UPnP-to-Jini 서비스의 설계 및 구현

한 상 숙[†] · 은 성 배[††] · 김 철 민[†††]

요 약

홈 네트워크의 발전에 따라 홈 네트워크에 연결되는 장치들 사이의 상호운용을 지원하는 홈 네트워크 미들웨어가 다양하게 개발되고 있다. UPnP와 JINI는 그 중의 대표적인 미들웨어로 개별적으로 연구개발이 이루어지고 있다. 따라서 홈 네트워크의 사용 확대를 위한 홈 네트워크 미들웨어 사이의 연동방안이 필요하다. 본 논문에서는 홈 네트워크 미들웨어인 UPnP와 Jini의 동작에 대하여 알아보았고, UPnP의 동작과정의 특성을 이용하여 UPnP 디바이스를 Jini 클라이언트가 사용할 수 있도록 하는 UPnP-to-Jini 서비스를 설계하고 구현하였다. 전등을 켜고 끄는 응용 예를 통하여 상호 연동됨을 보였다.

Design and Implementation of UPnP-to-Jini Service

Sangsuk Han[†] · Seongbae Eun[††] · Chulmin Kim[†††]

ABSTRACT

According to the development of home-network, home-network middleware supporting interoperability between devices connecting to home-network has been variously proposed. As a typical middleware, UPnP and Jini are developed individually. So, to extend the usage of home-network, it is necessary to have connections among various home-network middleware. In this paper, we study about home-network middleware UPnP and Jini, and design and implement UPnP-to-Jini service which allows Jini client to use UPnP device using the speciality of UPnP performance process. Using application of light on-off we implemented the connection of them.

키워드 : 홈 네트워크(Home-Network), 미들웨어(Middleware), 연동(Interoperability), UPnP, Jini

1. 서 론

홈 네트워크를 구성하는 기술은 크게 세 부분으로 나눌 수 있다. 첫째, 외부 네트워크와 홈 네트워크를 연결하는 xDSL, 케이블 모뎀, 전력선 통신, FTTH(Fiber to the Home)등의 기술이 있다. 둘째, 홈 네트워크 장치들을 연결하는 HomePNA, USB, Ethernet, IEEE1394, Bluetooth, WirelessLAN, HomeRF 등의 홈 네트워크 접속기술이 있다. 셋째, 연결된 장치들의 상호연결, 상호운용을 제공하는 UPnP(Universal Plug and Play), Jini, HAVi(Home Audio Video Interoperability), OSGi(Open Services Gateway Initiative) 등의 미들웨어기술이 있다.

이러한 홈 네트워크 미들웨어기술[1-3]은 홈 네트워크를 구성하는 장치들을 상호 연결시킬 뿐만 아니라, 다양한 환경에서 사용할 수 있도록 지원하여 사용자의 편의를 확대시킨다.

홈 네트워크에서 서로 다른 미들웨어기술을 적용하여 개발된 장치가 한 가정의 홈 네트워크에 연결될 수도 있다. 이 경우 서로 다른 미들웨어 사이의 호환성이 없어서 홈 네트워크에 접속된 장치를 원활하게 사용할 수 없는 문제가 발생할 수 있다. 이는 사용자의 불평을 초래하고 더 나아가 홈 네트워크 발전의 저해요소가 될 것이다.

따라서 홈 네트워크 미들웨어를 개발하는 사람들은 상이한 미들웨어 사이의 비 호환성 문제를 해결할 필요가 있다. 홈 네트워크 미들웨어기술을 통합하여 단일 표준화를 이루거나, 각각의 미들웨어기술을 연동시키는 방법이 있다.

현재까지 UPnP와 LonWorks의 상호연동 및 HAVi와 Jini의 상호연동 등 몇 가지의 연동지원기술이 제시되었다[4, 5]. 그러나 UPnP와 Jini 사이의 연동기술은 아직 제시된 바가 없다. UPnP와 Jini 미들웨어 기술은 마이크로소프트와 썬 마이크로시스템이 각각 주도하여 만든 미들웨어 기술로서 이 방면의 기술 중 영향력이 큰 기술이라 할 수 있다.

본 논문에서는 UPnP와 Jini 기술의 동작방법 및 특성을 알아보고 UPnP가 갖는 동작과정의 특징을 이용하여 UPnP

※ 본 논문은 한남대학교 2001년도 교비연구비의 지원을 받아서 수행한 결과입니다.
† 준 회 원 : 대전기능대학 멀티미디어과 교수
†† 정 회 원 : 한남대학교 정보통신공학과 교수
††† 준 회 원 : 한남대학교 대학원 정보통신공학과
논문접수 : 2003년 9월 24일, 심사완료 : 2003년 12월 17일

와 Jini 미들웨어기술의 상호 연동방법을 제시하였다. 그리고 제시된 방법을 적용하여 Jini 클라이언트를 통해 UPnP 디바이스를 제어할 수 있도록 지원하는 UPnP-to-Jini 서비스 응용예를 구현하였다.

2장에서는 UPnP와 Jini 미들웨어 기술의 개념 및 동작과정등 배경에 대하여 기술하였고, 3장에서는 UPnP와 Jini의 비교, 장·단점 등의 동작 특성에 대하여 기술하였고, 4장에서는 UPnP-to-Jini 서비스의 설계로, 연동장치, 구조 및 세부 기능 및 알고리즘에 대하여 기술하였고, 5장에서는 UPnP-to-Jini 서비스의 구현으로 구현 환경, 구현 내용 및 시연을 하였으며, 6장에서는 연구결과를 정리하고 연구결과를 통해 필요성이 도출된 향후 연구과제에 대해서 기술하였다.

2. 배 경

2.1 UPnP

UPnP 미들웨어기술은 1999년에 마이크로소프트, Intel 등의 회사가 모여 홈 네트워크를 위해 제안한 기술[6-9]이며, 구조상의 보편적 특성은 다음과 같다[10].

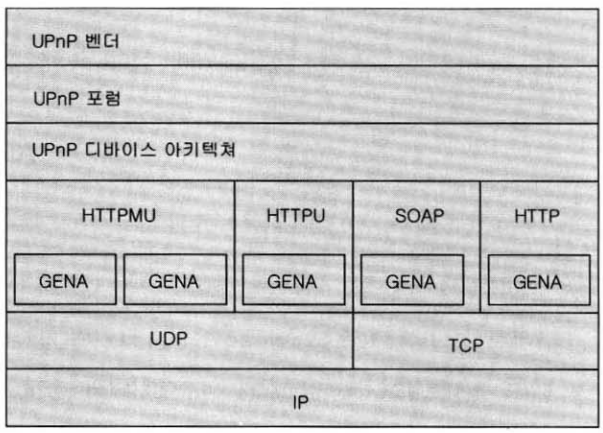
- Plug & Play
- 특정한 기기이기보다는 일반적인 프로토콜을 사용한다.
- 중요한 물리적 미디어와 전송으로부터 독립적이다.
- 어떤 프로그램 언어와 동작시스템에서도 구현될 수 있다.
- XML과 SOAP 등과 같은 HTTP와 다른 인터넷 기술에 지렛대 역할을 한다.
- 브라우저를 경유하여 기기의 사용자 인터페이스와 상호작용을 제어 할 수 있게 한다.
- 재래적인 응용프로그램의 제어를 가능하게 한다.
- 벤더들은 다양한 기기에 기반을 두고 있는 UPnP의 제어 프로토콜에 일반적으로 동의한다.
- 필요시마다 기본적인 제어 프로토콜을 일방적으로 확장할 수 있다.

UPnP가 동작하는 과정은 디바이스, 서비스, 제어점의 세 부분으로 나누어 설명될 수 있다. 디바이스는 서비스를 제공하는 장치를 의미하며, 제공할 서비스를 포함하고 있다. 서비스는 서비스 내용을 갖고 있어서 제어점의 요청에 응답하여 장치를 사용할 수 있도록 한다. 제어점은 서비스를 사용할 수 있도록 하는 일종의 프로그램이다. 제어점의 예로는 윈도우 운영체제에 포함된 UPnP 제어점을 들 수 있다.

UPnP 제어점을 통하여 네트워크에 참가한 디바이스를 발견하고, 디바이스가 제공하는 서비스를 사용할 수 있다. 이와 같은 과정은 실제적인 사용자의 제어와 관리를 제외하고는, 사용자의 개입 없이 UPnP 디바이스와 UPnP 제어점 사이의 네트워크 서비스를 위해 UPnP에서 정의한 프로토콜을 통하여 이루어지며, 인터넷 환경에서 정의한 프로토콜의 장점을 모두 수용하고 있다.

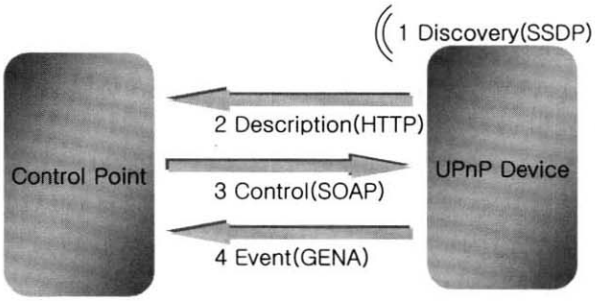
UPnP 디바이스가 네트워크에 참가하면 UPnP 디바이스는 가장 먼저 어드레싱 과정을 거친다. 어드레싱은 사용자에 의해 UPnP 디바이스가 네트워크에 접속될 때 네트워크에 연결되어 작동할 수 있도록 IP 주소를 획득하는 과정이다. IP 주소의 획득은 UPnP 디바이스에 설정된 고정 IP

주소를 사용하거나 DHCP(Dynamic Host Configuration Protocol) 또는 자동IP를 이용하여 통신 가능한 IP 주소를 설정하여 사용할 수 있다. 어드레싱 과정은 UPnP 디바이스의 동작을 위한 가장 기초적인 과정이다.



(그림 1) UPnP 프로토콜 스택

(그림 1)은 UPnP 프로토콜 스택을 나타내고, (그림 2)는 어드레싱 이후의 UPnP의 동작과정을 간략하게 도식하였다.



(그림 2) UPnP 동작과정

- 디스커버리 : 새로운 UPnP 디바이스가 네트워크에 참가한 것을 알리는 과정이다. SSDP(Simple Service Discovery Protocol)를 이용하여 HTTPMU(HTTP Multicast over UDP)를 통하여 멀티캐스트 메시지를 보낸다. 이 때 네트워크에 위치한 제어점 멀티캐스트

주소를 받은 디스커버리 메시지를 통하여 이후의 단계를 진행할 수 있다.

- 디스크립션 : 제어점은 네트워크에 참가한 UPnP 디바이스로부터 상세 정보를 얻기 위하여 UPnP 디바이스에게 디바이스 정보를 요청한다.
- 제어 : 제어점은 사용자가 선택한 UPnP 디바이스 제어 명령을 UPnP 디바이스에 전달한다. 전달되는 메시지는 SOAP 메시지로 작성된다. SOAP은 UPnP 디바이스에 전달 할 명령을 규격화하여 작성하도록 한다.
- 이벤트 : UPnP 디바이스가 다수의 제어점을 통해 제어받고 있을 때 제어점을 통해 변경된 상태변화를 이벤트를 통해 모든 제어점에 전달하여 제어점이 갖고 있는 디바이스 상태와 실제 디바이스 상태를 일치시킨다.
- 프리젠테이션 : UPnP 디바이스가 제어점에 제공하는 사용자 인터페이스로서, 제어점은 UPnP 디바이스가 제공하는 프리젠테이션 URL을 통해 사용자에게 HTML 페이지를 표현하여 친숙한 사용자 인터페이스를 제공하며, 더 쉽게 UPnP 디바이스를 사용할 수 있도록 한다.

예를들어, UPnP를 지원하는 장치인 UPnP light를 네트워크에 참가시키면 윈도우에서 제공하는 제어점을 통해 UPnP light가 네트워크에 참가했음을 알게 되고, 사용자는 제어점을 통해 UPnP light가 제공하는 점등, 소등, 밝기 조절 등을 할 수 있다.

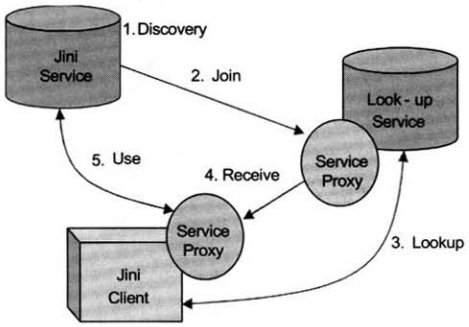
2.2 Jini

Jini는 Sun Microsystem에서 개발한 미들웨어로서 자바를 기반으로 하여 다양한 방식으로 네트워크에 접속된 장치나 소프트웨어를 동적으로 상호 작용하도록 하게 하는 기술이다. 자바 환경에서 동작하므로 운영체제나 기타 하드웨어 플랫폼에 구애받지 않는다. Jini는 분산 환경에서 각각의 장치 또는 소프트웨어간의 상호운용을 위한 방법으로 정의되었을 뿐만 아니라, 자바 소프트웨어 기반의 기술이기 때문에 소프트웨어 서비스를 제공하는데 좀 더 적합한 구조를 갖고 있다[11-15].

Jini는 크게 룩업 서비스, 서비스, 클라이언트 및 네트워크의 네 부분으로 나뉜다. 룩업 서비스는 Jini 서비스로부터 서비스 프록시를 받아 등록시켜서 클라이언트가 사용할 수 있도록 하는 서버의 역할을 담당한다.

Jini 서비스는 장치에 포함되며, 장치가 제공하는 기능을 의미하고, Jini 클라이언트는 Jini 서비스를 사용하는 부분이다. 마지막으로 네트워크는 세 부분의 유기적인 연결을 담당한다. Jini의 설계상으로 네트워크 전송 매체나 프로토콜에는 무관하지만 현재는 TCP/IP를 사용하도록 구현되어 있다.

Jini 네트워크는 룩업 서비스가 동작하고 있는 네트워크에서 Jini 서비스가 참가하여 Jini 네트워크 서비스가 시작된다. (그림 3)은 Jini 네트워크의 동작과정을 도식한 것이다.



(그림 3) Jini 동작과정

- 디스커버리 : Jini 서비스가 Jini 네트워크에 참가하면 Jini 서비스는 룩업 서비스를 찾는 작업을 하는 데 이를 디스커버리라 한다. 디스커버리는 유니캐스트 또는 멀티캐스트 메시지를 통하여 네트워크에 존재하는 룩업 서비스를 검색할 수 있다.
- 죠인 : 룩업 서비스를 찾으면 Jini 서비스는 자신이 제공하는 서비스 프록시를 룩업 서비스에 등록한다. 서비스 프록시는 Jini 클라이언트의 서비스를 의미하며, 서비스 프록시의 동작방법은 크게 세 가지 방법으로 나눌 수 있다.

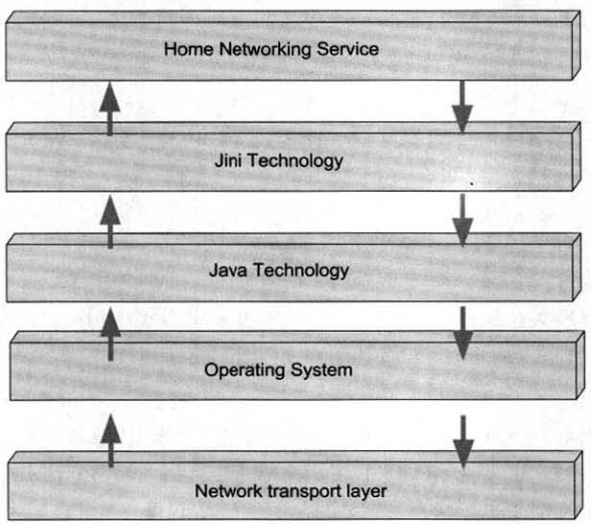
첫째, 서비스 프록시 자체가 Jini 서비스가 제공하는 서비스와 동일한 기능을 하는 경우이다. Jini 서비스가 제공하는 서비스 프록시는 자바 클래스 파일로서 Jini 클라이언트에 전달된 후 Jini 클라이언트 환경에서 수행된다.

둘째, Jini 클라이언트에 전달된 서비스 프록시가 Jini 서비스와 RMI 연결을 하는 RMI_Stub인 경우이다. Jini 클라이언트가 받은 RMI_Stub를 수행하면 실제 수행은 RMI로 연결된 Jini 서비스의 RMI_Skel을 통해서 Jini 서비스 측에서 수행된다.

셋째, 앞의 두 가지 방법이 동시에 사용되는 서비스 프록시이다. 이 때는 Jini 서비스에서 동작하는 부문과 Jini 클라이언트에서 동작하는 부분이 모두 존재한다. RMI는 자바에서 정의한 것으로 실행시간에 원격지의 자바 클래스 파일의 메소드를 호출할 수 있는 방법을 정의한 것이다. 메소드 호출은 클라이언트에서 이루어지지만 실제 호출에 대응하는 실행은 서버에서 이루어진다. RMI는 Jini의 근간을 이루는 기술이다[10].

- 룩업, 리스 : 룩업은 Jini 클라이언트가 룩업 서비스로부터 Jini 서비스를 검색하는 단계로 룩업을 통해 Jini 서비스를 획득하면 서비스 프록시를 얻는다. 이때 리스를 통해 획득한 서비스 프록시의 사용기간을 갱신하게 된다.
- 제어 : 서비스 프록시를 제공받으면 Jini 클라이언트에서 제공되는 사용자 인터페이스를 통해 Jini 서비스를 사용할 수 있다. 서비스 프록시의 성격에 따른 동작을 Jini 클라이언트와 Jini 서비스가 알아서 처리하게 된다. 이 경우에도 Jini 서비스 사용자는 내부적인 과정을 전혀 알 필요가 없이 사용자 인터페이스를 통해 Jini 서비스를 사용할 수 있다.

예를들어, Jini를 지원하는 Jini 프린터를 네트워크에 연결하면 Jini 프린터 서비스는 룩업 서비스에 자신의 서비스 프록시를 등록한다. 사용자는 워드프로세서를 통해 문서를 작성한 후 프린트가 필요할 때 워드 프로세서에 포함된 Jini 클라이언트를 통해 Jini 프린터의 서비스 프록시를 받아서 원하는 출력물을 얻을 수 있게 된다. (그림 4)는 Jini 소프트웨어 구조를 보여주고 있다.



(그림 4) Jini 소프트웨어 구조

3. UPnP와 Jini의 동작 특성

3.1 UPnP와 Jini의 비교

<표 1>은 UPnP와 Jini 미들웨어 기술의 특성을 비교한 것이다. UPnP와 Jini 미들웨어 기술은 사용자 입장의 작동 방법은 매우 유사하지만 실제 구현에 관련해서는 많은 차이점을 갖고 있음을 알 수 있다.

〈표 1〉 UPnP와 Jini 미들웨어 기술의 비교

	UPnP	Jini
목 표	네트워크 Plug & Play	네트워크 Plug & Play
사용자 인터페이스	HTML + 스크립트	자바 애플리케이션
서비스 관리	룩업 서비스 관리	중앙 서버가 없음
동작 과정	유 사	유 사
구현 언어	언어 독립적	자 바
기반 기술	개방형 프로토콜 기반	자바 객체, RMI 기반

3.2 UPnP와 Jini의 장 · 단점

<표 2>는 UPnP와 Jini의 장 · 단점을 비교한 것이다. UPnP의 경우 표에서 보는 바와 같이 여러 장점 가운데 가장 큰 장점은 Plug & Play가 된다는 점과 일반적인 프로토콜을 이용한다는 점이다. 또한 Jini의 경우에는 UPnP의 경우와 같이 Plug & Play를 장점으로 들 수가 있고, 특히 자바기반의 기술이기 때문에 JVM만 설치되어 있으면 운영체제, H/W 및 S/W 플랫폼에 무관하게 동작한다는 점을 가장 큰 장점으로 들 수가 있다.

한편 UPnP의 단점으로는 Windows 운영체제에 탑재하기 때문에 시스템 과부하가 있을 수 있고 패치를 통해서만 보안문제를 해결할 수가 있다. 또한 Jini의 단점으로는 Java 환경에서 사용하기 때문에 일반 가전제품의 경우 Jini 기술을 적용해야만 서비스가 가능하고, 서비스 실행 중에는 다양한 동작들을 얻기 어려운 점이 있으며, Java 환경에서만 구현이 가능하다는 점이다.

따라서 두 기술의 유사한 동작과정을 활용하여 어떤 한 부분의 연동을 이루게 되면 각각의 미들웨어가 갖는 단점을 극복할 수 있을 뿐만 아니라 두 기술의 장점도 극대화시킬 수 있게 되는 것이다.

〈표 2〉 UPnP와 Jini의 장 · 단점 비교

	UPnP	Jini
장점	• Plug & Play • 일반적 프로토콜 사용 • S/W에 구애받지 않음 • 웹 및 전화를 이용하여 제어 • 프로토콜을 확장하여 사용 가능 • 다른 미디어 접근 기술에 독립적 • 오래된 응용프로그램 제어 가능	• Plug & Play • JVM만 있으면 OS, H/W, S/W 플랫폼에 제한 없음 • 별도의 서버가 없음
단점	• Windows 운영체제에 탑재로 운영체제 과부하가 있을 수 있음 • 보안 문제 - 패치를 통해 해결	• 자바 기반에서만 구현 가능 (JVM이 반드시 필요) - 속도저하 • 네트워크에 연결되는 가전제품들은 Jini 기술을 적용해야만 서비스 가능 • 서비스 실행 중에는 서비스 프록시가 제공하는 다양한 동작들을 얻기 어려움 • 룩업 서비스가 존재해야 하므로 완벽한 분산 환경 제공 불가능

3.3 Jini-to-UPnP의 연동 가능성

UPnP-to-Jini의 연동은 UPnP에서 Jini로 제어점을 보내주면 Jini가 받아서 자바 소스 코드로 변환하여 처리 후 제어점을 UPnP로 반환할 수 있기 때문에 연동이 가능할 수가 있다.

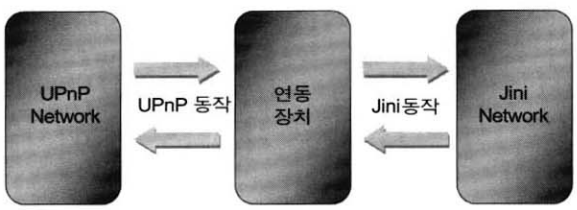
그러나 Jini-to-UPnP의 경우는 Jini의 서비스 목록을 먼저 보내준 후 자바 소스 코드를 UPnP에게 보내주기 때문에 UPnP가 자바 소스코드를 처리할 수가 없다. 따라서 UPnP-to-Jini는 연동이 가능하나 Jini-to-UPnP는 UPnP가 서비스 목록을 처리할 수 없는 한 연동이 불가능 하다고 할 수 있다.

4. UPnP-to-Jini 서비스의 설계

4.1 연동장치

UPnP와 Jini 미들웨어 기술의 상호연동을 위해 (그림 5)와 같은 연동 장치가 필요하다. 연동장치는 UPnP와 Jini 미들웨어 기술 사이에 위치하여 양쪽의 동작과정에서 일어나는 다양한 메시지를 변환하며, 서로 다른 미들웨어 기술을 활용하여 네트워크에 전송한다.

예를 들어 UPnP 네트워크에서 발생하는 디스커버리 메시지를 연동장치에서 Jini 룩업 메시지로 변환하여 Jini 네트워크로 전달할 경우, Jini 네트워크는 룩업에 대한 응답을 한다. 연동장치는 Jini 네트워크에서 발생한 응답을 UPnP 네트워크에 전달하면 디스커버리 단계가 이루어진다. 그러나 Jini와 UPnP가 갖는 구현 언어와 기반 기술의 차이점 때문에 연동장치의 개발이 양쪽 모두가 가능한 것은 아니다. Jini의 경우 Jini 서비스에서 제공하게 될 서비스에 관한 정보인 서비스 프록시가 자바로 작성된 클래스 파일이므로 연동장치가 실행 중에는 서비스 프록시가 제공하는 다양한 동작들을 얻기 어렵다는 문제점이 있다.



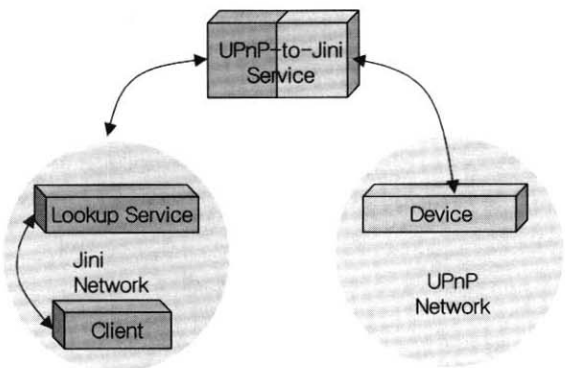
(그림 5) UPnP와 Jini의 연동장치 개념도

반대로 UPnP의 경우, UPnP 디바이스에 대한 다양한 정보들을 UPnP 동작과정에서 XML로 작성된 파일로 얻어낼 수 있고, 이것을 가공하여 Jini 미들웨어 서비스에 전달할 수 있다. 제어의 경우도 SOAP을 통해 텍스트 기반으로 작성되기 때문에 디스크립션에서 얻은 정보를 통해 SOAP 메시지를 작성할 수 있다. 이와 같이 실행 중에 UPnP 디바이스를 찾아 사용할 수 있도록 지원하는 것이 UPnP의 구성요소에 포함되어 있는 제어점이다. 제어점은 다양한 UPnP 디바이스가 제공하는 모든 서비스를 사용할 수 있도록 지원한다. UPnP의 동작이 XML, SOAP, GENA(General Event Notification Architecture), SSDP 등의 프로토콜을 기반으로 하여 동작하기 때문에 가능하다.

위와 같이 UPnP가 갖고 있는 동작과정의 특징을 이용하여 UPnP 디바이스를 Jini 클라이언트가 사용할 수 있도록 UPnP-to-Jini 서비스를 구현할 수 있다.

4.2 UPnP-to-Jini 서비스

UPnP-to-Jini 서비스는 UPnP와 Jini를 모두 인식할 수 있도록 작성되어야 한다. (그림 6)은 UPnP-to-Jini 서비스의 연동 개념도를 나타내고 있다. UPnP 디바이스는 UPnP-to-Jini 서비스를 통해 Jini 서비스로 변환되어 Jini 룩업 서비스에 등록된다. 사용자인 Jini 클라이언트는 룩업 서비스가 제공하는 서비스 프록시를 통해 제어명령을 하고, 이 제어명령은 UPnP-to-Jini 서비스를 통해 UPnP 제어 메시지로 변환되어 UPnP 디바이스에 전달된다.

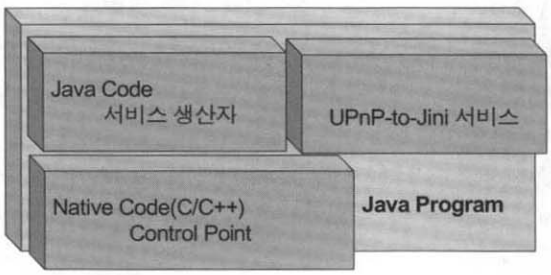


(그림 6) UPnP와 Jini의 연동 개념도

4.3 구조 및 세부 기능

(그림 7)은 UPnP-to-Jini 서비스의 기본구조를 나타내고 있다. UPnP-to-Jini 서비스는 크게 세 부분으로 구성된다. 전체 프로그램은 Jini를 지원하기 위하여 자바 응용프로그램으로 작성하고, 제어점부는 C/C++로 작성한다. 제어점부는 UPnP 디바이스를 검색하고 UPnP 디바이스로부터 제어정보를 얻는 부분이다. C/C++로 작성된 부분과 자바로 작성된 부분의 연결은 JNI(Java Native Interface)를 이용한다.

- 제어점부 : UPnP의 제어점과 같은 기능을 하는 부분이다. UPnP 디바이스가 제공하는 서비스에 대한 다양한 정보를 읽어 서비스 생산자부에 넘기는 역할을 담당한



(그림 7) UPnP-to-Jini 서비스의 구조

다. 이 때 UPnP-to-Jini 서비스의 생산을 위하여 UPnP 디바이스로부터 읽을 필수 정보는 다음과 같다

- UDN
- deviceType
- ServiceType
- ServiceId
- SCPDURL
- controlURL
- action name
- argument name
- argument relatedStateVariable
- serviceState name
- serviceState dataType

제어점부는 UPnP 네트워크에 참가한 UPnP 디바이스가 제공하는 디스크립션으로부터 (그림 7)에서 기술한 내용들을 읽은 후 서비스 생산자를 호출한다. 서비스 제어에 관련된 내용은 ActionList와 StateList에 저장되어 서비스 생산자가 Jini 서비스를 실행할 때 사용된다.

- 서비스 생산자부 : 제어점부를 통해 얻은 ActionList와 StateList를 이용하여 서비스 프록시를 생성하는 부분이다. UPnP-to-Jini 서비스를 위해 작성한 기본 Jini 서비스 템플릿 클래스를 이용하여 액션, 스테이트를 추가한다. 추가된 액션은 Jini 클라이언트가 UPnP-to-Jini 서비스를 통해 UPnP 디바이스에 제어 메시지를 전달할 수 있도록 작성되고, 스테이트는 Jini 클라이언트에 제공되는 사용자 인터페이스를 통해 UPnP 디바이스의 상태를 나타낸다. 생성된 UPnP-to-Jini가 제공하는 서비스 프록시는 RMI 통신을 지원하는 것으로 Jini 클라이언트 쪽으로 보낼 RMI_Stub를 포함하고 있다.
- UPnP-to-Jini 서비스부 : 서비스 생산자부에서 생성된 UPnP-to-Jini 서비스가 수행되는 부분으로서 Jini 서비스와 동일한 동작을 한다. UPnP-to-Jini 서비스는 서비스 프록시를 룩업 서비스에 등록한다. Jini 클라이언트로부터 받은 명령은 UPnP-to-Jini 서비스에 전달되고, 이것은 다시 제어점부로 전달되어 최종적으로 UPnP 디바이스에 전달된다.

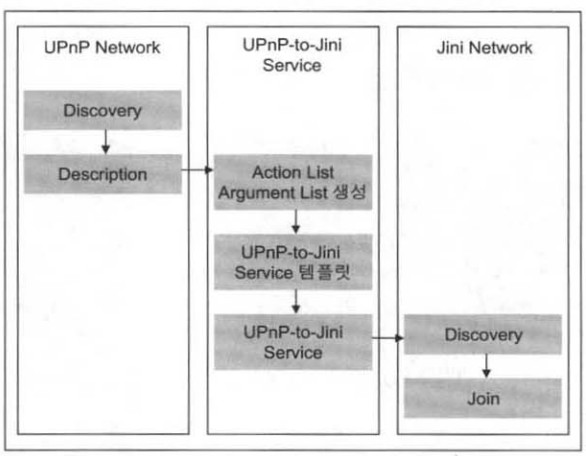
4.4 알고리즘

(그림 8)은 UPnP-to-Jini의 알고리즘으로서 UPnP가 초기화되는 과정으로부터 디바이스를 발견하고 Jini의 서비스 액션 과정을 거쳐 최종적으로 UPnP-to-Jini가 성립되는 것을 나타내고 있다.

```
UPnP Initialize ;
UPnP Event Handler Setting ;
while()
{
 if( Found Device? )
 {
  Device Description Download ;
  Parsing Device Description ;
  Get Action List ;
  Get Argument List ;
  Set Jini Service Action List ;
  Set Jini Service Argument List ;
  UPnP-to-Jini Service Start ;
 }
 if( Found Jini Control? )
 {
  UPnPSendAction ;
 }
}
```

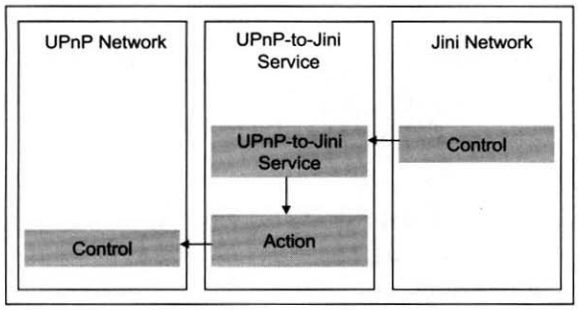
(그림 8) UPnP-to-Jini 알고리즘

(그림 9)는 UPnP 디바이스가 Jini 네트워크에 전달되는 과정을 나타내고 있다.



(그림 9) UPnP 디바이스가 Jini 네트워크에 전달되는 과정

(그림 10)은 Jini 클라이언트가 UPnP 디바이스를 사용하는 과정을 나타내고 있다.



(그림 10) Jini 클라이언트가 UPnP를 사용하는 과정

5. UPnP-to-Jini 서비스의 구현

5.1 구현 환경

본 논문에서 구현한 UPnP-to-Jini 서비스는 구현의 편의를 위하여 UPnP 디바이스가 제공하는 서비스 중 한 가지 액션에 대한 구현만을 하도록 구현상 제한사항을 두었다. 이러한 제한사항으로 인해 UPnP-to-Jini 서비스가 제공하는 기능은 매우 부족하다고 할 수 있으나, 제한적인 기능의 구현만으로도 Jini 클라이언트를 통해 UPnP 디바이스를 제어할 수 있음을 시연할 수 있다.

구현은 레드헷 리눅스 8.0 운영체제 환경에서 자바 및 C/C++ 컴파일러로 JDK 1.4.1_02와 gcc 3.2를 사용하였다. Jini API는 Sun Microsystem의 Jini SDK 1.2.1를 사용하였다. 제어점의 구현을 위하여 Linux SDK for UPnP Device 1.2.1을 사용하였다.

구현 및 시연을 위해 사용한 UPnP 디바이스는 Intel에서 제공하는 UPnP light 디바이스를 사용하였다. UPnP light 디바이스는 윈도우 환경에서 동작하는 응용프로그램으로 UPnP에서 정의한 디바이스와 동일한 동작을 하는 응용프로그램이다.

실행은 리눅스 또는 윈도우 운영체제 환경에서 이루어진다. UPnP-to-Jini 서비스는 리눅스 환경에서 동작하며, Httpd, Rmid, 룩업 서비스, Jini 클라이언트는 리눅스 또는 윈도우 환경에서 동작한다. 그리고 UPnP light 디바이스는 윈도우 응용프로그램이므로 윈도우 환경에서 동작한다.

5.2 구현 내용

구현 내용은 주 프로그램, 제어점부 및 UPnP-to-Jini 서비스부로 나누어 설명한다.

5.2.1 주 프로그램

UPnP-to-Jini 서비스를 수행하는 주 프로그램은 자바로 작성되었다. 주 프로그램은 JNI를 통해 연결한 StartUPnP 함수를 호출하여 제어점부를 실행하고, Jini 서비스로부터 오는 제어명령을 제어점부에 전달하기 위하여 SendAction 함수를 갖고 있다.

주 프로그램은 자바로 작성된 Jini 프로그램과 C/C++로 작성된 제어점 프로그램의 연결을 담당한다. 주 프로그램과 제어점과의 연결을 JNI 함수가 구현한다. 주 프로그램에서 호출된 JNI 메소드 내에서 다른 C/C++ 함수를 호출할 수 있다.

Java_UPnP2Jini_StartUPnP 함수를 통해 메인함수가 호출되어 제어점을 초기화하고 디스커버리가 가능하도록 한다. 또한 Jini 클라이언트쪽에서 전달된 제어동작에 대응하여 UPnP 액션을 호출하는 Java_UPnP2Jini_SendAction 함수를 정의하였다. Java_UPnP2 Jini_SendAction 함수는 매개변수를 입력 받아 SendAction 함수를 호출한다. 이를 통해 Jini 클라이언트의 명령이 UPnP-to-Jini 서비스를 거쳐 UPnP 디바이스에 전달된다.

5.2.2 제어점부

제어점부는 Linux SDK for UPnP Device 1.2.1에서 제공하는 UPnP와 XML 관련 함수들을 이용하며, C/C++ Shared Library로 작성하여 주 프로그램에서 JNI로 연결하여 실행할 수 있도록 하였다.

주 프로그램에서 UPnP 제어점을 시작하기 위하여 Main() 함수를 호출하면 Main() 함수는 UPnPStart 함수를 호출하여 UPnP 디스커버리 과정을 실행한다.

CtrlPointStart 함수 수행 중에 호출되는 UpnpRegisterClient를 통해 EventHandler가 등록된다. EventHandler 함수는 UPnP 수행과정에서 UPnP 디바이스로 송신되는 다양한 정보에 따라 발생되는 다양한 이벤트에 대응하여 수행되는 함수이다.

```
<? xml version = "1.0" encoding = "utf-8"?>
- <scpd xmlns = "urn:schemas-upnp-org:service-1-0" >
  + < specVersion >
  - < actionList >
    + < action >
    - < action >
        < name > SetTarget </ name >
        < argumentList >
          < name > newTarhetValue </ name >
          < direction > in </ direction >
          < relatddStateVariable > Target </ relatedStateVariable >
        </ argument >
      </ action >
  - < serviceStateTable >
    + < stateVariable sendEvents = "Yes" >
    + < stateVariable sendEvents = "no" >
    </ serviceStateTable >
  </ scpd >
```

(그림 11) UPnP light 디바이스 서비스 디스크립션

네트워크에서 디스커버리 메시지를 통하여 UPnP 디바이스의 참가 알림 메시지를 받게되면 EventHandler가 수행되고, ControlPointAddDevice, ControlPointAddService 및 ControlPointAddActionList를 호출하여 발견된 UPnP 디바이스에 대한 상세 디스크립션을 읽어 온다. 또한 Action-List를 검출하기 위한 CtrlPointAddActionList 함수의 일부를 나타내고 있다. CtrlPointAddActionList 함수는 (그림 11)과 같은 서비스 디스크립션 파일을 읽어서 UPnP 디바이스가 제공하는 서비스의 ActionList와 StateList정보를 저장한다. CtrlPointAddActionList에서 XML로 시작하는 함수들은 UPnP 디바이스로부터 얻은 XML 파일의 분석을 위하여 Linux SDK for UPnP Device 1.2.1에서 제공하는 함수들이다.

5.2.3 UPnP-to-Jini 서비스부

이는 Jini 서비스 API를 이용하여 작성된 자바 프로그램으로서 Jini 서비스와 동일한 기능을 갖고 있으며, Jini 서비스가 제공할 서비스 프록시의 내용을 나타내고 있다. 백엔드 프로토콜은 백엔드를 위한 RMI 자바 클래스 인터페이스이다. 백엔드는 컴파일할 때에 Backend_Skel.class, Backend_Stub.class의 두 가지 클래스 파일이 생성된다. 생성된 파일 중 Backend_Stub.class 파일은, Jini 클라이언트가 서비스 프록시를 사용할 때, UPnP-to-Jini 서비스가 제공하는 파일이다.

5.3 시 연

먼저 Jini 서비스를 위하여서는 Jini SDK 1.2.1과 Java Development Kit 1.4.1_02에 포함된 Httpd, Rmid 및 룩업 서비스를 동작시킨다. 또한 UPnP-to-Jini 서비스를 위한 Httpd도 동작시킨다. 동작명령은 (그림 12)와 같다.

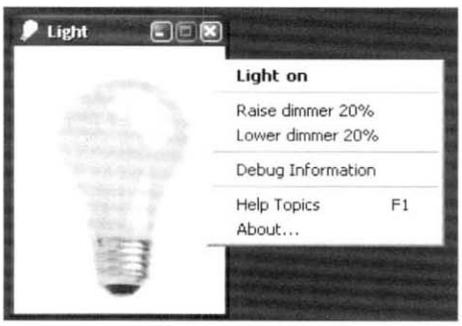
```
Httpd :
java -jar C:\Jini\lib\tools.jar -port 8080 -dir C:\Jini\lib -trees
Rmid :
rmid -J-Djava.security.policy=C:\Jini\policy
\policy.all
Lookup Sevice :
java -jar C:\Jini\lib\reggie.jar
http://192.168.0.2:8080/reggie-dl.jar
C:\Jini\example\ lookup\policy.all C:\Jini\lookup_log public
UPnP-to-Jini Service Httpd :
java -jar tools.jar -port 8085 -dir /home/root/Work/Jini/UPnP
-verbose
```

(그림 12) Httpd, Rmid 및 룩업 서비스의 구동

먼저 Httpd는 자바 파일을 포함한 룩업 서비스 또는 Jini 서비스의 요청에 따라 Jini 관련 파일을 제공한다. Rmid는 Jini 동작과정에서 사용되는 RMI를 위한 데몬 서버이다. 룩업 서비스는 Jini 서비스들을 등록하고 관리하는 서버이다.

다음으로는 UPnP light 디바이스를 수행한다. 윈도우 환경에서 수행한 UPnP light 디바이스의 동작화면은 (그림 13)과 같다. 그림의 오른편에 나타난 메뉴는 응용 프로그램에서 UPnP light를 제어할 수 있는 명령들이다. UPnP light 서비스는 프리젠테이션을 제외한 UPnP 디바이스가 갖고 있어야 할 모든 기능을 제공한다.

(그림 14)는 UPnP-to-Jini 서비스의 동작내용을 보이고 있다. 명령수행 후 나타나는 과정은 UPnP 제어점부에서 UPnP light 디바이스를 찾은 다음, UPnP-to-Jini 서비스의 수행과정을 나타낸 화면이다. UPnP 디바이스를 검색하여 UPnP 디바이스로부터 서비스, 액션네임 등의 필요정보를 읽어서 출력한 내용을 확인할 수 있고, 마지막으로 UPnP-to-Jini 서비스가 수행된 후에 룩업 서비스를 찾은 것을 확인할 수 있다.



(그림 13) UPnP light 디바이스

```
java-cp jini-core.jar:jini-ext.jar:sun-util.jar:./
-Djava.rmi.server.codebase=http://192.        168.0.122:8085      /
-Djava.library.path=./
-Djava.security.policy=policy.all UPnP2Jini
Initializing UPnP with IP Address = (null) Port = 0
UPnP Initialized (192.168.0.122:-16384)
Registering Control Point
Control Point Registred
.Found Device : [urn:schemas-upnp-org:device:
BinaryLight:1]
        Location : http://192.168.0.121:56463/
Found Service : urn:schemas-upnp-org:service:
SwitchPower:1
        ServiceId : urn:upnp-org:serviceId:
SwitchPower.0001
        SCPD URL : _SwitchPower.0001_scpd.xml
XML Read Success
Action Name : SetTarget
Argument Name : newTargetValue
Relate Variable : Target
State Variable Name : Target
Data Type : boolean
discovered a lookup service!
```

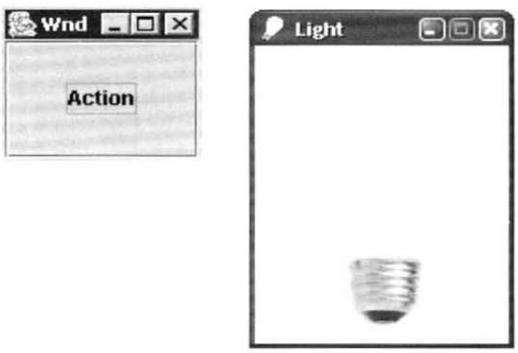
(그림 14) UPnP-to-Jini 서비스의 실행

(그림 15)는 Jini 클라이언트의 실행을 나타내며, UPnP light 서비스의 전등을 켜는 제어를 보인다.

```
java -cp jini-core.jar ; jini-ext.jar;sun-util.jar ; .\
  -Djava.security.policy = C:\Jini\policy\policy.all HelloWorldClient
Got a matching service.
Proxy calling backend…
```

(그림 15) Jini 클라이언트의 실행

(그림 15)의 실행 후에 UPnP-to-Jini 서비스로부터 받은 메시지가 출력된 것을 볼 수 있다. 즉 Jini 클라이언트를 수행하면 룩업 서비스로부터 UPnP-to-Jini 서비스를 다운받아 (그림 16)과 같은 창이 뜨고, 액션을 선택하면 UPnP-to-Jini 서비스에 명령이 전달되어 UPnP light 디바이스에 불이 켜짐을 확인할 수 있다.



(그림 16) Action의 실행화면

6. 결 론

이 장에서는 홈 네트워크 미들웨어 기술 중 UPnP 동작 과정의 특성을 이용하여 UPnP-to-Jini 서비스를 설계하고 구현하였다. 그리고 전등을 켜고 끄는 응용 예를 통하여 상호 연동됨을 보였다.

이 장에서 제시한 방법을 통해 홈 네트워크 환경에서 UPnP나 Jini의 활용도가 더 높아질 것으로 기대된다. 본 연구에서는 UPnP의 다양한 기능 중 UPnP-to-Jini 서비스를 설계하고 구현하였으나, 향후에 좀 더 복잡한 내용의 UPnP 디바이스를 지원할 수 있는 방안에 대한 연구가 필요하다고 본다.

참 고 문 헌

[1] 문경덕, 배유석, 김채규, "홈 네트워크 제어 미들웨어 개요 및 표준화 동향", 정보처리논문지, 제8권 제5호, pp.45-52. 2001.

[2] Amitava Duta Roy, "Networks for Homes," IEEE SPECTRUM, Dec., 1999.

[3] Brent A. Miller, Toby Nixon, Charlie Tai, Mark D. Wood, "Home Networking with Universal Plug and Play," IEEE Communication Magzine, Dec., 2001.

[4] 박동환, 구태연, 문경덕. "HAVi 홈 네트워과 연동을 지원하는 Jini 서비스의 구현", 정보과학회 2002년 추계학술대회, 제29권 제2호, 2002.

[5] 배대호, 박준호, 강순주, 최광호. "UPnP와 LonWorks의 상호연동을 위한 브리지 구조", 한국정보과학회 2003년 춘계학술대회, 제30권 제1호, 2003.

[6] T. R. Halfhill, "Sun's Jini : Science, Not Magic," Microprocessor report, pp.10-13, March, 1999.

[7] P. J. Gill, "Jini without the java : Microsoft faces off with sun with Universal Plug and Play : ERP vendors warm up to java," Components strategies, pp.17-18, Sep., 1999.

[8] G. Bhatti, Z. Sahinoglu, K.A. Peker, J. Guo, F. Matsubara, "A TV-Centeric Home Network to provide a Unified Access to UPnP and PLC Domains," Proceedings of th 2002 IEEE 4th International Workshop on Networked Applications, pp. 234-242, Jan., 2002.

[9] Dong-Sung, Kim, Jae Min Lee, Wook-Hyun Kwon, In Kwan Yuh, Gye Yeon Cho, "Design and Implementation of Home server System using UPnP middleware," Proceedings of the International Conference on Consumer Electronics, pp.106-108, June, 2002.

[10] B. A. Miller, T. Nixon, C. Tai, M. D. Wood, "Home networking with universal plug and play," IEEE Communication Magazine, Vol.39, No.12, pp.104-109, Dec., 2001.

[11] R. Gupta, D. P. Agrawal, S. Talwar, "Jini home networking : a step toward pervasive computing," IEEE Computer, Vol.35, No.8, pp.34-40, Aug., 2002.

[12] Jini, http://www.jini.org.

[13] Jini, http://www.sun.com.

[14] J. Waldo, "Alive and well : Jini technology today," IEEE Computer, Vol.33, No.6, pp.107-109, June, 2000.

[15] Jini Network Technology, http://java.sun.com.

한 상 숙

e-mail : sshan@tjpc.ac.kr

1987년 한밭대학교 전자계산학과(공학사)

1990년 청주대학교 전자계산학과(공학석사)

2004년 한남대학교 정보통신공학과(공학박사)

1995년~현재 대전기능대학 멀티미디어과 부교수

관심분야 : 실시간시스템, 멀티미디어처리

은 성 배

e-mail : sbeun@mail.hannam.ac.kr

1985년 서울대학교 컴퓨터공학과(공학사)

1987년 한국과학기술원 전산학과(공학석사)

1987년~1990년 한국전자통신연구소 TDX개발단 연구원

1990년~1995년 한국과학기술원 전산학과 (공학박사)

1995년~현재 한남대학교 정보통신공학과 교수

관심분야 : 실시간시스템, 멀티미디어 처리

김 철 민

e-mail : man0126@hotmail.com

2000년 한남대학교 정보통신공학과 (공학사)

2003년 한남대학교 정보통신공학과(공학석사)

2001년~2003년 (주)옥타컴 연구원

관심분야 : 실시간시스템, 멀티미디어 처리