OSGi와 Java RMI기반의 이기종 홈네트워크 미들웨어 간의 상호운용에 관한 연구*

한찬규○ 최형기
성균관대학교 정보통신공학부
hedwig@skku.edu○, hkchoi@ece.skku.ac.kr

The Enhanced Interoperability in Heterogeneous Home Networks using OSGi and Java RMI

Chan-kyu Han○ Hyoung-kee Choi
School of Information and Communication Engineering, Sungkyunkwan University

요 약

홈네트워크는 댁내의 가전기기들을 네트워크를 통해 공유하고 외부에서 제어가 가능하도록 발전해가고 있다. 이러한 홈네트워크를 사용하기 위해서는 기본적으로 미들웨어라는 운영체제 상단의 응용 기술이 필요하다. 현재까지 제안된 미들웨어들은 독립적이고, 단일의 표준안이 마련되고 있지 않아 상호운영에 큰 제약이 있었다. 본 논문에서는 게이트웨이 역할을 주로 수행하던 OSGi를 미들웨어 호환을 수행하도록 확장하고, Java의 RMI를 결합해서 홈네트워크를 구성하여 이기종 홈네트워크 간에서의 상호운용에 대한 발전방향을 제시한다. 제안기법은 홈네트워크 구성에 있어 기존의 호환기법보다 구현이 용이하다는 장점이 있다. RMI만을 통해 각 디바이스들을 제어할 수 있으므로 기존 기법에 비해 자원의 낭비가 감소한다. 또한 다른 미들웨어를 사용하는 가전기기의 경우에도 단일 네트워크로 결합할 수 있는 확장성을 보인다.

1. 서론

21세기 정보통신 패러다임은, 사용자가 장소에 상관없이 자유롭게 네트워크에 접속할 수 있는 정보통신 환경인 유비쿼터스(Ubiquitous)[14]로 진일보하고 있는 추세이다. 유비쿼터스의 일례로 가전, 통신 등의 통합 서비스를 창출하는 새로운 패러다임인 디지털 컨버전스가 조명 받고 있다. 이러한 일련의 디지털 컨버전스 네트워크 환경에서, 주목 받으며 등장한 것이 바로 홈네트워크(Home network)이다. 홈네트워크란 T.V., 냉장고, 에어컨 등 댁내의 가전제품을 네트워크로 연결하여 댁내부망을 구성하고, 인터넷을 통해 댁외부망과 연결한 시스템이다.

각 가전기기(디바이스)는 기업별, 기종별로 각기 다른 방식의 인터페이스 및 운영체제를 사용한다. 각기 다른 인터페이스를 사용하는 가전기기(디바이스)들이 홈네트워크를 구성하기 위해서는, 모든 디바이스 정보를 통합하여 관리하고 통일된 제어명령을 내리는 규약을 정의하는 것이 필수적이다. 이러한 규약으로써 미들웨어가 제안되었다. 미들웨어는 각 디바이스간의 상호발견, 구성 및 관리를 수행하여 댁내의 디바이스들을 단일의 네트워크로 결합하는 실질적인 역할을 수행한다.

현재까지 제안된 미들웨어로는, (1) Microsoft사의 Universal Plug and Play(UPnP)[2]와 (2) Sun사의 Java Intelligent Network Infrastructure(Jini)[3][11] 및 (3) 음향 및 영상 관련 장비 중심인 Home Audio Video Interoperability(HAVi)[4] 등이 존재한다. 동일한 미들웨어 내의 디바이스는 별도의 드라이버나 인터페이스의 까다로운 설정 없이 해당 미들웨어를 통해 제어할 수 있다. 그러나 서로 다른 방식의 미들웨어로 동작하는 디바이스들로서는 디바이스 간 규약이 다르기 때문에 단일의 홈네트워크를 구성하는 데 어려움이 존재한다. 서로 다른 미들웨어의 통제를 받는 디바이스 간에는 통신이 불가능하다. 따라서 이러한 개별 미들웨어간의 호환은 요구되는 연구개발 과정이라 할 수 있다.

홈네트워크 미들웨어 분야에서는 아직 단일의 표준안이 마련되어 있지 않다. 많은 기술들이 등장하고, 개별 기업간의 경쟁을 하고 있지만 단일 표준안이 채택되지 않고, 몇몇 미들웨어 간의 호환가능성만 제시되었다[1][2]. 미들웨어 표준화 작업은 1999년부터 본격화되기 시작하여, 각 미들웨어 제안인 UPnP, Jini, HAVi 등을 중심으로 컨소시엄을 구성하여 프로토콜 표준, 디바이스와의 인터페이스를 위한 논의가 진행 중이었다. 그러나 이러한 제안들은 다소 독립적이었기 때문에, 미들웨어간의 상호호환성을 보장하기 위해 표준 서비스 모델을 연구하는 단체로 Open Service Gateway Initiative(OSGi)[7]가 대두되었다. OSGi를 통하여 이기종 미들웨어 bridging 등 상호호환을 위한 연구 개발 및 특허가 진행되는 상황에서 현재 상호호환성(Interoperability)의 연구 및 개발은 상당히 미진한 상태이다[14]. 미들웨어간의 상호호환에 관련되어 현재까지 진행된 연구의 대부분은 UPnP를 Jini Service 형태로 운용하여 UPnP를 Jini 네트워크로 통합

* 본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터(ITRC) 지원사업의 연구결과로 수행되었음.

화하거나[1][2], UPnP와 Jini간의 Base Driver 등, 일종의 Bridge를 이용하는 형태였다[4][5]. 이와 같은 연구들은 제한된 수의 미들웨어 간 상호호환이 목적이기 때문에 확장성이 다소 결여되어 있다.

이에 본 논문에서는 OSGi의 응용된 형태로 미들웨어 간의 호환을 제안한다. OSGi는 미들웨어들을 통합화하는 역할을 주로 담당했다. 본 논문에서는 OSGi를 확장된 형식으로 운용하여 미들웨어 간의 통합을 꾀하므로, 이는 기존의 연구에 결여되어 있던 확장성을 증대시킨다. 또한 기존의 Base Driver보다 번들 Overhead가 적은 Java Remote Method Invocation(RMI)로 홈네트워크를 구성할 수 있음을 보여준다.

본 논문에서는 먼저 2장에서 홈네트워크 기반 지식과 미들웨어에 대한 관련 연구를 정리한다. 3장에서 본 논문에서 제안하는 시스템의 구조를 설명하고, 4장에서 제시된 구조의 실제 구현 및 결과물의 특성을 서술한다. 5장에서는 이 시스템의 정당성을 확장성 측면과 효율성 측면에서 다른 연구결과와 비교한다. 6장에서는 향후 과제를 언급하고, 본 논문의 결론을 맺는다.

2. 관련연구

홈네트워워크 미들웨어는 디바이스간의 통신규약을 통칭한다. 미들웨어에 관련된 연구로는 현재까지 대표적으로 두 가지 접근 방향이 있다. 첫째 미들웨어만을 단독으로 연구하는 분야에서는 UPnP와 Jini의 연구 및 개발이 진행되고 있다. 둘째 이기종 미들웨어 호환에 관한 연구는 대표적으로 Open Service Gateway Initiative(OSGi) 내에 Base Driver를 통한 호환과 Jini 미들웨어의 Jini Lookup Service(LUS)를 통한 미들웨어 간의 호환 연구 등이 진행되고 있다.

2.1 홈네트워크 미들웨어

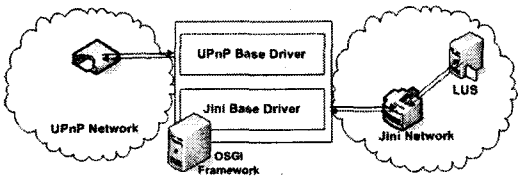
UPnP[2][9]는 TCP/IP 프로토콜을 이용하여 디바이스 간의 통신을 정의한다. 디바이스들은 멀티캐스트와 Extensible Markup Language(XML) 언어를 사용하여 통신한다. UPnP는 응용계층 프로토콜이므로 4계층 즉 전송계층 하단의 프로토콜에 관계 없이 구현이 용이하다.

Jini[3][9]는 Java 언어로 홈네트워크를 구성하기 위한 프레임워크이다. Jini는 디바이스의 발견 및 제어를 위해 서비스서버(Lookup Service:LUS)라는 시스템을 운용한다. LUS에서는 Jini 미들웨어의 모든 디바이스 목록이 등록되어 있다. Jini의 모든 디바이스는 LUS를 통해 서로를 발견하고 제어한다.

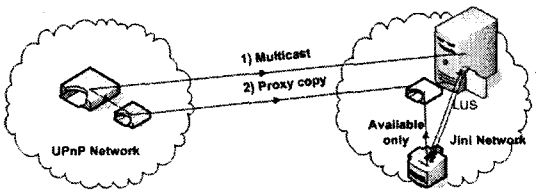
OSGi[7][9]는 홈네트워크 미들웨어 표준을 연구하고 호환을 위한 시스템을 제시하는 컨소시엄이다. OSGi 표준은 디바이스 간의 연결 및 제어를 담당하고 디바이스와 OSGi 프레임워크 간의 통신을 정의한다. OSGi가 제시하는 구조는 프레임워크, 번들, 서비스로 나뉘어진다. 서비스는 디바이스가 수행하는 기능을 의미하고, 프레임워크는 운영체제와 같은 역할을 수행한다. 번들은 프레임워크에 등록되어 서비스를 수행하는 Java Archive(JAR) 파일로써 운영체제의 어플리케이션과 같은 기능을 수행한다. 서비스는 번들의 형태로 제작되고 이러한 번들은 프레임워크 상에서 동작하여 디바이스의 기능 즉 서비스를 제공한다.

2.2 이기종 홈네트워크 미들웨어 간의 호환 기법

이기종 홈네트워크 미들웨어 호환에 관한 연구는 UPnP와 Jini 간의 호환 연구가 주로 진행되고 있다. 대표적으로 Base Driver를 통한 호환 기법과 Jini LUS를 통한 호환 기법에 대한 연구개발이 활발히 진행되고 있다.



(a) Interoperability using Base-Driver



(b) UPnP Client and Jini Service with LUS

(그림 1) 이기종 홈네트워크 미들웨어 간의 호환 기법

Base Driver를 통한 호환은 OSGi 프레임워크 상에 UPnP Base Driver와 Jini Base Driver를 설치하여 중계하는 기법이다[4][5]. 그림 1.a에서 보듯이 UPnP Base Driver는 OSGi를 통해서 오는 Jini의 메시지를 UPnP에 알맞게 변환하고, UPnP에서 나가는 메시지가 OSGi를 통할 수 있도록 변환한다. Jini Base Driver는 OSGi를 통해 오는 UPnP 메시지를 Jini에 맞게 변환하고, Jini의 메시지가 OSGi를 통하도록 변환한다. 그러나 Base Driver는 UPnP와 Jini 등의 한정된 미들웨어의 경우에만 연구되고 있어 다양한 미들웨어에 적용할 수 있는 확장성이 부족하다.

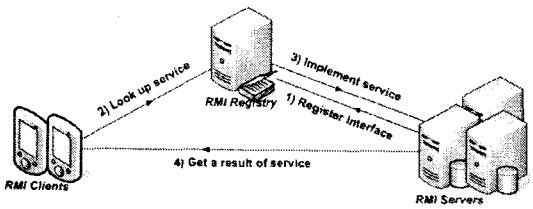
Jini의 LUS를 통해 이기종 미들웨어 호환을 도모하는 연구도 진행되고 있다[1][2][9]. 이 기법은 UPnP 디바이스를 LUS 관리 하에 두고 UPnP 디바이스가 Jini에 참여할 수 있도록 설정한다. 그림 1.b에서 보듯이 UPnP 디바이스는 멀티캐스트로 Jini LUS를 발견하고 자신의 프록시를 LUS에 복사하여 등록한다. Jini 미들웨어의 디바이스는 복사된 UPnP의 프록시를 통해 UPnP 서비스를 이용한다. 그러나 UPnP 디바이스가 LUS에 단방향으로 등록되어 있으므로 UPnP 디바이스는 Jini 디바이스를 인식할 수 없어 Jini 서비스를 이용할 수 없다. LUS를 이용한 호환기법은 양방향성이 제공되지 않는 단점이 있을 뿐만 아니라 UPnP 디바이스가 LUS를 발견하고 Jini에 참여하기 위한 부가적인 코드를 추가해줘야 한다는 비효율성이 존재한다.

3. RMI 기반의 호환 기법의 동작 과정

2장에서 이기종 홈네트워크 미들웨어 간 호환을 제공하는 기존의 기법들을 소개하였다. 기존의 기법들은 확장성이 부족하거나 구현이 용이하지 않다는 단점이 있었다. 이에 본 논문에서는 효과적으로 이기종 홈네트워크 미들웨어 간의 호환을 제공하고자 한다. Java Remote Method Invocation(RMI)[13]를 사용하여 홈네트워크 미들웨어를 구성하여 확장성을 높이고 보다 양질의 서비스를 제공할 수 있다. 본 장에서는 Java RMI의 구조와 통신 방식 그리고 RMI 기반의 이기종 홈네트워크 미들웨어 호환 기법에 관해 설명한다.

3.1 Java RMI

프로토콜의 상세가 복잡해질수록 소켓 프로그래밍의 구현 난이도가 높아지고 이에 따라 통신상의 에러가 발생할 확률도 높아졌다. 프로토콜 상세를 지켜 구현하는 것은 난해하고 통신상의 에러가 빈번하게 발생한다는 문제점을 해결하기 위해, 로컬함수를 호출하는 것과 같은 방식으로 원격함수를 호출하는 방식인 Remote Procedure Call(RPC)의 개념이 도입되었다. RPC란 네트워크 프로토콜 상세에 영향받지 않고 원격함수를 호출하는 방식으로 RPC기법을 사용하면 통신상의 에러율이 감소하고, 네트워크 프로그램 구현이 보다 용이해진다. 본 논문에서는 RPC 중 하나인 Java RMI를 이용하여 홈네트워크 디바이스 간의 통신을 정의하여 미들웨어를 구성한다.



(그림 2) RMI의 구조와 동작방식

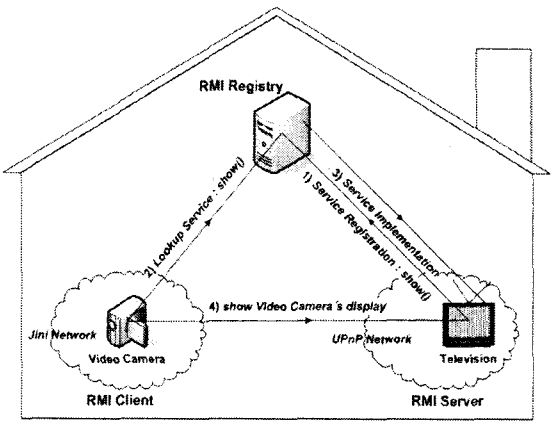
그림 2에서 보듯이 RMI는 다수의 RMI 서버들과 RMI 클라이언트들 그리고 RMI Registry로 이루어진다. RMI 서버는 서비스를 제공하는 함수의 모음이고, RMI 클라이언트는 RMI서버에 서비스를 요구한다. RMI Registry는 RMI 서버의 모든 서비스 목록을 보유하고 RMI 서버와 RMI 클라이언트 간의 통신을 중재한다. RMI 서버와 RMI 클라이언트는 원격함수 호출을 위해 서비스 즉 함수의 인터페이스를 미리 나눠가진다. 인터페이스란 RMI 클라이언트와 RMI 서버간의 주고받을 서비스, 즉 함수의 정의만을 담은 것으로 원격함수호출시 인터페이스에 맞춰 통신한다.

그림2는 RMI의 동작방식을 보여준다. 1) RMI서버는 자신이 제공할 수 있는 서비스들의 인터페이스를 RMI Registry에 등록한다. 2) RMI 클라이언트는 RMI Registry를 검색하여 원하는 서비스를 찾는다. 3) 서비스 검색이 끝나면 RMI 서버 측에서 함수가 실행되고, 4) RMI 서버는 서비스 수행의 결과를 RMI 클라이언트에 보고한다.

3.2 이기종 홈네트워크 미들웨어 호환에 관한 제안

본 논문은 RMI 통신을 응용하여 홈네트워크 미들웨어 구성을 제안한다. 각 디바이스가 채택하고 있는 미들웨어의 종류에 관계없이 홈네트워크를 구성할 수 있음을 보인다.

본 논문에서 제안하는 RMI 기반의 호환기법에서 홈네트워크 디바이스는 RMI서버와 RMI클라이언트가 된다. 서비스를 제공하는 홈네트워크 디바이스들은 RMI 서버가 되고, 서비스를 요구하는 홈네트워크 디바이스들은 RMI 클라이언트가 된다. RMI 서버가 되기 위해서는 자신의 서비스를 인터페이스로 만들어 두고 이를 RMI Registry로 등록하는 과정이 필수적이다. RMI 클라이언트가 되기 위해서는 RMI 서버의 인터페이스와 RMI Registry에서 서비스를 검색하는 과정이 필수적이다.



(그림 3) RMI 기반의 이기종 미들웨어 호환 기법 제안

RMI 기반의 이기종 미들웨어 호환 기법의 설명을 위해 홈네트워크 디바이스로 텔레비전과 비디오카메라를 가정한다. 비디오카메라는 텔레비전의 show(화면송출) 서비스를 사용해 비디오카메라의 입력을 텔레비전에 출력하는 일련의 과정을 예로 든다.

그림 3에서 보듯이 1) 텔레비전은 자신의 show 서비스의 인터페이스를 RMI Registry에 등록하고, 2) 비디오카메라는 RMI Registry에서 텔레비전의 show 서비스의 인터페이스를 찾는다. 3) 비디오카메라의 검색요청을 받으면 텔레비전은 show 서비스를 실제 수행하게 하고, 4) 비디오 카메라는 텔레비전의 show 서비스 수행의 결과값을 받는다. 즉 비디오 카메라가 찍은 화면이 텔레비전에 송출된다.

그림 3의 예에서 텔레비전은 show 서비스를 수행하므로 RMI 서버가 되고, 비디오 카메라는 텔레비전에 show 서비스를 요청하므로 RMI 클라이언트가 된다. 홈네트워크 디바이스는 서비스요청과 서비스수행의 관계가 다중적이

므로 RMI서버와 RMI클라이언트의 여부는 역동적으로 결정된다.

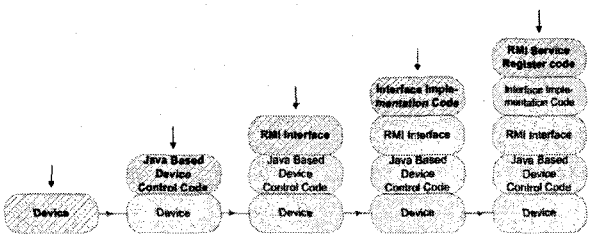
위와 같은 RMI 통신을 통하여 기존의 이기종 미들웨어 호환 기법보다 보다 효율적으로 이기종 홈네트워크 미들웨어를 호환할 수 있다. 기존에 Base Driver를 이용한 호환기법은 UPnP-Jini 외의 미들웨어로의 확장성이 부족하며, Jini LUS를 이용한 호환기법은 UPnP 디바이스에 덧붙이는 부가적인 구현의 난이도가 높아 비효율적이다. 이에 반해 본 논문에서 제안하는 RMI 기반의 이기종 미들웨어 호환 기법은 보다 손쉽게 확장성 있는 홈네트워크 미들웨어를 구성할 수 있다.

4. RMI 기반의 호환 기법 구현

3장에서 설명한 RMI 기반의 호환 기법으로 홈네트워크 디바이스를 구현하여 간단한 홈네트워크 미들웨어를 구성한다. 이를 위해 하드웨어인 웹캠과 소프트웨어인 미디어플레이어를 이용하였다. 본 논문에서는 RMI 및 OSGi등의 시스템을 이용하기 위해 Java 언어를 사용한다. 또한 웹캠과 미디어플레이어 등 모두 미디어와 관련된 처리를 요구하므로 미디어 제어 관련 Application Programming Interface(API)인 Java Media Framework(JMF)[12]를 이용하여 구현한다.

구현은 두 가지 단계로 분류한다. 첫째 홈네트워크 디바이스는 RMI 통신에서의 서버 역할로 해당 디바이스에 Java RMI 인터페이스와 실행코드를 구현한다. 둘째 RMI 클라이언트는 OSGi의 번들로 제작한다. 제어용 번들로는 OSGi 프레임워크 위에 RMI 클라이언트를 구현한다. 그 후 액티베이터[7]을 구현하고 제어용 번들에 서블릿 및 HTTP 서비스 번들을 추가적으로 구성한다.

4.1 홈네트워크 디바이스의 RMI 서버 구현



(그림 4) 홈네트워크 디바이스의 RMII 서버 구현

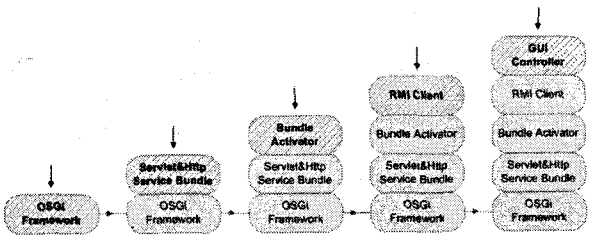
홈네트워크 디바이스의 RMI 서버 구현은 그림 4에서 보듯이 다섯 단계로 이루어진다. 1) Java 기반의 디바이스나 하드웨어가 필요하다. 2) JMF등을 이용하여 각 디바이스를 제어하는데 필요한 메소드를 구현한다3) java.rmi.Remote 패키지를 이용하여 Java RMI를 이용할 수 있도록 디바이스에 제어용번들의 인터페이스와 동일한 인터페이스를 추가한다. 4) 그 후에 UnicastRemoteObject 클래스를 상속받아 RMI 인터페이스를 동작하도록 하는 실제 구현코드를 추가한다. 5) 마지막으로 RMI Registry에 등록하는 Naming 클래스의 bind() 메소드를 사용하여 해당 홈네트워크 디바이스의 서비스를 등록한다.

(표 1) 미디어플레이어의 RMI 서버 구현

클래스	역할
MediaPlayerInterface	RMI 서버와 RMI 클라이언트 간의 공통된 인터페이스이다.
MediaPlayerImpl	*MediaPlayerInterface*에서 정의한 인터페이스를 실제로 수행한다.

위와 같은 구현 절차를 따라 미디어플레이어와 웹캠을 구현하였다. 미디어플레이어는 MediaPlayerInterface 클래스, MediaPlayerImpl 클래스로 구현하였고, 지면관계상 각 클래스의 상세코드는 생략하고, 역할만을 표 1과 같이 간략히 소개한다. 구현 1),2),3)단계는 MediaPlayerInterface 클래스를 참조하고, 구현 4),5)단계는 MediaPlayerImpl 클래스 구현을 참조한다.

4.2 OSGi 번들의 RMI 클라이언트 구현



(그림 5) OSGi 번들의 RMI 클라이언트 구현

홈네트워크 디바이스를 제어하는 OSGi 프레임워크 내에 번들 즉 RMI 클라이언트의 구현은 그림 5에서처럼 다섯 단계로 분류한다. 1) 구동할 OSGi 프레임워크는 필수적이다. 본 논문에서는 오픈소스로 운영되고 있는 Oscar OSGi 프레임워크를 사용한다. 2) 웹에서 사용할 수 있도록 서블릿과 HTTP 서비스에 관련된 코드를 구현한다. 3) 관련 서비스가 OSGi 프레임워크에서 활성화될 수 있도록 액티베이터[7] 코드를 구현한다. 액티베이터는 OSGi 번들 특유의 특성으로 서비스를 이용하고자 하는 사용자가 있을 때 해당 서비스를 가동시켜 주는 트리거 역할을 수행한다. 4) 디바이스는 홈네트워크 RMI Registry에 등록되어 있음을 가정하므로, 공통된 인터페이스를 통해 클라이언트가 서비스 받고자 하는 해당 디바이스를 Naming 클래스를 이용해 검색할 수 있도록 구현한다. 5) 마지막으로 웹 사용자를 위한 Graphic User Interface(GUI)를 추가한다.

(표 2) OSGi 번들의 RMI 클라이언트 구현

클래스	역할
MediaPlayerClient	RMI Registry에서 원하는 서비스를 검색한다.

클래스	역할
RMIServlet	HTTP Service 관련 구현으로 GUI를 위해 서블릿을 사용한다.
HttpActivator	사용자가 OSGi 프레임워크를 통하여 서비스에 접근하였을 때 번들을 활성화한다.

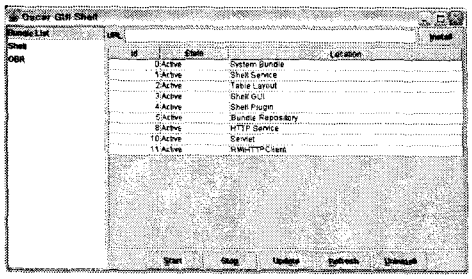
표 2는 위와 같은 과정의 구현코드를 미디어플레이어의 경우 간략하게 요약한 것이다. 세 개의 클래스로 이루어져 있는데, MediaPlayerClient 클래스는 구현 4) 단계와 RMIServlet 클래스는 구현 2),5)단계와 상응한다. HttpActivator는 구현 3)단계를 참조한다.

4.3 RMI 기반의 호환 기법 구현 결과

그림 6은 4장에서 구현한 RMI 클라이언트 번들을 Oscar OSGi 프레임워크에 올려 활성화 시킨 실행 예이다. RMIHTTPClient 번들은 댁내 디바이스 즉 RMI 서버의 서비스를 사용하는 RMI 클라이언트가 된다. 또한 사용자는 RMIHTTPClient라는 번들로 그림 7처럼 외부에서 댁내 디바이스를 제어할 수 있다.

제안한 시스템은 디바이스를 RMI Registry에 등록할 수 있도록 구현한 후에 OSGi에 제어 번들을 서블릿을 추가하여 구현한다. OSGi가 설치된 시스템에서, 디바이스 제어가 가능하며 네트워크가 가능한 웹브라우저가 장착된 모바일 디바이스에서도 제어할 수 있게 되는 것이다.

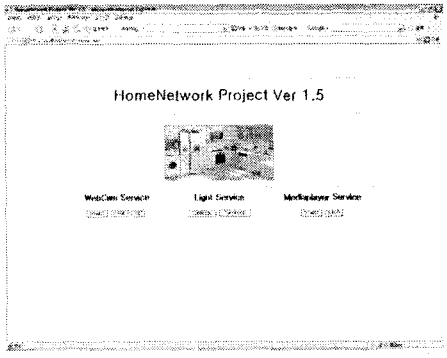
하나의 번들로 외부와의 연결 및 댁내에서의 제어가 가능하기 때문에 기존의 시스템보다 자원 낭비가 적고 다른 홈네트워크 미들웨어와의 통신이 필요한 경우 OSGi 상에서 클라이언트 번들과의 통신으로 쉽게 구현이 가능하다는 장점이 있다.



(그림 6) Oscar OSGi 번들 창

5. 비교 및 평가

본 장에서는 이기종 미들웨어 간의 호환을 제공하기 위한 기존의 연구기법들과 본 논문에서 제시한 시나리오를 번들의 구성측면, 이기종 미들웨어로의 확장측면에서 심도 있게 분석한다. 2장에서 설명하였듯이 이기종 미들웨어 간에 호환을 제공하기 위한 기존의 연구기법은 크게 두 가지가 존재한다.



(그림 7) 웹을 통한 OSGi 접근

첫째로 Jini의 LUS를 수정하여 다른 미들웨어의 디바이스들이 Jini 네트워크에 참여하도록 하는 기법은 Jini로의 단방향 서비스만 가능하다는 단점이 있다[1][2]. 또한 Jini와 UPnP같이 특정한 미들웨어들만의 연동만 가능하므로 확장성이 다소 부족하다.

두 번째는 OSGi 내에서 미들웨어에 대한 Base Driver를 구현함으로써 번들간의 상호작용을 통해 이기종 미들웨어 간의 호환을 도모하는 기법이다[4][5]. 이 기법의 경우 OSGi에 해당 미들웨어의 Base Driver를 운용하기 위해 해당 미들웨어를 제어하기 위한 제어번들을 추가적으로 운용하여야 한다. 댁내에서 OSGi에서 운용되는 수개의 번들을 제어하는 것은 사용자 입장에서 편의성을 제공한다는 원칙에 어긋난다. OSGi R3에서는 Jini Base Driver가 위에서 언급한 단점들 때문에 제약이 많다고 판단되어 요구사항명세에서 누락되기도 하였다.

본 논문에서는 비교를 위해 Base Driver를 이용한 홈네트워크 미들웨어 구성 또한 추가적으로 시도하였다. Jini의 경우에는, 당초 독일 회사인 ProSyst 에서 제공하는 Jini 드라이버 명세서를 통하여 제작하려 하였으나 명세서의 설명이 불충분하고, 또한 ProSyst 당사의 상용 번들을 이용해야 한다는 접근장벽이 있었다. 이에 Knopflerfish Open OSGi 에서 제공하는 Jini Driver version 0.1.0[10]] 을 사용하기로 하였다. 약 두 달간, Jini Driver를 적용하는 과정에서 많은 시행착오를 겪었다. 대표적인 것으로는 첫째, 정책문제(Policy)로 IO Permission, Runtime Permission, Network Permission 등 원 제작자의 시스템 환경에 최적화된 드라이버를 수정하는 일이 생각보다 쉽지 않았다. 둘째, 의존문제(Dependency)로 Jini 드라이버 번들을 돌리기 위해 Servlet, OSGi Service, Device Manager, httpBundle 등 다수의 Bundle repository가 필요하다는 단점이 있었다.

미들웨어 호환 연구 분야에 있어 최근 Per-service Implementation Effort를 줄이고자 하는 노력이 행해지고 있다[3]. 이와 같은 추세에, 번들의 오버헤드를 줄이고, 보다 다양한 미들웨어에 적용할 수 있도록 포괄적인 시스템을 제안하여 확장성을 높이는 본 논문의 연구목표는 적절한 것이라 할 수 있다.

이기종 미들웨어를 호환함에 있어 Base Driver를 이용하는 방법으로는 대표적으로 UPnP-Jini 간의 호환을 예로 들었다. 본 논문에서 제시한 시나리오와 Base Driver를 이용할 때의 OSGi 환경의 프레임워크 번들 Overhead를 비교해보면 표 3과 같다. OSGi 프레임워크에 공통적으로 탑재되어 구동하고 있는 번들은 System Bundle로써 OSGi의 기본적인 프로그램 구동을 위함이다. Base Driver를 이용하는 방법은 기본적으로 OSGi를 호환시스템과 게이트웨이로써 구동하기 위해 OSGi Service가 필요하다. 또한, UPnP 디바이스를 OSGi를 위해 발견하는 Generic Control Point 등이 필요하다. 또한 양 미들웨어 간의 Base Driver인 UPnP Base Driver, Jini Base Driver 각각 필요하다.

(표 3) 제안시스템과 Base Driver를 이용한 시스템의 비교

Evaluation	OSGi Bundle
Base Driver	System Bundle, Bundle Repository, Service Lookup, OSGi Service, Generic Control Point, UPnP Base Driver, Jini Base Driver
RMI-based Proposed System	System Bundle, Bundle Repository, Servlet, RMIHTTPClient(*)

본 논문에서 제안한 기법으로는, 4장에서 구현한 RMIHTTPClient 라는 번들 하나만 구동하면 독자적인 홈네트워크를 구성할 수 있다. 이 번들은 Base Driver를 구동하는데 필요한 모든 자바소스코드를 단일의 JAR 파일로 묶어 번들로 만든 것이 아니라, Base Driver 및 Gateway OSGi를 위한 OSGi Service 등 불필요한 번들은 모두 제거하여 RMIHTTPClient 라는 번들로 제작한 것이다. 적절한 라이프 사이클과 연계성을 지니는 번들을 최소한의 자원으로 만들어 내는 것은 OSGi 컨소시엄의 주요 관심사이다[3].

6. 결론 및 향후과제

홈네트워크 미들웨어는 각 디바이스간의 상호발견, 구성 및 관리를 수행하여 댁내의 가전기기들을 제어하는 실질적인 역할을 수행한다. 그러나 서로 다른 미들웨어로 동작하는 디바이스로는 통신이 어려워 홈네트워크를 구성하는데 어려움이 따른다. 이에 각 미들웨어 간의 호환을 제공하는 연구개발이 필수적이다.

본 논문에서는 기존의 Base Driver, Jini LUS등을 이용한 미들웨어 호환기법보다 구현이 용이하고 보다 확장성 있는 시스템을 제안한다. 상위레벨의 네트워크 통신방식인 RMI와 OSGi를 미들웨어로 응용, 확장한 방식으로 홈네트워크 구성을 제안하고 이를 시나리오 기반으로 구현하였다. 구현은 RMI 서버 역할을 수행하는 홈네트워크 디바이스에 RMI 통신코드를 추가하고, OSGi를 통해 댁내 디바이스를 이용할 RMI 클라이언트로 분류하여 작업하였다. 구현 결과, 웹캠과 미디어플레이어는 RMI서버 역할을 수행하여 RMI Registry를 통해 자신의 서비스를 등록한다. 웹 브라우저 유저는 서블릿을 통해 웹으로 접근한 뒤 OSGi에 등록된 RMI 클라이언트 번들을 통해 RMI 서버 즉 댁내 디바이스의 서비스를 이용할 수 있다.

본 논문은 OSGi의 역할을 게이트웨이에서 하나의 단일화 된 플랫폼으로 확장한 것으로 현재 OSGi 컨소시엄의 연구개발방향[7][9]의 흐름을 반영하였다. 본 논문에서 제시한 홈네트워크 구성방식은 RMi와 같은 유저레벨의 통신방식을 사용하고 별도의 게이트웨이가 필요 없어 리소스를 감쇄시킬 수 있다. 또한 사용자 측면에서 효율적으로 OSGi 프레임워크를 운용할 수 있고, 보다 다양한 미들웨어에 적용할 수 있도록 확장성을 강화시켰다.

향후에는 Extensible Markup Language(XML), C++ 등 다른 언어로의 확장문제를 해결하고 또한 Java 기반의 UPnP로의 확장 아키텍처에 관한 연구와 좀 더 다양한 디바이스 및 시나리오의 넓은 범주로의 확장 연구를 수행할 계획이다.

참고문헌

[1] Jan Newmarch, School of Network Computing Monash University "UPnP Service and Jini Client", ISNG 2005

[2] Jan Newmarch, School of Network Computing Monash University "A Custom Lookup Service for UPnP Services and Jini Clients", 2005

[3] J. Allard, V. Chinta, S. Gundala, G. G. Richard III "Jini Meets UPnP: An Architecture for Jini/UPnP Interoperability", Proceedings of the 2003 International Symposium on Applications and the Internet (SAINT 2003)

[4] Pavlin Dorbrev, David Famolari, Christian Kurzke, Brent A. Miller "Device and Service Discovery in Home Networks with OSGi", *IEEE Communications Magazine, August 2002*

[5] Haelyong Song, Jitae Shin, Hyoung-Kee Choi "Implementation and interoperability between Jini Middleware and OSGi in Remote-Controlled Home Networks," KICS Conference, November, 2005

[6] Ken Arnold, "The Jini Architecture: Dynamic Services in a Flexible Network", DAC 1999

[7] Sven Haiges, "OSGi Tutorial", October 2004

[8] Yu Jia, Jan Newmarch and Michael Geissler, "JINI/J2EE Bridge for Large-scale IP Phone Services", 10th Asia-Pacific Software Engineering Conference (APSEC 2003)

[9] Darragh O' Sullivan, BSc., "An Advanced Appliance Interaction Architecture", Master of Science in Computer Science, University of Dublin, Trinity College, September 2005

[10] Nico Goeminne, Jini Driver available at http://www.knopflerfish.org/repo/

[11] Jini Technology 1.0 API Documentation available at http://www-rohan.sdsu.edu/doc/jini/doc/api/index.html

[12] JMF 2.0 API Documentation available at http://java.sun.com/products/java-media/jmf/2.1.1/apidocs/

[13] Java RMI Tutorial, available at http://www.ccs.neu.edu/home/kenb/com3337/rmi_tut.html

[14] 홈네트워크 기술 및 표준화 특허 동향, 정보통신 기술, 정책 및 산업 주간기술동향 통권 1215호 available at http://kidbs.itfind.or.kr/WZIN/jugidong/1215/121506.htm