TMS800: 메타데이타를 사용한 방송 장비 관리 시스템

(TMS800: A Metadata-based Management System for Distributed Broadcasting Devices)

김 민 석 † 최 정 호 † 김 정 선 ††
(Minsuc Kim) (Jeongho Choi) (Jungsun Kim)

요 약 방송 시스템이 거대해지고 방송 장비들의 기능이 다양해지면서 방송장비들에도 네트워크 기능이 첨가되고 있다. 그러나 방송장비에 특화된 관리 시스템은 전무한 상태라 할 수 있다. 네트워크에 접속된 장비들은 네트워크에 접속하지 않을 때보다 훨씬 더 많은 기능을 수행해 낼 수 있는 반면에 관리하기 쉽지 않은 단점이 있다. 따라서 네트워크를 통한 자동화된 관리 시스템은 필수적이라 할 수 있다. 하지만 방송 장비를 네트워크상에서 관리하기 위해서는 기존의 SNMP 방식만으로는 한계점이 있다. 본 논문에서는 메타데이타를 사용하여 방송장비를 관리할 수 있는 시스템(TMS800)을 제안한다. TMS800은 네트워크에 분산된 각종 방송 장비의 상태 뿐만 아니라 영상정보도 이미지의 형태로 모니터링 할 수 있으며, 장비의 제어와 이상 발생 시 실시간으로 제공받는 기능을 제공한다.

키워드 : 메타데이타, 방송장비, 네트워크 관리 시스템, SNMP

Abstract As the scale of broadcasting systems gets bigger and the functionality of broadcasting devices becomes diverse, networking facilities are being incorporated within recent broadcasting devices. Although networked devices can perform many of the additional functionalities, an effective management becomes a difficult issue. Therefore, it is essential to provide an automated management system for monitoring and controlling distributed broadcasting devices across a network. SNMP (Simple Network Management System) is one of the enabling technologies that we could adopt when we build such a system. However, SNMP-based solution has its limitations. In this paper, we propose the TMS800 system, which is a metadata-based device management system on top of SNMP framework. The system is specifically designed for the management of distributed broadcasting devices. It makes it possible to monitor not only the status of devices, but also the videos in the form of still images. Remote control and real-time notification facilities are also provided.

Key words : metadata, broadcasting device, network management system, SNMP

1. 서 론

1990년 이후, 급격한 인터넷의 발전은 네트워크에 대한 전반적인 패러다임에 변화를 가져왔고, 그 결과 네트워크에 접속되는 디지털 장비들이 기하급수적으로 증가하기 시작했다. 특히, 방송 시스템이 거대해지고 방송 장비들의 기능이 다양해지면서 최근에는 방송장비들에도 네트워크 기능이 첨가되고 있다. 네트워크에 접속된 장비들은 네트워크에 접속하지 않을 때보다 훨씬 더 많은 기능을 수행해 낼 수 있는 반면에 관리하기 쉽지 않은 단점이 있다. 따라서 네트워크를 통한 자동화된 관리 도구는 필수적이라 할 수 있다. 하지만, 일반적인 관리 시스템은 많이 개발되어 사용되고 있는 실정인데 반해 방송 장비에 특화된 관리 시스템은 거의 전무한 상태라고 할 수 있다.

본 논문에서는 메타데이타를 사용하여 방송장비를 관리 할 수 있는 시스템[1](TMS800)을 제안한다. 관리를 하기 위해서는 피 관리 장비를 모니터링 할 수 있어야

† 비 회 원 : 한양대학교 컴퓨터공학과
minsuc@cse.hanyang.ac.kr
jhchoi@cse.hanyang.ac.kr
†† 종신회원 : 한양대학교 컴퓨터공학과 교수
jskim@cse.hanyang.ac.kr
논문접수 : 2005년 11월 23일
심사완료 : 2007년 6월 22일

1) TMS800은 (주)Televiewㅘ의 산학협동 과제의 산출물이다. TMS800의 모든 저작권은 (주)Televiewㅔ 있음을 밝힌다.

하고 피 관리 장비의 상태의 추적이 가능해야 하며 피 관리 장비에서 발생하는 예외적인 상황에 적절히 대처할 수 있어야한다. 이를 위해서 네트워크 관리에 표준 프로토콜로 쓰이는 SNMP를 사용하였다. 또한, 방송장비에 특화된 관리 시스템의 기능을 수행하기 위해서는 SNMP만으로는 불가능한 점들이 많다. 예를 들어, 현재 방송되는 화면을 모니터링 하기 위해서는 현재 화면을 갈무리하여 이미지 상태로 전송을 해야 하지만 SNMP 상에서는 오로지 바이너리로만 전송을 하기 때문에 그에 따른 다른 정보가 더 있어야 한다. 이를 위해 TMS800에서는 메타데이타를 사용하여 이 문제를 해결하였다.

본 논문의 2장에서는 관련연구를 언급하고 3장에서 TMS800의 설계 및 구현에 관한 것을 제시하며, 4장에서는 문제점 해결 방법과 향후 보완되어야 할 문제와 함께 결론을 맺도록 한다.

2. 관련 연구

2.1 SNMP Framework

SNMP Framework[1]는 네트워크 관리를 위한 여러 가지 표준을 명시하고 있다. SNMP Framework은 관리국/에이전트 모델로 구성되어 있다. 핵심적인 구성요소들은 다음과 같다[2,3].

- 관리국(Manager): 전송 규약에 명시된 여러 가지 오퍼레이션을 사용하여 에이전트에게서 값을 가져오거나 값을 세팅한다. 때로는 에이전트로부터 트랩을 받기도 한다.
- 에이전트(Agent): 관리해야 하는 여러 이질적인 장비들을 SNMP라는 같은 관점으로 볼 수 있게 해주는 일종의 소프트웨어다. SNMP를 사용하여 각 장비와 통신 한다는 의미는 각 장비의 에이전트와 통신한다는 뜻이다. 에이전트는 관리국에서 어떤 정보를 요구하면 MIB에서 그 정보를 찾은 다음 다시 관리국에게 보내주는 것이 에이전트에서 하는 가장 중요한 역할이다. 하지만, 항상 이런 수동적인 행위만을 하지는 않는다. 각 장비에 특별한 문제가 발생하거나 관리국으로 어떤 정보를 보내줘야 할 때 트랩을 발생시켜 관리국에 정보를 보내주기도 한다.
- 관리 정보 기반(MIB: Management Information Base)[4-6]: 피 관리 장비에서는 장비에 관한 여러 가지 정보들을 담아두어야 할 데이타베이스가 필요하다. MIB는 이러한 일을 수행하는 가상 정보 저장소라고 보는 것이 옳다. MIB의 안을 살펴보면 피 관리 장비에서 관리해야할 여러 가지 정보들을 볼 수 있다. 이런 정보들을 관리 객체라고 부르는데, 각각의 관리 객체들은 SMI(Structure of Management Information)[7]에 의해서 표현된다. SMI에서의 관리객체는 다음의 세 가지 요소로 구성된다.
  - 이름(Name): 관리 객체는 자신을 구분하기 위한 유일한 이름을 가진다. 즉, 이름이라는 것은 객체 식별자(OID)로서, 객체를 고유하게 지칭한다.
  - 구문(Syntax): 관리 객체의 데이타 형을 정의한다. ISO 8824에 정의되어 있는 메시지 서술 언어인 ASN.1(Abstract Syntax Notation One)[8]을 사용한다.
  - 인코딩(Encoding): 관리 객체와 관련된 정보를 이질적인 네트워크 환경에서 어떻게 인코딩/디코딩 할 것인지 명시한다. ISO 8825에 정의 되어 있는 BER (Basic Encoding Rules)[9]를 사용한다.
- 전송 규약(SNMP)[10]: SNMP(Simple Network Management Protocol)은 관리국과 에이전트가 통신하기 위한 표준 규약이다. 그림 1은 SNMP가 어떤 방식으로 관리국과 에이전트사이에 통신을 하는지 보여준다. SNMP의 오퍼레이션은 크게 Get, Set, Trap의 세 동작으로 구분해 볼 수 있다.
  - Get: 관리국에서 에이전트에게 관리 객체의 값을 요청할 때 쓰이는 오퍼레이션이다.
  - Set: 관리국에서 에이전트에게 관리 객체의 값을 세팅하라는 명령을 보낼 때 쓰이는 오퍼레이션이다.
  - Trap: 피 관리 장비 측에서 각 장비에 문제가 발생하거나 관리국으로 특정 정보를 보내줘야 할 때 에이전트에서 쓰이는 오퍼레이션이다.

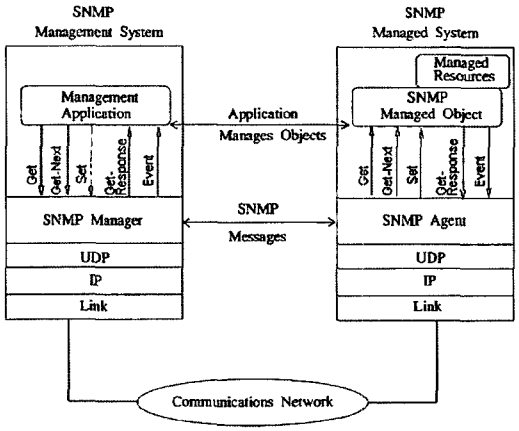

그림 1 SNMP Architecture(1990, IEEE)

SNMP의 기본적인 통신은 UDP상에서 161번 포트를 사용함으로써 이루어진다. 또한, TRAP은 162번 포트를 사용하게 된다.

2.2 NMS(Network Management System)

현재 네트워크를 관리하기 위한 범용툴들이 매우 광범위하게 연구되어지고 있고 현재 상용화되어 있는 응용프로그램도 상당수 존재한다.

- HP Openview[11]: 대표적인 상용 네트워크 관리 시스템이다. Hewlett Packard사에서 개발하였으며 네트워크 관리에 관한 전체적인 솔루션을 제공한다.
- OpenNMS[12]: GPL을 따르는 대표적인 네트워크 관리 시스템이다. 2007년 5월 현재 1.3.2버전까지 개발 중이며 계속적으로 시스템의 업그레이드가 진행 중이다.

위 두 가지 시스템 외에도 이와 유사한 방식의 네트워크 관리 시스템이 존재하고 있으며 연구 중에 있다. 하지만 방송용 시스템에 특화된 네트워크 관리 시스템은 전무한 상태이다.

따라서, TMS800은 방송용 장비를 좀 더 효과적으로 관리하기 위하여 다음과 같은 사항을 구현하였다.

- 그림이나 소리와 같은 ASN.1에 정의 되어 있지 않은 타입을 지원 할 수 있도록 메타데이타 빌더를 유연하고 견고하게 디자인 하였다. 특히, TMS800은 방송용 장비에서 필수적이라 할 수 있는 이미지를 지원하도록 구현되었다.
- 특정 OID의 값들에 임계치를 부여 할 수 있는 기능이 구현되었다. 따라서 사용자는 실시간으로 변하는 수많은 값들 중 임계치를 넘어가는 값들만을 빠르게 찾아서 볼 수 있는 편의성을 제공한다.
- 각 장비의 메타데이타에는 각 장비가 가지고 있는 수많은 값들의 기본 값을 포함 할 수 있다. 이 의미는 굉장히 복잡한 방송용 장비의 세팅을 방송용 장비를 개발하는 측에서 세팅을 할 수 있다는 뜻이고 방송 장비의 사용자는 방송 장비의 복잡한 부분까지 몰라도 되는 장점이 있다.
- MIB이외에도 방송용 장비를 다루기 위해서는 더욱 더 많은 정보가 필요하다. TMS800은 메타데이타를 통하여 MIB를 포함하여 방송용 장비를 좀 더 효과적으로 컨트롤 할 수 있는 방법을 제공한다.

위와 같은 사항을 구현하기 위해서는 메타데이타와 메타데이타 빌더가 필수적으로 고려되고 구현되었으며 신뢰성을 더하기 위하여 Java의 객체 직렬화[13] 기술을 사용하였다.

3. TMS800의 설계 및 구현

3.1 TMS800의 구현 목표

- 개별화 된 방송용 장비의 MIB 지원: 메타데이타를 관리하는 툴을 따로 분리하여 개별화되고 특화된 방송용 장비의 MIB를 지원한다. 또한, 각 OID별로 임계치를 설정할 수 있는 기능을 지원함으로써 방송용 장비 관리자가 매우 효율적으로 시스템을 관리 할 수 있도록 한다.
- 이미지를 직접 표현 할 수 있는 관리 객체 모니터링 화면 지원: 각 방송용 장비의 메타데이타에는 관관리객체가 이미지를 전송하는지에 관한 정보가 들어있다. TMS800은 이 정보를 바탕으로 해당 관리 객체가 이미지를 전달하는지를 판단하여 관리 객체 모니터링에 직접 이미지를 표현할 수 있는 기능을 지원하게 된다. 방송용 장비의 특성상 매우 유용하고 필수적인 기능이다.

3.2 TMS800의 개요

TMS800은 방송용 장비를 관리하기 위한 SNMP 어플리케이션이다. 관리를 하기 위해서는 피 관리 장비를 모니터링 할 수 있어야 하고 피 관리 장비의 상태의 추적이 가능해야 하며 피 관리 장비에서 발생하는 예외적인 상황에 적절히 대처할 수 있어야한다. 이를 위해서 TMS800에서는 장비의 현재 상태를 값으로 가져오기 위해 SNMP를 사용한다. 또한, 장비 측에서는 TMS800에 알려야 할 정보가 있을 수 있기 때문에 SNMP 트랩을 사용하여 TMS800측으로 값을 전달하기도 한다.

그림 2는 TMS800과 방송용 장비들과의 연결 상태를 도식화한 것이다. TMS800은 모든 장비들과 1:N의 관계를 유지하면서 주기적으로 각 에이전트에게 SNMP-GET 메시지를 보내게 된다. TMS800으로부터 SNMP-GET 메시지를 받은 에이전트는 SNMP-GET 메시지에 있는 관리 오브젝트의 OID에 해당되는 값을 TMS800으로 되돌려주게 된다. 마지막으로, 요청한 OID의 값을 받은 TMS800은 그 값을 화면에 보여주기도 하고, 그래프를 사용하여 시각적으로 사용자에게 리포트를 해주기도 하며, 특정 값을 넘은 OID에 관해서는 따로 관리를 해주기도 한다.

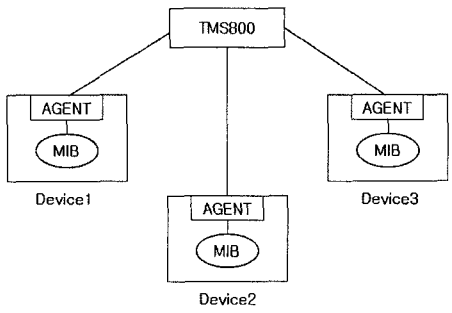

그림 2 TMS800과 장비들과의 연결

3.3 설계 시 고려 사항

3.1절에서 TMS800이 각 장비들과 어떤 과정을 거치며 통신을 하는지에 관해서 설명을 했다. 하지만, 그 같

은 과정을 거치기 위해서는 필히 수반되어야 하는 여러 가지 절차가 필요하다. 다음 열거된 항목들은 TMS800이 방송용 장비들을 관리하기 위해서 필요한 기본적인 사항들이다.

- TMS800이 각 에이전트로부터 OID의 상태를 가져오기 위해서는 해당 OID를 알고 있어야 한다. 즉, 각 장비의 고유 MIB를 이미 알고 있어야 한다는 뜻이다.
- 그림이나 소리와 같은 ASN.1에 정의 되지 않은 타입의 데이타를 표현 할 수 있어야 한다. 방송용 장비의 특성상 그림이나 소리와 같은 정보를 장비로부터 받아와야 하는 경우가 생기게 된다. 하지만, ASN.1에 정의된 타입만으로 불충분 할 때가 있다. 그림이나 소리 모두 OCTET STRING 타입을 사용해야 한다. 하지만, TMS800입장에서 다른 정보가 주어지지 않는다면 OCTET STRING 타입의 데이타만으로는 그 정보가 어떤 정보인지 판단할 방법이 없기 때문이다.
- 특정 OID에 의미를 부여 할 수 있어야 한다. 특정 OID는 의미 있게 다루어질 필요가 있다. 예를 들어, 어떤 값들은 그래프 차트를 사용해서 사용자에게 시각적으로 알려주어야 하며, 어떤 값들은 임계값을 설정해서 임계 범위를 벗어날 때마다 로깅을 남기거나 사용자에게 알려주어야 한다.
- 특정 OID에 임계값을 설정하기 위한 방법이 필요하다. 현재의 SMI에는 각 OID에 대한 임계값을 설정할 수 있는 부분이 존재 하지 않는다.
- SNMP나 장치에 관한 세팅을 사용자가 하지 않아도 동작할 수 있게끔 기본 설정이 존재해야 한다. TMS800을 쓰는 사용자는 방송 장비를 다루는 일에 종사하는 사람일 확률이 상당히 높다. 이런 사람들은 SNMP에 관해서 잘 알지 못할 가능성이 높다. 이런 사용자를 위하여 기본 세팅은 존재해야 한다.
- MIB 이외에도 장비를 올바르게 다루기 위해서는 다른 여러 가지 정보들이 필요하다. MIB에 장비를 컨트롤 할 수 있는 모든 기능을 넣기는 불가능 할 뿐만 아니라 넣는다 해도 그 기능을 다 넣기 위해서 장비의 MIB가 상당히 커져야하는 불편함을 감수해야 하기 때문이다. 이런 정보를 가장 잘 알고 있는 곳은 다름 아닌 방송 장비를 만드는 측이다. 방송 장비를 만드는 측에서 다른 정보를 제공할 수 있는 기능이 존재해야 한다.

본 논문에서는 위와 같은 문제점을 해결하기 위하여 메타데이타를 사용하여 해결하는 방법을 제안한다.

3.4 TMS800 메타데이타

3.2절에서 열거한 여러 가지 문제점들은 정보들을 기술할 수 있는 정보들을 사용함으로써 해결이 가능하다. 즉, 메타데이타를 따로 두어서 그 정보들을 모두 관리하게 된다면 훨씬 손쉽게 위와 같은 문제점들을 해결할 수 있게 된다.

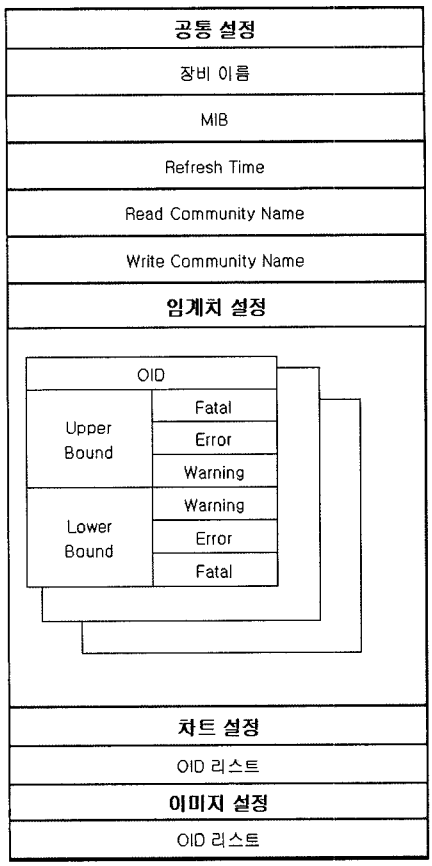

그림 3 TMS800 메타데이타 다이어그램

그림 3은 TMS800을 위한 메타데이타를 도식화 한 그림이다. 이런 메타데이타 설정은 크게 공통, 임계값, 차트, 이미지 설정으로 나눠 볼 수 있다.

- 공통 설정
  - 장비이름 : 해당 장비의 이름을 설정하는 부분이다.
  - MIB: 해당 장비의 MIB를 설정하는 부분이다. 3.2절에서 언급했던 것과 같이 장비를 SNMP로 모니터링 할 수 있기 위해서는 장비의 MIB를 알아야 한다. TMS800은 이 정보를 바탕으로 OID를 알아내어 장비를 모니터링 할 수 있다.
  - Refresh Time: 장비의 OID값들을 알아오기 위해서는 주기적으로 값을 장비의 에이전트에게 물어봐야 한다. 이와 같은 주기를 설정하는 부분이다.
  - Read/Write Community Name: SNMP로 통신을 하기 위해서는 Community Name을 알고 있어야 한다. 그 부분을 설정해 주는 부분이다.
- 임계값 설정: 임계값 설정은 각각의 OID를 중심으로

설정하게 된다. 각 OID는 크게 4가지 단계로 구분 할 수 있다. 그 단계는 Normal, Warning, Error, Fatal 이고 각각의 단계는 Upper Bound와 Lower Bound 를 가진다. 예를 들어, 1.3.6.1.2.1.1.6이라는 OID의 임계값 설정이 표 1과 같다고 하자.

표 1 임계값 설정의 예

OID		1.3.6.1.2.1.1.6
Upper Bound	Fatal	100
	Error	50
	Warning	25
Lower Bound	Warning	0
	Error	-50
	Fatal	-100

그런데 특정 시간에 1.3.6.1.2.1.1.6의 값이 75가 되었다면 그때 1.3.6.1.2.1.1.6의 상태는 Error가 되는 것이다. 이런 식으로 각각의 OID에 임계값을 설정하게 됨으로써 장비의 이상 현상 발생 시에 즉각적으로 대응 할 수 있게 되며 방송장비의 관리에 효율을 높일 수 있게 된다.

또한, 각각의 단계는 화면상에 색으로 구분하게 하여 사용자가 한눈에 알아 볼 수 있게 한다. Fatal은 빨간색, Error는 주황색, Warning은 하늘색, Normal은 흰색으로 색을 지정함으로써 화면상에서 유저가 한눈에 현재의 상태를 알아 볼 수 있게 할 수 있다.

- 차트 설정: 어떤 OID의 값은 시간상으로 변하는 것에 의미가 있기 때문에 계속 값을 보관하면서 시간상으로 변하는 것을 사용자에게 시각적으로 보여줄 필요가 있다. 이때 필요한 것이 시각적인 차트이다. 즉, 그래프를 사용하는 것이 유저 인터페이스에 있어서 상당히 효과적이다. 이와 같은 OID의 리스트를 설정해 주는 부분이다.
- 이미지 설정: 방송 장비의 특성상 현재 방송되는 화면을 실시간으로 모니터링 할 수 있는 기능이 필요하다. 일반적으로 생각해 볼 수 있는 방법은 SNMP 통신을 위한 포트 이외에 다른 포트를 할당해서 이미지를 받을 수 있는 채널을 할당하는 방법이다. 하지만, 이 방법은 그에 따른 오버헤드를 감수해야 한다. 따라서 TMS800에서는 기존의 SNMP 채널을 그대로 이용한다. 물론, SNMP가 UDP상에서 통신을 하기 때문에 UDP가 TCP에 비해 가지는 단점을 그대로 가지고 있지만, 모니터링 이라는 측면에서 중간에 이미지가 어느 정도 회손 되는 것은 용납되기 때문에 적절하다고 할 수 있다.

3.5 메타데이타 빌더

메타데이타 빌더는 메타데이타 파일을 만들어주는 TMS800의 보조 어플리케이션이다. 메타데이타를 사용하게 되면 3.2절에서 제기되었던 여러 가지 문제점들을 해결할 수 있다. 하지만 메타데이타를 손쉽게 만들 수 없다면 그에 따른 효용성은 상당히 감소하게 된다. 따라서 메타데이타 빌더는 메타데이타를 사용함에 있어서 필수적인 어플리케이션이라 할 수 있다.

메타데이타 빌더를 유연하고 견고하게 디자인하기 위해서는 두 가지 사항을 준수해야 한다.

① 메타데이타의 구성이 변경되어도 유연하게 대처할 수 있어야 한다.

② 생성되는 메타데이타 파일 포맷의 추가가 자유로워야 한다.

①을 만족시키기 위해서 객체의 직렬화[13]를 사용하였다. 객체의 직렬화를 사용하게 되면, 객체내부의 표현과 객체에서 필요한 부분을 파일로 생성하는 부분이 분리 된다. 따라서 차후에 메타데이타 다른 섹션을 추가한다 하더라도 메타데이타 파일을 생성하는 부분은 전혀 영향을 받지 않는다.

②를 만족시키기 위해서 Builder Pattern[14]을 사용하였다. Builder Pattern을 사용하게 되면 표현과 생성을 분리 할 수 있기 때문에 효율적이다. 즉, 다른 메타데이타 파일의 포맷을 추가하려 할 때에도 표현과 생성이 분리되어 있기 때문에 손쉽게 추가가 가능하게 된다. 그림 4는 이와 같은 구조를 UML 클래스 다이어그램으로 나타낸 것이다. TMS800MetadataBuilder Interface가 추상화된 부분이고 그것의 서브 클래스들이 실제로 메타데이타를 생성하는 concrete 클래스이다. TMS800-MetadataDirector 클래스가 한 개 이상의 TMS800MetadataBuilder 타입의 인스턴스를 가지고 있다가 build라는 메서드에 의해서 실제 작업을 하게 된다.

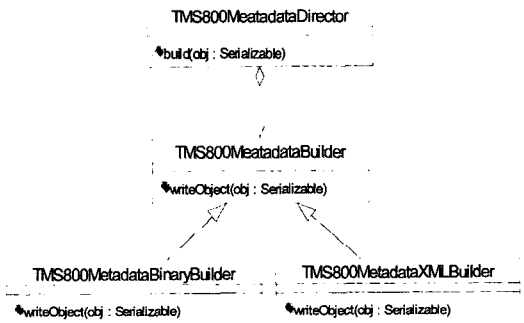

그림 4 TMS800 메타데이타 생성에 관한 UML 클래스 다이어그램

3.5.1 공통사항 설정

그림 5는 메타데이타 빌더의 공통 설정을 하는 화면이

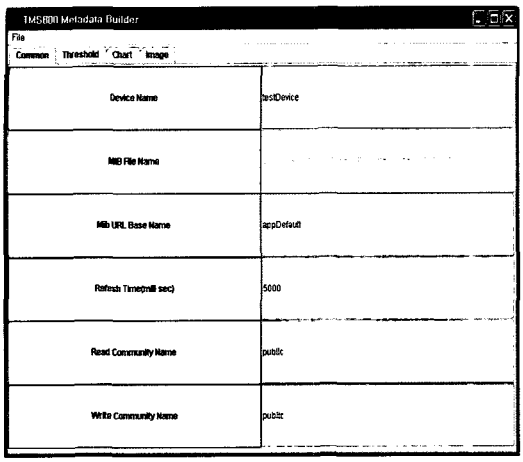

그림 5 메타데이타 빌더 공통 사항 설정 화면

다. 각각의 행은 장비의 이름이나 MIB 파일 경로, refresh time, community name을 설정하도록 되어 있다.

3.5.2 임계값 설정

그림 6은 임계값 설정을 수행할 수 있는 화면이다. 화면의 우측은 MIB 내용을 Tree 구조로 보여준다. 우측에서 특정 OID를 선택한 후 Select버튼을 누르거나 화면 왼쪽 상단의 리스트로 드래깅을 하게 되면 선택한 OID가 리스트에 추가된다. 추가된 OID는 왼쪽 아래에 보이는 설정창에서 임계값을 설정해 줄 수 있다.

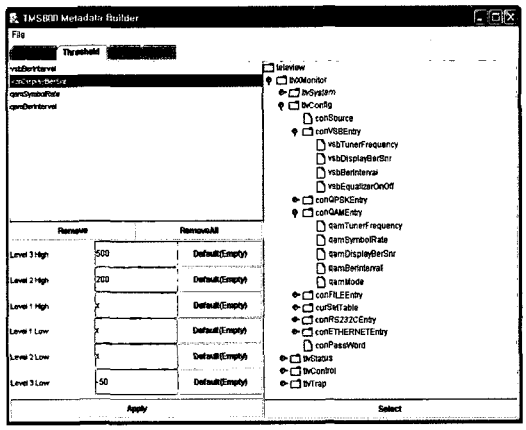

그림 6 메타데이타 빌더 임계값 설정 화면

3.5.3 차트와 이미지 설정

차트 설정과 이미지 설정은 비슷한 방식으로 진행된다. 오른쪽 화면에서 OID를 선택한 후 왼쪽 리스트에 추가하면 차트나 이미지가 설정된 OID가 된다.

3.6 TMS800의 구성

TMS800은 여러 가지 모듈들이 서로 통신하면서 어

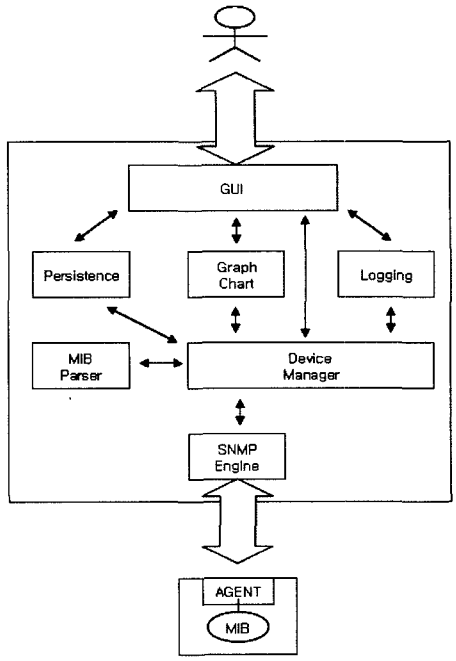

그림 7 모듈간의 협력 관계

플리케이션이 동작을 하게 된다. 그림 7은 모듈간의 관계를 그림으로 나타낸 것이다. 이번 절에서는 모듈간의 협력관계와 각 모듈의 특징을 설명함으로써 TMS800의 각 부품이라 할 수 있는 모듈들에 대해서 알아보고자 한다.

3.6.1 모듈간의 협력관계

TMS800은 모듈들끼리 유기적인 통신을 함으로써 방송장비에 대한 관리 작업을 수행하게 된다. 중심이 되는 모듈은 Device Manager라고 볼 수 있다. 그 이유는 Device Manager가 피 관리 객체들에 관한 정보를 가지고 있어서 다른 모듈들이 Device Manager를 참조하는 방식으로 모듈들 간의 통신이 이루어지기 때문이다.

유저가 GUI를 통하여 피 관리 장비를 추가하게 되면 Device Manager는 메타 파일 정보를 토대로 MIB Parser를 사용하여 MIB Tree를 구성하게 된다. 그리고 메타 정보에 기록된 Refresh Time을 바탕으로 지속적인 피 관리 장비의 관리 객체의 정보를 SNMP Engine에게 요청하게 된다. SNMP Engine이 SNMP Get 요청으로 에이전트에게 정보를 요청하고 나서 요청한 정보를 받게 되면 다시 Device Manager에게 해당 정보를 넘겨주는 식으로 모니터링에 대한 전반적인 작업을 수행한다.

장비에 관한 특정 관리 객체의 값을 변경하고 싶은 경우에는 유저가 GUI 모듈을 통하여 명령을 내리게 되면 Device Manager가 그에 관한 정보를 받게 된다. 그

후, Device Manager는 다시 SNMP Engine에게 다시 그 요청을 보내주게 된다.

에이전트로부터 트랩이 발생했을 때는 제일 먼저 SNMP Engine안에 있는 트랩 데몬이 그 정보를 받게 된다. 다시 그 정보는 Device Manager에게 보내지게 되고 Device Manager는 트랩에 대한 정보를 판단한 후 Logging 해야 하는 정보는 Logging 모듈로 정보를 보내주게 된다.

3.6.2 SNMP Engine 모듈

SNMP Engine은 TMS800에서 SNMP에 관한 처리를 담당하는 모듈이다. Device Manager가 주기적으로 각 장비의 현재 정보를 원하게 되면, SNMP Engine은 해당하는 장비로 SNMP Get 요청 메시지를 보내어 값을 가져와서 Device Manager에게 다시 값을 보내주게 된다. 또한, 장비로부터 TRAP이 발생했을 때 그 정보를 받아서 Device Manager에게 보내주게 된다.

SNMP Engine는 joeSNMP[15]라는 Java용 SNMP API를 사용하여 작성되었다. joeSNMP는 OpenNMS [12]라는 open NMS 제작을 위한 프로젝트에서 만들어진 것이다. joeSNMP는 현재 0.3.3버전까지 릴리즈 되었으며 SNMPV2c까지 지원을 하고 있다. 이벤트 기반의 동작을 하기 때문에 UDP를 사용하는 SNMP에서 효과적으로 사용할 수 있는 장점이 있다.

3.6.3 Device Manager 모듈

각 장비들은 TMS800안에서 DeviceMeatadataBean 클래스의 인스턴스로 존재하게 된다. 장비와 Device-MetadataBean 클래스의 인스턴스는 1:1의 관계를 갖게 되는데 DeviceMeatadataBean 클래스의 인스턴스를 관리해주는 역할을 Device Manager에서 담당하게 된다. 즉, Device Manager는 장비에 관한 정보를 관리하는 모듈이다. 또한, 각각의 모듈들이 장비의 상태와 정보를 원하게 되면 장비의 상태와 정보를 전달해주는 역할도 하게 되며 각 장비들이 제대로 동작하는지 관리 감독하는 역할도 하게 된다.

Device Manager에 장비를 등록 하는 과정은 Meatadata Builder에서 장비를 메타데이타화 하는 과정의 역순이다. Meatadata Builder에서 만들어진 데이타를 TMS800에서 읽어 들이면 Device Manager는 직렬화된 객체를 다시 원래 객체로 만든 다음 보관하게 되는 과정을 거친다.

3.6.4 MIB Parser 모듈

MIB Parser는 MIB 파일을 파싱해서 메모리로 읽어오는 역할을 한다. MIB Parser는 AgentAPI[16]라는 Java용 공개 API를 통해 만들어졌다. AgentAPI를 이용하여 MIB를 파싱하게 되면 MIB안에 정의되어 있는 managed object들은 트리형식으로 객체화되어 언제든지 접근가능하게 된다.

3.6.5 Persistence 모듈

TMS800은 상태를 유지할 필요가 있다. TMS800이 실행되는 머신을 재부팅 할 필요도 있을 것이고 불의의 사고로 머신이 작동을 하지 않을 수도 있다. 또한, 현재 상태 그대로 다른 머신에서 TMS800을 실행할 경우도 생길 수 있기 때문에 현재 어플리케이션의 상태를 그대로 유지할 필요가 생긴다. Persistence 모듈은 이와 같은 기능을 담당한다.

Persistence 모듈의 구현 방식도 메타데이타 빌더와 마찬가지로 객체의 직렬화를 통해서 구현되었다. 그림 8은 TMS800에서 Persistence를 지원하기 위한 Persistence 모듈의 클래스 다이어그램이다. 먼저 보아야 할 클래스가 TMZBean 클래스이다. TMZBean 클래스는 TMS800에서 관리하는 피 관리 장비를 추상화한 것이다. 따라서 TMZBean 클래스에서는 DeviceMetadata-Bean에 관한 레퍼런스를 가지고 있다. 거기에 실제 TMS800에서 사용자 GUI 세팅에 관한 부분이 추가되었다. 3.5.8절에서 GUI 모듈에서 설명하겠지만 사용자가 화면에서 보고 싶어 하는 것은 실시간으로 변하는 방송 장비의 관리 객체에 관한 정보이다. TMS800은 이 정보를 3개의 테이블을 사용해서 사용자에게 보여준다. 하지만, 모든 관리 객체에 관한 정보를 사용자에게 한 번에 보유 줄 수는 없다. 장비가 추가될수록 관리객체의 수는 한 화면에 보여 줄 수 없을 만큼 증가되기 때문이다. 또한, 특정한 시점에만 값을 보고 싶은 관리 객체도 있을 수 있기 때문에 어떤 관리 객체를 보고 싶은지는 사용자가 선택해야 할 문제이다. 이에 관한 사용자의 세팅을 보관하고 있는 필드 변수가 setupList, statusList, summaryList이다. 이게 관한 자세한 설명은 3.5.8절의 GUI모듈에 명시되어 있다. APPStatusBean은 HashMap을 상속하고 있는 TMZBean을 담는 컨테이너 클래스이다. id는 TMZBean이 참조하고 있는 DeviceMetadata-Bean의 id(피 관리 장비의 이름)를 사용한다. App-StatusBean에는 appConfiguration라는 필드 변수가 존재한다. 이 변수의 타입은 TMS800Configuration-DataBean 인데 이 클래스에는 TMS800 시스템의 전체적인 설정 정보가 보관되어 있다. 따라서 AppStatus-Bean이 직렬화의 과정을 거치게 되면 시스템의 전체적인 설정과 피 관리 장비의 모든 정보를 파일로 저장 할 수 있게 된다.

3.6.6 Graph Chart 모듈

Graph Chart는 장비의 특정 OID의 특정 시간 동안 값의 변화를 그래프로 나타내주는 모듈이다. Graph Chart를 설계할 때 가장 중요시해야 하는 점은 User Interface이다. Graph를 써서 Chart를 보여주는 것 자

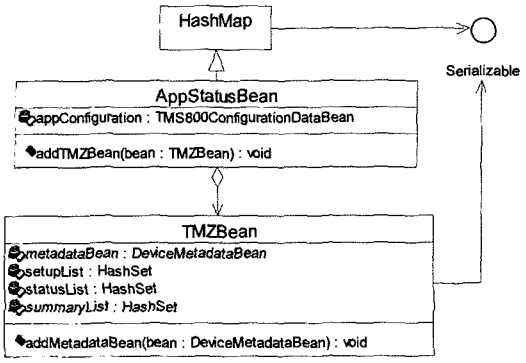

그림 8 Persistence 모듈의 클래스 다이어그램

체가 유저에게 시각적인 효과를 극대화 하는 것에 그 존재의 이유가 있기 때문이다. 이와 같은 사항을 준수하기 위해서 TMS800에서는 다음과 같은 사항들을 고려해서 graph chart 모듈을 설계하였다.

- 해당 OID에 임계값이 설정되어 있다면 그래프의 upper bound와 lower bound를 임계값의 가장 극단의 값으로 설정한다. 이렇게 하면 해당 OID의 값이 임계값을 초과할 때 그래프를 벗어나기 때문에 유저는 한눈에 임계값이 초과됐다는 것을 알 수 있다.
- 해당 OID마다 창을 따로 띄워서 볼 수 있게 했다. 단일 OID의 시간상의 값의 변화도 중요하지만 때로는 특정 OID들끼리 시간상의 값의 변화를 비교해야 할 때도 충분히 생길 가능성이 있기 때문이다. 따로 창의 띄우게 되면 비교하기가 훨씬 수월해진다.
- 창의 타이틀은 OID와 장비의 이름을 조합해서 만들어 준다. 창이 여러 개 띄워졌을 때 구별하기 쉽게 하기 위해서다.

Graph Chart는 JFreeChar[17]라는 Java용 공개 API로 만들어졌다. JFreeChart는 파이 차이나 바 차트 등 여러 가지 차트를 지원할 뿐만 아니라 특수한 용도에 맞게 차트를 커스터마이징해서 만들어 줄 수 있기 때문에 재사용 측면에서 효과적이다. 또한 차트에 필요한 여러 가지 요소들이 추상화가 잘 되어 있기 때문에 JFreeChart에서 지원하지 않는 기능이 있다 해도 새로운 클래스를 추가하기에 용이하다.

3.6.7 Logging

모니터링 시스템에 로깅 모듈은 상당히 중요한 역할을 차지한다. 중요한 사항부터 사소한 사항까지 유저가 원하는 사항들을 남겨줄 수 있는 기능을 로깅 모듈에서 담당하기 때문이다.

로깅 정보에 대한 관리를 위해서 DBMS를 사용하였다. 현재 TMS800에서는 사용자의 편의를 위하여 Microsoft사의 Access를 사용했지만 언제든지 변경이 가능하다.

```
public static synchronized void logMsg(
        int msgType,
        String equipmentName,
        String val,
        int alarmType)
```

그림 9 logMsg 메서드의 프로토타입

로깅에 관한 대부분의 사항은 TMS800Logging 클래스에서 담당한다. TMS800Logging 클래스가 하는 주된 작업은 DB Connection과 Device Manager로부터 로깅 데이타를 넘겨받는 것이다. 이중에 로깅을 남기기 위해 주로 사용하는 메서드가 logMsg이다. logMsg 메서드의 프로토타입은 그림 9와 같다.

logMsg 메서드에는 4개의 인자가 있다. 각각의 인자가 나타내는 것은 다음과 같다.

- msgType: 해당 로깅이 어떤 종류인지 명시한다. (예) 임계값 초과, 트랩 발생 등
- equipmentName: 로깅이 발생한 장비의 이름을 명시한다.
- val: 해당 로깅의 값을 넣어주는 부분이다. 사실 로깅의 종류가 다양하기 때문에 String타입으로 설정되어 어떤 종류의 값도 남길 수 있게 디자인 되었다.
- alarmType: 해당 로깅의 경고 수준(Normal~Fatal)

3.6.8 GUI

GUI는 효율적이면서 편해야한다. 즉, 사용자의 입력 행위는 최소화하고 출력되는 정보는 최대화해야 한다. 또한, 출력되는 정보 중 사용자가 필요로 하는 부분을 사용자는 최소한의 시간에 취할 수 있어야 한다. TMS800에서는 최대한의 정보를 최소한의 시간에 사용자에게 보여 주기 위해서 Table을 적극적으로 활용하였고 수많은 정보 중 사용자가 원하는 정보를 최소한의 시간에 얻게 하기 위해서 소팅과 컬러링을 적극적으로 사용하였다.

3.6.8.1 메인 화면

TMS800의 메인 화면은 크게 6부분으로 나눠 볼 수

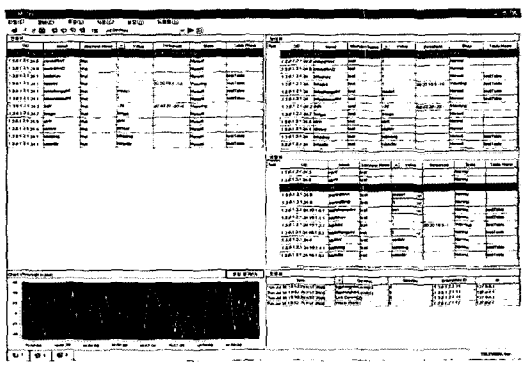

그림 10 TMS800 메인 화면

있다. 6개의 화면은 서로 크기가 정해져 있지 않고 경계점을 유동적으로 변경해서 화면 크기가 유기적으로 변경 가능하기 때문에 사용자는 특정 부분에 집중해서 화면을 구성할 수 있다. 다음은 6개 화면에 대한 설명이다.

- 메뉴: 상단에 위치한다. 파일, 장비, 로깅, 차트, 설정, 도움말의 카테고리로 나눠지고 각 카테고리는 세부적인 사항을 수행할 수 있는 부 메뉴로 구성되어 있다. 메뉴를 구성하고 디자인 하는데 있어서 가장 중점을 둔 것은 모든 기능을 메뉴에서 수행할 수 있어야 한다는 것이다. 도움말이 지원되기는 하지만 도움말이 없이도 사용자가 모든 기능을 쉽게 사용할 수 있게 하는 것이 좋은 인터페이스이다. 메뉴에서 모든 기능을 수행 할 수 있게 함으로써 사용자가 좀 더 쉽게 TMS800을 사용할 수 있게 하였다.
- Summary View: 중단 왼쪽에 위치한다. 각 행의 헤더를 누르게 되면 소팅이 되기 때문에 사용자가 원하는 정보를 빠르게 찾아볼 수 있다. 또한, 각 행은 언제든지 테이블에서 제외시킬 수도 있고 포함 시킬 수도 있기 때문에 특정 항목에 집중해서 값을 보여줄 수 있다. 행은 다음과 같이 총 7개의 항목으로 구성되어 있다.
  - OID: 관리 객체의 ID를 나타낸다.
  - Name: 관리 객체의 이름을 나타낸다.
  - Machine Name: TMS800에서 각 장비를 인식하는 유일한 이름이다. 즉, 메타데이타 빌더에서의 장비 이름은 장비를 제작하는 측에서 부여하는 이름이지만 이 이름은 TMS800을 사용하는 사용자가 부여하는 이름이다.
  - Value: TMS800이 SNMP의 Get 요청을 통해서 가져온 관리객체의 값이다.
  - Threshold: 관리객체에 임계값이 설정되어 있다면 이 항목에 값이 나타난다. 표시하는 방식은 가장 큰 임계값부터 가장 작은 임계값으로 콜론을 사용해서 구분해서 보여준다. 예를 들어 50:40:30:-10:-30:-50 이라고 표시되어 있다고 하자. 관리 객체의 값이 50을 초과 하거나 -50 미만 일 때 Fatal의 상태를 가지게 되어 해당 열은 배경색이 빨간색이 된다. 또한 30미만 -10 이상 일 때 Normal 상태를 가지게 되어 해당 열의 배경색은 흰색이 된다.
  - State: 해당 관리 객체의 상태를 나타낸다. Fatal, Error, Warning, Normal의 총 4가지 상태로 구분이 되며 각각의 상태로 인해 해당 열의 배경색이 결정된다. 각 상태의 바탕색은 Fatal은 빨간색, Error는 주황색, Warning은 하늘색, Normal은 흰색이다.
  - Table Name: 관리 객체가 MIB 설정에 의해 SEQUENCE의 Entry라면 해당 SEQUENCE의 이름이 나타난다.
- Status View: 중단 우측 위쪽에 위치한다. 기본적으로 Summary View와 같은 화면으로 구성되어 있다. Summary View와 구별되는 점은 view의 좌측에 TMS800에서 관리하는 모든 장비가 탭으로 구성되어 있어서 한 탭은 하나의 장비만을 집중적으로 보여준다는 점이다. 사용자가 특정장비에 관심이 많다면 Status View가 해답이 될 것이다.
- Setup View: 중단 우측 아래쪽에 위치한다. Summary View와 Status View가 SNMP Get Request에 관한 부분이었다면 Setup View는 SNMP Set Request에 관한 부분이라 할 수 있다. Setup View의 표현 방식도 Summary View나 Status View와 크게 다르지 않다. 다른 점은 Value 행에 있다. Value 행에서 해당 관리객체의 값을 사용자가 설정해 줄 수 있다. 값을 설정하는 방법은 두 가지로 나눠볼 수 있다. 해당 관리객체의 값에 이름이 정해져 있거나 값의 범위가 정해져 있다면 해당 Value 행에는 콤보 박스가 나타난다. 콤보 박스에서 해당 값을 선택하게 되면 TMS800이 해당 장비의 에이전트에 SNMP Set Request를 보내게 된다. 해당 관리 객체의 값에 이름도 없고 범위도 없다면 텍스트 필드가 나타나서 사용자가 직접 값을 입력할 수 있다. 사용자가 값을 입력한 후 엔터를 누르게 되면 SNMP Set Request가 발생하게 된다.
- Chart View: 하단 왼쪽에 위치한다. 사용자가 차트 메뉴에서 보고 싶은 차트를 선택하게 되면 TMS800은 해당 관리 객체에 관한 차트 창을 따로 띄워서 사용자에게 보여준다. 이런 식의 작업을 여러 번 하게 되면 화면에는 수많은 차트 창이 존재하게 된다. 특히, 사용자는 TMS800의 메인 창에 항상 관심을 가지게 될 것이고 항상 최대화해서 보려고 할 가능성이 높다. 이렇게 되면 기존에 띄워 놓은 차트 창들은 TMS800의 메인 화면에 가려서 보이지 않게 된다. 이런 단점을 극복하기 위해서 Chart View에서는 특정 차트를 보여 줄 수 있는 기능을 제공한다. 메뉴에서 차트를 Chart View에 도킹하라는 명령을 사용자가 내리게 되면 해당 Chart 창은 Chart View에 도킹 되어 다른 Chart 창이 화면 뒤쪽으로 가려지더라도 계속해서 볼 수 있게 된다.
- Trap View: 수많은 에이전트로부터 발생하는 트랩을 보여주는 부분이다. Trap View도 테이블을 사용해서 해당 정보를 보여주고 해당 열의 헤더를 통해서 소팅을 할 수 있다. Trap View의 열은 총 5개로 구성되어 있다.
  - Time: 트랩이 발생한 날짜와 시간을 보여준다.

- Generic: SNMP Framework에서는 트랩 중 일반적인 트랩 몇 가지를 정해 놓았다. 이런 트랩들이 발생할 경우에 이에 관한 정보를 표시하는 열이다.
- Specific: Generic외에 특별한 트랩이 발생할 경우 이에 관한 정보를 보여주는 열이다.
- Enterprise ID: 트랩이 발생한 관리 객체의 OID를 표시한다.
- IP: 트랩이 발생한 장비의 IP 주소를 나타낸다.

3.6.8.2 설정 화면

설정 화면에서는 3가지 View(Summary, Status, Setup)에서 사용자가 보고 싶은 관리객체를 설정해 주는 화면이다. 크게 4개의 탭으로 구성되어 있다.

- Summary View 설정: 그림 11은 Summary View의 설정 화면이다. 화면이 좌우로 구성되어 있다. 좌측 화면에는 장비를 선택할 수 있는 탭들과 탭에 의해 선택된 장비의 MIB Tree를 보여 준다. MIB Tree에서 관리 객체 하나를 선택한 후 추가 버튼을 눌러주면 우측 화면의 관리객체 리스트에 복사된다. 관리객체 리스트 박스에 있는 관리객체의 엔트리들이 Summary View에 실제적으로 보이는 관리객체들이다. 관리 객체 리스트 박스에 있는 관리객체의 엔트리를 삭제하고 싶을 때는 관리 객체 엔트리 하나를 선택한 후 삭제 버튼을 눌러주면 된다.

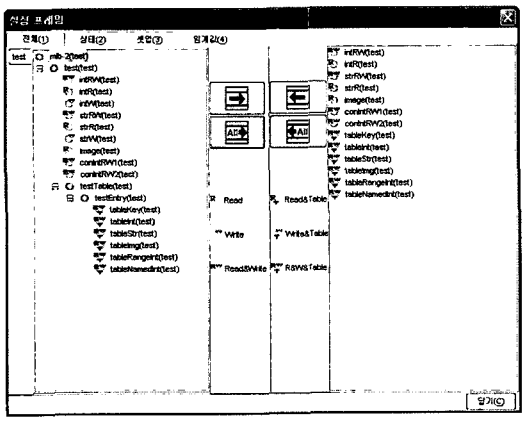

그림 11 Summary View 설정 화면

- Status View 설정: Status View 설정의 기본적인 구성은 Summary View 설정의 구성과 크게 다르지 않다. 한 가지 다른 점은 Status View는 장비별로 관리 객체를 구분해서 보여주기 때문에 Status View 설정의 우측 화면에 있는 관리 객체 리스트 박스도 장비에 따라서 따로 보여준다는 점이다. 관리 객체 리스트 박스에 추가된 관리 객체들을 Status View에서 볼 수 있다.
- Setup View 설정: Status View 화면과 동일하다. 관리 객체 리스트 박스에 추가된 관리 객체들을 Setup View에서 볼 수 있다.
- Threshold 설정: 그림 12는 Threshold 설정 화면이다. Threshold 설정도 다른 설정들과 마찬가지로 좌우로 구분 되어 있고 좌측에는 장비 탭과 MIB Tree가 나타나고 오른쪽에는 관리객체 리스트를 보여준다. 특이한 점은 관리 객체 리스트 아래에 임계값을 설정하기 위한 화면이 있어서 사용자는 관리 객체 리스트에서 하나의 엔트리를 선택한 후에 임계값을 설정한 후 적용 버튼을 누르게 되면 해당 관리 객체에 대한 임계값이 변경된다는 점이다.

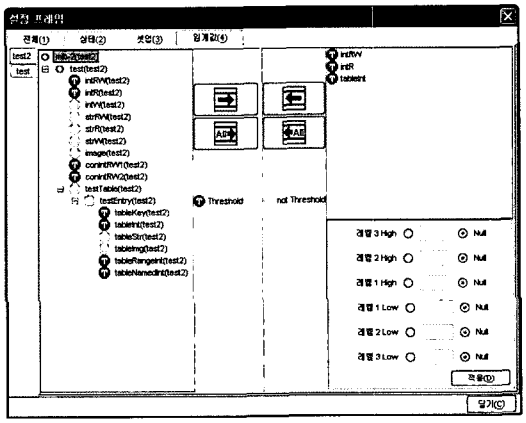

그림 12 Threshold 설정 화면

3.6.8.3 이미지를 표현하는 관리 객체 모니터링 화면

현재 방송중인 이미지를 보여주는 화면이다. 피 관리 장비의 메타 정보에는 특정 관리 객체가 이미지로 표현될 것인지에 대한 정보가 들어있다. 이에 관한 정보는 Device Manager에서 관리하게 되는데 GUI에게 이미지에 관한 정보를 알려주기 때문에 GUI는 같은 OCTET STRING 타입이라 하더라도 화면상에서 그림으로 보여줄 수 있다. 그림 13은 TMS800에서 실제로 이미지가 보이는 모습이다. Value 행을 보면 위의 열은 검은 원이 떠있는 것을 볼 수 있다.

OID	Name	Machine Name	Value	Threshold	State
6.1.2.1.25.1	image	testImage	[redacted]		Normal
6.1.2.1.24.1	intRW	testDevice	-7	40:30:20:10:-20:...	Warning

그림 13 TMS800에서 이미지를 보여주는 화면

4. 결 론

본 논문에서는 메타데이타를 사용해서 방송 장비를 관리할 수 있는 시스템에 대해서 설계하고 구현하였다.

3.1에서 언급하였던 두 가지 구현 목표는 다음과 같은 방법으로 해결이 가능하였다.

- 개별화 된 방송용 장비의 MIB 지원: 메타데이타 빌더를 사용하여 각 방송용 장비의 메타데이타에서 MIB를 보관하게 된다. 이 값은 후에 TMS800에서 직접 읽어 들여서 파싱 후 사용가능한 상태가 된다.
- 이미지를 직접 표현 할 수 있는 관리 객체 모니터링 화면 지원: 메타데이타 빌더에서 사용자는 각 OID별로 이미지 유무를 체크 할 수 있다. 이 체크된 사항은 값으로 만들어져서 메타데이타에 보관되고 후에 TMS800에서 이 값을 사용하여 OCTET STRING 타입의 값을 이미지로 보여줄것인지에 관한 사항을 판단하게 된다. 즉, 사용자 체크> 메타데이타> Device Manager> GUI 순으로 값이 전달되게 된다.

TMS800은 자바 플랫폼을 사용하여 이식성을 극대화 하였으며, 메타데이타를 사용함으로써 기존의 SNMP를 사용한 범용 관리 시스템에서 구현하기 쉽지 않았던 기능들을 쉽게 구현 할 수 있었다. 그리고 메타데이타를 사용자가 훨씬 편하고 신속하게 만들기 위해서 메타데이타 빌더를 설계하고 구현하였다. 메타데이타 빌더는 향후 변경에 유연하게 대처하도록 객체의 직렬화 기능과 Builder Pattern을 사용하여 디자인 하였다. 또한, 방송 장비 관리 시스템은 기능적인 요소들을 분리하여 모듈화 함으로써 향후 기능이 효과적으로 추가 될 수 있도록 하였다. SNMP Engine 모듈이 SNMP 커뮤니케이션에 관한 모든 것을 담당하였다면 GUI 모듈은 사용자와의 커뮤니케이션에 관한 모든 것을 담당하였다. 또한, Persistence 모듈은 어플리케이션의 영속성을 책임지고 있으며 Logging 모듈은 시스템의 모든 로깅에 관한 데이타를 유지 관리하는 기능을 하였고 Graph Chart모듈은 각 관리객체의 값을 시각적으로 보여 주는 기능을 하게 되었다. Device Manager 모듈은 각 방송장비의 메타데이타들을 통합 관리하는 책임을 가지고 있으면서 다른 모듈들에게 필요한 데이타를 보내주는 역할도 함께 담당하였다. 마지막으로, MIB Parser 모듈은 Device Manager 모듈로부터 각 장비의 MIB를 받아 시스템에서 사용 할 수 있게 파싱하는 역할을 수행하였다.

TMS800은 현재 오디오 기능을 지원하지 않는다. 즉, 현재 방송중인 오디오를 보내 줄 수 있는 에이전트가 있다 해도 현재 시스템에서는 지원 할 수 없다는 말이다. 앞으로 오디오까지 지원하게 된다면 좀 더 실용적인 방송장비 관리 시스템이 될 수 있을 것이다.

참 고 문 헌

[1] J. Case, R. Mundy, D. Partain, B. Stewart, Introduction and Applicability Statements for Internet-Standard Management Framework, RFC 3410, December 2002.

[2] Miller, Mark A., Managing internetworks with SNMP, M&T Book, 1993.

[3] Douglas Mauro, Kevin Schmidt, Essential SNMP, O'REILLY, 2001.

[4] K. McCloghrie, M. Rose, Management Information Base for network management of TCP/IP-based internets, RFC 1156, May 1990.

[5] M. Rose, K. McCloghrie, Concise MIB definitions, RFC 1212, March 1991.

[6] K. McCloghrie, M. Rose, Management Information Base for Network Management of TCP/IP-based internets:MIB-II, RFC 1213, March 1991.

[7] M. Rose, K. McCloghrie, Structure and identification of management information for TCP/IP-based internets, RFC 1155, May 1990.

[8] International Organization for Standardization, Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation-Third Edition; Amendment 2, ISO/IEC 8824-1, 2005-07-15.

[9] International Organization for Standardization, Information Technology - ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)-Third Edition; Amendment 1,ISO/IEC 8825-1, 2004-10-15.

[10] J. Case, M. Fedor, M. Schoffstall, J. Davin, Simple Network Management Protocol (SNMP), RFC 1157, May 1990.

[11] http://www.openview.hp.com

[12] http://www.opennms.org/index.php/Main_Page

[13] Sun Microsystems, Java Object Serialization Specification-version 1.5.0, 2004.

[14] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, Design Patterns - Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995.

[15] http://sourceforge.net/projects/joesnmp/

[16] http://nms.estig.ipb.pt/agentapi.web/index.jsp

[17] http://www.jfree.org/jfreechart/index.php

김 민 석

2004년 2월 한양대학교 전자계산학과 학사. 2006년 2월 한양대학교 컴퓨터공학과 석사. 2006년 1월~현재 LG전자 홈넷 사업팀 근무

최 정 호

2002년 2월 한양대학교 전자계산학과 학사. 2004년 2월 한양대학교 컴퓨터공학과 석사. 2003년 12월~현재 한화 S&C CS센터 근무

김 정 선

1986년 서울대학교 컴퓨터공학과 졸업(학사). 1988년 Iowa State University 전기 및 컴퓨터공학과 졸업(석사). 1994년 Iowa State University 전기 및 컴퓨터공학과 졸업(박사). 1994~1996년 한국전자 통신연구원(ETRI) 선임연구원. 1996년~현재 한양대학교 전자컴퓨터공학부 부교수. 관심분야는 Parallel/Distributed Processing, Distributed Object Computing, Component Based Development