2001년 한국정보처리학회 추계 학술발표논문집 제8권 제2호

클라이언트/서버 환경에서 자바를 이용한 클라이언트 모니터링시스템 설계 및 구현

전도현*, 이선동**, 윤인모*, 김찬수*
*호남대학교 소프트웨어공학과
**호남대학교 정보통신공학과
e-mail:iyoon@honam.honam.ac.kr

Design and Implementation of Client Monitoring System Using Java in Client/Server Environment

Dohyeon Jeon*, Seondong Lee**, Inmo Yoon* , Chansoo Kim*
*Dept of Software Engineering, Honam University
**Dept of Information & Communications Engineering, Honam University

요약

본 논문에서는 클라이언트(Windows)/서버(Linux) 환경에서 Java와 C/C++ 언어를 이용한 원격 제어 시스템을 설계·구현하여 Windows 기반의 그룹 내에(기업, 학교 등) 연결된 컴퓨터사이의 통신, 원격 제어, 모니터링, 시스템의 사양, 사용되고 있는 응용프로그램 등의 관리가 가능한 시스템을 개발하였다.

1. 서론

최근 하드웨어와 소프트웨어의 급진적인 발전으로 인한 소형 컴퓨터의 성능 향상과 가격 하락으로 보급량이 증가하고, 통신기술 발달에 힘입어 LAN/WAN 및 인터넷 등을 통한 컴퓨터의 이용이 증가함에 따라 정보 통신을 비롯한 여러 분야에서 클라이언트/서버를 기반으로 하는 애플리케이션들이 개발되고 있는 추세이다.[1] 이에 본 논문에서는 네트워크을 통해 연결된 그룹 내의 모든 클라이언트 PC의 자원(CPU, MEMORY, 프로세스 정보 등) 사용 현황 및 작업 현황을 모니터링하며 필요에 따라 클라이언트를 종료하거나 입력 장치를 제어함으로써 각 클라이언트에 대한 원격 관리가 가능한 시스템을 설계·구현하였다.

본 논문의 구성은 2장에서는 본 논문에서 사용한 JNI와 RMI에 대한 용어에 대해 간략히 설명하고 3장에서는 본 시스템을 설계하고 구현한 과정에 대해 설명한다. 4장에서는 본 연구의 결과와 향후 과제에 대해 기술한다.

2. 용어 설명

2.1 JNI(Java Native Interface)

JNI는 다른 언어로 작성된 코드를 자바에서 호출하도록 만들어진 규약이다. 현재는 C/C++에 대한 호출만을 지원한다. 이는 자바의 가장 큰 장점인 '플랫폼 독립적' 이라는 부분을 해치는 규약이긴 하지만 자바의 속도 문제와 특정 플랫폼에서 제공하는 고유의 서비스의 기능을 모두 포함할 수 없고, 특수한 목적으로 제작된 하드웨어를 자바에서 제어해야 할 필요가 있을 때 자바만으로 해결하기는 힘들기 때문에 JNI가 필요하다. 지나치게 JNI에 의존하는 것은 자바가 갖고 있는 많은 장점들을 해치는 결과를 초래할 수 있지만, 특정 부분에 적절하게 사용한다면 JNI는 자바의 장점과 C/C++의 장점을 골고루

이용할 수 있는 길을 제공한다.
JNI는 다음과 같은 경우에 주로 사용된다.

1) 속도 문제가 있는 계산 루틴
2) 자바에서 하드웨어 제어
3) 자바에서 지원되지 않은 특정 운영 체제 서비스
4) 기존의 프로그램에서 자바가 제공하는 서비스를 이용[7][8]

2.2 RMI(Remote Method Invocation)

RMI는 네트워크 상으로 다른 자바 애플리케이션과 통신할 수 있는 자바 애플리케이션을 만들기 위해 사용된다. 즉 어떤 어플리케이션에서 멀리 떨어진 곳에 위치한 어플리케이션의 메소드를 호출하거나 이 애플리케이션의 변수에 접근할 수 있게 하고, 객체를 네트워크를 통해 주고받을 수 있게 하는 것을 의미한다. 또한 이렇게 멀리 떨어진 곳에 위치하는 어플리케이션의 경우 완전히 다른 시스템과 다른 자바 환경에서 동작할 수도 있다는 것이다. RMI를 사용하면 어떠한 프로토콜을 사용해서 통신할 것인지를 미리 알지 않더라도 다른 자바 프로그램과 통신을 할 수 있다.[7][8]

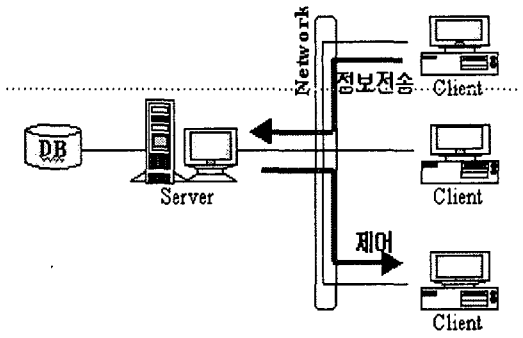
3. 클라이언트와 서버 설계 및 구현

3.1 설계 및 구현 환경

본 시스템의 서버 운영체제로는 Linux를 사용하였으며 클라이언트 운영체제로는 Windows98을 사용하였다. 서버측에서는 Java 개발 도구인 JDK1.3의 AWT와 스윙 패키지 안에 있는 기본 컴포넌트를 사용하여 GUI를 구성하고 이벤트를 처리하였으며, Java 언어로 구현이 어려운 클라이언트 정보(CPU, MEMORY, PROCESS 등) 획득과 시스템 제어(시스템 종료, 입력장치 제어 등)에 있어서는 C와 C++로 작성된 코드를 JNI를 이용하여 자바클래스화 한 후 RMI을 통해 자바에서 호출하거나 접근이 가능하게 하였다.

3.2 시스템 구성

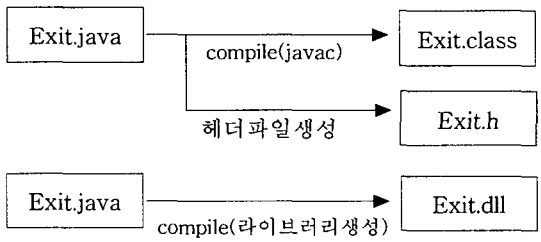
[그림 1]은 본 시스템의 대략적인 구성도를 나타낸다. 클라이언트는 부팅 시 자동 실행되며 클라이언트에 설치된 애플리케이션과 각종 하드웨어/프로세스 정보를 서버의 요청이 있을 때 전송한다. 클라이언트는 View가 없고 마치 Daemon처럼 활동하게 되며, 사용자는 로그인 폼에서 인증절차를 거치고 나서 해당 클라이언트를 사용할 수 있다. 서버에서는 항상 waiting상태에서 클라이언트 측에서 부팅이 되면 그 때부터 Monitoring을 시작하고 log file을 작성하기 시작한다.



[그림 1] 시스템 구성도

3.3 시스템 제어 구현

[그림 4]처럼 자바 소스 파일을 작성하고 컴파일한다. 그리고 JNI 스타일의 헤더 파일을 생성한다. Exit.java 파일에 보면 native로 선언된 메소드가 있는데 이는 헤더 파일의 함수 원형이 된다. 만들어진 헤더파일을 [그림 3]처럼 include하는 실제 구현 라이브러리를 작성하고 컴파일하면 .dll 확장자의 라이브러리 파일이 생성된다. 작성된 라이브러리 파일을 자바 가상 머신이 읽어들이는 것은 자바 소스 코드에 있는 System.loadLibrary("exit"); 루틴이다.



[그림 2] JNI 구현 과정

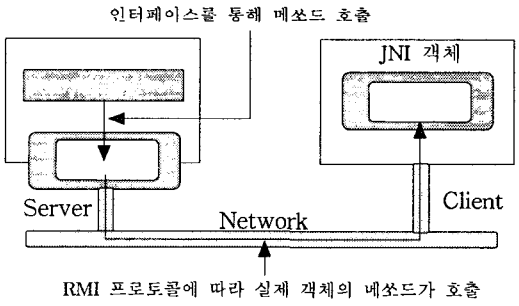
```
#include <jni.h>
#include "Exit.h"
#include <windows.h>

JNIEXPORT void JNICALL Java_Exit_exit(
JNIEnv * env, jobject obj){
 ExitWindowsEx(EWX_SHUTDOWN | EWX_FORCE, 0);
}
```

[그림 3] C 소스코드

```
class NativeExit{
  public native void exit();
  static{
    System.loadLibrary("exit");
  }
}
class UseExit{
  public UseExit(){
    new NativeExit().exit();
  }
  public static void main(String [] args){
    new UseExit();
  }
}
```

[그림 4] Java 소스코드

그런 후 [그림 5]에서처럼 서버와 클라이언트에 자바 RMI의 원격 객체 인터페이스(클라이언트)와 지역 객체(서버) 객체를 구현하여 서버에서 클라이언트의 공유라이브러리를 접근하고 제어할 수 있도록 한다.

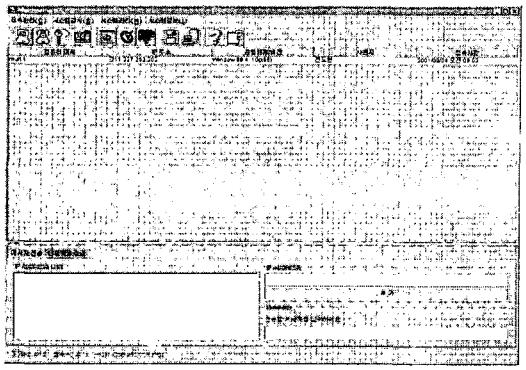


[그림 5] RMI 통신 구성도

자바의 원격 객체 통신에서 호출될 원격 객체를 서버 객체, 원격 객체를 사용하는 지역의 객체를 클라이언트 객체라고 부른다.[7] 하지만 본 시스템에서는 클라이언트에 호출될 원격 객체를 선언하고 이를 서버에서 접근하기 때문에 반대의 개념으로 적용되었다.

3.4 메인화면

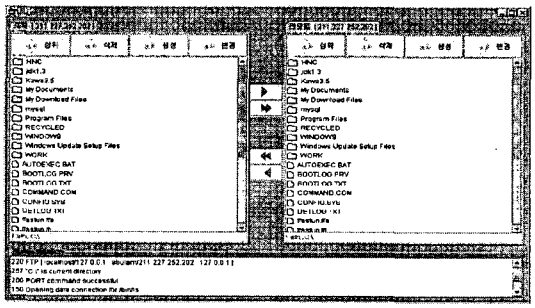
본 프로그램을 실행시키면 [그림 6]과 같은 화면이 나타난다. 서버 메인 화면에서는 전체 사용자와 사용자에 대한 대략적인 시스템 정보를 볼 수 있다. 메인 화면 아래에는 클라이언트에 경고 메시지를 전송하거나 클라이언트와의 네트워크 연결이 정상적으로 작동하는지를 테스트할 수 있는 탭이 구성되어 있다. 그 아래의 상태바에서는 현재 서버에 접속된 클라이언트의 수와 현재 선택되어진 클라이언트를 표시한다.



[그림 6] 서버측 메인 화면

3.4.1 파일 전송

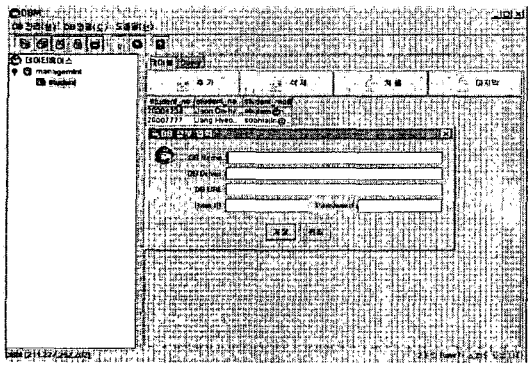
[그림 7]은 파일전송 부분이다. 서버와 클라이언트에 대한 모든 파일 정보들을 한 다이얼로그 안에서 동시에 볼 수 있다. 따라서 사용자는 간편하게 전송할 파일과 대상 경로를 선택할 수 있다. 파일 전송은 서버에서 클라이언트로, 클라이언트에서 서버로의 양방향이 가능하며 또한 다중 파일 전송이 가능하여 네트워크로 연결된 그룹내의 모든 클라이언트에 파일을 동시에 전송할 수 있다.



[그림 7] 파일전송 다이얼로그

3.4.2 DBM(Database Manager)

[그림 8]은 그룹내의 클라이언트 사용자와 클라이언트 관리를 쉽게 하기 위한 데이터 베이스 관리 툴 이다. MySQL 뿐만 아니라 메뉴에서 데이터 베이스 추가항목을 선택하여 다른 데이터베이스 서버에도 접속할 수 있도록 하였으며, 동시에 여러 DB를 접속한 후 데이터 처리가 가능하다.



[그림 8] DBM 화면

4. 결론 및 향후과제

본 연구에서 주안점을 맞춘 것은 자바로 구현이 어려운 시스템 제어 부분에서 JNI와 RMI을 이용한 서버에서의 클라이언트의 접근과 제어였다. 속도에 있어서는 다른 언어로 구현한 Application 보다는 조금 떨어지지만, 예상되는 프로그램을 충분히 수용할 수 있는 서버와 클라이언트의 경우 그룹내의 클라이언트에 대한 원격관리가 용이하도록 구현하였다. 향후 과제로는 그룹 외부에서도 인터넷을 통한 측 웹 브라우저를 이용하여 그룹내의 클라이언트를 모니터링할 수 있도록 확장하고 클라이언트 제어 기능 등이 있다.

참고문헌

[1] 이준연, 김대현, 김영찬, "클라이언트/서버 응용의 연산 부하 측정을 위한 시뮬레이터", 정보과학회 논문지(C), 제5권 제2호, 1999년 4월

[2] 유철중, "Java 통합 개발 환경에서 기능 컴포넌트들의 상호연동 기법, 한국정보처리학회 논문지 제5권 제11호, 1998년 11월

[3] 장경수, 신동렬, "JNI를 이용한 MMS 구현", 한국정보처리학회 논문지 제7권 제11호, 2000년 1월

[4] 민병준, 최재영, 김정국, 김문회, "실시간 객체지향 프로그램의 실행시간을 감시하는 모니터의 설계 및 구현", 한국정보처리학회 논문지, 제7권 제12호, 2000년 12월

[5] 김희철, 신필섭, 박영진, 이용두, 자바를 기반으로 한 글로벌 인터넷 컴퓨팅 환경", 한국정보처리학회 논문지, 제6권 제9호, 1999년 9월

[6] Matthew Robinson, Ravel Vorobiez 저 / 번역서 "Swing", 인포북

[7] http://www.javaland.co.kr/index.jsp

[8] http://www.javastudy.co.kr