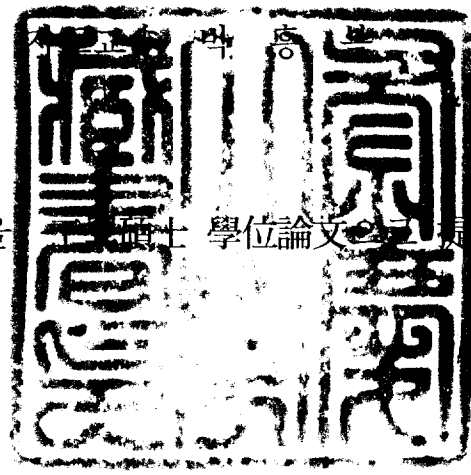
공학석사 학위논문

분산 자원 관리를 위한 무선 원격 제어 시스템의 설계 및 구현



이 論文을 碩士 學位論文으로 提出함

2005년 8월

부경대학교 산업대학원

전 산 정 보 학 과

허 지 훈

허지훈의 공학석사 학위논문을 인준함

2005년 6월 17일

주 심 이학박사 윤 성 대 

위 원 공학박사 여 정 모 

위 원 이학박사 박 흥 복 

목 차

표 목 차

그 림 목 차

Design and Implementation of Wireless Remote Control System for Management of Distributed Resources

Ji-Hoon Heo

Dept. of Computer and Information Graduate School of Industry
Pukyong National University

Abstract

As the Internet business has been growing rapidly, Internet users or Internet service providers have requested stable service. Along with them, the protection of data such as the management of computer information system and the prevention of barriers has become an important element.

The number of operating personnel has not changed while the number of system has been greatly on the increase so that many users have wanted to manage the system in a systematic and effective manner and to control the application that could

monitor the system performance anywhere and anytime.

This study has embodied the remote control system that manages the scattered resources and service using JMX (Java Management Extensions) in the mobile environment and it has also suggested the method of connecting it with the remote terminal. Furthermore, the embodiment of the functions such as "shotcuts", "control the letters" and "history" minimizes the problems of input of the existing mobile phone.

Through the system suggested in this study, system managers are able to manage the system safely through their own regardless of time and space with the aid of mobile phone.

1. 서론

미래에는 컴퓨터가 사람들의 생활에 편리함과 보다 많은 여가 시간을 줄 것이라는 예측에도 불구하고 그 반대의 상황이 벌어지고 있는 것 같다. 컴퓨터는 CPU의 발전에 힘입어 빠른 속도로 발전하였고, 산업 환경도 컴퓨터만큼이나 빠르게 발전하였다. 이런 발전과 변화의 한가지로 인터넷을 들 수 있다. 인터넷과 웹이 세계화되어감과 동시에 점점 더 복잡해지고 인터넷 활용 인구가 늘어남에 따라 24시간 연속된 컴퓨터 서비스가 필요한 것이다[1].

이와 같은 인터넷 비즈니스의 발달로 컴퓨터와 네트워크는 더욱 복잡하고 다양해짐에 따라 중앙 집중적 작업 환경을 분산 작업환경으로 바꾸게 하였고, 분산 환경에서 작업하면서 관리자가 관리해야할 전산 시스템도 크게 증가하게 되었다. 컴퓨터 시스템은 매우 복잡하고, 변화하므로 항상 CPU, 메모리와 같은 성능이나 프로세스 상태를 주기적으로 모니터링 해야 한다[2].

회사에서 관리해야 할 시스템들은 SUN, HP, IBM 등 다양한 종류의 서버들이 있다. 그리고 거기서 동작하는 웹 서버나 웹 애플리케이션서버와 각종 DB 및 서비스까지 관리해야 할 요소들은 다양하면서도 많다. 관리자가 이 시스템들을 관리할 때 콘솔이나 터미널 접속을 통해 top 명령이나 vmstat 명령을 통해 현재 시스템의 CPU나 메모리 사용률을 확인 한다. 각종 데몬을 확인하기위해 프로세스가 정상적으로 떠 있는지, 리스닝 포트가 정상적으로 열려 있는지, 그리고 에러 로그를 확인함으로써 현재 상태를 파악할

수 있다. 이처럼 각각의 시스템은 그 시스템의 특성에 맞게 특화된 방법으로 관리하기 마련이다. 그러나 시스템은 분산되어 있고 다양하며 계속 변화하고 각각 다른 방법으로 접근해야 한다. 이런 시스템을 정적인 방법으로 관리한다면 관리 시스템을 만드는 사람은 관리해야 할 모든 요소들을 미리 알고 있어야 하고, 추가되고 변경되는 사항에 대해서 계속 업데이트해야 한다. 결국 이런 동적인 환경에서는 동적인 시스템 관리가 필요하다[12].

최근 무선 이동통신기기의 발달로 모바일 단말기를 이용한 원격 제어 시스템이 발달하고 있어, 시간과 공간의 제약이 없는 모바일 단말기를 이용한 원격지 시스템 관리의 필요성이 점점 증가하고 있다[13,14]. 그리고 모바일 단말기의 특성상 제한된 디스플레이와 사용자의 입력의 불편을 해결하기 위한 연구도 진행되고 있다[3].

본 논문에서는 JMX(Java Management Extensions) 기술을 사용하여 분산 자원 관리를 위한 원격 제어 시스템을 제안하고 원격지 시스템에 직접 접속을 해야 할 경우를 위해 원격지 터미널 접속에 대한 방법을 제안한다. 그리고 클라이언트 환경은 모바일 단말기를 사용하여 시간적, 공간적 제약을 없앴으며 모바일 단말기의 특성상 사용자 입력이 용이하지 않기 때문에 바로가기, 문자제어, 히스토리를 통해 사용자 입력이 편리하게 설계를 하였다.

제시된 모델은 순수 자바 코드로 구현되기 때문에 어떤 시스템이든지 호환 가능하고, 컴포넌트 형식으로 만들어져서 확장성이 뛰어나다. 우리가 관리 하고자 하는 관리 대상으로는 시스템 자원 또는 네트웍 상태, 애플리케이션 등 시스템에 사용 가능한 모든

것이 될 수 있다.

제안하는 시스템을 통해 관리자는 자신의 모바일 단말기를 이용해 분산된 원격지 시스템을 모니터링 및 제어 하고 문제가 발생되었을 때 즉각적으로 조치할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 JMX와 텔넷 네트워크 가상 터미널 그리고, 원격 제어에 대해 살펴보고, 3장에서는 분산 자원 관리를 위한 무선 원격 제어 시스템의 설계부와 모듈에 관해 설명하고, 4장에서는 제안된 설계와 모듈을 통해 구현하고 결과를 분석한다. 5장에서는 결론 및 향후 연구 과제에 대해 언급한다.

2. 관련 연구

이 장에서는 분산 자원 관리를 위한 자바의 JMX와 텔넷 네트워크 가상 터미널에 대해서 소개하고 원격 제어에 대한 관련 연구들을 소개한다.

2.1 JMX 구조

JMX[4]는 썬 마이크로시스템즈에서 제안한 자원 관리 프레임워크이다. JMX는 동적인 관리라는 시장의 요구와 관리 솔루션의 설계자와 개발자들에게 적절한 툴을 제공하기 위해서 만들어 졌다. 그리고 관리 대상과 관리하는 시스템간의 결합이 유연해 진다. 애플리케이션이나 서비스 개발자는 관리 시스템이 어떻게 구현되어 있는지 신경 쓰지 않고 애플리케이션과 서비스를 관리 가능하도록 만들 수 있고 관리 시스템 개발자는 많은 요소들을 동적으로 관리할 수 있다[5]. JMX는 Instrumentation level, Agent level, Distributed services level과 같이 3레벨의 모델로 그림 1[4]과 같이 구성되어 있다. 이는 관리 대상과 관리하는 시스템간의 종속성을 제거함으로써 유연한 구조를 만들 수 있다.

• Instrumentation level : JMX는 이 레벨에서 MBean(Managed Bean)을 통해 리소스를 관리할 수 있게 한다. 여기서 리소스는 애플리케이션, 서비스의 구현과 디바이스 등이 될 수 있다. JMX는

자바가 절반 이상이므로 자바로 구현되거나 자바 래퍼(Java Wrapper)가 있어야 한다. MBean은 다른 레벨에 대해서 전혀 알 필요가 없이 특정 패턴이나 인터페이스를 따라 만들어주기만 하면 된다. Standard MBean은 JavaBeans에서 파생된 디자인 패턴을 따르는 자바 객체이고, Dynamic MBean은 런타임시에 더 유연성을 제공하는 인터페이스를 따른다.

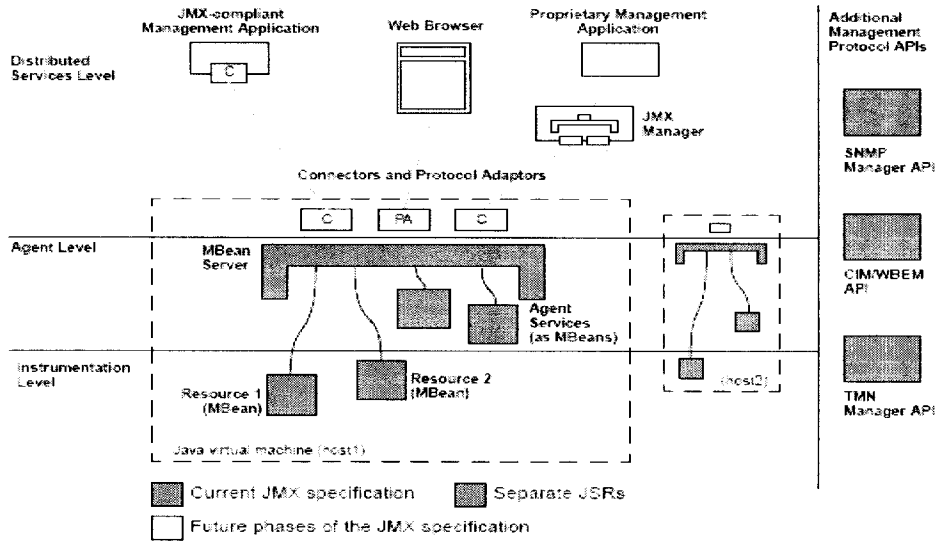


그림 1. JMX 구조

• Agent level : 에이전트는 직접 리소스를 관리하고 원격의 관리 시스템이 그 리소스들을 다룰 수 있도록 한다. 에이전트는 MBean 서버와 MBean을 다루기 위한 에이전트 서비스로 이루어져 있다. 그리고 원격의 관리 시스템이 접근할 수 있도록 하나 이상의 프로

토콜 어댑터나 커넥터가 필요하다. MBean 서버는 MBean의 레지스트리이다. 모든 관리 프로그램은 MBean 서버를 거쳐야 MBean을 다룰 수 있다.

• Distributed services level : JMX매니저를 구현하기 위한 인터페이스를 제공한다. 높은 수준의 프로토콜을 제공하거나 여러 에이전트의 관리 정보를 통합해 사용자의 비즈니스 로직과 관련된 논리적인 뷰를 제공한다. 이 레벨에서는 에이전트와 MBean을 좀 더 쉽게 접근하고 추상화된 뷰를 제공하는 인터페이스를 포함한다.

2.2 텔넷 네트워크 가상 터미널

텔넷은 원격지 터미널 접근을 위해 TCP 서비스를 사용하는 네트워크 응용 프로그램이다. 텔넷 클라이언트는 사용자 단말기에서 수행되고 그 단말기 창에서 데이터를 제공하는 것이 일반적이다[6]. 텔넷 프로토콜은 NVT(Network Virtual Terminal)를 정의한다. 텔넷 NVT는 텔넷 연결을 통해 전송되는 출력 데이터를 만들어내는 키보드와 입력 데이터를 표시하는 프린터로 구성된 양방향 문자 장치이며 사용자 터미널을 아주 쉽게 사용할 수 있는 리모트 터미널 접근을 제공하기 위해 고안되었다. 그림 2는 NVT에서 텔넷 연결의 시작과 끝의 개념을 보여준다. 그림에서, 클라이언트 텔넷은 NVT와 로컬 터미널 프로세스 사이에서 트랜슬레이션을 수

행한다. 트랜슬레이션은 텔넷 NVT ASCII 코드로부터 내부 데이터로 또는 내부 데이터로부터 텔넷 NVT ASCII 코드로 변환하도록 고안 되어 있다. 또한 그것은 텔넷 NVT 명령어 순서로부터 또는 텔넷 NVT 명령어 순서로 변환하는 호스트 제어 문자를 포함한다[7].

서버 텔넷은 NVT와 호스트 프로세스 사이에서 동일한 트랜슬레이션 형식을 수행한다.

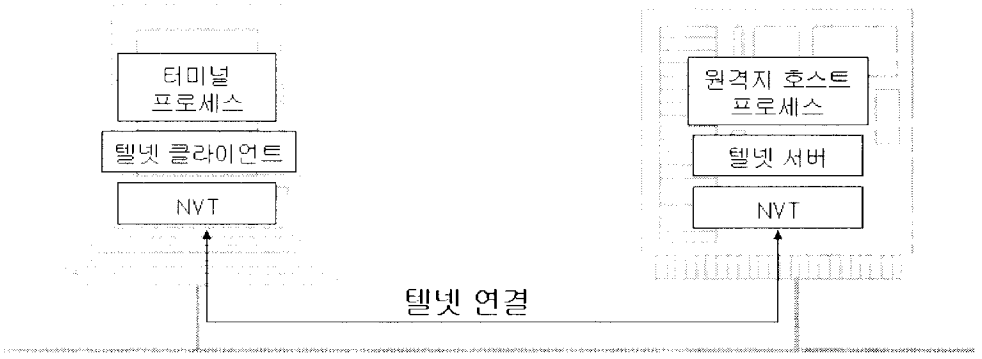


그림 2. 텔넷 모델

2.3 원격 제어

인터넷이 세계화되어감과 동시에 점점 더 복잡해지고 인터넷에서는 각 나라간의 국경이나 시간 구분이 무의미해지고 있다. 이에 컴퓨터 시스템도 매우 복잡해졌으며 관리자들이 주기적으로 시스템을 관리해야 할 요소도 많아졌다.

기존 웹상에서의 시스템 관리는 인터넷이 연결된 컴퓨터에서만 가능하기 때문에 상당한 제약이 따랐다. 그러나 무선 인터넷 기술의 급속한 발달은 무선 인터넷상의 휴대단말기를 이용하여 시스템을 관리할 수 있는 여건을 제공하게 되었다[15].

[10]에서는 휴대폰을 사용해서 원격 PC의 데스크탑을 제어할 수 있는 시스템을 구현하였다. 시스템은 다수의 원격 PC상에서 실행되는 VNC 서버, 실행중인 다수의 VNC 서버를 모니터링하는 Monitor, 원격 PC들을 제어하는 휴대폰 상의 Mobile Viewer로 구성되고, VNC 서버와 Mobile Viewer와의 통신을 위해 기존의 RFB(Remote Frame Buffer) 프로토콜을 모바일 환경에 맞게 수정한 MRFB(Mobile RFB) 프로토콜을 사용하였다.

[11]에서는 클라이언트/서버 기반 원격 제어를 위한 실시간 모니터링 시스템을 구현하였다. Pure Native Java로 구현하며 온실 환경에 적용을 하였으며 인터넷을 통한 원격지의 환경 정보 상황을 모니터링 및 제어할 수 있다. 웹 서버와 제어 서버, 웹 서버에서 애플릿이 제공되는 클라이언트로 구성되고, 클라이언트인 온실 사용자들은 자신의 온실을 제어 및 모니터링 하기 위해 클라이언트/서버의 기본 통신 모델인 소켓을 이용한다. 제어 서버는 원격지 모니터링을 위해 온실 환경 정보를 클라이언트로 송신하고 클라이언트로부터 수신된 명령어를 수행한다. 웹 서버는 클라이언트의 애플릿과 웹 페이지를 생성해주며 제어 서버와 클라이언트 중간 매개체 역할을 한다.

[8]에서는 American University Help Desk에서 PC 문제해결을

위해 현지 전문가의 방문에 대한 요구가 증가하는 문제를 해결하기 위해 Lotus Sametime 원격 제어 솔루션을 사용한 사례를 연구했다. Lotus Sametime은 웹을 통해 클라이언트와 접근 및 채팅이 가능한 협력 도구이다. 실시간 채팅, 온라인 모임, 원격 제어와 스크린 공유를 제공하여 전문가가 문제해결을 위해 간단한 허가만으로 클라이언트의 컴퓨터에서 일어나고 있는 상황을 볼 수 있다.

[13]에서는 PKI 기반의 실시간 무선 원격제어 시스템을 구현하였다. 모바일 통신기기(휴대폰, PDA, Smart Phone, webpad 등)를 이용하여 PKI를 기본으로 하는 무선 원격제어 시스템(W-RCS)을 제안했다. 모바일 장비(PDA), 원격 제어 서버, 제어 대상 시스템의 3-Tier 구조이며 원격 제어 서버는 원격지 시스템의 모니터링 결과와 실행명령에 대한 출력결과를 단말기에 출력하는 웹서버와 통신, 세션, 로그, 그리고 사용자 및 시스템 정보를 구성하는 코어엔진과 에이전트에 의한 자원의 모니터링, 관리, 제어명령을 전달하는 응용프로토콜 인터페이스로 구성되어 있다.

[16]에서는 무선 인터넷 기술을 이용한 리눅스 원격관리 시스템을 구현하였다. C와 CGI로 구현된 클라이언트/서버 구조이며 무선 LAN에서 연결된 PC에서 사용될 웹 브라우저용과, PDA에 구성된 브라우저용, 그리고 휴대폰의 마이크로 브라우저용으로 나눈다. 이 시스템에서 사용 가능한 관리기능으로는 사용자 관리, 그룹 관리, 프로세스 관리, 네트워크 관리, 시스템 탐색기등을 구현하였다. 그러나 관리기능이 미흡하며 휴대폰용은 모니터링 기능만 구현되어 있고 C언어는 이기종 시스템에 대해서는 별도의 컴파일 과정이 필

요하다.

[17]에서는 WAP(Wireless Application Protocol) 기술을 통해 시스템 관리자가 무선 인터넷으로 시스템을 관리하는 기능을 설계하고 구현하였다. 관리자의 휴대폰으로 웹에 접속하여 명령어를 전달하는 무선 시스템과 넘겨받은 명령어를 수행하고 결과 값을 가공하여 휴대폰으로 넘겨주는 유선 시스템 구조로 되어 있다. 이러한 기능을 담당하는 WAP 서버는 통신 모듈, 명령어 모듈, 결과처리 모듈로 구성되어 있으며 관리 기능으로는 시스템 사양 정보, 프로세스 관리, 사용자 계정 관리, 라우팅 설정, 네트워크 모니터링, 네트워크 사용정보를 확인할 수 있다. 그러나 log 확인, 파일 편집 및 삭제 등과 같은 세부적인 제어를 할 수가 없으며 WAP 시뮬레이터를 사용해서 실제 단말기에서의 동작 유무를 확인 할 수 없다.

따라서 본 논문은 이기종 시스템 지원을 고려해 모든 환경을 플랫폼에 독립적인 자바로 사용하여 구현한다. 또한 분산된 시스템을 관리하기 위하여 중앙 제어 서버를 두고 다수의 관리 대상 시스템을 모니터링 및 제어한다. 그리고 세부적인 관리를 위해 원격지 시스템에 직접 접속을 해야 할 경우를 위해 휴대폰을 통해 원격지 터미널에 접속할 수 있게 한다. 또한 휴대폰의 제한된 디스플레이의 특성과 사용자 입력이 용이하지 않기 때문에 기능별 관련 메뉴를 나타내고 바로가기, 문자 제어, 히스토리를 통해 사용자 입력이 편리하게 메뉴를 구성하였다.

3. 제안 시스템의 설계

분산 자원 관리를 위한 무선 원격 제어 시스템은 Java로 구현하며 분산 시스템 환경에 적용하기로 한다. 본 시스템은 모바일 단말기를 통해 원격지의 분산된 시스템의 자원을 모니터링 및 제어할 수 있고 모니터링 및 제어 동작 결과를 모바일 단말기를 통해 실시간으로 확인 가능하다. 본 장에서는 분산 자원 관리를 위한 무선 원격 제어 시스템을 설계한다.

3.1 시스템의 구성

본 논문에서 제안하는 시스템 구조는 그림 3과 같이 표현되며 클라이언트, 중앙 제어 서버, 관리 대상 시스템으로 구성된다. 분산 시스템의 가장 두드러진 특징은 다수의 노드들이 존재하며 이들을 중앙에서 관리할 필요가 있다는 점이다[18].

중앙 제어 서버는 분산 컴퓨팅 환경에서 서로 다른 기종간의 서버와 클라이언트를 연결해주는 기존에 존재하는 미들웨어(middleware)를 사용한다.

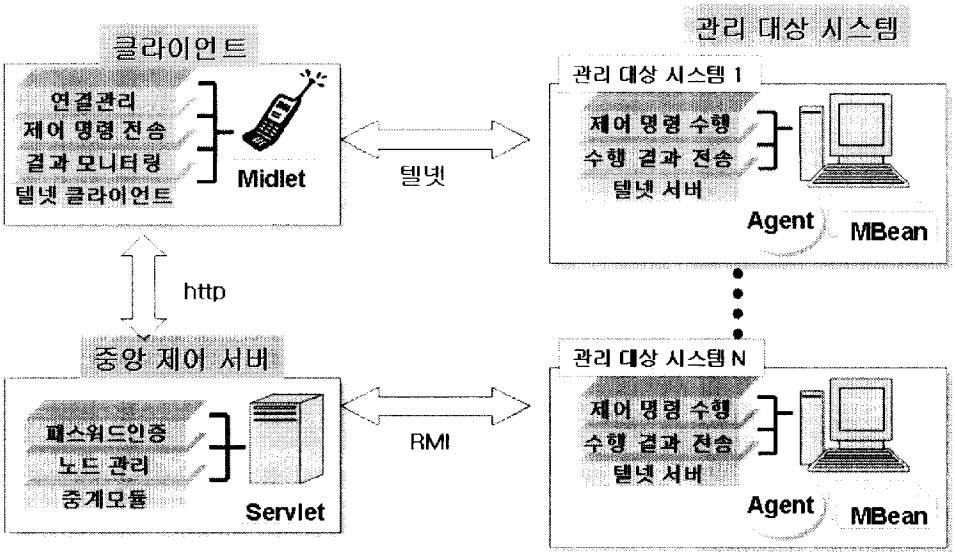


그림 3. 시스템 구성도

클라이언트인 모바일 운영자는 자신의 모바일 단말기로 http로 중앙 제어 서버에 인증을 요청 한다. 인증이 되면 클라이언트는 중앙 제어 서버에 시스템 제어요청을 하게 된다. 관리 대상 시스템에 직접 접속을 해야 할 경우 텔넷 프로토콜을 통해 원격지 터미널에 접속을 해서 관리할 수 있다.

중앙 제어 서버는 모바일 단말기에서 요청받은 명령을 관리 대상 시스템의 Agent에 요청을 전달하고, 관리 대상 시스템으로부터 받은 결과를 모바일 단말기에 전송한다. 중앙 제어 서버와 Agent와의 통신은 자바 RMI(Remote Method Invocation)의 기본 통신 포트인 1099를 사용한다. 자바 RMI는 분산 시스템 환경하에서 서로 다른 하드웨어와 소프트웨어 플랫폼에서 발생하는 문제를 투명

하게 처리하고 자바 가상 머신 위에서 작동중인 클라이언트 응용 프로그램이 원격 자바 가상 머신에서 구현된 서버 객체를 로컬 객체와 같은 방법으로 호출해서 사용할 수 있도록 지원해 준다[9].

관리 대상 시스템은 중앙 제어 서버로부터 요청받은 명령을 Agent가 MBean에 명령을 전달하고 MBean은 명령을 수행하게 된다. 그리고 수행 결과를 Agent에 결과를 전송하고 Agent는 그 결과를 다시 중앙 제어 서버로 보낸다. 클라이언트가 터미널 접속을 시도할 경우 텔넷 서버가 사용자 인증을 거친후 접속을 허용한다. 그림 4는 이러한 동작과정을 나타낸다.

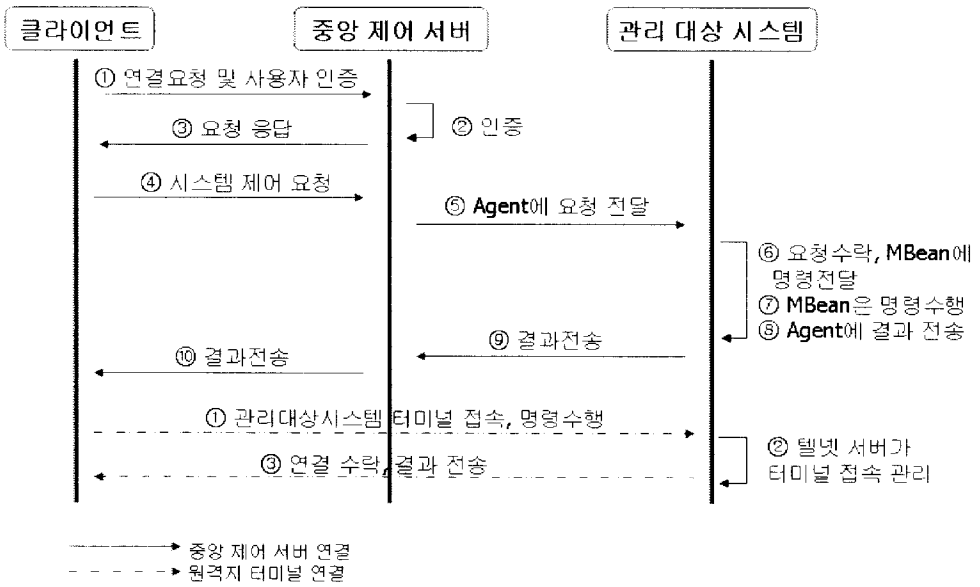


그림 4. 동작 과정

3.2 클라이언트 설계

클라이언트는 자신의 모바일 단말기를 통해 중앙 제어 서버로 연결을 할 것인지 관리 대상 시스템으로 터미널 접속을 할 것인지를 선택한다. 클라이언트 구성은 그림 5와 같다.

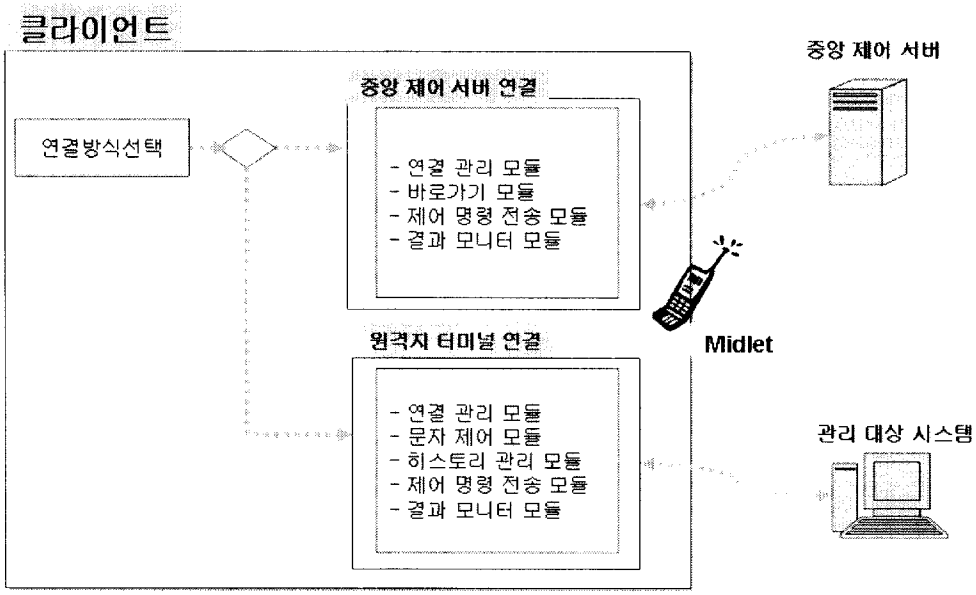


그림 5. 클라이언트 구성

중앙 제어 서버 연결은 연결 관리 모듈, 바로가기 모듈, 제어 명령 전송 모듈, 결과 모니터 모듈로 구성되며 클라이언트가 중앙 제어 서버로 연결해 여러 관리 대상 시스템을 모니터링 및 제어할 수 있게 한다. 각 모듈의 역할은 다음과 같다.

• 연결 관리 모듈 : 중앙 제어 서버로 접속하기위한 주소와 암호를 관리한다.

• 바로가기 모듈 : 사용자가 저장한 제어명령을 바로가기로 관리하여 인증과 동시에 명령 바로 수행한다.

• 제어 명령 전송 모듈 : 사용자가 입력한 제어명령을 중앙제어서버 또는 원격지 터미널에 전송한다.

• 결과 모니터 모듈 : 관리 대상 시스템 에서 수행한 결과값을 관리자 화면에 출력한다.

원격지 터미널 연결은 연결 관리 모듈, 문자 제어 모듈, 히스토리 관리 모듈, 제어 명령 전송 모듈, 결과 모니터 모듈로 구성되며 클라이언트가 관리 대상 시스템으로 터미널 연결을 할 수 있게 해준다. 클라이언트는 터미널 접속을 통해 파일 및 디렉토리 관리나 파일의 편집등과 같은 좀 더 자세한 제어를 할 수가 있다. 각 모듈의 역할은 다음과 같다.

• 연결 관리 모듈 : 원격지 터미널 연결을 위한 주소, 사용자 아이디, 패스워드를 관리 한다.

• 문자 제어 모듈 : 모바일 단말기로 입력하기 힘든 특수문자를

처리하기 위한 모듈이다.

• 히스토리 관리 모듈 : 관리 대상 시스템에서 입력한 명령어를 저장시켜 빠르게 재실행하기 위한 모듈이다.

• 제어 명령 전송 모듈 : 사용자가 입력한 명령(스트링)을 ACSII 코드 값으로 변환 시켜 전송한다.

• 결과 모니터 모듈 : 관리 대상 시스템으로 부터 받은 결과값(ASCII코드)을 스트링으로 변환해 화면에 출력한다.

3.3 중앙 제어 서버 설계

중앙 제어 서버는 웹 서버와 연결 관리 모듈, 노드 관리 모듈, MBean 관리 모듈, 제어 명령 관리 모듈, 제어 명령 수행 모듈로 이루어진 엔진과 RMI 클라이언트로 이루어지며 그림 6과 같다. 클라이언트의 요청과 응답결과는 웹 서버가 처리하며 엔진으로 명령을 전달하고 결과를 전송받아 클라이언트로 전송하게 된다. 클라이언트가 웹 서버에 접속한 후 관리 대상 시스템을 선택하면 웹 서버로부터 요청을 받은 엔진은 요청내용을 RMI 클라이언트로 보내게 되고 RMI 클라이언트는 요청내용을 관리대상 시스템으로 보내고 관리요소를 받게된다. 여기서 응답 받은 관리요소가 클라이언트가 실제 모니터링 및 제어하게 되는 항목이다. 클라이언트는

응답받은 관리요소를 통해 제어 명령을 웹 서버를 통해 전송하면 엔진이 이를 다시 RMI 클라이언트로 보내고 RMI 클라이언트는 관리 대상 시스템에 명령을 요청하고 처리결과를 받는다.

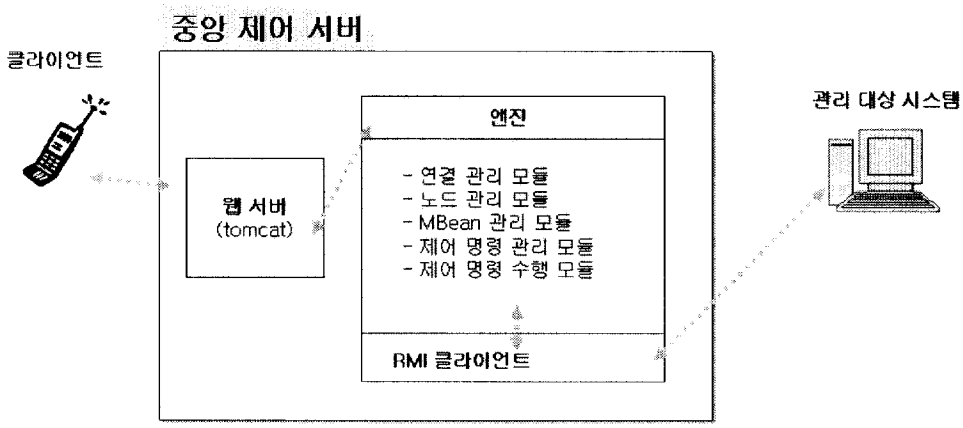


그림 6. 중앙 제어 서버 구성

엔진은 아래와 같이 5가지의 모듈로 구성된다.

• 연결 관리 모듈 : 호스트와 패스워드 인증을 담당하며 RMI 클라이언트를 실행하여 RMI 서버(관리 대상 시스템)에 연결한다.

• 노드 관리 모듈 : 관리 대상 시스템의 Agent로 부터 노드를 등록해서 관리한다. 노드 정보는 텍스트 형태로 저장이 되며 클라이언트의 요청이 있을때 노드 정보를 보여주기 때문에 노드가 증가

하여도 중앙 제어 서버에 부하가 걸리진 않는다.

• MBean 관리 모듈 : 사용자의 요청에 의해 Agent에 등록된 MBean들을 RMI 서버에 요청해 사용자에게 리스트로 나타내준다.

• 제어 명령 관리 모듈 : MBean에 의해 등록된 관리요소를 Agent로 부터 받아 사용자에게 리스트로 나타내준다.

• 제어 명령 처리 모듈 : 제어명령 리스트에 나타난 명령을 Agent로 전송하고 처리내용을 웹서버로 전송해 준다.

3.4 관리 대상 시스템 설계

관리 대상 시스템은 중앙 제어 서버로 부터의 요청을 처리하는 Agent와 클라이언트의 터미널 접속을 관리하는 텔넷 서버로 구성된다. 관리 대상 시스템 구성은 그림 7과 같다.

지금까지 대부분의 분산 시스템 관리는 하나의 중앙 관리자 노드에서 폴링(polling) 기법을 이용하여 관리 대상 노드들로부터 성능 데이터를 수집하고 처리하는 중앙 집중형 관리 방식이 주로 사용되어 왔다. 그러나, 대규모 분산 시스템에서는 관리 대상 노드의 환경이 더욱 다양하기 때문에 수집된 정보의 의미적 이질성을 처리하기 위해 많은 부하가 중앙 관리 노드에 집중되는 현상이 있었다.

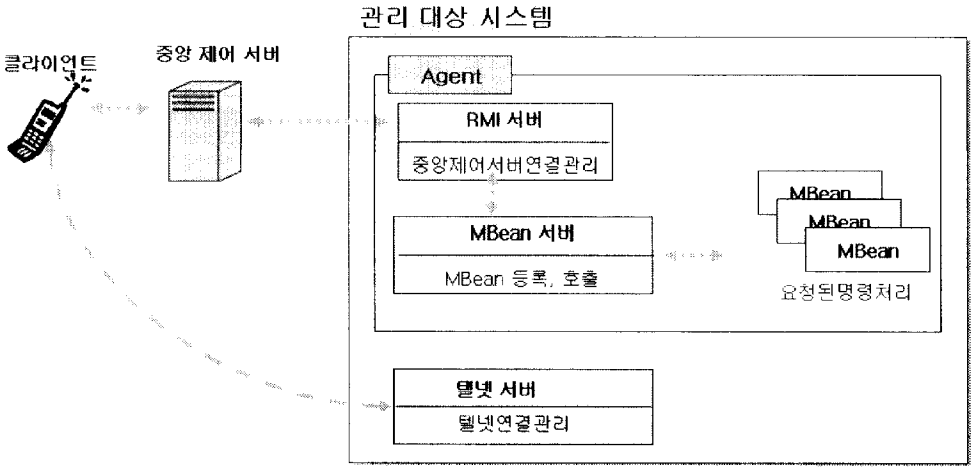


그림 7. 관리 대상 시스템 구성

따라서 원격지에 있는 관리 대상 노드의 상태를 중앙 제어 서버에서 폴링하는 대신에, 에이전트를 파견하여 관리 대상 시스템에서 지역적으로 요청된 명령을 처리하고 적절한 조치를 취할 수 있게 하였다. 이런 방식으로 수집된 성능 데이터는 현재 상태를 보다 정확히 반영하며, 필요한 데이터만 수집할 수 있다.

Agent는 중앙 제어 서버와 통신을 하며 RMI 서버, MBean 서버, MBean 으로 구성되며 아래와 같다.

• RMI 서버 : 중앙 제어 서버에서 요청한 관리요소를 MBean 서버에 요청하고 그 처리 결과를 다시 중앙 제어 서버로 전송하는 역할을 한다.

• MBean 서버 : MBean의 설정 파일인 MLet 에 등록된 MBean을 등록, 관리하며 RMI 서버의 요청이 있을때 등록된 MBean 내용을 전송한다.

• MBean : 실제 요청된 명령을 처리하는 부분이며. 표 1과 같이 관리 대상 시스템별로 관리요소를 정의한다.

표 1. 관리요소의 정의

관리대상	MBean Name	동작	함수
운영체제	Linux	CPU 상태	getCpustate()
		메모리 상태	getMemstate()
		네트워크 상태	getNetstate()
		가상메모리 상태	getSwap()
		디스크 사용량	getDiskusage()
		하드웨어 상태	getDiag()
		시스템 가동 시간	getUptime()
		시스템 로그	getSyslog()
		ping 테스트	ping()
		프로세스 상태	getProcess()
		kill 프로세스	killProcess()
		시스템 재시작	sysReboot()
		시스템 정지	sysShutdown()
웹 서버	Apache	버전정보	getVersion()
		접속로그	getAccesslog()
		에러로그	getErrorlog()
		웹서버 시작	start()
		웹서버 정지	stop()
		웹서버 재시작	restart()
데이터베이스 서버	Mysql	버전정보	getVersion()
		에러로그	getErrorlog()
		데이터베이스 시작	start()
		데이터베이스 정지	stop()
		데이터베이스 재시작	restart()

텔넷 서버는 Windows 서버 계열이나 유닉스, 리눅스 계열에 기본 탑재된 텔넷 서버를 사용하며 클라이언트의 터미널 접속을 관리한다. Windows 서버 계열은 텔넷 서버가 기본적으로 시작되어 있지 않기 때문에 서비스에서 텔넷 서버를 시작해 주어야 한다.

4. 구현 및 결과 분석

본 장에서는 3장에서의 설계를 바탕으로 분산 자원 관리를 위한 무선 원격 제어 시스템을 구현하고 결과를 분석한다.

4.1 구현 환경

본 논문에서 중앙 제어 서버와 관리대상 시스템은 JMX API 1.2와 JMX Remote API 1.0을 사용하였고 모두 플랫폼 독립적인 Java로 구현하였다. 클라이언트 단말기는 SAMSUNG SCH-E170을 사용했으며 SK-VM 1.3 기반으로 구현하였다. 중앙 제어 서버의 운영체제는 Red Hat Linux 9를 사용하고 Tomcat 5.5 웹 서버를 사용했다. 2대의 관리대상 시스템을 두었으며 관리 대상 시스템 1의 운영체제는 Windows 2000 서버, 관리 대상 시스템 2는 Red Hat Linux 9를 사용했다.

4.2 구현

본 논문에서는 모바일 환경하에서 원격지의 분산된 시스템을 모니터링 및 제어를 수행한다. 클라이언트인 시스템 관리자는 자신의 모바일 단말기를 이용해 언제 어디서나 관리 하는 시스템을 모니터링 할 수 있고, 중앙 제어 서버를 통해 정의된 관리요소 제어할 수 있고 원격지 터미널 연결을 통해서 관리 시스템에 접속을

하여 명령을 수행할 수 있다.

4.2.1 클라이언트 구현

중앙 제어 서버의 명령요청과 바로가기는 메인클래스인 MRC 클래스가 담당한다. 중앙 제어 서버 연결을 위해 클라이언트는 원격지 주소와 암호를 입력한 후 중앙 제어 서버로 전송 한다. 인증이 되면 클라이언트는 중앙 제어 서버로 접속을 하게 되고. 클라이언트는 중앙 제어 서버를 통해 명령을 전송하게 된다. requestAction() 메소드에 의해 클라이언트가 제어명령을 요청하면 그 명령을 전송하는 역할을 하는 것이 sendRequest() 메소드이다. sendRequest() 메소드는 중앙 제어 서버로 전송할 값(requestStr)을 매개변수로 받으며 호출을 해서 post 방식으로 값을 요청하게 된다. postmsg 변수에 요청 명령문을 저장하고 os 변수에 요청 스트림을 생성하고 os.write(postmsg)로 요청 명령문을 전송하게 된다. is 변수에 응답스트림을 생성하고 중앙제어 서버로 부터 응답이 오면 response 변수에 스트링형으로 변환한다.

응답을 처리하는 processResponse(String) 메소드는 쓰레드에서 호출되는 함수로 서버에서 전송된 값을 매개변수로 받는다. 중앙 제어 서버로부터 자료를 받는 listExpected는 호스트, 노드, MBean, Action과 같은 리스트형 자료가 올 때 true가 되면 Action을 실행한 값은 false가 된다. 따라서, 호스트, 노드, MBeans, Action은 선택 리스트로 보여지며, 관리요소를 실행한 화

면은 사용자 인터페이스를 담당하는 MMCanvas에 의해 출력된다. 그림 8은 명령 요청과 응답을 처리하는 주요 부분이다.

```
// sendRequest 메소드
private String sendRequest(String requestStr) {
  hcon = (HttpConnection)Connector.open(url); // HTTP로 중앙제어서버에 연결
  hcon.setRequestMethod(HttpConnection.POST); // Request방식을 Post로 함
  byte postmsg[] = requestStr.getBytes(); // 요청 명령문
  os = hcon.openOutputStream();// 요청 스트림 생성
  os.write(postmsg);  // 요청명령문 전송
  is = hcon.openInputStream();// 응답 스트림 생성
  len = ((HttpConnection)hcon).getLength();
  if (len != -1) {    // 요청에 대한 응답이 오는 시점
  }
  response = b.toString(); // 요청을 String형으로 변환
}

// processResponse 메소드
private Displayable processResponse(String s) {
  if (listExpected) {  // 중앙제어 서버에서 자료 받음
        // listExpecte가 true : 노드,domain, Mbean, Action 리스트가 수신된 경우
     .....
      }
  } else {    // Action 으로 처리된 값이 수신 될 경우는 MMCanvas 화면에 출력
  }
}
```

그림 8. 명령 요청과 응답

클라이언트가 자주 사용되는 명령은 바로가기 기능을 통해서 인증과 동시에 명령을 즉각 수행할 수 있다. handleAddShortcut() 메소드는 매개변수로 Displayable을 받으며, 관리요소 리스트 화면에서 '바로가기저장' 버튼으로 활성화된다. 바로가기는 10개까지 저장되며 같은 바로가기가 있는지 검사하게 된다. 관리요소를 선택하게 되면 RecordStore상에 호스트/도메인:MBean/Action으로 저

장된다. 그림 9는 바로가기를 수행하는 handleAddShortcut() 메소드의 주요부분이다.

```
private void handleAddShortcut(Displayable d) {
  int is = ((List)mmScreen[state]).getSelectedIndex();
  String s = ((List)mmScreen[state]).getString(is);
  StringBuffer actbuf = new StringBuffer();
  actbuf.append(currentHost).append('/')
  .append(currentDomain).append(':')
  .append(currentMBean).append('/')
  .append(s);
  s = actbuf.toString();
  // host/domain:Mbean/Action 내용으로 String을 만듬.
      if (database.getNumRecords() == 11) {
       // 바로가기는 10개까지 저장가능
      } else {
          RecordEnumeration re = database.enumerateRecords
          (new ShortcutFilter(s), null, false);
          if (re.hasNextElement()) {
              // 같은 바로가기가 있는지 검사
          }
          re.destroy();
          database.addRecord(s.getBytes(), 0, s.getBytes().length);
          // RecordStore에 저장 시킴
      }
}
```

그림 9. 바로가기

클라이언트가 관리 대상 시스템에 직접 접속을 해야할 경우 원격지 터미널 연결을 통하여 관리 대상 시스템에 접속할 수 있게 한다. 모바일 단말기의 특성상 특수문자를 제어하기가 힘들기 때문에 이러한 불편을 해결하고 히스토리 기능을 통해 명령어의 빠른 재실행이 가능하다[6].

연결과 명령을 전송하는 Trans 클래스의 Trans() 생성자는 클

라이언트가 관리 대상 시스템에 터미널 접속을 하기 위해 23번 포트로 소켓 연결을 하게 된다. 그리고 수신을 위해 is 변수에 수신 스트림 객체를 생성하고, 송신을 위해 os 변수에 송신 스트림 객체를 생성한다. 그리고 send() 메소드는 스트링형으로 받은 사용자 입력값을 바이트형으로 변환하여 서버로 전송한다. 이는 텔넷 프로토콜 정의에 의해 사용자의 입력 값을 텔넷 서버로 전송시 ASCII코드 형태로 전송해야 하기 때문이다[7].

그림 10은 연결과 명령전송을 담당하는 Trans() 생성자, send() 메소드의 주요부분이다.

```
// Trans() 생성자
public Trans(TForm c1, String hosts[]) throws IOException
  {
   connection = (StreamConnection)Connector.open("socket://" + hosts[1] + ":"
+ hosts[2], 3, false);
   // 23번 포트, 관리대상시스템 주소로 소켓 연결
   is = connection.openInputStream(); // 수신 스트림 객체 생성
   os = connection.openOutputStream();  // 전송 스트림 객체 생성
  }

// send() 메소드
public void send(String s)
  {
   try
    {
     os.write(s.getBytes());  // 서버에 쉘명령어 전송
    } catch(IOException ioexception) {
      quit();
      }
}
```

그림 10. 명령 전송

TForm 클래스는 연결 관리와 문자 제어, 메뉴 관리를 담당하며 그림 11과 같다. commandAction() 메소드는 문자 제어와 심볼을 위해서 사용된다. ctrlist는 문자 제어 리스트 객체이고, symbollist은 심볼 리스트 객체이다. 리스트 객체에서 값을 선택해서 s1변수에는 서버에 전송될 값이 들어가고, s2 변수에는 사용자에게 나타나는 값이 들어간다. ctrlist의 s1 변수에는 "CTRL-C", "CTRL-Z"와 같은 문자 처리를 위해 자바의 문자형 리터럴(Character Literal)로 표현했다.

message() 메소드는 클라이언트가 입력한 명령어를 서버로 전송할 때 이 메소드를 같이 호출하며 히스토리에 10개까지 저장하며 10개가 넘어가면 마지막으로 저장된 값을 삭제하면서 저장이 된다.

```
// commandAction 메소드
public void commandAction(Command command, Displayable displayable){
  if(displayable == ctrlist) {
      int j3 = ctrlist.getSelectedIndex();
       s1 = "" + specialsymbol[j3]; // 전송되는 값(' ', '₩t', '₩177', '₩033',
'₩003', '₩032', '₩b', '₩r')
       s2 = specialkey[j3]; // 화면상에 표시되는 기호("Space", "TAB", "DEL",
"ESC", "CTRL-C", "CTRL-Z", "Backspace", "Return")
    } else
       if(displayable == symbollist) {
         int k3 = symbollist.getSelectedIndex();
         s1 = "" + specialsymbol3[k3]; // 전송되는 값('/', '₩₩', '&', '|', '<', '>',
'*', '_', '~', '$', '₩'', '"', '#')
         s2 = specialsymbol2[k3];      // 화면상에 표시되는 기호("/ Slash", "₩₩
Back Slash", "& Ampersand", "| Vertical Bar", "< lower", "> greater", "*
Asterisk", "_ Underline", "~ Tilde", "$ Dollar Sign", "' Single Quotation", "₩"
Double Quotation", "# Sharp")
        }
}
// message 메소드
private void message(String s1, String s2)
  if((i1 = a3.indexOf(s1)) != -1) {
     a3.removeElementAt(i1);
     historylist.delete(i1);
    } else
       if(a3.size() == 10) {
         a3.removeElementAt(9);
         historylist.delete(9);
        }
         a3.insertElementAt(s1, 0);
         historylist.insert(0, s2, null); // List 객체인 historylist에 s2를 추가
}
```

그림 11. 문자 제어

4.2.2 중앙 제어 서버 구현

중앙 제어 서버는 클라이언트의 요청에 의해 여러 관리 대상 시스템으로부터 제어명령을 전송하고 그 결과를 클라이언트에게 전송하는 중계 역할을 하게 된다.

MMServlet 클래스는 클라이언트로부터 제어명령을 요청받고 관리 대상 시스템으로 전송하고 관리 대상 시스템의 노드 관리와 호출한 관리요소를 클라이언트에 전송한다. 그림 12의 init() 메소드는 프로퍼티 파일에 등록된 패스워드를 비교를 해서 사용자 인증을 거치고 관리 대상 시스템 주소를 읽어오고 RmiConnectorClient() 메소드를 통해 RMI 클라이언트 객체를 생성하고 RmiConnectorAddress() 메소드를 통해 관리 대상 시스템의 주소를 생성하게 된다.

processRequest() 메소드는 get 방식, post 방식으로 호출되면서 실행되며 클라이언트로부터 패스워드르 받아 인증을 처리하며, 노드 리스트 처리, 도메인리스트 처리, MBean 리스트 처리, Action 리스트 처리를 해서 바이트 형태로 계층별로 클라이언트에게 메뉴를 보여 준다.

```
// init() 메소드
public void init(ServletConfig config) throws ServletException {
 String hosts = servletResources.getString("mrc.hosts"); //관리대상시스템 읽기
 passwd = servletResources.getString("mrc.passwd"); //패스워드 읽기
 StringTokenizer st = new StringTokenizer(hosts);
  while (st.hasMoreTokens()) {
   RmiConnectorClient rcc = new RmiConnectorClient();
   // RMI 클라이언트 객체 생성
   RmiConnectorAddress address = new RmiConnectorAddress
    (host, ServiceName.RMI_CONNECTOR_PORT,
    ServiceName.RMI_CONNECTOR_SERVER); // 관리대상시스템의 RMI 주소 생성
     try {
         rcc.connect(address) ; // RMI 어드레스로 연결
         } catch (Exception e) {// 예외처리 }
}

// processRequest() 메소드
 protected void processRequest(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, java.io.IOException
{
    response.setContentType("text/plain"); // response 타입 정의
        ServletOutputStream sos = response.getOutputStream(); //응답스트림생성
        byte[] barr = null; // 클라이언트로 전송할 byte 정의
        String checkPwd = request.getParameter("pwd");// 패스워드 수신
        String whatNext = request.getParameter("nodes"); // 노드 수신
        if (whatNext != null) {
         if(!checkPwd.equals(passwd)) {// password 비교}
        } else {
            whatNext = request.getParameter("host");
            if (whatNext != null) {
             //host, domain, mbean, action 계층별 메뉴 리스트
              whatNext = request.getParameter("host");
              whatNext = request.getParameter("domain");
              whatNext = request.getParameter("mbean");
              whatNext = request.getParameter("action");
              .....
             }
       }
}
```

그림 12. 연결 및 메뉴관리

4.2.3 관리 대상 시스템 구현

관리 대상 시스템은 실제 클라이어트의 명령을 처리하게 되는 MBean으로 구성이 되고 이 MBean들을 등록하고 처리결과를 중앙 제어 서버로 전송하는 Agent로 구성된다.

Agent 클래스는 중앙 제어 서버와의 통신을 위한 RMI 서버 객체의 생성과 관리 대상 시스템의 MBean 들을 등록하게 되는데 Agent가 구동이 되고 객체를 생성할 때 제일 처음 MBeansServerFactory에서 MBean 서버를 생성하고 RMI 서버를 생성한다. 그 후 mlet을 MBean 서버에 등록하여 관리하려는 도메인을 등록해 놓는다. mlet에는 MBean에 의해 정의된 관리요소가 들어가 있다. 그림 13은 Agent() 생성자의 주요부분이다.

```
public Agent() {
 server = MBeanServerFactory.createMBeanServer(); // Mbean 서버 생성
 rmiConnector = new RmiConnectorServer(); // RMI 서버 객체 생성
 mlet = new MLet(); // Mlet 객체 생성
 try {
     String domain = server.getDefaultDomain();
     server.registerMBean(mlet, new ObjectName(domain +
":type=MLetService")); // mlet객체를 Mbean에 등재
 } catch (Exception e) { // 예외처리 }
}
```

그림 13. 객체 생성

그림 14에서 initCommnunication() 메소드는 MBean 서버에 RMI 서버를 등록하고 rmi.Connector.start()를 통해 클라이언트의 요청을 받을 준비를 한다.

```
private void initCommunication() {
try {
      ObjectInstance rmiConnectorInstance =
      server.registerMBean(rmiConnector, null);
      // MBean서버에 RMI 서버객체 등재
     } catch(Exception e) { //예외처리 }
        rmiConnector.start();  // RMI 서버 객체 시작( RMI클라이언트 요청 받을 준
비 됨
 }
```

그림 14. RMI 서버

그림 15의 MBeand을 등록하는 loadMBean() 메소드는 Agent생성자에서 mlet을 MBean 서버에 등록한 후에 관리할 도메인을 프로퍼티 파일에서 읽어서 인스턴스에 등록한다.

```
private void loadMBeans(String mBeanURL) {
try {
      Set loadedMBeans = mlet.getMBeansFromURL(mBeanURL);
      // m-let.properties파일에서 Mbean항목 등재
      Iterator i = loadedMBeans.iterator();

     } catch (ServiceNotFoundException snfe) {
      // 예외처리
    }
}
```

그림 15. MBean 등록

MBean은 Agent로부터 실제 요청된 명령을 처리한다. 그림 16은 실제 요청된 명령을 처리하는 주요 메소드를 나타낸 것이다. MBean 이름은 Linux 이며 리눅스 시스템의 CPU 사용량, 메모리 사용량, 프로세스 제어 등과 같은 메소드들로 구성이 되어 있다. 메소드 내에서 직접 처리하기 힘든 명령어는 쉘스크립트로 작성을 하여 처리를 하였다. getCpustate.sh의 쉘스크립트 내용은 /usr/bin/top n 1 | grep 'CPU states' 과 같이 되어 있는데 이렇게 하는 이유는 java.lang.Runtime 클래스의 exec 메소드로는 | 와 같이 파이프가 있는 명령문은 한번에 실행하지 못하기 때문이다. 다음은 그림 16은 관리 요소를 정의한 Linux MBean의 일부분이다.

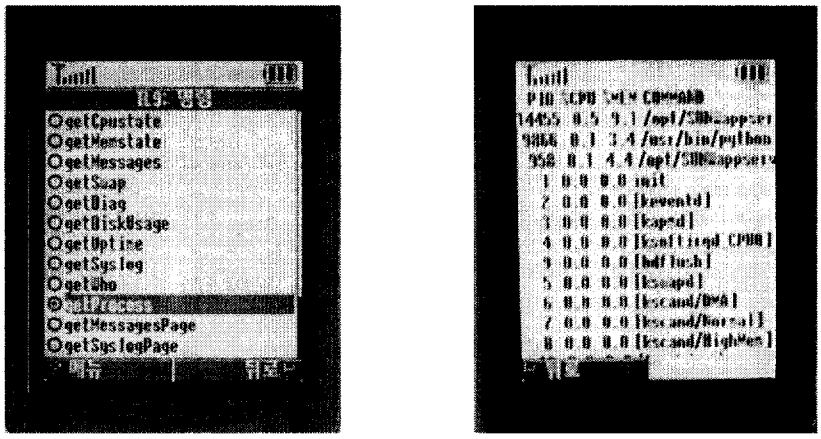
```
public class Linux implements LinuxMBean {

  public byte[] getCpustate() {   // CPU 사용량 검사
      return MBeanHelper.executeCommand("/opt/mrc/script/getCpustate.sh");
  }
  public byte[] getProcess() { // 프로세스 상태 확인
      return MBeanHelper.executeCommand("/bin/ps -eo pid,%cpu,%mem,args
--sort -%cpu");
  }
  public byte[] sysShutdown() { // 시스템 종료
      return MBeanHelper.executeCommand("/sbin/shutdown -y -i0 -g0");
  }
}
```

그림 16. 관리 요소 정의

4.3 실행화면

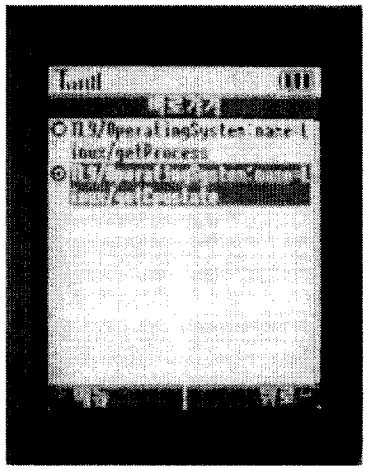
중앙 제어 서버 연결은 JMX기반으로 구현되어 MBean에 의해 운영체제, 웹 서버, 데이터베이스 서버로 관리 요소가 정의되어 관리자가 oneclick으로 시스템을 관리할 수 있게 하였다. 그림 17은 MBean에 의해 정의된 관리 요소 중 현재 실행중인 프로세스를 확인하는 getProcess 함수를 수행한 결과이다.



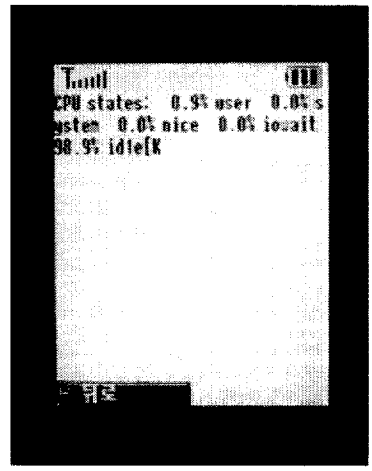
(a) 명령 리스트 (b) 프로세스 확인

그림 17. 프로세스 확인 화면

화면에 나타나지 않는 부분은 방향키로 위, 아래, 좌, 우 스크롤을 통해서 전체 내용을 확인할 수 있다. 또한 모든 명령에는 자주 사용하는 기능을 바로가기로 만들 수가 있어 초기 화면에서 자주 사용되는 명령을 바로 실행해서 결과를 얻을 수 있다. 그림 18은 바로가기 기능을 통해 현재 CPU 사용량을 확인하는 화면이다.



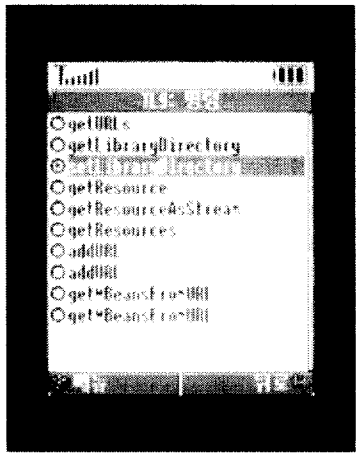
(a) 바로가기 저장



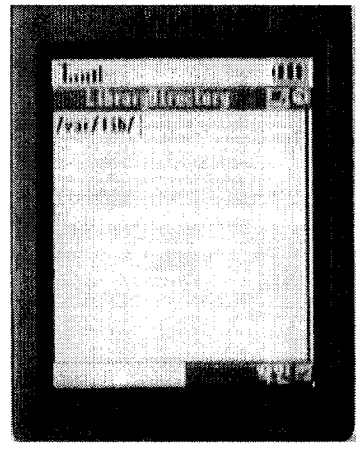
(b) CPU 사용량 확인

그림 18. 바로가기 실행 화면

그림 19는 MBean 서버에 의해 동적으로 등록된 MBean들이 메모리에 등록되지 전에 쓰여지는 라이브러리 디렉토리 위치를 확인하고 변경하는 화면이다.



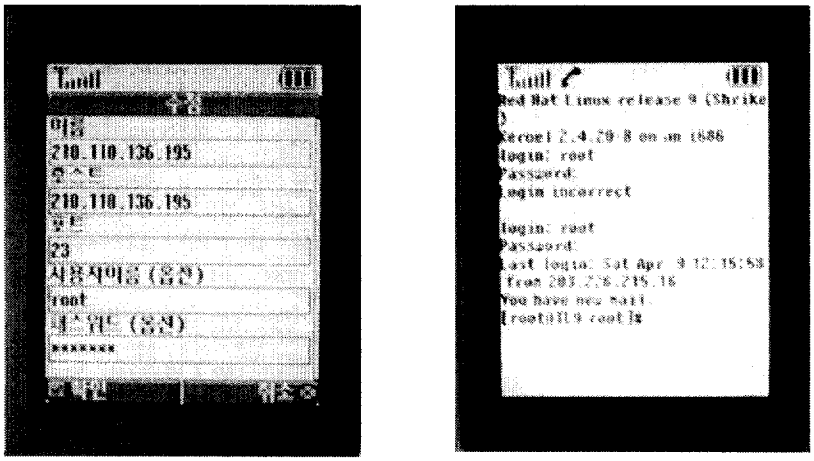
(a) MBean 제어 명령



(b) 라이브러리 위치 지정

그림 19. 라이브러리 디렉토리 정의

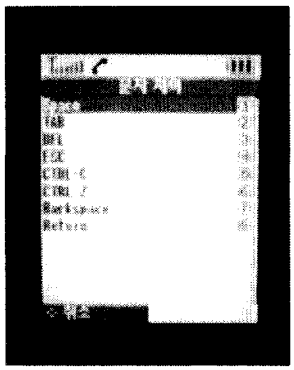
MBean에 의해 정의 되지 않은 요소나 원격지 시스템에 직접 접속을 해야 할 경우 원격지 터미널에 접속하여 보다 자세한 제어가 가능하다. 그림 20은 호스트 연결 관리를 통해 원격지 터미널에 접속한 화면이다. 사용자이름과 패스워드를 저장해 놓으면 별도의 로그인 과정없이 자동으로 로그인이 이루어 진다.



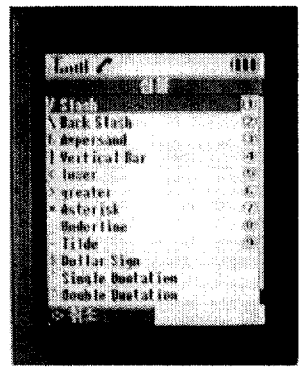
(a) 연결 정보 입력 (b) 텔넷 로그인

그림 20. 원격지 터미널 접속

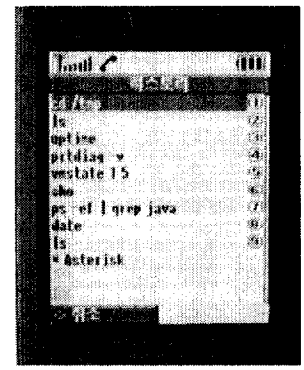
그림 21은 원격지 터미널에 접속 후 문자 제어를 처리하는 메뉴를 나타내고 있다. 문자 제어는 텔넷에서 자주 사용되는 기능이지만 휴대폰으로 입력하기 힘든 제어문자를 정의하였다. 심볼도 텔넷에서 필수적인 심볼들을 정의 하였으며 히스토리는 사용자가 입력했던 명령어들을 다시 빠르게 재입력할 수 있게 함으로써 모바일 단말기의 입력의 제약을 최소화 하였다.



(a) 문자 제어



(b) 심볼



(c) 히스토리

그림 21. 문자 제어

4.4 결과분석

제안하는 분산 자원 관리를 위한 무선 원격 제어 시스템은 분산 시스템 환경에 적용하였으며 플랫폼에 독립적인 자바를 이용하여 이식성을 강화 시켰다. 또한 PDA보다 대중적인 휴대폰을 기반으로 구현하여 추가적인 장비의 구입을 최소화 하였다. 또한 중앙 제어 서버를 통해 분산된 다중 시스템을 관리할 수 있으며 MBean에 의해 사전 정의된 관리요소 이외에 보다 세부적인 제어가 필요할 경우 관리 대상 시스템에 터미널 접속을 할 수가 있어 콘솔에서 작업하듯이 시스템을 관리할 수 있다. 표 2는 기존 시스템과 제안하는 시스템을 비교 분석한 결과를 나타내고 있다.

표 2. 기존 시스템과 비교

비교 항목	제안하는 시스템	기존 시스템[17]	기존 시스템[16]	기존 시스템[13]
사용자 단말기	휴대폰 (SCH-E170)	Simulator(Open wave WAP edition 4.0)	PC, Simulator(Siemens S45)	PDA(Compaq iPAQ 3850)
구현언어	Java	C, Java	C	C, Java
이기종 운영체제 지원	플랫폼 독립적	플랫폼 종속적	플랫폼 종속적	플랫폼 종속적
다중 시스템 관리	가능	불가능	불가능	가능
원격지 시스템 로그인	터미널 접속	불가능	불가능	불가능
모니터링 및 제어 항목	시스템에서 제공되는 모든 기능 사용 가능	프로세스, 사용자계정, 라우팅설정 등 제한적	휴대폰은 모니터링만 가능	사전 정의된 관리요소만 가능

5. 결론 및 향후 과제

인터넷과 웹이 세계화되어 가면서 동시에 점점 더 복잡해지고 다양해지고 있다. 이와 함께 시스템도 점점 증가하고 24시간 중단 없는 서비스를 위해 시간적, 공간적 제약없이 시스템을 관리하는 일은 무엇보다도 중요한 요소로 자리잡게 되었다.

본 논문에서는 분산 자원 관리를 위한 무선 원격 제어 시스템을 설계 및 구현하였다. 본 논문은 모바일 단말기를 사용하여 관리자가 언제, 어디서든지 자신의 시스템을 관리할 수 있도록 하였다. 그래서 본 연구의 기반이 되는 JMX 기술을 응용한 관리모듈과 Agent와 통신하는 중앙 제어 서버를 통해 다수의 시스템을 관리할 수 있게 하였다. 그리고, 원격지 시스템에 직접 접속을 해야할 경우를 위해 원격지 터미널에 접속할 수 있는 방법도 설계 및 구현하였다. 또한 모바일 단말기의 제한된 화면 크기를 위해 기능별, 계층적 메뉴를 구성하였으며 사용자 입력이 용이하지 않기 때문에 바로가기 기능과 특수문자 제어, 히스토리를 통해 사용자 입력의 불편을 최소화 하였다. 그리고 클라이언트, 중앙 제어 서버, 관리 대상 시스템 애플리케이션을 운영체제에 독립적인 자바로 구현함으로써 이식성과 재사용성을 고려하여 구현하였다.

향후 연구 과제로써 중앙 제어 서버와 관리 대상 시스템간 RMI 통신은 SSL(Secure Sockets Layer)에 기반한 RMI 소켓을 사용한 방법과 원격지 터미널 연결은 모바일 단말기에서 SSH(Secure Shell) 기반의 터미널 접속에 대한 연구가 계속 되어야 할 것이다.

참 고 문 헌

[1] Evan Marcus, Hal Stern, "Blueprints for High Availability", John Wiley & Sons, 2000.

[2] Sybase Whitepaper, "iAnywhere Mobile Manager", 2002.

[3] O. Buyukkokten, H. Grarcia-Molina, A. Paepcke, T. Winograd, "Power Browser: efficient web browsing for PDAs", Conference on Human Factors in Computing Systems, p.430-437, 2000.

[4] Sun Microsystems, "Java Management Extensions Instrumentation and Agent Specifications, v1.2", October 2002.

[5] Sun Microsystems,"http://java.sun.com/j2se/1.5.0/docs/guide /jmx/tutorial/tutorialTOC.html".

[6] Barron Housel, lan Shields "Accelerating telnet performance in wireless networks", International Workshop on Data Engineering for Wireless and Mobile Access, 1999.

[7] Postel, J., and J. Reynolds, "Telnet Protocol Specification", RFC 854, USC Information Sciences Institute, May 1983.

[8] Sean Stockburger, "Virtual Onsite Support: Using Internet Chat and Remote Control to Improve Customer Service" Proceedings of the 30th annual ACM SIGUCCS conference on User services, p.143-147, November 2002.

[9] Sun Microsystems, "Java RMI Specification",

http://java.sun.com/j2se/1.4.2/docs/guide/rmi/spec/rmiTOC.html.

[10] 천희자, "휴대폰을 사용한 이미지 기반의 원격 PC 데스크탑 제어 시스템의 설계 및 구현" 한국정보처리학회 학술대회지, VOL. 11 NO. 02, pp.575-578, 2004. 11.

[11] 김대업, "Client/Server 기반 원격 제어를 위한 실시간 모니터링 시스템 설계 및 구현", 부경대학교 석사학위, 2002. 2.

[12] 이정원, "동적인 시스템 관리의 세계로", 마이크로소프트웨어, 2004.

[13] 이문구, "PKI를 기반으로 한 실시간 무선 원격제어 시스템의 구현", 한국정보보호학회논문지 제 13권 3호, pp.71-79, 2003. 6.

[14] 나승원, "무선 환경에서 자원 관리를 위한 모바일 에이전트의 설계 및 구현", 한국정보과학회 학술발표논문집(B), pp.151-153, 2003. 4.

[15] 박홍진, "무선 기반 성능 모니터링 시스템 구현" 한국정보과학회 봄 학술발표논문집(A), 31권 1호, 2004. 4.

[16] 김찬수, "무선 인터넷 기술을 이용한 리눅스 시스템 원격관리", 호남대학교 정보통신연구 Vol.11, p.207-218, 2001.

[17] 최원석, "WAP 기반 서버 관리 시스템의 설계 및 구현" 한국정보처리학회 춘계학술대회, VOL.09 NO.01, pp.1479-1482, 2002. 4.

[18] 김진향, "JMX를 기반한 분산된 자원을 관리하는 모니터 시스템에 관한 연구", 건국대학교 석사학위, 2001. 2.

감사의 글

짧지않은 2년 반동안의 대학원 생활동안 기쁘고, 힘들었던 많은 시간들이 있었습니다. 주위에서 많은 분들이 함께 해주었기 때문에 졸업을 할 수 있었던 것 같습니다.

부족한 제자 질책과 격려를 아끼지 않으시며 논문을 지도하여 주신 박흥복 교수님께 진심으로 감사드립니다. 바쁘신 와중에 논문을 심사하여 주신 윤성대 교수님, 여정모 교수님께도 깊은 감사를 드리며 배움의 길을 열어주신 산업대학원 전산정보학과 모든 교수님들께도 감사드립니다.

논문교정에 많은 도움을 주신 서정희 선배님, 연구실 생활동안 많은 도움을 받은 희정이, 연구실 체질인 은영이, 많은 나이에도 불구하고 연구실의 모범이 되어 주신 김영완 선생님과 영진씨, 학교 수업에 많은 시간을 함께 한 동기 유경씨께 깊은 감사를 드립니다. 그리고 짧은 시간이었지만 산업대학원 1년차 영철이, 연구실의 학부생중 유일한 남아 왕근이, 항상 생기발랄한 혜성이, 민희 모두 고맙습니다.

함께 밤을 지새워가며 프로그램 개발에 많은 도움을 주신 한국건설CALS협회 오승규 과장님 정말 감사드립니다. 그리고 학교 수업에 차질을 빚지 않게 많은 도움을 주신 직장 동료분들께도 깊은 감사를 드립니다.

늘 뒤에서 사랑하는 아들 묵묵히 지켜봐주신 아버지, 어머니 그리고, 하나뿐인 동생을 지극히 이뻐해주는 누나 모두 사랑합니다.

2005년 6월

허지훈