서비스와 이에 관련된 정보 교환을 지원해 주는 공통 관리 정보 규약이 정의되어 있다. 셋째, 시스템 관리 기능: OSI 시스템 관리를 수행하는데 필요한 기능들을 정의하고 있다. 넷째, 관리 정보 모델: OSI 환경에서는 객체 지향 기법을 사용하여 네트워크 자원을 관리하고 있다. 이를 위해 객체에 대한 개념과 객체들로 구성된 관리 정보 베이스(MIB)등이 정의되어 있다. 다섯째, 계층 관리: OSI layer들에 관련된 관리 정보, 서비스, 기능들이 정의되어 있다.

3.1 OSI 관리 기능 분야 및 관리 구성

OSI 관리 기능 분야는 성능 관리, 장애 관리, 구성 관리, 요금 관리 및 보안 관리로 구성되는데 이와 같은 관리 기능 분야들은 모든 통신 네트워크 요소에 공통적으로 적용되는 기능 분야들이다. 이들 관리 기능 분야들을 구현하기 위해서 OSI는 시스템 관리 기능(System Management Function)들을 정의하고 있으며, 시스템 관리 기능을 수행하기 위해서는 다른 시스템과의 통신이 필요한데, 이를 위해 CMISE 서비스를 사용하게 된다.

이들 서비스들 간의 관계를 나타내는 OSI의 구조적 모델에서 중요한 요소는 다음의 4가지가 있다. 첫째, 시스템 관리 응용 프로세스(SMAP): 하나의 시스템 내에서 네트워크 관리 기능 수행을 전담하는 요소로서 다른 시스템의 SMAP와 조정 역할을 한다. 둘째, 시스템 관리 응용체(SMAE): 네트워크 관리 센터 및 그 밖의 시스템과의 통신을 전담하는 요소이다. 셋째, 계층 관리 객체(LME): 네트워크 관리에 필요한 정보를 저장하고 있는 요소이다. 넷째, 관리 정보 베이스(MIB): 네트워크 관리에 필요한 정보를 저장하고 있는 요소이다.

3.2 공통 관리 정보 서비스와 공통 관리 정보 규약

공통 관리 정보 서비스(Common Management Information Service: CMIS)는 OSI 시스템 관리를 위해 제공되는 서비스로, 관리 프로세스가 이를 이용하여 OSI 관리를 수행한다.

이들 서비스는 연계 서비스와 관리 정보 전달 서비스로 나눌 수 있다. 연계 서비스(Association Service)는 CMIS 사용자가 다른 시스템의 CMIS 사용자와 연계를 설정하기 위한 것으로 서비스 활동의 첫 단계이다. 이들은 ISO 8649에 정의 되어 있다. 관리 정보 전달 서비스에는 관리 통지 서비스와 관리 동작 서비스의 두 가지가 있다.

관리 통지 서비스는 관리 객체로부터 발생된 사건을 상대방 관리자에게 통지하는 서비스로 M-EVENT-REPORT가 있으며, 관리 동작 서비스는 관리 객체에게 관리 동작을 수행 시키기 위한 서비스로 M-GET, M-CANCEL-GET, M-SET, M-ACTION, M-CREATE 그리고 M-DELETE 등이 있다.

이들 서비스는 공통 관리 규약(CMIP)을 통하여 이루어 진다. 즉, 공통 관리 정보 규약 기계(Common Management Information Protocol Machine: CMIPM)는 CMISE-서비스-사용자에 의해 제기된 CMISE요구(Request)와 응답(Response) 서비스 프리미티브에 대한 PDU들을 전송하는 역할을 담당한다. 또한, 상대방 시스템의 CMIPM에 의해 보내진 PDU를 받아들이고 CMIS 지시(Indication) 및 확인(Confirm) 서비스 프리미티브 처리를 위해 CMISE-서비스-사용자에게 PDU를 전달한다. 그러나, 수신된 PDU에서 오류가 발견되면, CMISE-서비스-사용자에게 전달되지 않고 상대방으로 반송된다. 위와 같은 작업을 수행하기 위해 CMIP는 ISO 9072에서 정의한 원격 동작 서비스 요소(ROSE)의 RO-INVOKE, RO-RESULT, RO-ERROR 및 RO-REJECT 서비스들을 사용한다.

4. TMN 신호망 관리

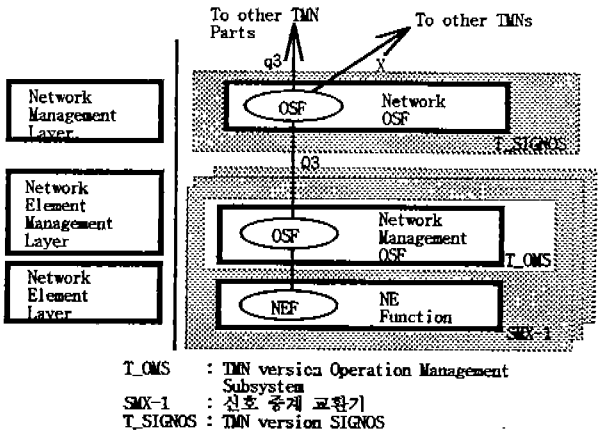
본 장에서는 OSI의 시스템 관리와 ITU-TS의 TMN 권고안을 중심으로 국내 신호망 관리에 대한 네트워크 관리 구조와 TMN 수용이 가능한 새로운 신호망 관리 시스템인 T_SIGNOS 시스템 구조에 대하여 모델을 제시 한다.

4.1 기능 참조 모델

신호망 관리를 위한 OMAP은 OSI의 시스템 관리 권고안인 X.700 시리즈와 TMN 권고안인 M.3000의 관리 기본 원칙을 근간으로 정의되며 이에 따른 신호

망 관리를 위한 기본적 요구 사항은 다음과 같다.

첫째, 네트워크의 운용 관리를 위하여 TMN 인터페이스가 제공되어야 한다. 둘째, 통신망 통합 관리를 수행할 수 있도록 다른 TMN 망과 통일된 인터페이스를 제공하여야 한다. 셋째, 신호망 프로토콜 자체에 정의된 기존 관리 기능과 TMN은 합병이 가능하고 이의 확장성이 보장되어야 한다. 이들 기본 원칙과 요구 사항에 따른 신호망 관리 기능 참조 모델에 대한 물리적 실현 방안은 현재 ETRI의 지능망 연구부에서 개발된 국내 신호망 형상을 고려해 볼 때 신호



(그림 4.1) 기능 참조 모델에 대한 국내구축 신호망의 TMN 매핑

(Fig. 4.1) The TMN mapping of the Signalling network for FRM

망 기능 참조 모델은 그림 4.1과 같이 매핑 된다. 즉, T_SIGNOS에는 네트워크 관리 기능을 수행하는 OSF가 단 하나 존재하고, 네트워크 구성 요소(NE:Network Element)에 대한 관리 기능을 수행하는 T_OMS는 SMX-1에 존재한다. 네트워크 구성 요소(NE) 기능을 수행하는 NEF는 SMX-1마다 하나씩 존재한다. 또한, T_SIGNOS와 T_OMS는 Q3 인터페이스를 사용하고, T_OMS의 OSF와 NEF가 서로 다른 SMX-1에 존재할 경우 이들은 Q3 인터페이스를 사용한다.

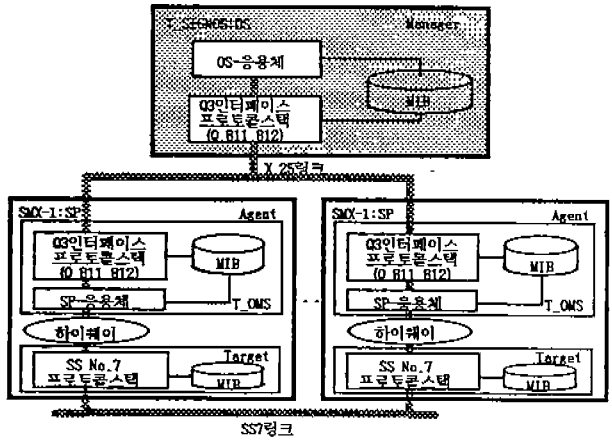
4.2 TMN 신호망 관리 구조와 T_SIGNOS 시스템 구조

TMN 신호망 관리 구조는 기본적으로 OSI 시스템 관리 모델인 관리 객체 모델을 채용하며 현재 구축된 국내 신호망 형상을 고려해 볼 때 그림 4.2와 같이 설계된다.

T_SIGNOS는 TMN 버전의 네트워크 관리 OS로 OSF 기능을 수행하는 신호망 관리 시스템(SIGNOS)으로서 관리자 역할 기능을 수행 한다. T_OMS는 NEMF 기능을 수행하는 TMN 버전의 OMS로서 SMX-1 별로 하나씩 존재 하며 T_SIGNOS와는 피관리자 관계로 동작하고, SMX-1의 타겟 시스템과는 관리자 관계를 가진다.

T_SIGNOS와 T_OMS간 통신은 Q.811, Q.812에 권고된 Q3 인터페이스를 사용하는데 Q.811은 OSI의 프로토콜 스택을 기준으로 하위 계층 프로토콜에 대하여 정의되어 있으며 Q.812에서는 계층 4 이상의 상위 계층 프로토콜에 대하여 정의하고 있다. T_SIGNOS와 T_OMS간 통신 프로토콜 구조는 그림 4.3과 같다.

SMX-1 시스템 중 타겟 시스템 부분은 T_OMS와는 피관리자 관계를 가지며 각 계층의 No. 7 신호망



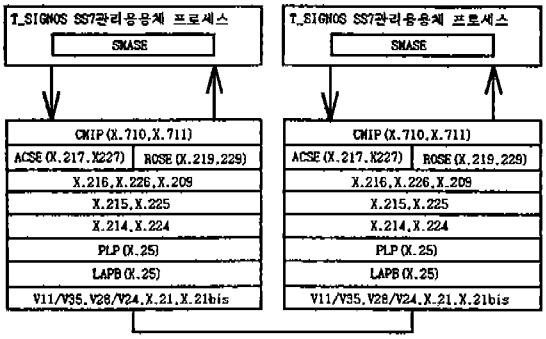
(그림 4.2) TMN 신호망 관리 구조

(Fig. 4.2) TMN Signalling network management structure

프로토콜 중 프로토콜 내의 관리 기능이 직접 수행되는 부분이다.

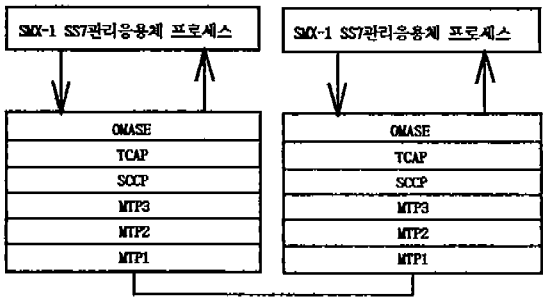
T_OMS와 타겟 시스템과의 q3 참조점은 동일한 물리적 신호점에 존재 함으로 구현 의존적이며, SMX-1

신호 중계 교환기에는 물리적 매체로 고속 하이웨이가 실장 된다.



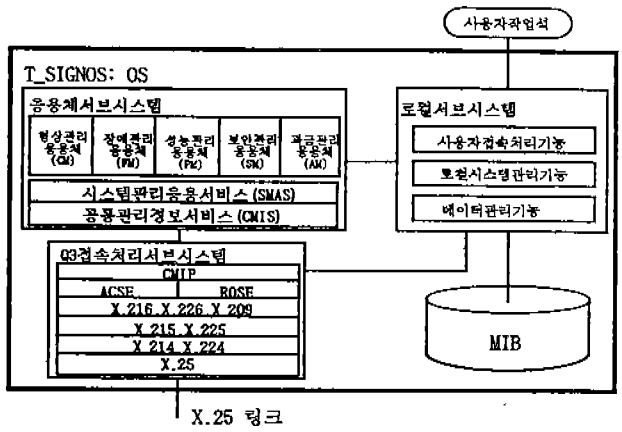
(그림 4.3) T_SIGNOS와 T_OMS간 통신 프로토콜 구조
(Fig. 4.3) The protocol structure between T_SIGNOS and T_OMS

신호망 관리를 위한 신호점과 신호점 간에는 통신 프로토콜 스택은 OMASE가 추가되어 그림 4.4의 형태를 갖는다.



(그림 4.4) SP간(SMX-1 <-> SMX-1) 통신 프로토콜 구조
(Fig. 4.4) The protocol structure between SPs

앞에서 제시한 TMN 신호망 관리 구조(그림 4.2)에서 SIGNOS의 TMN 버전 시스템 구조는 그림 4.5와 같다. T_SIGNOS는 논리 구조상 응용체 서브 시스템, 로컬 처리 서브 시스템, Q3 접속 처리 서브 시스템의 크게 3개의 서브 시스템으로 구성한다.



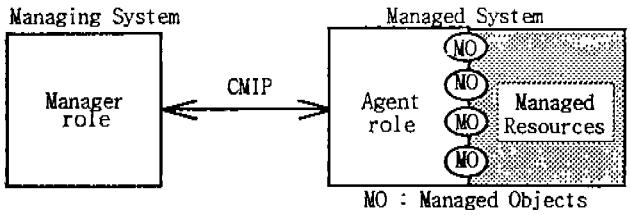
(그림 4.5) TMN version SIGNOS 시스템 구조
(Fig. 4.5) The system structure of TMN version SIGNOS

응용체 서브 시스템은 TMN에서의 5개 기능 영역을 근간으로 구성되는 응용체 프로세스 집합체 이며 각 응용체 프로세스는 서비스 관리 응용 서비스(SMAS)와 공통 관리 정보 서비스(CMIS)를 통하여 Q3 접속 처리 서브 시스템의 공통 관리 정보 프로토콜(CMIP)과 직접 접속 한다. Q3 접속 처리 서브 시스템은 피관리자와의 Q3 인터페이스를 위한 처리 기능을 수행하는 서브 시스템으로 응용체 서브 시스템의 응용체와는 CMIP의 프리미티브를 통하여 서로 접속한다. 로컬 시스템은 T_SIGNOS의 자체 기능으로 사용자 접속 처리 기능, 로컬 시스템 관리 기능, 관리 정보 및 관련 데이터 베이스를 관리하는 데이터 관리 기능을 갖는다.

5. TMN 신호망 관리 정보 모델링

이 장에서는 신호망 관리의 중요한 요소가 되는 관리 정보 모델을 제시한다. 그림 5.1에서 볼 수 있듯이 관리자와 피관리자 사이의 관리 정보 교환은 피관리자 시스템 내에 있는 관리 대상인 여러 자원들 간의 내부 통신에 의해 이루어 질 수 있으며 이러한 관리 자원들은 관리 객체에 의해 표현된다.

ISO에서는 OSI 시스템 관리 정보 모델을 객체 지향 모델링(Object-Oriented Modeling) 기법을 이용하



(그림 5.1) 관리 정보 교환
(Fig. 5.1) Management Information exchange

여 권고안 X.720과 X.721에 제시하였고, 본 논문에서도 TMN 신호망 관리 정보 모델의 제시를 위해 ISO에서 제시한 모델을 근간으로 한다.

5.1 신호 중계 교환기의 관리 기능

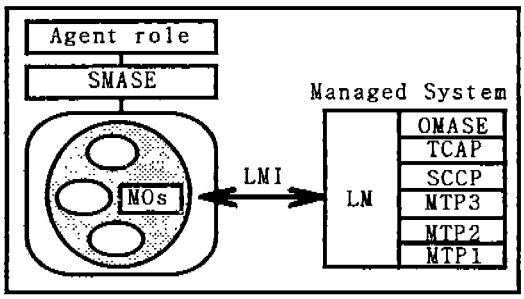
신호 중계 교환기는 No.7 신호 방식에서의 레벨 1에서 3까지가 메시지 전송부(MTP:Message Transfer Part)로서 신뢰도가 높고 상황에 따라 여유 있게 루트를 선택하여 신속하게 신호 메시지를 상대방에게 보내는 기능을 가지고 있다. MTP의 상위 레벨을 총칭해서 레벨 4로 불리며 그 중 제일 하위에 위치하는 것이 신호 접속 제어부(SCCP:Signalling Connection Control Part)이다. SCCP는 신호망 내에 있어서 OSI 모델의 Layer 3에 해당하는 데이터 전송 기능을 제공한다.

레벨 4중에서 호 제어에 직접 관계하지 않고 SCCP를 이용해서 하나의 신호 중계 교환기와 다른 신호 중계 교환기, 또는 신호 중계 교환기와 서비스 제어 노드 간의 보수/운용 정보나 서비스 처리 정보 전송을 담당하는 곳이 응용부이며 그 중 운용 정보와 서비스 정보를 적절히 처리/관리 하는 기능을 제공하는 것이 처리 기능 응용부(TCAP:Transaction Capabilities Application Part)이다.

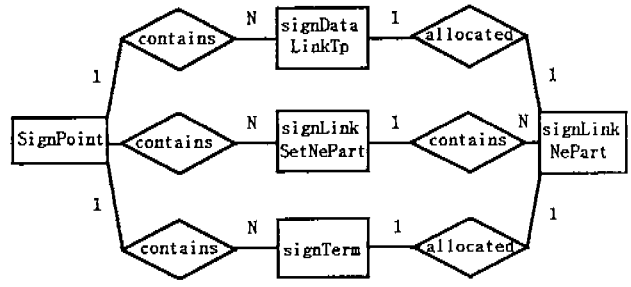
위와 같은 신호 중계 교환기의 각 레벨과 관련한 기능은 관리 목적을 위해서 관리 객체로써 표현 하며, 각 관리 객체는 각 레벨과 관련한 정보를 내부 통신 또는 계층 관리 인터페이스(Layer Management Interface)를 통해 수집/저장 한다. 그림 5.2는 신호중계 교환기 내의 관리 객체와 각 레벨과의 관계를 보여 준다.

5.2 관리 객체 클래스의 정의

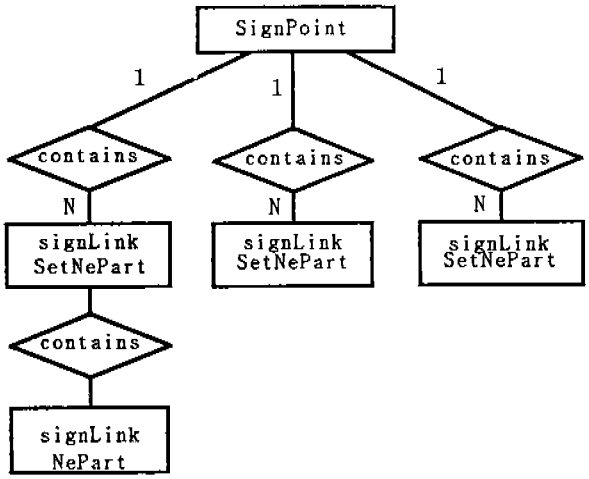
이 절에서는 앞 절에서 언급된 신호 중계 교환기 내에 관리 되어야 할 기능들에 대한 관리 객체 클래스를 정의한다. 각 관리 객체 클래스들은 신호망 내에서 신호 중계 교환기로써의 역할에 대한 No.7 기능 관점과 신호망 관리를 위해 고려되어야 할 관리 관점을 중심으로 정의 한다. 각 클래스의 형식적인 정의(Formal Description)는 권고안 X.722의 GDMO(Guidelines for the Definition of Managed Objects)에서 정의 된 템플릿(Template)들을 이용한다. 또한, 각 관리 객체 클래스의 정의는 OMAP 권고안 Q.751에서 정의하고 있는 관리 객체 클래스들에 근거한다.



(그림 5.2) 관리 객체와 각 레벨과의 관계
(Fig. 5.2) The relationship between MO and each level



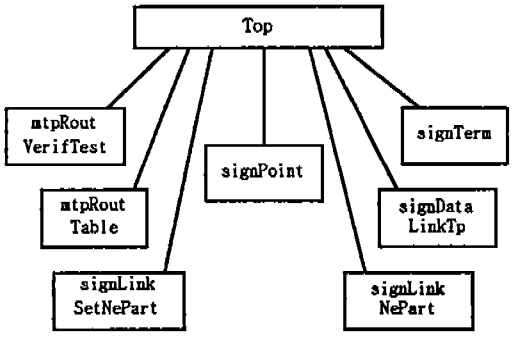
(그림 5.3) 신호중 교환기와 신호링크와의 Entity Relationship Diagram
(Fig. 5.3) The Entity relationship Diagram between SMX and SignLink



(그림 5.4) SMX의 관리대상들간의 Entity Relationship Diagram
(Fig. 5.4) The Entity Relationship Diagram among MOs of SMX

신호 중계 교환 내에서 관리되어야 할 자원들의 논리적인 엔터티(Logical Entity)들을 서로의 Entity Relationship에 따라 그림 5.3과 그림 5.4와 같이 표현된다.

위의 각 엔터티들은 관리 객체 클래스에 의해 표현되며, 각 관리 객체 클래스들의 상속 트리(Inheritance Tree)는 그림 5.5와 같다.



(그림 5.5) 상속 트리
(Fig. 5.5) Inheritance Tree

(1) 신호 중계 교환기(SP : Signalling Point)

No.7 기능 : 신호 중계 교환기는 신호망에 존재하는 한 노드로서 다음 3 종류로 분류한다. 첫째, SEP (Sinalling End Point) : 이 유형의 신호 중계 교환기는 MTP와 ISUP 기능을 소유한다. 그러나, 신호망의 중간 노드(Intermediate Node)로 작동하지는 않는다. 둘째, STP(Signalling Transfer Point) : 이 유형의 신호 중계 교환기는 MTP 기능만을 소유하고 있으며, 신호망에서 메시지를 전달하는 중간 노드로 작동한다. 셋째, STEP(Signalling Transfer and End Point) : 이 유형의 신호 중계 교환기는 STP와 SEP 유형의 기능을 소유한다.

관리 관점 : 관리 목적을 위해 각 신호 중계 교환기는 lock/unlock 또는 restart되며 OS로부터 신호 중계 교환기의 유형(SP-Type)이 무엇인지에 대한 질의에 응답 한다. 또한, 신호 중계 교환기와 관련된 장애(Fault)의 발생시 OS에게 통지(Notification)를 보낸다.

속성(Attribute)-signPointId : 관리 객체 클래스의 실체(instance)인 MO의 ID를 저장한다. pointCode : 메시지 전송부(MTP) 신호 절차에서 이용되는 point code를 나타낸다. spType : 신호 중계 교환기의 유형(SEP, STP, STEP)을 표시한다. state : OSI 관리 상태(OSI Management State)를 표시한다. userPartsPresent : 신호 중계 교환기가 지원할 수 있는 사용자부(User Part)들의 집합을 표시한다.

통지 : stateChange : 'state' attribute 값이 변경될 경우 발생한다. stpRestarted : 메시지 전송부(MTP)가 재시동될 경우 발생한다.

(2) 메시지 전송부 루팅 정보(MTP routing information)

No.7 기능 : 각 신호 중계 교환기는 임의의 목적 신호 중계 교환기에 도달하기 위해 거쳐야 하는 인접 신호 중계 교환기에 관한 메시지 전송부 루팅 정보를 저장하는 테이블을 가진다.

관리 관점 : 새로운 신호 중계 교환기가 신호망에 첨가되거나 트래픽의 관리를 목적으로 OS에 의해 메시지 전송부 루팅 정보는 변경될 수 있어야 한다.

속성 : mtpRouteTableId : 관리 객체 클래스의 실체인 MO의 ID를 저장한다. routeList : 관련 신호 중계 교환기로부터 도달 가능한 여러 루트 정보를 저장한

다. administrativeState:관리 객체 클래스의 상태(state)를 표시하며 관리 객체의 사용 여부에 따라 'locked' 또는 'unlocked'의 상태 값을 갖는다.

통지:adminStateChange:관리 객체의 상태 속성 값이 변경 될 경우 발생한다.

(3) 메시지 전송부 루트 검증 실험(MRVT:MTP route verification test)

No.7 기능:이 실험의 기능은 신호망 전반에 걸쳐 각 신호 중계 교환기가 소유하고 있는 메시지 전송부 루팅 정보의 일치 여부를 판별하며 권고안 Q.753에 정의 된 시험 절차에 따라 신호망 내에 loop의 존재 여부, 너무 긴 경로는 없는지의 여부, 존재하지 않는 목적 신호 중계 교환기의 존재 여부, 신호 관계에 있어서 양방향성(bi-directional)이 유지되는지의 여부를 알아낸다. 신호 중계 교환기에서 MRVT 실험이 시작되면, 한 목적 신호 중계 교환기에 도달 할 수 있는 모든 가능한 인접 신호 중계 교환기에게 MRVT 메시지를 보내며 그 인접 신호 중계 교환기들로부터 MRVA (MTP Route Verification Acknowledgment) 메시지가 올 때까지 기다린다. MRVA 메시지를 통해 목적 신호 중계 교환기와 MRVT 메시지가 거친 모든 중간 노드들과 관련한 정보를 얻을 수 있다.

관리 관점:OS는 한 신호 중계 교환기 내에서 MRVT 실험을 명령하고 이에 대한 결과를 얻을 수 있어야 한다.

속성:mtpRouteTestId:관리 객체 클래스의 실체인 MO의 ID를 저장한다.

통지:mtpRouteTestResult:실험 결과를 통지해 주며 다음의 결과들을 나타낸다. success, detected Loop, excessiveLengthRoute, unknownObjectInstance, route-Inaccessible, processingFailure, unknownInitiatingSP, timerExpired, spNotAnSTP.

동작:startMtpRouteTest:이 동작은 실험을 위해 지정된 목적 신호 중계 교환기에 대한 MRVT 실험을 시작한다.

(4) 신호 링크(Signalling Link)

No.7 기능:신호링크의 기능은 신호 중계 교환기들 간에 신호 메시지를 전달한다. 신호 링크는 신호 데이터 링크(Signalling Data Link)와 신호 터미널(Signalling Terminal)로 이루어진다.

관리 관점:NE에 존재하는 신호 링크의 일부에 대응하는 관리 객체 클래스가 정의되며, OS는 신호 링크의 양끝에 대응하는 NE-part 객체를 관리함으로써 완전한 신호 링크를 관리할 수 있다. 신호 중계 교환기에 있는 각 신호 링크 NE-part는 신호 터미널과 신호 데이터 링크 종단점(Signalling Data Link Termination Point)이 다음 3 가지 유형에 따라 구성된다. 첫째, signLinkNePartType-0:고정된 방법에 의해 구성된다. 둘째, signLinkNePartType-1:경우에 따라 유동적으로 구성된다. 셋째, signLinkNePartType-2:고정된 방법에 의하여 신호 데이터 링크 종단점 만으로 신호 링크 NE-part를 구성한다. OS는 신호 링크 NE-part의 상태를 검색 및 수정할 수 있어야 한다. 예를 들어 입력 버퍼나 출력 버퍼와 관련된 Congestion Threshold들은 OS에 의해 변경될 수 있어야 한다.

속성:signLinkNePartId:관리 객체 클래스의 실체인 MO의 ID를 저장한다. state:본 관리 객체의 상태를 표시한다. signDataLinkTpId:신호 링크 NE-part를 구성하는 신호 데이터 링크 종단점의 ID를 표시한다. signTermId:신호 링크 NE-part를 구성하는 신호 터미널의 ID를 표시한다. maxCapacitySL:신호 링크 NE-part의 최대 용량을 표시한다. congThreshReceive:입력 버퍼에 대한 Threshold 값을 나타낸다. cong-ThreshTransm:출력 버퍼에 대한 Threshold 값을 나타낸다.

통지:stateChange:본 관리 객체의 상태가 변경 될 경우 발생한다.

동작:replaceSignTerm:현재 신호 터미널을 바꾼다. replaceSignDataLinkTermTp:신호 데이터 링크 종단점을 바꾼다. signLinkTest:링크 실험을 실시한다.

(5) 신호 링크 그룹(Signalling Link Set)

No.7 기능:두 인접한 신호 중계 교환기들 사이에 신호 링크들의 집합을 신호 링크 그룹이라 한다. 인접한 신호 중계 교환기 사이에 이러한 링크 그룹이 하나 이상 존재할 수 있다.

관리 관점:신호 링크 그룹의 NE-part에 대응하는 관리 객체 클래스가 정의되며, OS는 신호 링크 그룹 NE-part의 상태를 검색할 수 있어야 하며 오류 수정 방법(Error Correction Method)을 변경할 수 있다.

속성:signLinkSetNePartId:관리 객체 클래스의 실체인 MO의 ID를 저장한다. state:이 관리 객체의 상태를 표시한다. adjPc:이 링크 그룹에 연결된 신호 중계 교환기의 point code를 표시한다. maxCapacityLS:신호 링크 그룹의 최대 용량을 표시한다. errorCorrMethod:신호 링크 그룹에서 사용하고 있는 오류 수정 방법.

통지:stateChange:본 관리 객체의 상태가 변경 될 경우 발생한다.

(6) 신호 터미널(Signalling Terminal)

No.7 기능:신호 터미널은 No.7 레벨 2 기능부인 신호 데이터 링크의 기능에 오류 발견(Error Detection), 메시지 재전송, Alignment절차 동의 기능을 덧붙인다.

관리 관점:관리 목적을 위해 신호 터미널은 lock 또는 unlock될 수 있어야 하며, 장애 발생시 통지를 발생할 수 있어야 한다.

속성:signTermId:관리 객체 클래스의 실체인 MO의 ID를 저장한다. state:본 관리 객체의 상태를 표시한다.

통지:stateChange:본 관리 객체의 상태가 변경될 경우 발생한다.

(7) 신호 데이터 링크(Signalling Data Link)

No.7 기능:신호 데이터 링크는 메시지 전송부의 레벨 1을 구성하며 신호 전송용의 양 방향 전송로를 의미한다.

관리 관점:관리 목적을 위해 신호 데이터 링크 종단점만이 고려되며, OS를 통해 상태 정보의 검색이 가능해야 하며 장애 발생시 통지 기능을 수행할 수 있어야 한다.

속성:signDataLinkTpId:관리 객체 클래스의 실체인 MO의 ID를 저장한다. state:본 관리 객체의 상태를 표시한다. adjPc:본 관리 객체 클래스에 의해 표현된 자원과 연결된 신호 중계 교환기의 Point Code를 표시한다.

통지:stateChange:본 관리 객체의 상태가 변경될 경우 발생한다.

6. MIB 구현

신호망 관리 시스템을 개발하기 위하여 크게 다음과 같은 작업 영역으로 나눌 수 있다. 첫째, 관리 객체들을 저장할 관리 정보 베이스(MIB)를 구축하는 영역, 둘째, 관리 객체들에 적용할 7가지의 공통 관리 정보 서비스(Event report, Get, Set, Action, Create, Delete, Cancel-get)들을 구현 하는 영역, 셋째, 이들 공통 관리 정보 서비스를 사용하여 시스템 관리 기능(System management function)들을 구현하는 영역, 넷째, 시스템 관리 기능을 이용하여 5가지의 TMN 관리 기능(성능 관리, 장애 관리, 구성 관리, 요금 관리, 보안 관리)들을 구현하는 작업 영역으로 나눌 수 있다. 본 논문에서는 위의 작업 영역 중 관리 객체들의 포함 관계로서 표현되는 MIB의 프로토타입을 구축한다. MIB를 구축하기 위해서는 앞 장에서 정의한 신호망 관리 객체 클래스들의 실체인 관리 객체를 생성하여 이들 간의 포함 관계를 트리 형태로 표현함으로써 구축된다.

현재 OSI의 표준안에 맞추어 망 관리를 위하여 사용되는 패키지에는 ISODE와 OSIMIS가 있는데 MIB를 구축하는데 필요한 기술적인 정보를 얻기 위하여 이들 패키지를 Sun Workstation에 이식하고 소스 코드를 분석하였다.

6.1 환경 구축

ISODE는 OSI 프로토콜에서 응용 프로그램의 발전을 가속화 하기 위해서 유럽의 RACE와 미국의 Internet에서 개발 하였으며, 이것은 DDN(Defense Data Network)과 OSI 모두를 잘 지원하고 있기 때문에 널리 이용되고 있다. ISODE에서는 기본적으로 각 계층의 서비스 프리미티브들을 제공하는데, 이 프리미티브들은 하위 계층의 SAP(Service Access Point)를 통하여 상호 통신을 한다. 응용 계층은 ACSE, ROSE, RTSE에 대하여 각각 acsap, rosap, rtsap 프리미티브들이 있고 각 응용 프로그램 FTAM, VT등은 이를 이용하여 자신의 서비스를 제공한다. 그리고 Presentation 계층, Session 계층, Transport 계층은 각각 psap, ssap, tsap의 함수들이 제공된다. 그 하부에 tsapd daemon프로세스가 있어서 incomming connection을 감시하다가 이 연결 요청들을 각 응용 프로그램에 dispatch한다. OSI 각 계층과 ISODE에서 제공하는 함수와의 관계는 그림 6.1과 같다.

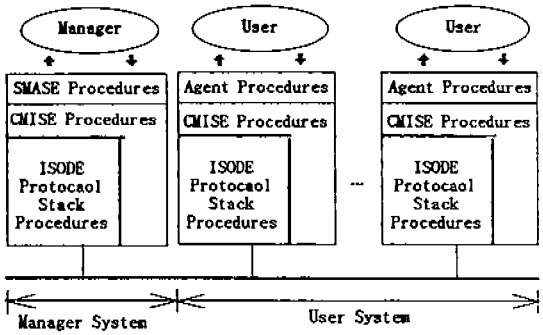
OSIMIS는 유럽의 연구 프로젝트의 한 부분으로

UCL에서 개발하였다. 이것은 OSI의 관리 기능들을 모두 포함하는 것은 아니고 이 시스템을 사용하여 네트워크 관리 기능들을 쉽게 개발하는데 도움을 주는 하나의 개발 도구이다. 특히, 객체 지향 응용 프로그램 인터페이스를 제공하므로 네트워크 관리 응용 프로그램 개발자들은 관리 정보 하부 구조의 자세한 사항을 고려하지 않고도 OSI의 공통 관리 정보 서비스/통신 규약(CMIS/CMIP)을 사용하여 네트워크 관리 시스템을 개발할 수 있다.

OSI protocol	ISODE protocol
Application	acsap rosap rtsap
Presentation	psap
Session	ssap
Transport	tsap
Network	tsapd

acsap : Association Control Service Access Point
rosap : Remote Operation Service Access Point
rtsap : Reliable Transfer Service Access Point
psap : Presentation Service Access Point
ssap : Session Service Access Point
tsap : Transport Service Access Point
tsapd : Transport Service Access Point Daemon

(그림 6.1) OSI protocol과 ISODE protocol stack
(Fig. 6.1) The ISODE protocol stack for the OSI protocol stack



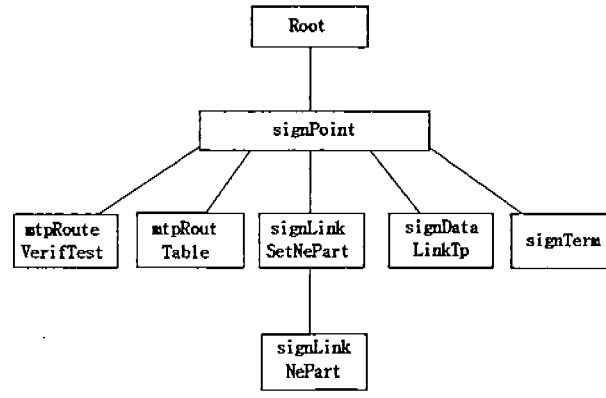
(그림 6.2) OSI 계층과 ISODE, OSIMIS와의 매핑
(Fig. 6.2) The mapping between OSI layers and OSIMIS

OSIMIS는 ISODE가 제공하는 서비스를 사용하여 OSI의 CMIS를 사용할 수 있도록 응용 인터페이스를 제공한다. ISODE와 OSIMIS의 관계를 OSI 계층에 매핑 시키면 그림 6.2와 같다.

6.2 구현 모델

MIB는 정의된 클래스의 실체(관리 객체)들이 노드를 형성하는 트리 구조로 이루어지며 트리를 구성하는 각 노드는 부모와 자식 노드를 갖는다. 이들 노드들 간의 계승 관계는 없고 각 관리 객체가 각자의 특성을 가지며 필요에 의해 부모/자식 노드의 관계가 성립된다. 한 클래스의 관리 객체가 같은 클래스 또는 다른 클래스의 관리 객체를 포함할 수 있는데 이 관계를 포함(containment)이라 한다. 이러한 포함 관계는 MIB를 구축하면서 트리 구조로 이루어지는데 이 트리를 containment tree라고 부른다. 이 트리를 구성하는 노드인 관리 객체들은 객체 식별자에 의하여 식별되며, RDN(Relative Distinguished Name)과 DN (Distinguished Name)이 있다. RDN은 특정 노드에서 자식 노드를 구별하기 위한 목적으로 사용되는 식별자로서 하나의 식별자 이름과 하나의 식별자 값으로 구성된다.

모든 노드는 자신의 RDN을 갖는데 다른 노드와 같은 RDN을 가질 수도 있다. 같은 RDN을 식별하기 위한 목적으로 사용되는 것이 DN이다. DN은 Containment Tree의 루트로부터 그 노드까지의 경로를



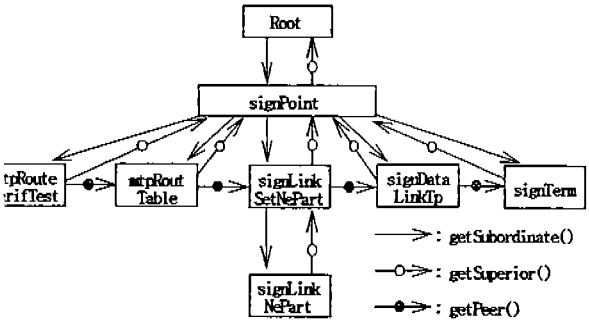
(그림 6.3) MIB 구현 모델
(Fig 6. 3) MIB implementation model

이루는 RDN들의 순열로 이루어 진다. MIB를 구축하기 위한 구현 모델은 그림 6.3과 같다.

6.3 MIB를 위한 Data Structure

그림 6.3의 구현 모델에서 MIB를 구성하는 각 관리 객체의 자료 구조를 정의하는 데 있어서 고려해야 할 사항들은 여러 가지가 있다. 먼저, 구축된 MIB 내에서 원하는 관리 객체를 탐색할 수 있는 메소드가 필요하다. 즉, 트리 구조의 MIB 내에서 각각의 노드로 표현되는 관리 객체들 중 찾고자 하는 관리 객체를 RDN 또는 DN으로 찾는 작업, 선택된 관리 객체에 대한 정보를 얻는 작업등 MIB를 관리하는데 필요한 가장 기본이 되는 수단이 필요하다. 이를 위해서 먼저 트리를 검색하기 위한 출발점이 되는 _root 관리 객체에 대한 포인터를 전역(global) 이면서 정적(static) 변수로 생성한다. _root에서부터 트리의 각 노드(관리 객체)의 subordinate를 탐색하기 위한 메소드를 getSubordinate( ), 각 노드의 superior를 탐색하기 위한 메소드를 getSuperior( ), 또한 트리 탐색의 용이성을 부여하기 위하여 각 노드의 형제 노드(sibling)를 탐색하기 위한 메소드를 getPeer( )로 각각 정의하였다. 이들 메소드의 동작을 도식화 하면 그림 6.4와 같다. 각 노드를 탐색할 때 비교 대상이 되는 관리 객체의 RDN 또는 DN을 얻기 위한 메소드를 각각 getRDN( )과 getDN( )으로 정의하였다.

위와 같은 메소드를 사용하여 원하는 관리 객체를 선택하고, 선택된 관리 객체에 대한 정보 얻기, 속성의 추가, 값들의 변경 등의 작업을 수행하기 위한 메



(그림 6.4) 트리를 탐색하기 위한 메소드
(Fig. 6.4) The Mothods for Tree Travers

〈표 6. 1〉 관리 객체의 정보를 수정 및 얻기 위한 메소드
〈Table 6. 1〉 The methods to set and to get the MO's Information

메소드	기능
initializeMIB( )	MIB를 초기화 한다.
getRoot( )	MIB의 root에 대한 포인터를 얻는다.
setClass(char*, int, int)	관리 객체의 이름을 저장한다.
setAttr(char*, char*)	관리 객체의 속성 이름을 저장한다.
getMO(RDN)	RDN 관리 객체 포인터를 얻는다.
getNAttrs( )	관리 객체의 속성들의 수를 얻는다.
getAttrName(int)	속성값 중 int번째 이름을 얻는다.
setAttrVAL(char*, void*)	관리 객체의 속성값을 저장한다.
getAttrVal(char*)	관리 객체의 속성값을 얻는다.

소드가 필요하다. 이에 대한 몇 가지 메소드를 표 6.1에 정리 하였다.

앞에서 설명한 메소드를 구현하기 위한 구조체 및 클래스를 정의하면 다음과 같다.

(1) RDN을 위한 구조체

```
typedef struct rdncomp {
        char* rdn_at;
        char* rdn_av;
 struct rdncomp* rdn_next;
} rdncomp, *RDN;
```

rdn_at에는 관리 객체 식별자 이름이 저장되고 rdn_av에는 관리 객체에 부여된 값이 저장된다. rdn_next에는 포함 트리(MIB)에서 자식 노드, 즉, subordinate 관리 객체의 RDN을 가리키는 포인터 이다.

(2) DN을 위한 구조체

```
typedef struct dncomp {
        RDN dn_rdn;
        struct dncomp* dn_parent;
} dncomp, *DN;
```

dn_rdn은 이 관리 객체의 RDN 값을 가리키는 포인터이고 dn_parent는 이 관리 객체의 부모 관리 객체, 즉, superior 관리 객체를 가리키는 포인터이다.

RDN 포인터를 따라가면서 subordinator들의 RDN을 검색할 수 있고, dn_parent를 따라 root에 도달하면 DN이 만들어 진다.

(3) 관리 객체를 생성하기 위한 클래스

```
class MOClassInfo {
        friend class MO;
        char*     _moClass;
        int _nattrs;
        char** _attrNames;
public:
        int setClass(char*, int, int);
        int setAttr(char*, char*);
        MO* getMO(RDN);
        int getNAttrs( );
        char* getAttrName(int);
        MOClassInfo( );
        ~MOClassInfo( );
};
```

MOClassInfo 클래스는 모든 관리 객체들이 갖는 일반적인 정보를 포함하며 모든 관리 객체 클래스는 이 클래스의 유일한 실체(static instance)를 공유한다. 유일한 실체만 존재하도록 하는 것은 다음에 설명될 Top클래스에서 static으로 정의되며 initializeClass 메소드에 의해서 초기화 된다. _moClass는 관리 객체의 식별자(이름)이다. setClass( ) 메소드는 파라메터로 전달된 관리 객체 이름을 _moClass에 저장하고 얼마나 많은 속성(_nattrs)들이 존재하는지 검사하여 사용될 기억 장소를 확보하는 일을 한다. 클래스에 존재하는 속성의 기억 장소가 확보되면 속성의 이름(_attrNames)을 저장해야 하고 또한 원하는 값을 저장할 수 있어야 한다. 이 수단을 setAttr( ) 메소드가 제공한다. 클래스의 모든 속성의 값을 주기 위해서는 이 메소드를 속성의 수 만큼 반복적으로 사용해야 한다.

```
class MO {
        friend class Top;
        static MO* _root;
        RDN       _rdn;
        DN        _dn;
        MO*       _superior;
        MO*       _subordinate;
        MO*_peer;
        int_nbindings;
        MO**_nameBinding;
public:
        static int    initialiseMIB( );
        static MO* getRoot( );
        MO*       getMO(DN);
        MO*       getSubordinate(RDN);
        MO*       getSuperior( );
        MO*       getPeer( );
        int       setRDN(char*);
        RDN       getRDN( );
        DN        getDN( );
        int       setAttrVal(char*, void*);
        void*     getAttrVal(char*);
        MO( );
        ~MO( );
};
```

MO 클래스는 모든 관리 객체 클래스의 superclass이다. 프로그램의 매인 함수 안에서 처음에 반드시 initialiseMIB( )를 사용하여 MIB를 초기화 해야 한다. 이 메소드는 다음에 설명될 Top클래스의 실체를 생성하여 그것을 MIB의 루트로 초기화 한다. getRoot ( ) 메소드는 MIB의 루트(_oot)를 얻을 수 있는 수단을 제공한다. getSubordinate(RDN) 메소드를 사용하여 MIB에서 자식 노드 중 RDN에 해당하는 subordinate관리 객체의 정보를 얻을 수 있다. 또한 getSuperior ( ) 메소드를 사용하여 바로 위 레벨의 관리 객체 정보를 얻을 수 있다. 이러한 메소드를 사용하여 MIB 트리의 전체를 탐색할 수 있다.

```
class Top:public MO {
        friend class  MO;
        static MOClassInfo _top;
        static int initialiseClass( );
        Top( );
};
```

MO 클래스의 정의에서 "friend class Top" 선언은,

Top 클래스로 생성되는 모든 관리 객체들이 MO클래스에서 정의한 속성과 메소드들을 사용할 수 있다는 것은 의미한다. 이 클래스를 정의한 목적은 모든 클래스와 관리 객체들이 이 클래스의 특성(MIB 루트 등)을 공유하도록 하는데 있다. initialiseClass( ) 메소드는 Top 클래스의 첫번 째 실체(관리 객체)가 생성될 때 실행 되는 메소드이다. 이 메소드가 하는 일은 관리 객체의 이름과 식별자를 MIB에 저장하고, 속성의 이름과 식별자들을 저장하여 MIB를 초기화 하는 일이다.

Top 클래스는 MO 클래스로부터 유도 되고 MO 클래스는 MOClassInfo 클래스의 특성(속성, 메소드)을 사용할 수 있으므로, Top 클래스로부터 유도 되는 모든 클래스는 MOClassInfo 클래스의 특성을 사용할 수 있고 MO 클래스의 특성과 Top 클래스의 특성을 모두 물려 받아서 이 세 가지의 클래스에서 정의한 메소드들을 사용할 수 있다. 새로운 클래스의 관리 객체들은 이 메소드들을 사용하여 MIB에 등록되며 초기화 된다.

(4) MIB 초기화 및 구축

MIB의 구축을 위하여 먼저 MIB의 루트를 나타낼 Top 클래스의 관리 객체를 생성 및 초기화 하고, 앞에서 정의한 클래스들로부터 관리 객체들을 생성하여 MIB에 등록함으로써 MIB 구축이 완료 된다.

MIB의 루트를 생성하고 이를 초기화 하기 위하여 클래스 MO에서 정의한 메소드 initialiseMIB( )를 실행시킨다. 즉, MO::initialiseMIB( )를 호출하면 이 메소드는 클래스 Top으로부터 관리 객체를 생성하여 그 포인터를 _root라는 속성에 할당한다(_root = new Top;). 다음에는 MIB의 루트를 초기화 하여야 하는데 이 작업은 initialiseClass( )를 호출함으로써 이루어진다. 즉, Top::initialiseClass( )를 호출하면 이 메소드는 _root->setClass("root", 0, 0)을 호출하여 루트의 관리 객체의 이름을 "root"로 저장하고 속성값들을 NULL로 설정함으로써 초기화 된다.

앞에서 설계한 MIB 구현 모델을 구축하기 위해서는 다음 세 단계를 거친다. 첫째, signPoint 클래스의 관리 객체를 생성하여 _root에 링크를 설정한다. 둘째, stpRouteVerifTest, mtpRouteTable, signLinkSetNePart, signData-LinkTp, signTerm 클래스로부터 관리 객체들을 생성하여 signPoint 관리 객체에 순서대로 링크를 설정한다. 셋째, signLinkNePart 클래스의 관리 객체를 생성하여 signLinkSetNePart 관리 객체에 링크를 설정함으로써 MIB의 구축이 완료된다. 예를 들어, signPoint 클래스의 관리 객체를 생성하여 MIB의 루트에 링크를 설정하는 과정은 다음과 같다. 먼저 관리 객체를 생성 한다(sp = new Top). 생성된 관리 객체에 이름을 부여하고 5개의 속성들의 이름과 값들을 저장할 기억 장소를 확보한다(sp->setClass ("signPoint", 0, 5)). 다음에는 속성들을 위해 확보된 기억 장소에 이름을 설정한다. 이를 위하여 setAttr( ) 메소드를 사용하는데, 이 메소드를 속성의 개수 만큼 반복하여 사용한다. 즉, sp->setAttr("signPointID", 0), sp->setAttr("pointcode", 1), sp->setAttr("spType", 2), sp->setAttr("state", 3), sp->setAttr("userPartsPresent", 4)와 같이 호출한다. 속성의 이름을 저장한 후에는 이들 속성들에 값을 설정해야 하는데 이것은 setAttrVal( ) 메소드에 의하여 이루어 진다. 즉, sp->setAttrVal ("signPointID", SP01), sp->setAttrVal("spType", SEP(0)), sp->setAttrVal("state", unlocked(1))등과 같이 수행 한다. 이로써 signPoint 관리 객체가 하나 생성되었고 속성의 이름과 값들이 설정되었다. 다음에 해야 할 작업은 이 관리 객체를 루트에 연결하는 작업이다. 이 작업은 _root->_namebindings[_nbindings] = sp와 _root->_nbindings^{++} 그리고 sp->_superior = _root, sp->_subordinate = NULL, sp->_peer = NULL을 수행함으로써 이루어 진다. 이로써 하나의 관리 객체를 생성, 초기화하여 MIB에 등록시키는 작업이 완료 되었다. 이 작업 과정을 모든 관리 객체에 반복 적용함으로써 목적한 MIB의 구현 모델을 구축할 수 있다.

7. 결론 및 향후 연구 방향

망 관리의 표준화를 위하여 ITU-TS, ISO, NM Forum 그리고 ANSI 위원회를 중심으로 활발한 연구가 진행되고 있다. 특히, 신호망 요소들의 TMN 접속을 위한 신호망 관리에 대한 연구가 ITU-TS SG XV에서 이루어 지고 있다. 현재, 국내에서는 ETRI와 KT에 의해서 향후 전송 망의 기반을 이룰 SDH(Synchronous Digital Hierarchy) 망의 TMN 접속을 위한 연구

가 진행되고 있으나, 국내 환경에 적합한 규격 및 표준화 활동은 미진한 상태이다.

이에 본 고에서는 신호망의 관리를 목적으로 개발된 신호망 관리 시스템(SIGNOS)을 TMN 접속을 통하여 TMN 기본 원칙과 요구 사항에 따르는 신호망 관리 기능 참조 모델을 제시하였고, 이를 근거로 TMN 신호망 관리 구조 및 T_SIGNOS 시스템 구조를 제안하여 진화된 신호망 관리 모델을 제시하였으며 신호망에서 관리 대상이 되는 관리 객체에 대한 MIB를 구현하였다. 향후 연구 방향은, CMIS를 사용하여 앞에서 제시한 T_SIGNOS를 운영하기 위한 시스템을 개발할 예정이다.

참 고 문 헌

[1] ITU-TS Recommendation Q. 750:Overview of Signalling System No.7 management

[2] ITU-TS Recommendation Q.751:Signalling System No.7 Managed Object

[3] ITU-TS Recommendation Q.753:Signalling System No.7 Management Functions MRVT, SRVT, and CVT and DEFINITION of the OMASE-User

[4] ITU-TS Recommendation Q.754:Signalling System No.7 Management ASE Definitions for MRVT, SRVT, and CVT

[5] ITU-TS Recommendation M.3010:Principles for a TMN

[6] ITU-TS Recommendation M.3020:TMN Interface Specification Methodology

[7] ITU-TS Recommendation M.3100:Generic Network Information Model

[8] ITU-TS Recommendation M.3180:Catalogue of TMN Management Information

[9] ITU-TS Recommendation M.3200:TMN Management Services-Overview

[10] ITU-TS Recommendation M.3300:TMN Management Capabilities Presented at the F Interface

[11] ITU-TS Recommendation M.3400:TMN Management Functions

[12] ITU-TS/ISO STANDARDS X.700/ISO 7498-4: Management Framework Definition for OSI

[13] ITU-TS/ISO STANDARDS X.701/ISO 10040: Systems Management Overview

[14] ITU-TS/ISO STANDARDS X.710/ISO 9595: Common Management Information Service Definition

[15] ITU-TS/ISO STANDARDS X.711/ISO 9596: Common Management Information Protocol Specification

김 태 형

1992년 한남대학교 전산학과 졸업(학사)

1994년 충남대학교 전산과학과(이학석사)

1996년~현재 충남대학교 컴퓨터과학과 박사과정

관심분야:통신망 관리, OS, 데이타베이스

고 병 도

1981년 숭실대학교 전산학과 졸업(학사)

1983년 숭실대학교 전산학과(석사)

1993년~현재 충남대학교 컴퓨터과학과 박사과정

1983년~현재 한국전자통신연구소 광대역서비스연구실장, 정보처리 기술사

관심분야:통신망 관리 및 구조 기술, 데이타베이스, 시트템 엔지니어링, B-ISDN

류 재 철

1985년 한양대학교 산업공학과 졸업(학사)

1988년 Iowa State University 전산학과(석사)

1990년 Northwestern University 전산학과(박사)

1991년~현재 충남대학교 컴퓨터과학과 조교수

관심분야:통신망 관리, 컴퓨터 및 통신 보안, 분산처리