Non-IP 장치 제어를 위한 새로운 UPnP 브리지 구조

A Novel Architecture of UPnP Bridge for Non-IP devices

강정석*, 최용순*, 김성훈*, 이광국*, 박홍성**
(JeongSeok-Kang, YongSoon-Choi, SeongHoon-Kim, KwangKoog-Lee, HongSeong-Park)

Abstract - **This paper presents an architecture of UPnP Bridge that allows controlling Non-iP devices from UPnP control point, without modification to Non-IP device or UPnP control point implementations. UPnP devices must provide SSDP discovery, SOAP control and GENA event processes. To represent Non-IP devices to UPnP devices, UPnP Bridge provides these functionalities on behalf of Non-IP devices. We provides two method to interoperability between UPnP and Non-IP devices, Message Field Description, Non-UPnP Proxy devices. And solution to integrate heterogeneous networking standards(RS232C, CAN, IEEE1394, USB) is provided.**

Key Words : UPnP Bridge, MFD, Message Field Description, Non-UPnP Proxy Device, UPnP

1. 개 요

UPnP[1]는 TCP/IP 기반 자신의 독자적인 기술을 사용하여 개발됨으로써 다른 미들웨어와 상호 연동이 힘들어 모든 분산 시스템 환경에 적용하기에 제한이 있다. 또한 모든 임베디드 컴퓨팅 장치는 자원이나 비용 등에 문제로 인해 IP 프로토콜을 요구하지 않는다. 이러한 IP 기반이 아닌 장치와 통신을 위해서는 브리지가 필요하다. [6][7]

현재 UPnP 장치와 IP 기반이 아닌 장치간의 상호 운용에 대한 연구가 활발히 진행 중에 있다.

- Jini Meets UPnP: An Architecture for Jini/UPnP Interoperability[2]
- Controlling Non IP Bluetooth Devices in UPnP Home Network[3]
- IEEE1394/UPnP Software Bridge[4]
- Integrate LonWorks and UPnP devices by OSGi gateway and Universal Remote Console[5]

하지만 위의 연구는 서로 다른 미들웨어(Corba, Jini, LonWork 등)간 상호연동에 관한 것이다. 어떠한 미들웨어 없이 개발된 IP 기반이 아닌 장치에 대해서는 상호연동을 제공하지 못한다. 또한 서로 다른 네트워크 간 모든 서비스를 상호연동하지 못하므로 특정 장치에 대해 정적으로 대리자 역할을 하는 객체를 코딩을 통해 삽입하여야 한다. [4]는 IEEE1394 네트워크 인터페이스를 가진 장치와 UPnP의 상호연동을 위한 브리지이다. 하지만 다양한 네트워크 인터페이스를 지원하지 못하고 상호 연동을 위한 서비스를 동적으로 추가 시키지 못한다.

본 논문에서는 UPnP와 IP 기반이 아닌 장치간 상호연동을 위해 고려해야 할 사항에 대해 알아보고 이를 다루기 위한 새로운 UPnP 브리지를 제안한다. 제안하는 브리지 구조는 UPnP와 다양한 네트워크(RS232C, CAN[8], IEEE1394[9], USB[10])와 연결된 IP 기반이 아닌 장치간 상호연동을 위해 두 가지 방법을 제안한다. 첫 번째는 사용자가 코딩을 하지 않고도 자신의 장치를 동적으로 UPnP 네트워크상에 연결할 수 있도록 XML 타입의 Message Field Description(이하 MFD)을 정의하여 자신의 메시지 포맷을 표현하는 것이다. 두 번째는 일반적인 브리지 기술로 사용자가 자신의 장치를 제어할 수 있는 객체를 코딩을 통해 브리지 내에 구현하는 것이다.

본 논문은 2장에서 UPnP 브리지 구조에 대해 설명하고 3장에서 실제 구현 예를 통해 제안한 브리지의 성능을 검증한다. 마지막으로 4장에서 결론을 맺도록 한다.

2. UPnP 브리지 구조

제안하는 브리지는 그림 1에서 보는 바와 같이 UPnP 네트워크(TCP/IP기반)와 TCP/IP 기반이 아닌 다른 네트워크(RS232C, CAN, USB, IEEE1394)상의 물리적/기술적 차이를 극복하기 위하여 상호 통신한다. IP 기반이 아닌 장치가 연결 시 장치 타입과 매핑되는 가상의 UPnP 장치가 생성된다. 가상의 UPnP 장치는 UPnP 미들웨어를 통해 UPnP 네트워크와 통신을 함으로써 IP 기반이 아닌 장치 대신 UPnP 장치의 기본적인 기능(존재 공지, 제어, 이벤트)을 수행한다. 가상의 UPnP 장치는 실제 IP 기반이 아닌 장치와 통신을 위한 두 가지 방법이 있다. 첫 번째는 XML 타입의 MFD을 이용하여 UPnP 메시지를 IP 기반이 아닌 장치 메시지로 변환하는

저자 소개
* 강원대학교 전자통신 공학과({sleeper82, libris, bs9901, 21thbomb}@control.kangwon.ac.kr
** 강원대학교 전기전자공학부(hspark@kangwon.ac.kr)

MFD 메시지 변환기를 이용하는 방법과 두 번째는 실제 IP 기반이 아닌 장치와 직접 통신을 하는 Non-UPnP Proxy Device 객체를 이용하는 방법이다.

2.1 UPnP 브리지 내의 각 모듈

2.1.1 Virtual UPnP Device

Non-UPnP장치를 대신하여 UPnP 네트워크와 통신하면서 UPnP 장치의 기본적인 기능(존재공지, 제어, 이벤트)을 담당하고 내부에 UPnP 서비스 모듈을 가지고 있다.

2.1.2 Non-UPnP Proxy Device

가상의 UPnP 장치 객체로부터 UPnP 메시지를 수신하여 그에 적합하게 실제 Non-UPnP 장치를 제어하는 객체이다. 사용자는 Non-UPnP Proxy Device 객체를 개발 시 메시지 변환 및 제어 부분을 구현한다.

2.1.3 Virtual UPnP Device Manager

가상의 UPnP 장치 객체를 관리한다. Plug and Play 기능을 가진 네트워크 인터페이스의 버스 이벤트를 처리하고 Plug and Play 기능을 가지고 있지 않은 네트워크 인터페이스에 대해서는 주기적으로 "Keep Alive" 메시지를 전송함으로써 Non-UPnP 장치의 연결 여부를 확인하여 Non-UPnP 장치의 타입과 동일한 가상의 UPnP 디바이스 객체를 생성/소멸한다.

2.1.4 Network Interface Abstraction Layer

아래와 같은 세 모듈을 통해 다양한 네트워크 인터페이스를 채널로 추상화한 계층으로서 상위의 Non-UPnP Proxy Device 객체를 개발하는 사용자에게 편의성을 제공한다.

- IO Device Manager : 각 네트워크 인터페이스를 IO Device로 보고 IO Device의 동작(동기, 비동기)에 대한 관리를 한다.
- IO Channel Table : 유일한 채널 식별자와 채널 객체를 저장하고 있는 테이블이며 각 채널 객체는 실제 디바이스의 주소를 가지고 있다.
- IO Channel Scheduler : 각 채널 별로 우선순위에 따라 메시지를 전송하는 모듈로서 위급한 메시지의 경우 먼저 메시지 전송이 가능하다.

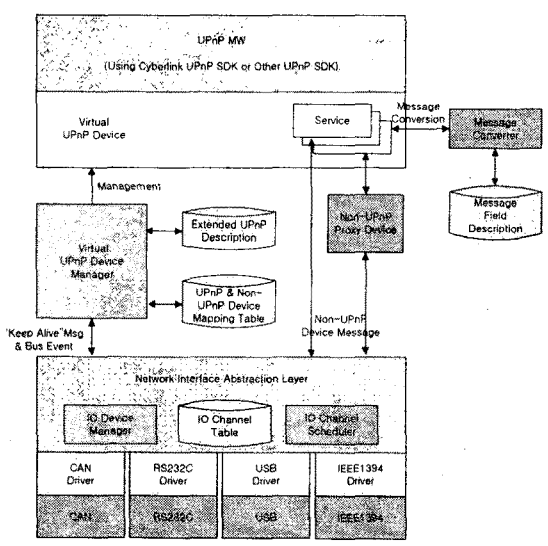

그림 1. UPnP 브리지 구조

2.2 UPnP와 Non-UPnP 장치간 상호연동

Non-UPnP 장치 개발자는 자신의 장치를 UPnP 네트워크와 연결하기 위해 장치에 대한 정보를 표준 UPnP 장치 기술과 브리지에서 정의한 확장된 UPnP 서비스 기술로 표현하여 브리지에 등록한다.

```
<actionList>
  <action ActCmd="Non UPnP Action Command">
    <name>Action Name</name>
    <argumentList>
    </argumentList>
  </action>
</actionList>
<serviceStateTable>
  <stateVariable sendEvents="yes" SVarID="Non UPnP Variable ID">
    <name>Status</name>
    <dataType>i4</dataType>
  </stateVariable>
</serviceStateTable>
```

그림. 2. 확장된 UPnP 서비스 기술

그림 2에서 보는 바와 같이 확장된 UPnP 서비스 기술은 표준 UPnP 서비스 기술의 "Action" 및 "StateVariable" 요소에 속성으로 실제 Non-UPnP 장치의 제어 명령과 상태변수 식별자를 추가한 것이다. UPnP 제어기에서 SOAP 형식의 제어 명령이 오면 브리지에서는 UPnP 제어 명령의 이름과 매핑된 Non-UPnP 제어명령(ActCmd의 값)을 Non-UPnP 장치에 전송한다. 이 때 아래와 같은 두 가지 방법을 통해 Non-UPnP 장치를 제어한다.

2.2.1 Message Field Description

일종의 프로토콜 규약서인 MFD XML 파일로 각 필드에 대한 길이와 역할을 기술한다. MFD의 구조는 크게 Packet Information과 Packet Structure로 나누어진다.

Packet Information에는 Argument 타입과 Message 타입을 기술하며 패킷의 가변 길이 여부 및 전체 크기, 그리고 패킷 바이트 배열의 정렬방식(Little Endian 및 Big Endian)을 지정한다.

Packet Structure는 패킷의 Message타입 별 필드를 세부적으로 정의하며 각 필드의 Placeholder(삽입 가능한 값)를 나열 할 수 있다. 각 필드는 필드의 의미를 나타내는 Description과 Message Converter에서 데이터 Mapping을 위한 Placeholder Type ID, 그리고 크기 정보를 포함한다.

MFD를 파싱하여 UPnP 브리지에서 메시지 변환을 위한 정보를 얻어 상호 변환을 가능하게 한다.

Message Converter는 MFD 정보의 필드 정의를 참고하여 각 메시지의 바이트 배열을 생성하고 정보를 채운다. 그리고 Non-UPnP 로부터 들어오는 바이트 배열의 데이터를 파싱하여 적절한 UPnP API를 호출하고 필요한 인자를 전달하는 역할을 한다.

2.2.2 Non-UPnP Proxy Device

실제 Non-UPnP 장치의 메시지 변환 및 제어역할을 하는 대리자 객체이다. 브리지에 Non-UPnP 장치가 접속 시 가상의 UPnP 장치 관리자는 확장된 UPnP 서비스 기술을 이용하여 가상의 UPnP 장치 객체를 생성한다. 가상의 UPnP 장치 객체는 자신과 매핑된 Non UPnP Proxy 장치 객체를 생성하면서 UPnP 네트워크 상에 자신의 존재를 알린다.

UPnP 제어기는 아래 그림처럼 Non UPnP Motor Controller를 UPnP 네트워크상에 있는 다른 장치들처럼 제어할 수 있다.

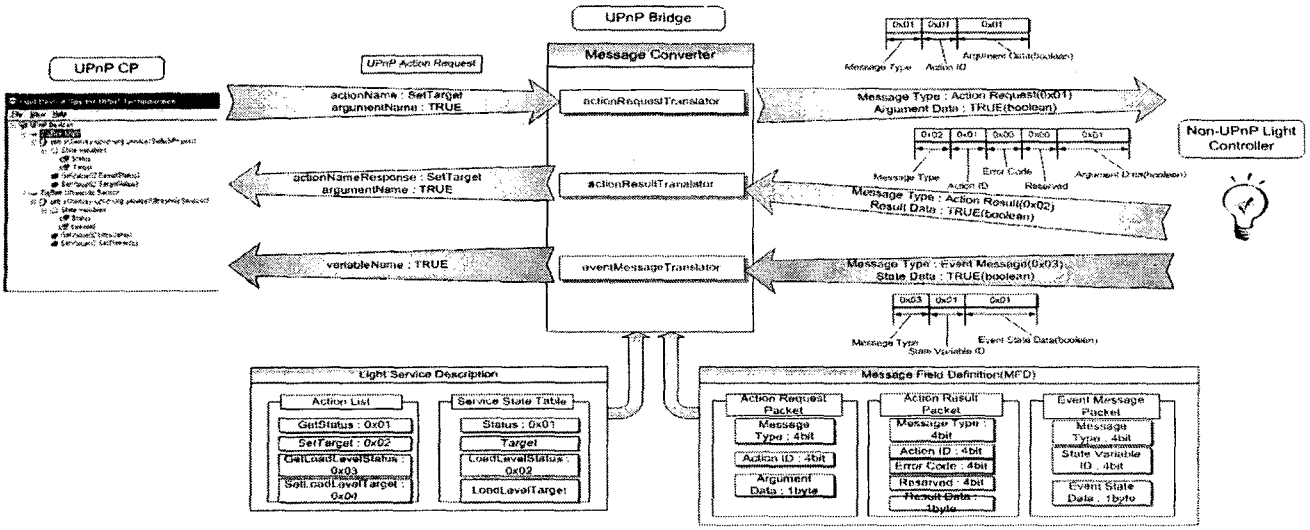

그림 4. MFD를 이용한 제어/이벤트 메시지 변환 흐름도

3. 구 현

우리는 제안한 브리지의 가능성을 검증하기 위해 그림 3에서 보는 바와 같이 테스트 베드를 구성하였다. Linux 기반 UPnP 브리지의 RS232C 포트에 Non-UPnP 초음파 센서(CPU: Atmega128L)를 연결하고 Windows 기반 UPnP 브리지의 RS232C 포트에 Non-UPnP 전등 제어기(CPU:Atmega128L)와 CAN 네트워크 인터페이스에 모터 제어기 역할을 하는 PC(CPU:Pentium4, 256RAM, Windows)를 연결하였다. 각 Non-UPnP 장치의 정보를 UPnP 기술로 표현하여 브리지에 등록하고 메시지 포맷을 MFD으로 표현하여 브리지에 등록하였다. UPnP 제어기는 Intel Device Spy를 사용하여 Non UPnP 장치인 전등 및 모터 제어와 초음파 센서에 이벤트 요청을 하여 센싱 된 값을 확인 할 수 있었다.

그림 4는 RS232와 연결된 Light Controller를 MFD를 통해 제어 및 이벤트 메시지를 변환하는 흐름을 나타낸다.

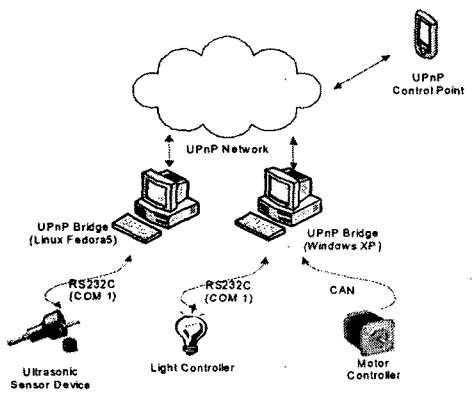

그림 3. 테스트 베드 구성도

4. 결 론

우리는 UPnP와 다양한 네트워크 인터페이스(RS232C, CAN, USB, IEEE1394)와 연결된 IP 기반이 아닌 장치의 상호 연동을 위해 새로운 브리지를 제안하였다. 실험 결과에서 보았듯이 Non-UPnP 장치 개발자는 자신의 장치를 수정하지 않고 장치의 메시지 포맷을 정의하는 MFD와 장치의 정보를 정의하는 확장된 UPnP 기술을 통해 코딩이 필요 없이 동적으로 UPnP 네트워크와 쉽게 상호연동 할 수 있었다.

본 논문에서 제안하는 브리지는 자원이나 비용 등의 문제로 인해 미들웨어를 제공하지 않는 임베디드 장치를 제어하는 UPnP 기반의 홈 네트워크 및 로봇 분야 등에 이용 가능할 것이다.

참 고 문 헌

[1] Universal Plug and Play Specification, www.upnp.org
[2] J. Allard, V.Chinta, S.Gundala, G.G.Richard III, "Jini Meets UPnP: An Architecture for Jini/UPnP interoperability", Application and the Internet, 2003
[3] Sun-Mi Jun, Nam-Hoo Park, "Controlling non IP Bluetooth devices in UPnP home network", Advanced Communication Technology, 2004
[4] Donghee Kim, Jun Hee Park, Poltavets Yevgen, KyeongDeok Moon, YoungHee Lee, "IEEE1394/UPnP Sortware Bridge", Consumer Electronics, IEEE Transactions on, 2005
[5] Yu-Hsiang Sheng, Wen-Wei Lin, Chia-Shou Tu, Kuen-Ming Lee, "Integrate LonWorks and UPnP devices by OSGi gateway and universal remote console", Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference
[6] Miller.B.A, Nixon.T, Tia.C, Wood.M.D, "Home networking with Universal Plug and Play", Communications Magazine IEEE 2001
[7] Dong-Sung Kim; Jae-Min Lee; Wook Hyun Kwon; In Kwan Yuh, "Design and implementation of home network systems using UPnP middleware for networked appliances", Consumer Electronics, IEEE Transactions on
[8] CAN, www.can-cia.org
[9] IEEE1394, IEEE std 1394a-2000
[10] USB, www.usb.org