유비쿼터스방송 환경의 멀티 프로토콜 에이전트 통합 네트워크에 관한연구

*정창덕 **김대영 ***김도형
고려대학교
jcd1234@paran.com

A study on An Integrated Network Management System Using Multi-Protocol Agents in Ubiquitous Broadcasting Environment

*Jung, Chang-Duk **Kim, Dae-Young ***Kim, Do-Hyung

< Abstract >

Integrated management model (SNMP/SMI) for lubiquitous egacy remote communication network or service make possible combination of various architecture. However, legacy management system cannot be applied some problems such as inefficient, complexly, implement and large network by reason of integration of voice and data, wired and wireless, and service area between service provider. For improve this, supplied JMX(Java Management eXtensions) on network management technology from SUN. JMX is integrated architecture for existing network management and monitoring. In this paper, we design and implement for integrated network management through multi-protocol agent using JMX.

Ⅰ. 서 론

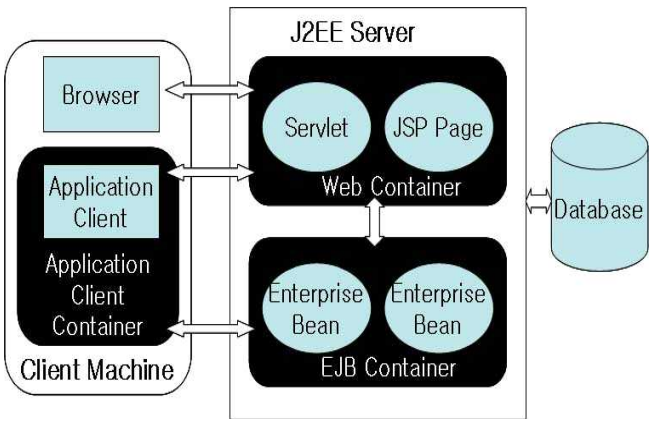
유비쿼터스환경의 네트워크는 기존의 공중사설망, 이동전화망, 패킷 망을 통합하여 수용할 수 있는 패킷 기반의 개방형 통신망이며, 네트워크 구조를 서비스 계층, 제어 계층, 전달 계층으로 분리하여 정의하고 있으며, 각 계층 간의 인터페이스는 개방 표준인터페이스를 정의하여 제공하고 있다. 이런 인터페이스의 정의는 Parlay 그룹에서 정의하고 있으며, 작업그룹에 의해 생성된 UML(Unified Modeling Language)은 JAIN(Java API for Integrated Network)에 의해 API(Application Program Interface)로 제공한다. 이러한 API로 인하여 다양한 네트워크(PSTN, Broadband, Wireless, IP, Satellite)에서 응용을 동작할 수 있도록 하고, 벤더와 서비스 제공자들에게 독립적으로 동작할 수 있도록 한다. 그러나 네트워크 통합을 추구하면서 문제가 되고 있는 것이 네트워크 관리이다. 기존 원격통신 네트워크나 서비스를 위한 관리는 통합된 관리 모델(SNMP/SMI)을 사용함으로써 다양한 기반 구조의 통합을 가능하게 하였다. 그러나 음성과 데이터의 통합, 유선과 무선의 융합, 사업자간 서비스 영역 구분으로 인하여 현 관리 시스템들은 비능률적이고, 복잡하며, 구현 및 큰 네트워크에는 적용하기 어렵다. 이를 실현하기 위하여 SUN에서 네트워크 관리 기술인 JMX(Java Management eXtensions)를 제공하고 있다. JMX는 자바 프로그래밍 언어에서의 어플리케이션과 네트워크 관리 및 모니터링을 위한 아키텍쳐, 디자인 패턴, API, 그리고 서비스를 정의한다. 또한, JMX 스펙은 자바 코드에 적용해 스마트 자바 에이전트를 만들고, 분산 관리 미들웨어 및 관리자를 구현하며, 이들 솔루션을 현존하

는 관리 및 모니터링 시스템으로 부드럽게 통합하는 수단을 업계의 모든 자바 개발자들에게 제공한다. 또한, 현존하는 표준 관리 및 모니터링 기술에 대한 여러 Java API가 JMX 스펙을 언급하고 있다. 따라서 본 논문에서는 JMX를 활용하여 통합 네트워크 관리를 할 수 있는 시스템을 설계 및 구현한다.

Ⅱ. 관련연구

2.1 J2EE(Java 2 Enterprise Edition)

J2EE는 J2SE(Java 2 Standard Edition) 기반이며, 컴포넌트를 이용한 분산환경 애플리케이션을 위한 표준이며, 애플리케이션 개발을 위한 자바 확장 API의 집합이다. <그림 1>은 J2EE 서버와 컨테이너의 구조를 보여주고 있다. 각각의 역할은 다음과 같다.



<그림 1> J2EE Server와 Containers

J2EE Server : J2EE의 런타임 부분으로써, EJB와 웹 컨테이너를 제공한다.

EJB(Enterprise JavaBeans) container : J2EE 애플리케이션을 위한 엔터프라이즈 빈의 실행을 관리하며, 엔터프라이즈 빈과 컨테이너가 J2EE 서버상에서 실행된다.

Web container : J2EE 애플리케이션을 위한 JSP 페이지와 서블릿 컴포넌트의 실행을 관리하며, 웹 컴포넌트와 그들의 컨테이너가 J2EE 서버상에서 실행된다.

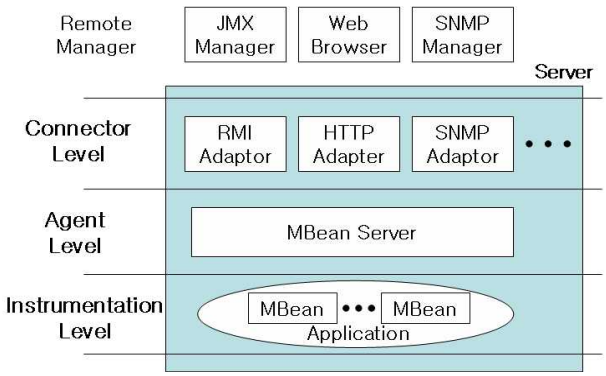
Application client container : 애플리케이션 클라이언트 컴포넌트의 실행을 관리하며, 애플리케이션 클라이언트와 그들의 컨테이너가 클라이언트에서 실행된다.

Applet container : 애플릿의 실행을 관리하며, 웹 브라우저와 자바 플러그인으로 구성된다.

J2EE 인터페이스는 데이터베이스를 위해 JDBC를, 디렉토리를 위해서는 JNDI를, 트랜잭션을 위해서는 JTA를, 메시징을 위해서는 JMS를, 전자우편시스템을 위해서는 JavaMail을, 그리고, CORBA와의 접속을 위해서는 JavaIDL을 각각 포함한다.

2.2 JMX(Java Management Extesions)

JMX는 관리와 관련한 단일하면서도 개방된 기술로 모든 산업에 곧바로 배포해 적용할 수 있도록 정의한 규격으로 기본적으로 레거시 시스템에 적용하기 쉽고, 새로운 관리 솔루션을 구현한 뒤에 이를 추가하는 좋은 확장성을 가지고 있다. 또한 JMX는 DMTF(Distributed Management Task Force)라는 표준화 단체에서 정의하는 WBEM(Web-Based Enterprise Management) 표준을 지키고 있으며, 웹 기반 분산, 모듈화 관리 장치나 애플리케이션 네트워크 관리 등에 대한 소프트웨어를 쉽게 구축할 수 있는 도구를 제공한다. JMX는 기본적으로 <그림 2>와 같이 세 개의 계층으로 구성되어 있다.



<그림 2> JMX의 구조

- 적용 레벨 (Instrumentation level)

JMX로 관리 가능한 자원을 구현하기 위한 스펙을 제공한다. 이러한 자원은 어플리케이션이나 서비스의 구현, 장치, 사용자 등이 될 수 있다. 이러한 자원은 JMX 호환 어플리케이션이 관리할 수 있도록 적용되어야 한다. 이렇게 주어진 자원의 적용은 하나 이상의 표준이거나 동적인 MBean(Managed Beans)에 의해 제공된다.

- 에이전트 레벨 (Agent level)

에이전트를 구현하기 위한 스펙을 제공한다. JMX로

관리 가능한 자원을 관리하는 표준화된 에이전트를 정의하기 위해 적용 레벨 위에 개발되어 적용 레벨을 사용한다. 또한, MBean 서버와 MBean들을 다루는 여러 서비스들(Timer, Monitoring, 등등)로 구성된다. 따라서, JMX 에이전트는 JMX로 관리 가능한 자원이 상주한 머신에 자바 가상 머신이 사용 가능하다면, 같은 머신에 임베디드할 수 있고, 사용할 수 없는 환경이면 중개자 역할을 할 수 있다.

- 분산 서비스 레벨 (Distributed services level)

JMX 관리자를 구현하는 인터페이스를 제공한다. 즉, 에이전트나 에이전트의 계층구조와 함께 작업할 수 있는 관리 인터페이스와 컴포넌트를 정의한다. 이들 컴포넌트들은 커넥터를 통해 에이전트와 JMX로 관리 가능한 자원을 투명하게 상호작용하는 관리 어플리케이션을 위한 인터페이스를 제공하며, 고수준 관리 플랫폼에서부터 수많은 JMX 에이전트까지 관리 정보를 분산시킬 수 있다.

그림 2와 같이 관리자 레벨과 다른 에이전트 및 적용 레벨의 조합은 완전한 관리 솔루션을 디자인하고, 개발하는데 필요한 완전한 아키텍처를 제공한다. 따라서, JMX 기술은 유일하게 이식성, 관리 기능의 on-demand 배치, 동적 및 이동성 서비스, 그리고 보안을 모두 제공해 준다.

2.3 JDMK (Java Dynamic Management Kit)

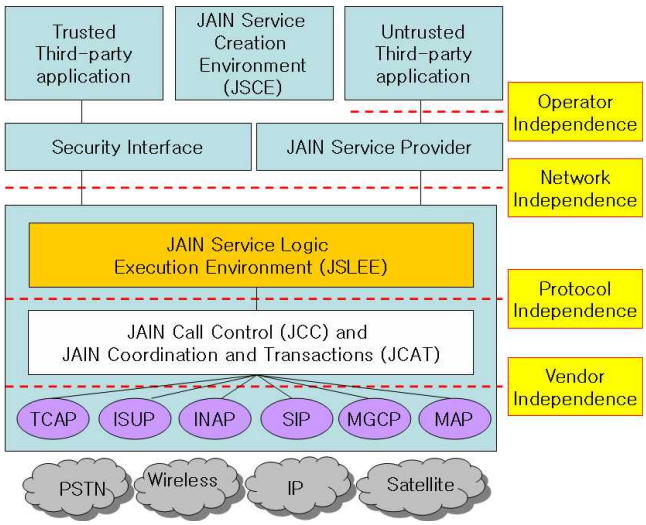
JDMK는 JMX와 호환되어 관리 어플리케이션을 설계하고 구현하기 위한 개발 툴이며 다중 네트워크 프로토콜을 통해 접근할여 관리할 수 있는 자바 API(Application Programming Inter- face) 이다. 또한, 자바 기술기반 어플리케이션을 통하여 자바 관리 오브젝트의 관리를 위한 프레임워크를 제공하며, 분산 관리 시스템을 설계하기 위한 완벽한 아키텍처를 제공한다.

JDMK는 SNMP 에이전트 개발과 관리자를 위한 툴킷을 제공한다. 이 툴킷은 SNMP MIB 컴파일러, 자바 소스 코드로 SNMP MIB을 컴파일하기 위해 사용하는 mibgen 툴과 기존 connector와 함께 사용하기 위한 프락시 오브젝트 생성자인 proxygen 툴이 포함되어 기존 프로토콜과 함께 사용할 수 있다.

2.4 JAIN (JAVA APIs for Integrated Network)

JAIN은 차세대 망에서 서비스를 개발해서 도입하는데 필요한 중요한 기술로서, 표준화된 자바 인터페이스를 통해서 유선, 무선, 패킷 망을 넘나드는 서비스 혹은 어플리케이션들에 JAIN 컴포넌트를 실현시키기 위한 환경을 제공하는 API의 집합이며, 네트워크 로직과 서비스 로직을 분리하여 유선, 무선, 패킷 망을 통합시킨다.

JAIN은 JAVA기술을 사용함으로써 차세대 통신 서비스를 보다 빠르고, 쉽게, 저비용으로 개발할 수 있도록 하며, JAVA 어플리케이션이 공중망 자원으로 접근할 수 있도록 한다. 특성화된 폐쇄적 시스템으로부터 발전된 통신망 시장을 서비스가 신속하게 생성되고 도입될 수 있는 통합된 단일 네트워크 구조를 갖도록 한다. JAIN의 처리 구조와 독립성은 <그림 3>에서 표현하고 있다.



<그림 3> JAIN의 구조

지능망의 SCE(Service Creation Environment)와 SLEE (Service Logic Execution Environment)에 JAVA를 추가하여 JSCE (JAIN Service Creation Environment)와 JSLEE (JAIN Service Logic Execution Environmnet)를 정의하였다. JSPA(JAIN Service Provider API)에 의해 제 3 응용 어플리케이션들을 수용한다. JSPA에 의해 오퍼레이터 독립을 제공할 수 있다. JAIN SLEE는 이벤트 타입(RequestEvents, ResponseEvents and TimeoutEvents)에 의해서 이벤트들을 식별하며, JAIN (Java API for Integrated Network) Service Provider APIs(SPA)는 Parlay 그룹의 목적인 개방형 기술을 개발 표준으로 제정하여 네트워크 운영자 API를 제공한다. 이러한 API로 인하여 다양한 네트워크(PSTN, Broadband, Wireless, IP, Satelite)에서 어플리케이션을 동작할 수 있도록 하고, 벤더와 서비스 제공자에게

독립적으로 동작할 수 있도록 해준다.

Ⅲ. 유비쿼터스환경의 방송 통합망 관리 시스템 설계

3.1 설계 원칙

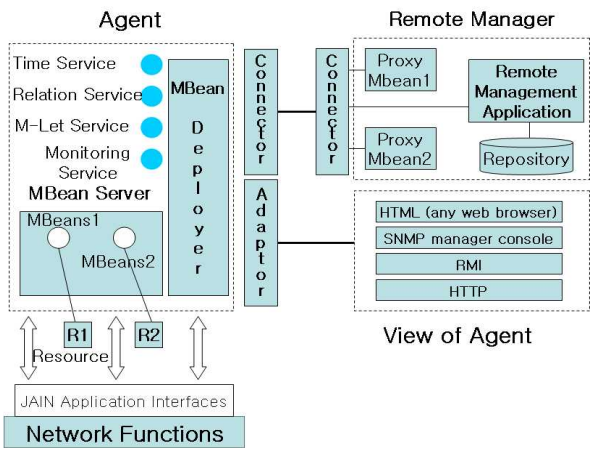
통합 네트워크 관리 시스템을 설계하기 위하여 고려할 점은 다음과 같다.

네트워크 통합 관리를 위하여, 어떤 네트워크에서든지 관리를 할 수 있어야 한다. 따라서, 하위 계층을 분리시켜야 하며, 하위 계층과 통신을 하기 위한 인터페이스가 필요하다. 기존 SNMP와 같은 프로토콜과 통합이 가능한 구조가 되어야 한다. 여러 서비스들이 지원 가능한 구조를 가져야 한다. 어플리케이션이든지 하드웨어이든지 모두 관리를 할 수 있는 구조를 가져야 한다. 응용계층에서 자원 구성들에 대한 정보를 가지고 있어야 한다.

위와 같은 설계 원칙을 따르도록 시스템이 설계되어야 한다. 본 논문에서 설계한 시스템은 이 모두를 만족시켜주는 시스템이다.

3.2 시스템 설계

<그림 4>는 본 논문에서 설계한 구조이며, 3.1절에서 제시한 설계 원칙을 따르는 통합 네트워크 관리 시스템이다.



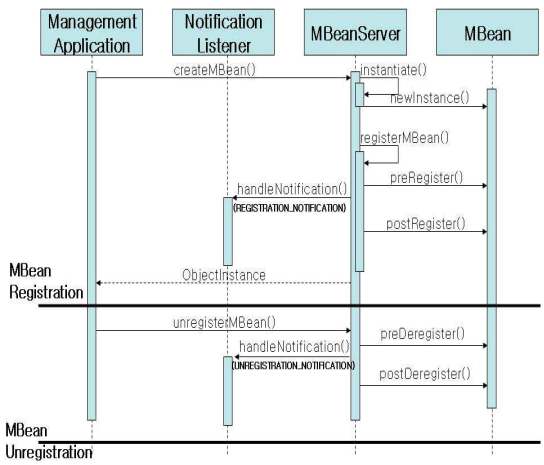
<그림 4> 통합 네트워크 관리 시스템

여러 네트워크를 관리하기 위해서 하위 계층과는 JAIN 어플리케이션 인터페이스를 통하여 통신하여 어플리케이션 계층에서 자원을 구성할 수 있도록 하였다. JAIN은 표준화된 자바 인터페이스를 통해 개방 소프트스위치 구조를 구축하고, 서비스를 제공하는 기술로, 네트워크 기반의 로직과 서비스 기반 로직을 분리해 어떤 망이든 서비스 제공이 가능하다.

에이전트는 MBean 서버에 등록되어 있는 MBean들을 관리하기 위해 time, monitoring, alert, event등의 서비스들을 지원해 준다. 또한, 다른 프로토콜(SNMP, HTTP)과 함께 통신하기 위해서는 JMX adaptor를 사용하며, 다른 전송기술(Java RMI)등과 연결하기 위해서는 connector를 사용함으로써 쉽게 확장이 가능하고, 존재하는 기존 기술들과 독립적으로 통신이 되도록 하였다. 또한, 프락시는 JDMK에서 제공되는 proxygen 툴을 사용하는데, proxygen은 기존 connector와 함께 사용하기 위한 프락시 오브젝트로써, 원격에서 MBean을 관리할 수 있도록 해주며, 모든 통신을 핸들링하게 해준다.

<그림 4>에서 보듯이, 통합 네트워크 관리 시스템은 웹을 통하여 관리하게 위한 HTML Adaptor와 SNMP 등과 같은 프로토콜을 지원하기 위한 Adaptor, 원격지 PC 자원과 연동하기 위한 Connector로 구분되어 있다.

MBean을 등록하고 해제하는 과정의 시퀀스 다이어그램을 <그림 5>에서 보여주고 있다. MBean 생성을 위해 createMBean() 메소드를 이용하여, 인스턴스를 생성하고, MBean을 registerMBean() 메소드를 이용하여 MBean Server에 등록하고, 등록이 되었다는 것을 Notifacation Listener를 통해 알려줌으로써, MBean을 등록하고, MBean을 해제하는 과정도 unregisterMBean() 메소드를 사용하여 MBean 등록과정과 유사하며, MBean을 unregister 하게 된다.



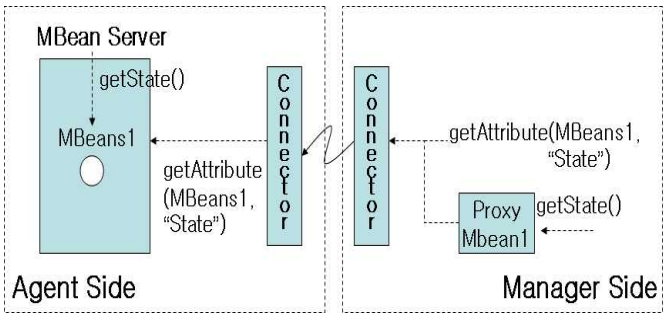
<그림 5> MBean 등록 및 해제 과정

MBean 서버에 등록된 MBean들은 <표 1>과 같은 관리 도구를 제공한다.

<표 1> 관리도구가 제공하는 기능

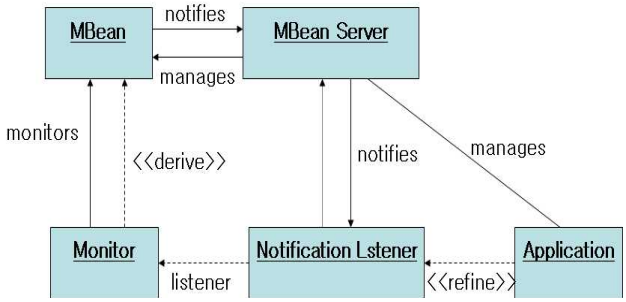
제공 기능	설 명
MBeanServer 연결	네트워크 상에 있는 MBeanServer 와 연결한다.
MBeanServer 정보	MBeanServer 에 등록한 MBean 정보를 본다.
MBean 정보	MBean의 오퍼레이션 정보를 보여준다.
이벤트 핸들러 등록	MBean이 발생하는 이벤트를 받을 수 있도록 핸들러를 등록한다.
오퍼레이션 실행	MBean의 오퍼레이션을 원격에서 구동한다.

<그림 6>은 MBean 서버를 통하여 어플리케이션에 의해 관리되는 MBean을 보여주는 시퀀스 다이어그램으로 MBean을 지속적으로 모니터링 하다가, MBean의 상태가 바뀌면, 등록된 리스너에게 통보해주는 그림이다.



<그림 6> MBean의 시퀀스 다이어그램

<그림 7>은 좀 더 세분화 하여 관리 동작이 원격 관리 어플리케이션으로부터 에이전트 측의 MBean으로 전달되는 것을 보여주고 있다. 관리 어플리케이션이 직접적으로 프락시 오브젝트상의 getState 메소드를 invoke하고, 프락시 오브젝트는 Connector 인터페이스를 통해 에이전트에게 request를 전송한다. 그러면, 에이전트의 Connector 인터페이스가 이 요청을 받아서 MBean 서버를 통하여 MBean의 상태를 읽는 과정을 보여준다. 그 다음에 역으로 가는 것도 동일한 방법을 통하여 전달된다. 또한 MBeans1에서 이벤트가 발생하면, JMX Agent를 통해 원격 관리자에게 이벤트를 알리게 된다.



<그림 7> 원격 관리 동작

3.3 멀티 프로토콜

멀티 프로토콜을 이용하여 통합 네트워크 관리를 위해 본 논문에서의 Adapter와 Connecter를 사용하였다. 각 모듈의 역할은 다음과 같다.

- HTML Adaptor Module

중앙 관리 시스템에 웹을 이용한 접근을 제공하기 위한 모듈이다. 관리자가 시스템에 접근하기 위해 거쳐야 할 인증과정은 저장소에 저장된 관리자 정보와 비교하여 일치할 때 접근을 허락하게 된다. 이 모듈을 통하여 레거시 시스템의 SNMP 에이전트와 원격지 JMX 에이전트에 요청한 정보에 대한 결과를 관리자가 보게 된다.

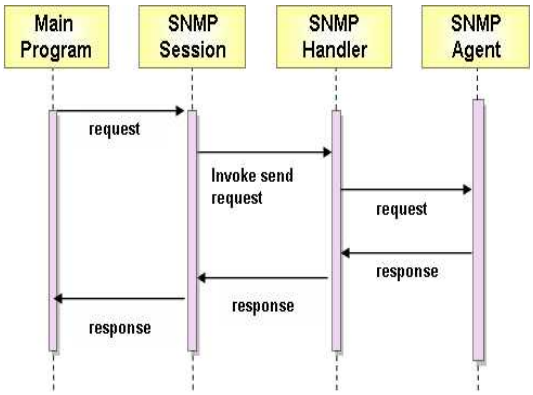
- SNMP Adaptor Module

레거시 시스템의 네트워크 장비에서 동작하고 있는 SNMP 에이전트와의 연동을 지원하기 위한 모듈이다. JDMK (Java Dynamic Management Kit)에 있는 SnmpAdaptor를 사용해도 되지만 상용이기 때문에 OpenNMS API를 사용하여 구현하였다. 그림 8은 SNMP API를 이용하여 데이터를 받는 순서를 나타내고 있다. SNMP 에이전트와 통신을 담당하는 SnmpSession에 데이터 요청 명령을 내리면 SnmpSession은 SnmpHandler를 통해 SNMP 에이전트에 데이터를 요청하게 된다. SnmpHandler는 단순한 인터페이스이므로 이것을 구현하여 SnmpSession에 등록해야 한다.

- Remote Agent Connector Module

원격지 에이전트와의 통신을 위한 모듈이다. 원격 에이전트도 중앙 관리 시스템과 마찬가지로 MBean들이 모여 MBean Server를 이루고, HTTP Adaptor Module도 존재하여 웹을 통한 접근이 가능하긴 하지만 데이터 전송을 위해서는 서버/클라이언트 구조가 되어야 하기 때

문에 이 모듈이 존재해야 하며, 서버에 해당된다.



<그림 8> SNMP를 통해 데이터를 받는 예

Ⅳ. 유비쿼터스환경의 통합 망 관리 시스템 구현

4.1 구현

통합 관리 시스템과 원격 에이전트는 일반 PC 상에서 구현 및 테스트하였으며, OS는 윈도우XP를 사용하였고, API로는 JDK 1.4.2, JMX 1.2, JMX Remote API 1.0, JDMK 5.0를 활용하여 구현하였다. 결론적으로 시스템 동작 환경은 PC나 서버 등 자바 가상 머신만 동작하고 있는 장비라면 어떤 곳에서도 동작한다.

<표 2> MBean Server와 HtmlAdaptor 생성 코드

```
// MBean 관리를 위한 MBeanServer 생성
System.out.println("₩n₩tCREATE the MBeanServer.");
MBeanServer server = MBeanServerFactory.createMBeanServer();
// HTML Adaptor 생성 및 시작
System.out.println
  ("₩n₩tCREATE, REGISTER and START a new HTML adaptor:");
HtmlAdaptorServer htmlAdaptor = new HtmlAdaptorServer();
ObjectName htmlObjName = null;
try {
// 8082 포트를 통해 서비스 제공
  htmlObjName = new ObjectName("Adaptor:name=htmlAdaptor
                    ,port=8082");
  System.out.println("NOTE: HTML adaptor is bound on " + "TCP
                     port 8082. ");
  System.out.println("₩tOBJECT NAME = " + htmlObjName);
  server.registerMBean(htmlAdaptor, htmlObjName);
} catch(Exception e) {
  System.out.println("₩t! Could not create the HTML adaptor !!!");
    e.printStackTrace();  return;
}
htmlAdaptor.start();
```

<표 2>는 웹 환경에서 관리하기 위한 HrmlAdaptor를 생성하는 코드와 MBean들을 관리하기 위한 MBeanServer를 생성하는 코드이다. HtmlAdaptor를 생성하여 MBeanServer에 등록하면 웹 페이지를 통해 프로그램이 동작하는 PC나 서버등의 시스템에 접속할 수 있게 되며, 구현한 MBean들을 등록할 수 있게 된다.

예를 들어 IP 주소가 128.134.64.35이고, 포트가 9092를 사용하는 시스템에 프로그램이 동작할 땔, http://128.134.64.35:8082 로 접속하면 등록된 MBean 들의 목록을 볼 수 있다.

<그림 9>는 Html Adaptor가 구현되어 있는 BaseAgent 프로그램을 실행시킨 화면이다.

<그림 9> BaseAgent 프로그램 실행 화면

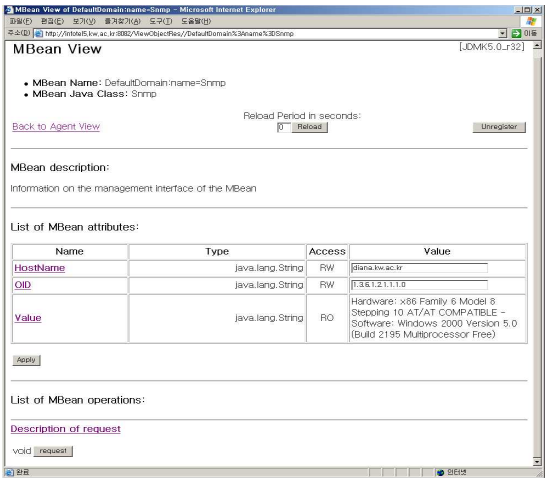
성공적으로 MBeanServer를 생성하고, Html Adaptor를 생성 후 등록하고 9092 포트를 사용하여 시작한다.

<그림 10>은 웹 페이지로 제공되는 사용자 인터페이스이다. 사용자가 원하는 기능을 하도록 구현한 MBean들을 추가/삭제할 수 있는 형태로 되어 있다. 새로운 MBean을 등록하고자 할 때는 Admin 버튼을 누른 후 등록하면 된다.

<그림 10> HTML 사용자 인터페이스

기존 네트워크와 함께 동작하기 위해서 SNMP Adaptor

와 원격 에이전트와 통신하기 위한 JMX Agent Connector를 구현하였다. 그림 11은 SNMP Adaptor를 구현한 MBean을 MBeanServer에 추가하여 SNMP 에이전트에서 원하는 정보를 얻어오는 것을 보여주고 있다. hostname이 diana.kw.ac.kr인 Windows 2000 Server에서 동작하고 있는 SNMP Agent의 하드웨어 정보를 얻어오기 위해 Object ID를 1.3.6.1.2.1.1.1.0로 설정하여 요청한 결과를 보여준다.



<그림 11> SNMP Agent와의 연동

4.2 분석

본 논문에서 설계하고 구현한 시스템은 향후 확장되는 시스템에 영향을 미치지 않고 쉽게 관리가 가능 하도록 구현하였다. 크게 통합 관리 시스템과 SNMP 에이전트와 연동, 원격지 에이전트와 연동으로 구성된다.

기존 시스템에서 발생하고 있는 문제점은 첫째, 네트워크 자원을 관리하기 위해 관리자가 직접 관리콘솔에 접근하여 네트워크 자원을 관리하고 있는 것이다. 둘째, 파일의 추가, 변경 있을 경우 원격지 사용자가 FTP 프로그램을 이용하여 파일을 다운로드 받아 사용하고 있으나, 원격지 사용자의 최신 파일의 업데이트가 이루어지고 있는지에 대한 관리가 불가능한 것이다. 본 논문에서는 SNMP와 연동, 원격지 에이전트와 연동을 이용하여 위와 같은 문제점을 해결 하였다. 또한 네트워크 장치와 어플리케이션을 하나로 통합하여 관리할 수 있도록 하였다.

본 시스템은 JAVA 기반의 JMX 기술을 이용하여 통합 관리 환경을 구축하였기 때문에 여러 이점이 있는데 그 첫 번째가 JAVA 언어로 구현 할 수 있는 동작은 모두 구현할 수 있다는 점이다. 두 번째로는 본 논문에서 구현하는데 사용한 MBean은 Standard MBean으로 JMX에서 제공하는 MBean 형태 중 하나인데, 실시간으로 MBean에 기능을 추가 혹은 동작 시키는 Dynamic MBean이나 JMX를 이용한 시스템 개발을 도와주는 Model MBean을 이용하게 되면 좀 더 동적이고 유용한 시스템을 구축할 수 있다는 것이며, 세 번째로는 MBean 단위로 설계 및 구현하기 때문에 모듈화가 쉽고 MBean 추가를 통한 기능 추가가 용이하다.

Ⅴ. 결론

기존의 통신망은 유무선 망이 분리된 수직적인 망 구조로 이질적인 망들이 혼재하여 존재하고 있으며, 이러한 이유로 타 망과의 연동이 취약하여 망의 개방화가 용이하지 않았지만, 본 논문에서 설계한 시스템은 네트워크 계층과 응용 계층을 서로 분리시킴으로써 각 기술이 독립적으로 발전할 수 있도록 하여 기술의 발전으로 인해 변화하는 네트워크와는 무관하게 서비스는 독립적으로 개발되고 이용할 수 있도록 하였다. 즉, 다양한 네트워크에서 어플리케이션을 동작할 수 있게 하고, 벤더와 서비스제공자들에게 독립적으로 동작할 수 있도록 하여 통합 네트워크를 관리할 수 있는 구조를 갖게 하였다.

본 논문에서는 JMX기술을 이용하여 네트워크 장치와 어플리케이션에 대한 관리를 하나로 통합하여 관리할 수 있는 시스템을 설계하고 구현하였다. JMX를 이용한 시스템 통합관리는 기존 시스템에 대한 연동과 확장에 유연한 구조와 웹에서 관리할 수 있는 환경을 제공한다. 현재 회사에서 네트워크 관리는 정해진 콘솔에서만 관리가 가능하여 위치적인 한계와 시간적인 한계를 가지고 있었다. 그러나 네트워크 장치와 어플리케이션의 통합관리는 관리자에게 위치적인 한계와 시간적인 한계를 극복할 수 있게 하였다.

통합 관리 시스템은 SNMP 에이전트와 연동하여 네트워크 장치에 대한 정보를 확인하고, 원격지 컴퓨터의 상태 모니터링이나 데이터 갱신을 하기 위해 원격지 컴퓨터의 JMX 에이전트와 연동한다. 관리 항목이 추가 및 삭제 될 때에는 MBean 서버에 추가항목을 생성하고, 삭제도 동일한 과정으로 접근하여 삭제한다. 네트워크 장치에 대한 자료와 원격지 시스템에 대한 자료의 분석은 장애 발생 가능성이 있는 장치에 대한 사전대처가 가능하고, 향후 확장되는 시스템 설계와 어플리케이션에 대

한 기초 자료로 활용이 가능하다.

향후 과제로는 미들웨어를 사용하여 네트워크 관리뿐만 아니라 OSS(Operation Support System)를 지원(QoS, 과금)하는 연구가 필요하다.

참고문헌

Sun Microsystems Inc. "Java Dynamic Management Kit 5.1 Tools Reference Guide" 2008.

Sun Microsystems Inc. "Java Dynamic Management Kit 5.1 Getting Started Guide" June 2007.

Julio Guijarro, HP Labs. "Framework for managing large scale component based distributed application using JMX" May 2002.

Bahman Kalali, "Management of EJB Components Using JMX" March 15, 2002.

Sun Microsystems Inc. "Java Management Extensions White Paper" revision 01, June 1999.

Tony G. Thomas "JMX Boosts J2EE Application Management" 2002.

Victor A. Villagra, "An Approach to the Transparent Management Instrumentation of Distributed Applications" 2001.

Vijay Machiraju, Akhil Sahai, And van Moorsel, "Web Services Management Network" 2002.

R.Bhat, R.Gupta, "JAIN Protocols APIs", IEEE Communications Magazine, January 2000.

Benjamin G. Sullins, Mark B. Whipple "JMX In Action" 2001.

H.Kreger "Java Management Extensions for application management" IBM Systems Journal, Vol 40, No 1, 2001

Frank Moreno, "Total e-Business Management : managing Application Server Components" EAI Journal, March 2002

http://mx4j.sourceforge.net/ MX4j's Homepage, open source JMX for Enterprise Computing.