non-Java 디바이스를 위한 Jini-SNMP의 설계 및 구현

손상혁[O] 이성동 김원태 안광선
경북대학교 컴퓨터공학과
{hanson, holyeast, wtkim, gsahn}@knight.ce.knu.ac.kr

Design and Implementation of Jini based SNMP system for non-Java device

Sang-Hyuck Son[O] Seong-Dong Lee Kwang-Seon Ahn
Dept. of Computer Engineering, Kyungpook National University

요 약

네트워크 관리에 있어서 Jini 기반의 SNMP 관리 구조는 Java의 장점을 수용하면서 더 많은 특징을 제공한다. 우선 Jini의 네트워크 플러그 앤 플레이 특징을 이용하여 관리자가 명시적으로 관리하고자 하는 대상을 지정할 필요가 없이 자동적으로 인식할 수 있다. 그리고 리스와 원격 이벤트를 사용하기 때문에 폴링으로 인한 자원의 낭비를 막을 수 있다. 하지만 Jini 기반의 네크워크 관리 구조는 자바 가상 머신을 탑재한 Java 디바이스만을 대상으로 하고 자바 가상 머신을 가지지 않는 non-Java 디바이스를 고려하지는 않는다. 이미 개발된 기존의 시스템이나 시스템 자원의 부족으로 자바 가상 머신을 탑재하지 못하는 시스템까지도 수용할 수 있어야 한다. 따라서 이 논문에서는 non-Java 디바이스까지 수용할 수 있는 Jini 기반의 SNMP 시스템을 설계하고 구현한다.

1. 서 론

SNMP[1]는 구조가 간단하고 오버헤드가 적으며 개발이 쉽고 확장성이 뛰어나다. 하지만 SNMP 기반의 네트워크 관리에서는 폴링으로 인한 통신 오버헤드가 크고 낮은 보안성과 이식성 등의 단점을 가진다. 이를 해결하기 위해서 Java나 Jini 기반의 SNMP[2, 3]를 이용하여 해결하고자 하는 연구가 있었다. 특히 Jini 기반의 경우에는 Jini[4, 5, 6]가 Java로 개발되었기 때문에 Java의 여러 장점을 그대로 수용하면서 더 많은 특징을 가진다. 일반적인 네트워크 관리에서는 에이전트가 설치되어 있는 디바이스를 자동적으로 인식할 수 있는 방법이 없으므로 명시적으로 관리자가 이를 지정해야 한다. 하지만 Jini에서는 네트워크 플러그 앤 플레이(Plug and Play)를 지원하여 서비스 가능한 상태의 에이전트들이 자동적으로 등록함으로써 이것을 피할 수 있다. 그리고 리스와 원격 이벤트의 개념을 이용하여 폴링에 의한 자원의 낭비를 막을 수 있다. 하지만 Jini 기반의 네트워크 관리에 있어서 non-Java 디바이스를 수용하지 못하는 단점을 가진다. 자바 가상 머신을 탑재하지 않은 기존의 시스템이나 시스템 자원의 부족으로 자바 가상 머신을 탑재하지 못하는 시스템까지도 수용할 수 있어야 한다.

이에 본 논문에서는 non-Java 디바이스를 수용할 수 있는 Jini 기반의 SNMP 시스템를 설계하고 구현한다. 본 논문의 구성은 다음과 같다. 2장에서는 Jini와 Jini 네트워크에서 non-Java 디바이스를 통합하기 위한 서로게이트(surrogate) 방법에 대해서 알아보고, 3장에서는 본 시스템의 구성과 동작 과정을 설명하고 끝으로 4장에서 결론을 맺는다.

2. 관련 연구

2.1 Jini

Sun Microsystems에서 제안한 기술로서 Jini는 네트워크상의 플러그 앤 플레이를 지원한다.

(1) 디스커버리(Discovery)

어떤 서비스나 클라이언트가 Jini 네트워크에 참가하기 위해서는 먼저 하나 또는 그 이상의 Jini 룩업 서비스의 레지스트라(registrar)를 획득해야 한다. 여기서 필요한 과정이 디스커버리 프로토콜이며 레지스트라(registrar)는 룩업 서비스의 프락시 역할을 한다. 서비스측에서는 서비스 등록을 위해, 클라이언트 측에서는 서비스 검색을 위해 레지스트라를 이용한다.

(2) 조인(Join)

어떤 서비스가 네트워크에 접속하게 되면 디스커버리 프로토콜을 이용해서 룩업 서비스를 찾고 그 룩업 서비스에 자신을 등록한다. 먼저 서비스의 프락시 객체와 속성(attribute)를 포함하는 서비스 아이템(service Item)을 만들고 룩업 서비스 레지스트라의 register()를 통해 그 서비스 아이템 등록하게 된다.

(3) 리스(Lease)

분산 환경에서는 예고 없이 협력하는 그룹의 하나의 멤버가 동작하지 않거나 멤버들 사이의 연결 실패로 인해 자원이나 서비스가 결코 반환되지 않는 결과를 초래할 수 있다. 이것을 해결하기 위해 리스는 어떤 자원이나 서비스에 대한 사용을 어떤 시간동안 허가하고 그 시간이 지나면 그 자원이나 서비스를 반환하게 한다.

(4) 원격 이벤트(distributed Event)

원격 이벤트의 목적은 하나의 자바 가상 머신에 있는 객체가 다른 자바 가상 머신의 객체에서 일어날 수 있는

이벤트에 대해 통보(notification)을 받을 수 있게 하는 것이다. 이벤트 소비자(event consumer)는 이벤트 생산자(event generator)에 이벤트 리스너 스텁 객체를 등록한다. 해당 이벤트가 발생하면 이벤트 생산자는 그 이벤트에 등록된 이벤트 리스너 스텁 객체를 이용하여 이벤트 소비자에 이벤트 발생을 알려준다.

2.2 서로게이트

서로게이트 모델[6]은 자바 가상 머신이 없는 non-Java 디바이스들을 Jini 네트워크에 참가시키기 위해서 현재 Jini Device Architecture에서 제안하고 있는 세 가지 방법중의 하나이다. 서로게이트 호스트(surrogate host)의 자바 가상 머신이나 자바 어플리케이션 환경(Java application environment)을 이용하여 디바이스가 Jini 네트워크에 참가할 수 있도록 하는 방법이다. 네트워크에 연결된 디바이스는 먼저 서로게이트 호스트를 찾고 자신의 서로게이트를 등록한다. 서로게이트는 클라이언트가 사용할 서비스 코드와 서로게이트가 디바이스와 통신할 때 필요한 코드를 포함한다. 디바이스가 서로게이트 호스트에 등록하게 되면 서로게이트 호스트는 서로게이트를 실행시킨다. 서로게이트는 디바이스를 대신하여 룩업 서비스에 등록하고 리스를 갱신하는 역할을 수행한다. 클라이언트와 디바이스는 서로게이트를 통해 간접적으로 연결된다.

3. 설계 및 구현

Jini 기반의 SNMP는 자바 가상 머신을 탑재하지 않은 non-Java 디바이스를 수용하지 못하는 단점을 가진다. 따라서 본 논문에서는 Jini Device Architecture의 서로게이트 모델을 이용하여 non-Java 디바이스를 수용하는 Jini 기반의 SNMP 시스템을 설계하고 구현한다.

3.1 구조

본 논문에서 구현한 시스템은 크게 관리자, 룩업 서비스, 에이전트, 서로게이트 호스트(surrogate host)로 나눌 수 있고, 다시 에이전트는 Java 디바이스 에이전트와 non-Java 디바이스 에이전트로 구분한다.

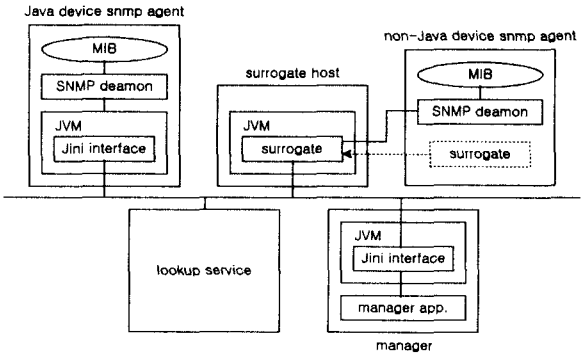

그림 1 시스템 구성

Java 디바이스 에이전트는 SNMP 네트워크 관리를 위한 MIB(Management Information Base)와 SNMP 데몬, 그리고 Jini 네트워크에 참여하기 위한 기능을 수행하는 Jini 인터페이스로 구성된다. non-Java 디바이스 에이전트는 직접 Jini 네트워크에 참여할 수 없기 때문에 서로게이트를 전달할 서로게이트 호스트를 필요로 한다.

3.2 동작 과정

SNMP 에이전트들은 관리자가 자신들의 서비스를 이용하도록 하기 위해서 룩업 서비스를 찾고 자신의 서비스 프락시 객체를 등록한다. 관리자는 이 에이전트들의 서비스를 이용하기 위해서 룩업 서비스를 찾고 원하는 서비스를 검색하여 그 서비스의 프락시 객체를 받아온다. 이 프락시 객체를 통해서 관리자는 룩업 서비스를 거치지 않고 직접 에이전트들에게 네트워크 관리에 필요한 요구를 전달하고 그 응답을 받는 것이다. 이 과정은 Java 디바이스 에이전트와 non-Java 디바이스 에이전트에 따라 차이가 있으므로 세부 동작 과정은 이 두 가지 에이전트를 구별하여 설명한다.

(1) 관리자와 Java 디바이스 에이전트의 동작

Java 디바이스 에이전트의 경우에는 Jini 네트워크에 직접적으로 참여할 수 있으므로, 먼저 디스커버리 프로토콜을 이용하여 룩업 서비스를 찾는다. 룩업 서비스는 그에 대한 응답으로 자신의 프락시 객체인 레지스트라를 전달한다. 에이전트는 이 레지스트라를 통해서 자신이 제공할 서비스의 프락시 객체와 속성을 등록한다. 관리자는 에이전트가 등록한 서비스들을 사용하기 위해 에이전트와 마찬가지로 디스커버리 프로토콜을 이용하여 룩업 서비스를 찾는다. 이에 대한 응답으로 관리자는 룩업 서비스의 프락시 객체인 레지스트라를 받는다. 이 레지스트라를 통해서 원하는 에이전트의 서비스를 검색하고 그 서비스의 프락시 객체를 받아온다. 관리자는 이 프락시 객체의 메서드를 호출함으로써 에이전트에 요구를 전달하고 그 응답을 받을 수 있는 것이다. 이 동작 과정을 시퀀스 다이어그램으로 표현하면 그림 2와 같다.

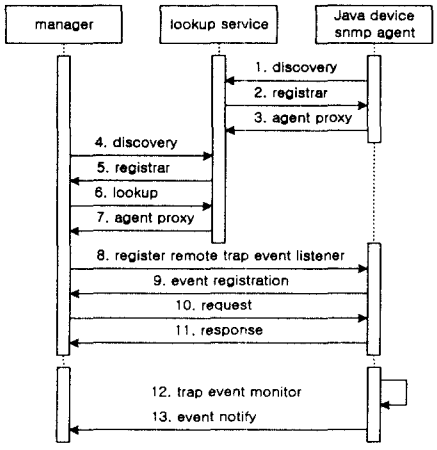

그림 2 관리자와 Java 디바이스 에이전트간의 동작 과정

관리자는 트랩 메시지(trap message)를 받기 위해 해당 에이전트에 원격 이벤트 리스너(remote event listener)를 등록해야 한다. 에이전트는 트랩이 발생하게 되면 이 리스너를 통해 관리자에게 통보한다.

(2) 관리자와 non-Java 디바이스 에이전트 동작

non-Java 디바이스 에이전트의 경우에는 Jini 네트워크에 직접 참여하지 못하기 때문에 서로게이트 호스트가 필요하다. 먼저 에이전트는 디스커버리 프로토콜을 이용하여 서로게이트 호스트를 찾는다. 찾은 서로게이트 호스트에 대해 에이전트는 등록(registration)을 요구하고 이에 대한 응답으로 서로게이트 호스트는 에이전트로부터 서로게이트를 받아와서 수행시킨다. 수행된 서로게이트는 자신의 에이전트가 제공하는 서비스를 룩업 서비스에 등록하고 자신의 에이전트가 존재하는지를 계속 감시하는 기능을 한다. 또 관리자의 요구를 받아 에이전트에 전달하고 그에 대한 응답을 관리자에 전달한다. 관리자는 룩업 서비스 검색을 통해 원하는 서비스의 프락시 객체를 받아오고 그것의 메서드를 호출함으로써 에이전트의 서비스를 이용하는데, 실제로 관리자는 에이전트가 아니라 서로게이트와 통신을 하며 트랩에 대한 원격 이벤트 리스너도 서로게이트에 등록한다. 관리자와 non-Java 디바이스 에이전트사이의 동작 과정을 시퀀스 다이어그램으로 표현하면 그림 3과 같다.

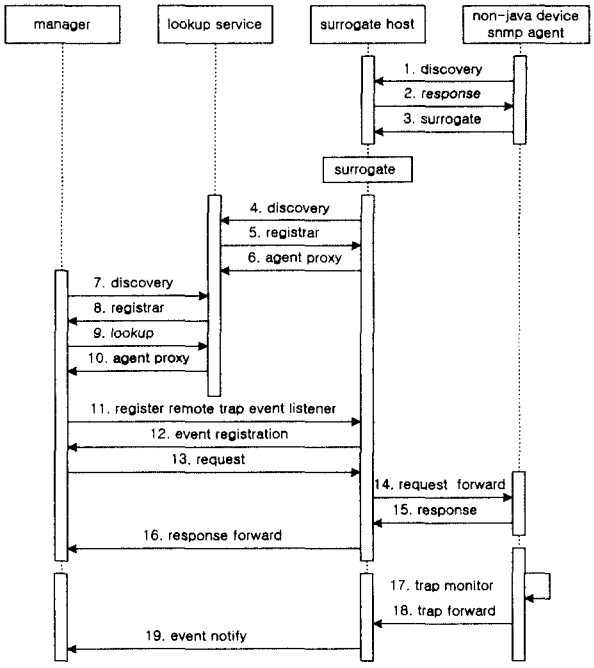

그림 3 관리자와 non-Java 디바이스 에이전트간의 동작 과정

서로게이트 호스트와 디바이스간의 통신은 Jini 네트워크에서 사용하는 RMI[8]를 사용할 수 없으므로 다른 별도의 프로토콜이 필요하다. 먼저 에이전트가 등록을 요구하고 서로게이트 호스트가 디바이스로부터 서로게이트를 받아오는 과정에서는 HTTP, FTP, TFTP 등을 사용한다. 그리고 관리자의 RMI를 이용한 서비스 요구에 대해서 서로게이트는 SNMP 게이트웨이(gateway) 역할을 한다. 관리자의 에이전트 프락시 객체에 대한 RMI 호출을 SNMP 요구로 변환하여 에이전트에 전달하고 그 응답을 받아 관리자에 다시 돌려준다. SNMP 트랩도 Jini 이벤트로 변환하여 등록된 이벤트 리스너를 이용해서 관리자에 전달하는 것이다. 마찬가지로 서로게이트와 디바이스가 존재하는 지를 서로 감시하는 과정에서도 RMI 스텁을 통한 리스를 이용할 수 없다. 그 대신에 서로게이트가 수행된 후에 지속적으로 일정한 시간마다 디바이스의 존재를 감시하는 방법을 사용한다. 디바이스는 서로게이트가 주기적으로 자신의 존재를 감시하는 지를 체크함으로써 서로게이트의 존재를 확인할 수 있다.

4. 결 론

기존의 Jini 기반의 SNMP 시스템은 Java의 장점을 수용하면서 에이전트 디바이스를 자동적으로 인식하고 폴링에 의한 자원의 낭비를 막는 부가적인 특징을 가진다. 하지만 자바 가상 머신이 없는 non-Java 디바이스를 관리할 수 없는 단점을 가진다. 따라서 본 논문에서는 서로게이트 모델을 이용하여 non-Java 디바이스를 수용할 수 있는 Jini 기반의 SNMP 시스템을 설계하고 구현하였다. 이미 개발된 기존의 시스템이나 네트워크나 시스템 자원의 제한으로 자바 가상 머신을 가지지 못하는 시스템을 Jini 기반의 SNMP 시스템에 수용할 수 있을 것이다.

5. 참고 문헌

[1]. RFC 1157, *A Simple Network Management Protocol(SNMP)*, May. 1990.

[2]. J. Park, N. Ban and T. Kim, "Java-based Network Management Environment", *IEEE*, 1998.

[3]. S. Lee, "Design and Implementation of Jini-based SNMP system", *KISS*, 2000

[4]. Sun Microsystems, *Jini Specifications v.1.1* , Available at http:// www.javasoft.com/jini/specs/, 1999.

[5]. W. Edward, *Core Jini 2E*, Prentice Hall, 2000

[6]. S. Oaks and H. Wong, *Jini in a Nutshell*, O'Reilly & Associates, 2000.

[7]. Jini Community, *Surrogate Project*, Available at http://developer.jini.org/exchange/projects/surrogate/

[8]. Sun Microsystems, *Java Remote Method Invocaton Specification*, Available at http://http://java.sun.com/j2se/1.3/docs/guide/rmi/, 1998.