다양한 Non-IP 장치를 위한 UPnP 브리지 구조

준회원 강 정 석*, 최 용 순*, 정회원 박 홍 성**

An Architecture of UPnP Bridge for Non-IP Devices with Heterogeneous Interfaces

Jeong-Seok Kang*, Yong-Soon Choi* ***Associate Members***,
Hong-Seong Park** ***Regular Member***

요 약

본 논문은 다양한 네트워크 인터페이스들로 연결된 Non-IP 장치들과 UPnP 네트워크 장치간의 상호 연동을 위한 새로운 UPnP 브리지 구조를 제안한다. 다양한 Non-IP 장치를 UPnP와 손쉽게 연동하기 위해 Non-IP 장치 대신 브리지 내에 가상의 UPnP 장치를 제공하여 일반적인 UPnP 장치의 기능을 하도록 한다. 또한 Non-IP 장치의 정보를 표현한 Non-IP 장치 기술(記述)과 Non-IP 장치들의 다양한 메시지 프로토콜을 UPnP 메시지 프로토콜로 변환하기 위해 각 Non-IP 장치의 메시지 변환용 XML 기반 메시지 필드 기술(記述) 방법과 Non-IP 장치의 서비스들을 정의하고 두 네트워크 간 제어 명령 및 상태 정보를 매핑 시키기 위한 확장된 UPnP 서비스 기술(記述) 방법을 정의한다. 위의 세 가지 기술(記述)들을 이용하여 자동으로 Non-IP 장치 메시지를 UPnP 메시지와 상호 변환이 가능하도록 하는 메시지 변환 모듈을 제공함으로써 Non-IP 장치 개발자는 부가적인 실행 프로그래밍이 없이 쉽게 동적으로 새로운 타입의 Non-IP 장치를 UPnP 네트워크와 연결할 수 있다. 본 논문에서는 실제 환경에서 RS232와 CAN으로 연결된 테스트베드를 구축하여 브리지의 동작성을 검증한다.

Key Words : UPnP Bridge, Message Field Description, Virtual UPnP Device, Extended UPnP Service Description, Non-IP Sink

ABSTRACT

This paper presents an architecture of UPnP Bridge for interconnecting Non-IP devices with heterogeneous network interfaces to UPnP devices on UPnP networks. The proposed UPnP Bridge provides a Virtual UPnP device that performs generic UPnP Device's functionalities on behalf of Non-IP device. This paper defines 3 types of descriptions, Device Description, Message Field Description, and Extended UPnP Service Description in order to reduce the amount of effort required to connect a non-IP device with a new interface or message format to UPnP network. By these three types of descriptions and Message conversion module, developers for Non-IP devices can easily connect the devices to UPnP network without additional programming. So UPnP control point controls Non-IP devices as generic UPnP device. Some experiments validate the proposed architecture, which are performed on a test bed consisting of UPnP network, the proposed bridge, and non-IP devices with CAN and RS232 interfaces.

※ 본 연구는 BK21 및 산업자원부의 지원에 의해 수행되었습니다.
* 강원대학교 전자통신공학과 산업정보통신 연구실({sleeper82, libris}@control.kangwon.ac.kr)
** 강원대학교 IT특성화학부 전기전자공학부 산업정보통신 연구실(hapark@kangwon.ac.kr)
논문번호 : KICS2007-07-319, 접수일자 : 2007년 7월 19일, 최종논문접수일자 : 2007년 11월 28일

Ⅰ. 서 론

UPnP[1]는 동적인 분산 컴퓨팅 환경을 위한 서비스 기반 미들웨어로서 프로그래밍 언어와 운영체제에 독립적이다. 다양한 컴퓨팅 장치간의 통신을 위해 범용 프로토콜인 TCP/IP와 다양한 웹 기술(HTTP, SSDP, SOAP, GENA)을 사용함으로써 물리 계층과 전송 계층에 독립적으로 구현이 가능하며 사용자는 쉽게 컴퓨팅 장치를 연결 할 수 있어 홈 네트워크[6], 로봇[7] 등에 사용되고 있다.

하지만 UPnP는 표준 IP 프로토콜을 채택하였기 때문에 IP 프로토콜을 장착하지 않은 Non-IP 장치와 상호연동이 어렵다. 그래서 다양한 네트워크 인터페이스를 사용하는 분산 시스템 환경에서 UPnP를 사용하기에 제약이 존재하며, 이를 해결하기 위해서는 UPnP와 Non-IP 장치간 상호연동을 위한 브리지가 필요하다[5][6]. 현재 UPnP와 Non-IP 장치간 상호 운용에 대한 연구는 다음과 같다: UPnP와 Bluetooth 장치간 상호 연동을 위하여 네트워크 간 장치 기술(記述) 및 메시지 변환을 위한 UPnP-Bluetooth 프락시(proxy) 서버를 사용하는 방법이 있다[2]. UPnP 컨트롤 포인트에서 IEEE1394에서 정의한 AV장치를 제어하기 위하여 실제 IEEE1394 장치를 제어 할 수 있는 가상 장치 모듈을 기반한 소프트웨어 브리지 방법이 있다[3]. 그리고 LonWorks 장치를 UPnP 네트워크와 연결하기 위하여 실제 LonWorks 장치를 대신하여 UPnP와 통신하면서 두 네트워크 간 메시지를 변환하는 C# 언어로 구현된 변환기 모듈을 이용하는 방법[4]이 있다. 하지만 위의 연구들은 특정 네트워크 인터페이스만 지원하고 그에 적합하도록 설계되어 있어, 다양한 네트워크 인터페이스를 사용하는 분산 시스템에는 많은 브리지 혹은 변환기가 설치되어야 하기 때문에 적합하지 않다. 또한 UPnP와 하나의 특정 프로토콜간의 상호연동만 가능하므로 사용자가 자유롭게 개발한 다양한 메시지 프로토콜을 가질 수 있는 일반적인 Non-IP 장치와의 UPnP 브리지의 구현은 어려울 수 있다.

현재 대부분의 UPnP 브리지[2][3][4]는 Non-IP 장치를 제어하고 두 네트워크간 메시지를 변환하는 프락시 또는 가상 장치 형태의 모듈이 Non-IP장치마다 별도로 존재하므로 새로운 타입의 Non-IP 장치를 연결 할 때 마다 별도의 모듈을 개발하여야 한다. 이 때 브리지 실행 중에 동적으로 Non-IP 장치와 UPnP를 연결하기 위한 별도의 모듈을 장착할 수 없어 불편하다. 이러한 관점에서 각 Non-IP 장치마다 두 네트워크간 메시지 변환 역할을 하는 모듈의 수월한 개발과 동적 로딩 기술이 필요하다.[12]

본 논문은 RS232C, CAN[8], IEEE1394[9], USB[10]와 같은 다양한 네트워크 인터페이스들과 연결된 다양한 메시지 프로토콜을 가진 일반적인 Non-IP 장치들과 UPnP 네트워크상의 상호연동을 위한 새로운 UPnP 브리지 구조를 제안한다. 제안된 구조의 특징은 UPnP와 Non-IP 장치간 일 대 다 메시지 변환을 손쉽게 하기 위해 다양한 Non-IP 장치의 메시지 형식을 표현할 수 있는 XML 형태의 메시지 필드 기술(記述)과 이를 통해 자동으로 메시지 변환을 수행하는 텍스트 메시지 변환기(앞으로는, 이를 XML 기반 메시지 변환기로 칭함)를 제공한다는 것이다. 제안한 UPnP 브리지는 기존의 UPnP 브리지에서 사용하는 바이너리 형태의 직접적인 메시지 변환기를 제공함과 동시에 동적 로딩 기술을 통해 런 타임시 XML 기반 메시지 변환기를 추가 및 삭제가 가능하도록 하여 불필요한 메시지 변환기로 인한 메모리 낭비를 줄였다. 특히 후자의 방법은 한 개의 Non-IP 장치에 대해 일반적인 UPnP 장치의 기능을 수행하는 가상의 UPnP 장치를 제공하고, Non-IP 장치의 서비스를 표현하고 두 네트워크 간 제어 명령 및 상태 정보를 매핑 시키기 위한 확장된 UPnP 서비스 기술(記述)을 정의하여 텍스트 메시지 변환기에 의해 자동으로 메시지 변환이 이루어진다. 이러한 메시지 필드 기술(記述) 방법과 서비스 기술(記述) 방법을 통해 브리지 내에 추가적인 프로그램이 없이 쉽게 동적으로 새로운 타입의 Non-IP 장치를 UPnP 네트워크와 연결할 수 있다. 제안하는 브리지 방법의 유효성을 RS232C와 CAN 인터페이스와 물리적으로 연결된 임의의 메시지 프로토콜을 가진 장치와 UPnP 네트워크상의 장치간의 동작을 통하여 보여준다.

본 논문은 Ⅱ장에서 UPnP 브리지 설계 시 고려사항과 구조를 설명하고 Ⅲ장에서 UPnP와 Non-IP 장치의 상호운용을 위한 기술(記述) 방법과 동작에 대해 설명한다. 그리고 Ⅳ장에서 제안된 UPnP 브리지의 동작성을 검증하기 위한 실험과 정성적 평가에 대해 설명하고 마지막으로 Ⅴ장에서 결론을 맺도록 한다.

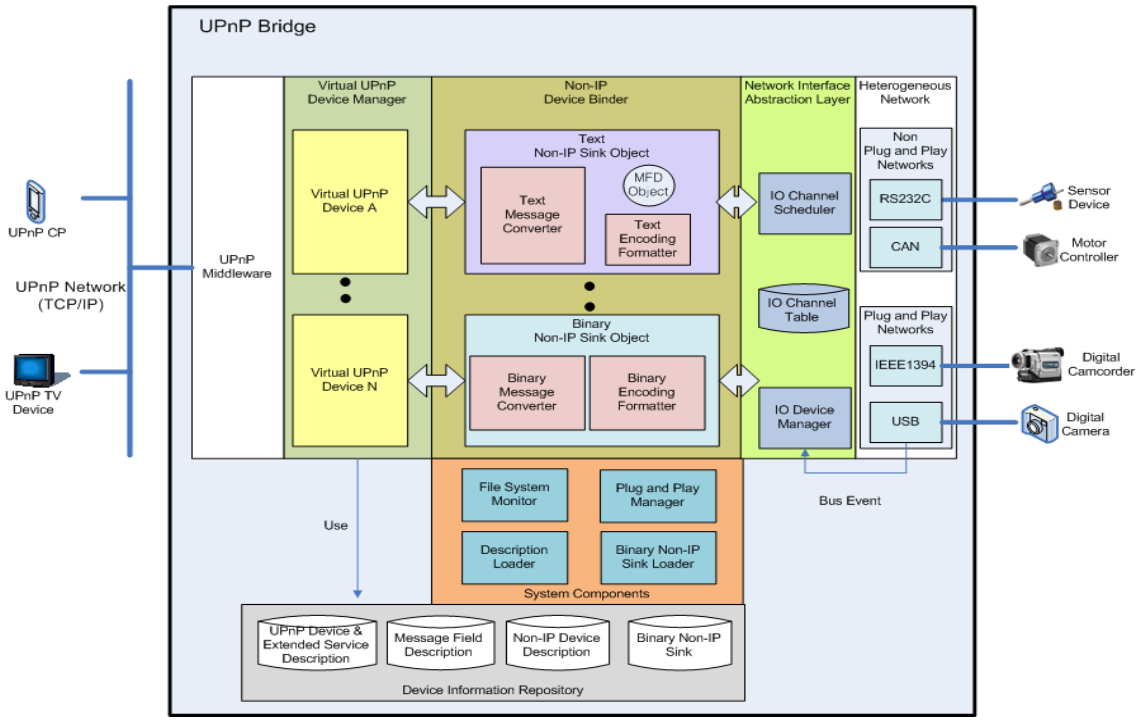


그림 1. UPnP 브리지 제안 구조

Ⅱ. 제안한 UPnP 브리지 구조

2.1 UPnP 브리지 설계 시 고려사항

2.1.1 Non-IP 장치 기술(記述)

UPnP는 XML 형태의 기술을 통해 장치의 정보를 표현하고 이를 통해 UPnP 장치의 발견 및 통신 상대를 결정한다. UPnP 브리지는 다양한 네트워크 인터페이스와 연결된 Non-IP 기반 비표준 장치나 IEEE1394나 USB 표준 단체에서 정의한 표준 장치(A/V, Storage 등)를 UPnP 기술(記述)로 표현하여야 한다.

2.1.2 메시지 포맷 변환

UPnP는 제어나 이벤트를 위한 기본 메시지 형식들이 SOAP이나 GENA 등의 웹 프로토콜을 통해 정의 되어 있다. UPnP 브리지는 비표준 장치나 다른 단체에서 정의한 표준 장치에 맞추어 UPnP 메시지 타입을 변환하여야 하고 그 반대의 경우에도 변환하여야 한다.

2.1.3 Non-IP 장치에 대한 플러그 앤 플레이 기능

UPnP는 주기적인 존재 공지(Presence announcement)를 통해 UPnP 장치에 PnP 기능을 제공한다. UPnP 브리지는 PnP 기능을 가지고 있지 않은 네트워크 인터페이스(RS232C, CAN)와 연결된 Non-IP 장치에 대해 PnP 기능을 제공하여야 한다.

2.1.4 다양한 네트워크 인터페이스의 추상화

다양한 기능을 수행하는 컴퓨팅 장치들을 연결하는 네트워크 인터페이스는 각 장치의 특징에 따라 다양하다. CAN의 경우 대역폭은 작지만 실시간성 데이터 전송을 위한 컴퓨팅 장치들간의 연결을 위해 사용되고 IEEE1394의 경우 대용량의 멀티미디어 데이터 전송을 위해 주로 사용된다. UPnP 브리지는 이러한 다양한 네트워크 인터페이스를 제공하여야 하고 Non-IP 장치 개발자가 다양한 네트워크 인터페이스의 특성을 고려하지 않고 개발할 수 있도록 네트워크 인터페이스 추상화가 필요하다.

2.2 UPnP 브리지 구조 및 내부 컴포넌트

제안하는 UPnP 브리지는 그림 1에서 보는 바와 같이 TCP/IP 기반 UPnP 네트워크와 Non-IP 기반 네트워크를 연결하기 위해 다음과 같은 주요 9개의 컴포넌트로 구성되어 있다.

2.2.1 가상 UPnP 장치(Virtual UPnP Device,VUD)

실제 각 Non-IP 장치를 대신하여 UPnP 네트워크와 통신하면서 일반적인 UPnP 장치의 기본적인 기능(존재 공지, 제어, 이벤트)을 수행하면서 서비스를 제공한다.

- 존재 공지(Presence announcement) - UPnP 네트워크상에 Non-IP 장치를 대신하여 주기적으로 자신의 존재를 알리고 UPnP 장치 및 서비스 기술(記述) 요청에 대한 응답을 한다.
- 제어(Control) - UPnP 컨트롤 포인트로부터 제어 메시지를 수신하면 자신과 연결(Binding)된 Non-IP 싱크(Sink)를 통해 실제 Non-IP 장치를 제어한다.
- 이벤트(Event) - Non-IP 싱크(Sink)를 통해 변환된 실제 Non-IP 장치의 상태 값을 수신 받아 이벤트를 신청한 모든 UPnP 컨트롤 포인트에 전송한다.

2.2.2 가상 UPnP 장치 관리기 (Virtual UPnP Device Manager, VUDM)

가상 UPnP 장치를 관리하며, Non-IP 싱크 연결기(Sink Binder)로부터 새로운 Non-IP 장치에 대한 연결이나 해제 이벤트가 오면 가상 UPnP 장치 관리기는 같은 타입의 가상 UPnP 장치를 생성(혹은 소멸)함으로써 UPnP 네트워크 상에 등록(혹은 해제) 한다.

2.2.3 Non-IP 싱크(Non-IP Sink, NIS)

실제 Non-IP 장치와 직접 통신 하면서 제어 및 이벤트 메시지를 송수신한다. 메시지 포맷기(Encoding Formatter)를 통해 Non-IP 장치의 메시지를 직렬화/역직렬화 하고 메시지 변환기(Message Converter)를 이용하여 UPnP와 Non-IP 장치간 메시지를 상호 변환한다. 동일한 메시지 프로토콜을 가지는 Non-IP 장치는 하나의 NIS를 공유할 수 있고 다음과 같이 크게 두 가지 타입이 있다.

- 텍스트형 NIS - Non-IP 장치의 메시지 프로토콜을 정의한 메시지 필드 기술(記述)과 텍스트 메시지 포맷기를 통해 Non-IP 장치 메시지를 직렬화/역직렬화하고 확장된 UPnP 서비스 기술(記述)과 텍스트 메시지 변환기를 통해 메시지를 상호 변환한다.
- 바이너리형 NIS - Non-IP 장치의 메시지 프로토콜이 복잡할 경우 텍스트형 NIS는 메시지를 변환하는데 오버헤드가 있다. 바이너리형 NIS는 Non-IP 장치 메시지의 직렬화/역직렬화와 두 네트워크 간 메시지 변환을 사용자가 직접 코딩을 통해 구현한 후 DLL 형태로 저장하므로 이러한 오버헤드가 줄어든다. 또한 기존의 방법과 다르게 새로운 Non-IP 장치 타입 추가 시 동적으로 브리지에 등록이 가능하다.

2.2.4 Non-IP 싱크 연결기(Non-IP Sink Binder, NISB)

NIS를 관리하며 각 NIS와 가상 UPnP 장치를 연결한다. Non-IP 네트워크에 Non-IP 장치가 연결/해제 되면 Non-IP 장치의 목적지 주소와 매핑되어 있는 NIS를 생성/소멸한다. 또한 Non-IP 장치가 연결/해제 되었음을 가상 UPnP 장치 관리기에게 알린다.

2.2.5 플러그 앤 플레이 관리기(Plug and Play Manager, PnPM)

플러그 앤 플레이 기능을 가진 네트워크 인터페이스(IEEE1394, USB)는 물리적 버스 이벤트를 통해, 플러그 앤 플레이 기능을 제공하지 않는 네트워크 인터페이스(RS232C, CAN)에 대해서는 타이머를 이용하여 주기적인 "Keep Alive"메시지 송수신을 통해 Non-IP 장치의 연결 여부를 확인한다.

2.2.6 파일시스템 모니터(File System Monitor)

Non-IP 장치 정보들을 저장하고 있는 장치 정보 저장소(Device Information Repository, DIR)의 상태를 감시한다. 장치 정보 저장소에 Non-IP 장치 정보에 대한 삽입/삭제 이벤트가 발생하면 이벤트의 타입과 장치 정보를 저장한 파일명을 플러그 앤 플레이 관리기에 알린다. 파일시스템 모니터를 통해 실제 Non-IP 장치의 메시지를 변환하는 NIS를 동적으로 브리지에 추가 할 수 있다.

2.2.7 기술(記述) 로더(Description Loader)

XML 형태의 Non-IP 장치의 기술(記述)들을 동적으로 로딩하여 XML 파서(Parser)를 이용하여 파싱한다. 불필요한 Non-IP 싱크(NIS) 의 생성을 방지하기 위해 브리지는 사전에 Non-IP 장치 기술(記述)에 대한 정보를 로딩 하거나 텍스트형 NIS를 생성하기 위해 메시지 필드 기술(記述)을 로딩한다.

2.2.8 바이너리형 Non-IP 싱크 로더(Binary Non-IP Sink Loader)

사용자가 직접 구현한 DLL 형태의 Non-IP 싱크(NIS) 객체를 동적으로 로딩하여 시스템의 중단 없이 새로운 Non-IP 장치의 메시지를 변환하는 NIS 객체를 추가 할 수 있다.

2.2.9 네트워크 인터페이스 추상화 계층(Network Interface Abstraction Layer)

다음과 같은 세 모듈을 통해 다양한 네트워크 인터페이스를 채널로 추상화 한 계층이다.

- 입출력 장치 관리기 - 각 네트워크 인터페이스의 데이터 송수신을 담당하는 입출력 장치를 관리한다. 입출력 장치의 생성/소멸과 동작(동기, 비동기)을 결정한다.
- 입출력 채널 테이블 - Non-IP 장치와 연결을 유지하는 채널 객체를 저장하고 있는 테이블이며 각 채널 객체는 실제 네트워크 인터페이스의 정보(타입, 목적지 주소 등)를 가지고 있다.
- 입출력 채널 스케줄러 - 각 채널 별로 우선순위에 따라 데이터를 송수신 하는 모듈로서 위급한 메시지의 경우 먼저 전송이 가능하다.

Ⅲ. UPnP와 Non-IP 장치 통합을 위한 記述과 동작

UPnP와 Non-IP 장치의 상호연동을 위해서 UPnP 브리지는 Non-IP 장치 기술(記述), 표준 UPnP 장치 기술(記述), 확장된 UPnP 서비스 기술(記述)과 메시지 필드 기술(記述)이 필요하다.

3.1 상호연동을 위한 장치 기술(記述)

3.1.1 Non-IP 장치 기술(記述)

그림 2에서 보는 바와 같이 Non-IP 장치 기술(記述)은 XML 형식으로 실제 Non-IP 장치의 정보(이름, 물리적 주소 등)를 표현한다. PnP 요소는 Non-IP 싱크(Sink)의 생성 시점을 결정한다. NIS 요소는 NIS의 타입과 파일 이름을 나타낸다.“Off”일 경우 Non-IP 장치 기술(記述)을 장치 정보 저장소에 저장 시 NIS 요소의 Type 속성을 보고 Non-IP 싱크(Sink)를 생성한다. “On”일 경우 Non-IP 싱크 연결기는 플러그 앤 플레이 관리기를 통해 주기적인 메시지를 Non-IP 장치 기술(記述)에서 명시한 주소(IEEE1394, USB는 제외)로 보내서 응답이 오면 Non-IP 싱크를 생성한다. Name 요소는 표준 UPnP 장치 기술(記述)의 파일 이름과 매핑

```
<Non IP Device Info>
    <Name> Non UPnP Device Name </Name>
    <Network Interface>
        <Type> RS232C or CAN or IEEE1394 or USB </Type>
        <Address> address </Address>
    </Network Interface>
    <PnP> On or Off </PnP>
    <NIS Type="Text or Binary"> File Name </NIS>
</Non IP Device Info>
```

그림 2. Non-IP 장치 기술

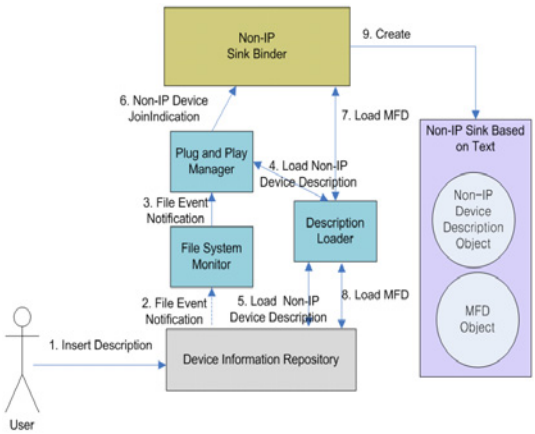


그림 3. 텍스트 형태의 새로운 NIS의 동적 생성

되어 있다. Non-IP 싱크 관리기는 Non-IP 장치 기술(記述)을 통해 Non-IP 싱크를 생성하고 가상 UPnP 장치 관리기에게 Non-IP 장치의 이름과 연결 여부를 알리면 가상 UPnP 장치 관리기는 Non-IP 장치의 이름을 가지고 표준 UPnP 장치 기술(記述)을 찾아 가상 UPnP 장치를 생성한다. 그림 3은 Non-IP 장치 기술(記述)을 통해 새로운 Non-IP 장치를 제어할 수 있는 텍스트 형태의 Non-IP 싱크가 동적으로 생성되는 과정을 나타낸다. 사용자가 장치 정보 저장소에 Non-IP 장치 기술(記述) 파일을 저장하면 파일 시스템 모니터를 통해 플러그 앤 플레이 관리기는 파일 이벤트를 수신하고 기술(記述) 로더를 통해 Non-IP 장치 기술(記述)을 로딩한다. PnP 요소를 파싱하여 “Off”일 경우 Non-IP 싱크 연결기에 새로운 Non-IP 장치가 연결되었음을 알린다. Non-IP 싱크 연결기는 기술(記述) 로더를 이용하여 메시지 필드 기술(記述)을 로딩하여 텍스트 형태의 Non-IP 싱크를 생성한다.

3.1.2 확장된 UPnP 서비스 기술(記述)

그림 4와 같이 확장된 UPnP 서비스 기술(記述)은 Non-IP 장치의 제어 명령과 상태 변수를 UPnP 장치와 매핑 시키기 위해 표준 UPnP 서비스 기술(記述)의 Action과 StateVariable 요소에 속성으로 실제 Non-IP 장치의 제어 명령(ActCmd)과 상태 변수(SVarID) 식별자를 추가한 것이다. 텍스트 메시지 변환기는 이 기술(記述)을 통해 두 네트워크 간 메시지를 변환한다.

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

그림 4. 확장된 UPnP 서비스 기술(記述)

3.1.3 메시지 필드 기술(記述)

Non-IP 장치는 기능이나 네트워크 인터페이스의 특성 또는 개발자에 따라 다양한 메시지 포맷을 가질 수 있다. UPnP 브리지는 다양한 메시지 포맷을 가진

```
<MFD version="0.7" xmlns="urn:schemas-mfd-0-7">
<PacketInfo>
  <FieldUnit>Size unit for type decision</FieldUnit>
  <TypeFieldStart>Start field position for type decision</TypeFieldStart>
  <TypeFieldEnd>End field position for type decision</TypeFieldEnd>
  <MessageTypeList>
    <MessageType>
      <TypeID>Type ID of message for Converter</TypeID>
      <Value>Value of type field</Value>
      <Direction>Direction of sending message</Direction>
    </MessageType>
  </MessageTypeList>
  <ArgumentTypeList>
    <ArgumentType>
      <Name>Name of argument</Name>
      <Type>Type ID of argument for Converter</Type>
    </ArgumentType>
  </ArgumentTypeList>
  <MaximumPacketSize>Maximum packet size in option </MaximumPacketSize>
  <ByteAlign>Byte align to target device</ByteAlign>
</PacketInfo>
<PacketStructure>
  <Packet>
    <Name>Message name</Name>
    <PacketType>Message variable</PacketType>
    <PayloadList>
      <Payload>
        <Name>Field name</Name>
        <PayloadType>Type ID of field for Converter</PayloadType>
        <Value>
          <PlaceHolderList>
            <PlaceHolder>
              <Name>Placeholder name</Name>
              <Value>Value of placeholder</Value>
              <Type>Type ID of placeholder field for Converter</Type>
            </PlaceHolder>
          </PlaceHolderList>
        </Value>
        <SizeUnit>Size unit of field</SizeUnit>
        <Size>Size of field</Size>
      </Payload>
    </PayloadList>
  </Packet>
</PacketStructure>
</MFD>
```

그림 5. 메시지 필드 기술(記述)

표 1. MFD 요소 설명

요소	설명
FieldUnit	메시지 타입 필드의 길이 단위
TypeFieldStart	메시지 타입 필드의 시작 위치
TypeFieldEnd	메시지 타입 필드의 마지막 위치
MessageTypeList	메시지의 타입들을 정의
TypeID	메시지 변환을 위한 메시지 타입별 고유 ID
Value	각 장치별 메시지 타입의 값
Direction	메시지의 전송 방향 정의
ArgumentTypeList	가능한 데이터 타입을 정의
ByteAlign	바이트를 표현하는 순서에 대한 정의
Packet	메시지 타입별 상세히 필드를 정의
PacketType	현재 메시지가 가변길이 타입인지 고정길이 타입인지 구분
PayloadType	실제로 MFD를 해석하는 NIS가 인식하는 필드 타입
PlaceHolderList	필드내에 들어갈 수 있는 값 리스트
SizeUnit	필드의 길이 단위
Size	필드의 길이

Non-IP 장치를 UPnP 네트워크와 연결하기 위해서 이러한 다양한 메시지 포맷을 분석 할 수 있어야 한다. 이러한 역할을 하는 것이 Non-IP 싱크이며 메시지 필드 기술(記述)은 Non-IP 장치의 메시지 포맷을 XML 형태로 표현한 것이다. 메시지 필드 기술(記述)은 Non-IP 싱크가 효율적으로 메시지를 변환할 수 있게 하기 위해서 다양한 메시지의 필드를 정의하고 의미를 부여한다. 메시지 필드 기술(記述)은 그림 5에서 보는 바와 같이 크게 두 부분으로 나누어지며 각 요소별 자세한 설명은 표 1과 같다.

- PacketInfo
  메시지 타입 판별을 위한 메시지 타입 리스트와 데이터 타입 정보 등 메시지에 필수적인 정보를 정의한다.
- PacketStructure
  각 메시지 타입별 구조를 기술하는 부분으로 각 메시지는 여러 필드를 가지고 있으며 각 필드마다 필드의 의미, 길이 등의 내용을 기술(記述)한다.

그림 6은 Non-IP 장치에서 발생한 상태 이벤트 메시지를 확장된 UPnP 서비스 기술(記述)과 메시지 필드 기술(記述)를 이용해 UPnP 메시지로 변환하는 과정이다.

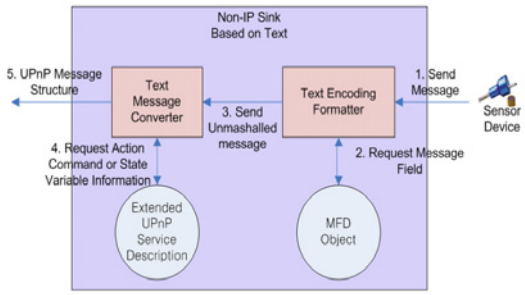


그림 6. 메시지 변환 과정

텍스트 Non-IP 싱크는 Non-IP 네트워크에 연결된 센서 장치로 부터 센싱 값에 대한 이벤트 메시지를 수신하면 텍스트 메시지 포맷기를 통해 Non-IP 센서 장치의 이벤트 메시지를 역직렬화 하여 텍스트 메시지 변환기로 전송한다. 텍스트 메시지 변환기는 확장된 UPnP 서비스 기술(記述)에서 StateVariable 요소를 분석하여 UPnP 메시지로 변환한 후 자신과 연결된 가상 UPnP 장치로 전송한다.

3.2 기본적인 동작 흐름

제안한 UPnP 브리지는 UPnP와 Non-IP 장치의 상호 연동을 위해 UPnP의 기본 동작인 서비스 발견, 존재 공지, 제어 그리고 이벤트 기능을 제공한다.

3.2.1 존재 공지 및 서비스 발견

플러그 앤 플레이 관리기는 주기적으로 Non-IP 네트워크상에 “Keep-Alive” 메시지를 송신하고 Non-IP 장치가 UPnP 브리지에 연결되면 새로운 Non-IP 장치의 연결여부를 Non-IP 싱크 연결기에게 알린다. Non-IP 싱크 연결기는 메시지 필드 기술(記述)을 로딩하여 Non-IP 싱크를 생성한 후 가상 UPnP 장치 관리기에게 새로운 Non-IP 장치의 연결을 알린다. 가상 UPnP 장치 관리기는 표준 UPnP 장치 기술(記述)과 확장된 UPnP 서비스 기술(記述)을 이용하여 생성된 Non-IP 싱크와 바인딩 할 수 있는 가상 UPnP 장치를 생성한 후 UPnP 미들웨어에 등록한다. 그런 다음 가상 UPnP 장치는 자신의 존재를 UPnP 네트워크 상에 알린다. UPnP 컨트롤 포인트는 자신이 관심 있어 하는 UPnP 장치를 찾기 위해 SSDP 프로토콜을 이용하여 Discovery 메시지를 UPnP 네트워크 상에 전송하면 UPnP 브리지 내의 가상 UPnP 장치는 Discovery에 대한 응답을 한다. 그 뒤 UPnP 컨트롤 포인트가 가상 UPnP 장치에 대한 자세한 정보(UPnP 장치

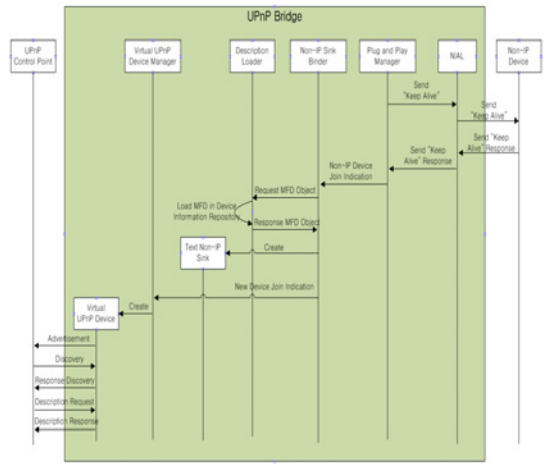


그림 7. Non-IP의 존재 공지 과정

및 서비스 기술(記述))를 얻으면 가상 UPnP 장치를 제어 할 수 있다. 그림 7은 사용자가 PnP 요소가 “On”인 Non-IP 장치의 기술(記述)을 장치 정보 저장소에 저장한 후 Non-IP 장치의 연결 시 UPnP 네트워크 상에 Non-IP 장치의 존재를 공지 하는 과정이다.

3.2.2 제어 요청과 응답

그림 8은 UPnP 제어 과정에 대한 순서이다. 이 과정에서 중요한 동작은 SOAP 형식의 UPnP 제어 메시지를 Non-IP 장치의 제어 메시지 포맷에 맞게 변환하는 것이다. 가상 UPnP 장치가 UPnP 제어 메시지를 수신하면 이를 분석하여 자신과 바인딩 되어있는 Non-IP 싱크에게 알린다. Non-IP 싱크는 메시지 필드 기술(記述)과 확장된 UPnP 서비스 기

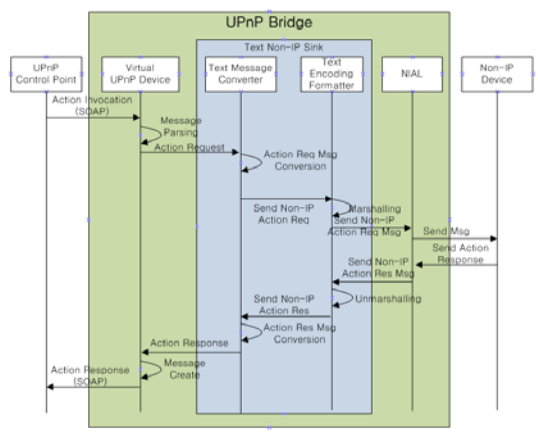


그림 8. Non-IP 장치의 제어 과정

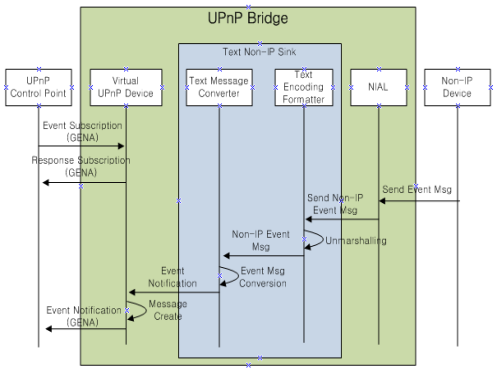


그림 9. Non-IP 장치의 이벤트 전송 과정

술(記述)을 통해 실제 Non-IP 장치의 메시지를 생성한 후 전송한다. 그 뒤 Non-IP 장치는 명령을 수행하고 응답 메시지를 UPnP 브리지에 전송한다. 최종적으로 UPnP 컨트롤 포인트는 가상 UPnP 장치로 부터 실제 Non-IP 장치에 대한 응답 메시지를 수신한다.

3.2.3 이벤트

그림 9에서 보는 바와 같이 UPnP 컨트롤 포인트는 가상 UPnP 장치 서비스의 상태에 대한 구독 요청을 하고 가상 UPnP 장치는 자신과 바인딩 되어 있는 Non-IP 싱크를 통해 Non-IP 장치로부터 변경된 상태 변수의 값을 수신 하면 구독을 요청한 모든 UPnP 컨트롤 포인트에게 이벤트 메시지를 전송한다. 이 과정에서 중요한 동작은 Non-IP 장치의 이벤트 메시지 포맷을 GENA 형식의 UPnP 이벤트 메시지 포맷에 맞게 변환 하는 것이다.

Ⅳ. 구현 및 평가

4.1 구현

4.1.1 테스트베드 환경

제안한 UPnP 브리지의 동작성을 검증하기 위해 그림 10과 같은 테스트 베드를 구축하였다. 제안한 UPnP 브리지는 C++ 언어를 이용하여 PC에 구현하였고 CyberLink[11]사에서 제공하는 개방형 UPnP 스택을 사용하였다. 사물과의 거리 측정을 위한 Non-IP 초음파 센서 장치는 RS232C 네트워크 인터페이스를 통해, 호스트 PC상에 구현된 전등을 제어하는 역할을 하는 Non-IP 전등 제어 장치 에뮬레이터는 CAN 네트워크 인터페이스를 통해 UPnP 브리지에 연결 하였다. 또한 Non-IP 초음파 센서와

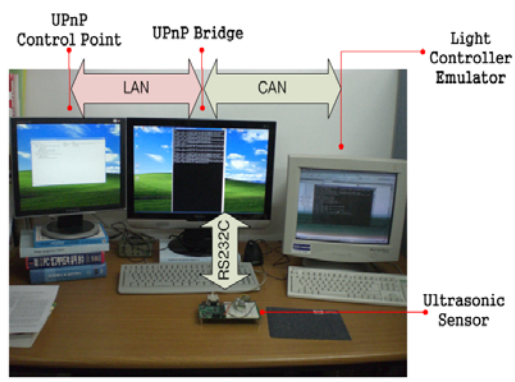


그림 10. 테스트 베드 환경

```
<actionList>
  <action ActCmd="1">
    <name>GetUltrasonic</name>
    <argumentList>
      <argument>
        <name>Ultrasonic</name>
        <direction>out</direction>
        <relatedStateVariable>
          UltrasonicState
        </relatedStateVariable>
      </argument>
    </argumentList>
  </action>
    .....
</actionList>
<serviceStateTable>
  <stateVariable sendEvents="yes" SVarID="1">
    <name>UltrasonicState</name>
    <dataType>ui2</dataType>
  </stateVariable>
    .....
</serviceStateTable>
```

(a) 초음파 센서의 확장된 서비스 기술 예

```
<Non IP Device Info>
  <Name> Ultrasonic Sensor </Name>
  <Network Interface>
    <Type> RS232C </Type>
    <Address> COM1 </Address>
  </Network Interface>
  <PnP> Off </PnP>
  <NIS Type="Text"> UltrasonicSensorMFD </NIS>
</Non IP Device Info>
```

(b) 초음파 센서의 Non-IP 장치 기술 예

그림 11. 초음파 센서의 기술(記述) 예

전등제어 장치에 대한 UPnP 장치 및 서비스 기술(記述), Non-IP 장치 기술(記述)을 UPnP 브리지에 등록하였다. 그림 11는 Non-IP 장치 기술(記述)과 확장된 UPnP 서비스 기술(記述)에 대한 것이다. 또한 위의 두 장치에 대한 메시지 변환을 위한 텍스트 형태의 Non-IP 싱크를 생성하기 위해 그림 12와 같은 메시지 필드 기술(記述)을 저장하였다. Non-IP 장치 제어를 위한 UPnP 컨트롤 포인트는

```
<PacketInfo>
  <FieldUnit>bit</FieldUnit>
  <TypeFieldStart>0</TypeFieldStart>
  <TypeFieldEnd>4</TypeFieldEnd>
  <MessageTypeList>
    <MessageType>
      <TypeID>UPnP_MT_ACTIONREQUEST</TypeID>
      <Value>0x03</Value>
      <Direction>out</Direction>
    </MessageType>
    ……
  </MessageTypeList>
  <ArgumentTypeList>
    <ArgumentType>
      <Name>none</Name>
      <Type>UPnP_PAYLOADDATA_NONE</Type>
    </ArgumentType>
    <ArgumentType>
      <Name>ui2</Name>
      <Type>UPnP_PAYLOADDATA_UI2</Type>
    </ArgumentType>
  </ArgumentTypeList>
  <ByteAlign>LittleEndian</ByteAlign>
</PacketInfo>
<PacketStructure>
  <Packet>
    <Name>ActionRequest</Name>
    <PacketType>Fixed</PacketType>
    <PayloadList>
      <Payload>
        <Name>MessageType</Name>
        <PayloadType>TYPE_FIELD</PayloadType>
        <SizeUnit>bit</SizeUnit>
        <Size>4</Size>
      </Payload>
      <Payload>
        <Name>ActionID</Name>
        <PayloadType>UPnP_ACTIONID</PayloadType>
        <SizeUnit>bit</SizeUnit>
        <Size>4</Size>
      </Payload>
  ……
  </Packet>
  <Packet>
    <Name>ActionResult</Name>
    ……
  </Packet>
  <Packet>
    <Name>EventMessage</Name>
  ……
  </Packet>
```

그림 12. 초음파 센서의 MFD 예

TCP/IP를 통해 UPnP 브리지와 연결되어 있다. 각 장치의 사양은 표 2와 같다.

표 2. 장치 사양

장치	사양
초음파 센서	MCU: Atmega128L, RAM: 8Mbytes
전등 제어 장치 에뮬레이터	CPU: 1.6GHz Pentium 4, RAM: 512Mbytes
UPnP 브리지	CPU: 2.13GHz Intel Core2, RAM: 1024Mbytes
UPnP 컨트롤 포인트	Intel사의 Device Spy Software, CPU: 2.0GHz Pentium 4, RAM: 512Mbytes

4.1.2 Non-IP 장치 제어 및 이벤트 시나리오

이번 절에서 우리는 UPnP 컨트롤 포인트가 제안한 UPnP 브리지를 통해 어떻게 Non-IP 장치를 제어하고 이벤트 메시지를 수신하는지 설명한다. 여기에서 사용되는 RS232C와 연결된 초음파 센서 장치의 실제 메시지 포맷은 그림 13과 같다.

그림 14는 UPnP 컨트롤 포인트에서 UPnP 기술(記述) 요청, 제어 그리고 이벤트 요청에 대한 결과 화면을 보여준다. Non-IP 초음파 센서와 전등 제어 장치가 UPnP 브리지에 연결되면 이들은 가상 UPnP 장치 형태로 UPnP 네트워크상에 연결 되면서 자신의 존재를 알린다. UPnP 컨트롤 포인트는 두 가상 UPnP 장치에 대한 자세한 정보를 알기 위해 UPnP 기술(記述)들을 요청한다. 그림 14a는 가상의 UPnP 전등 제어 장치의 표준 UPnP 장치 기술(記述)을 나타낸다. UPnP 컨트롤 포인터는 Non-IP 장치에 의존적인 Non-IP 장치 기술(記述)에 대해서는 알 필요가 없다. 그 다음 UPnP 컨트롤 포인트는 각 가상 UPnP 장치의 서비스에 상태 변수의 변경에 대한 구독을 요청한다. 이제 UPnP 컨트롤 포인트는 일반적인 UPnP 장치처럼 두 가상 UPnP 장치를 제어할 수 있고 상태 이벤트 값을 수신 받을 수 있다.

UPnP 컨트롤 포인트는 “SetLight(true)” 명령을 가상의 UPnP 전등 제어 장치에 전송한다. 가상의 UPnP 전등 제어 장치는 자신과 연결된 Non-IP 싱크를 통해 메시지를 변환한 후 실제 Non-IP 전등 제어 장치에 제어 메시지를 보내고 전등의 불이 켜진다. 그 뒤 Non-IP 전등 제어 장치는 UPnP 브리

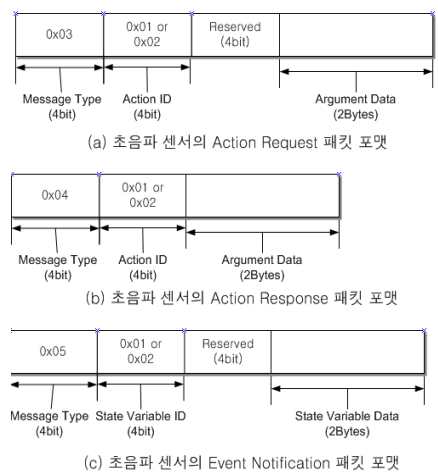
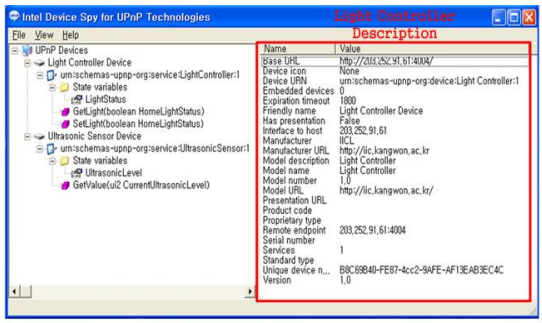
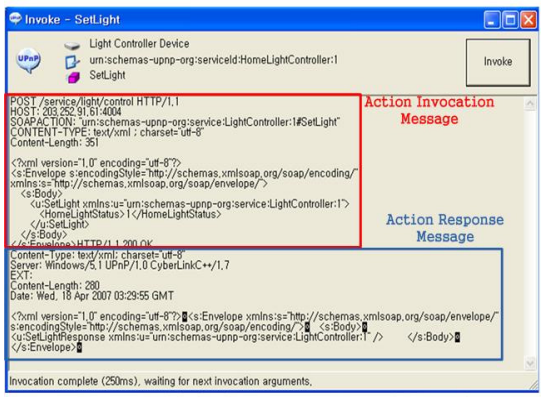


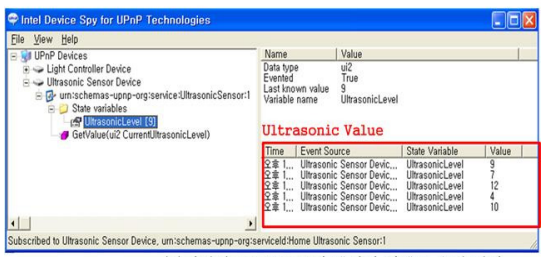
그림 13. 초음파 센서의 패킷 포맷



(a) 가상의 UPnP 전등 제어 장치 기술 화면



(b) 가상의 UPnP 전등 제어장치의 제어 화면



(c)가상의 UPnP 초음파 센서의 이벤트 수신 화면

그림 14. 시나리오에 의한 결과 화면

지에 명령에 대한 응답을 하고 결국 UPnP 컨트롤 포인트는 가상의 UPnP 전등 제어 장치로부터 명령에 대한 응답을 수신한다. 그림 14b는 위의 명령에 대한 요청과 응답 메시지를 보여준다. 실제 Non-IP 초음파 센서 장치가 초음파 센싱 값에 대한 변화를 UPnP 브리지에 통보하면 가상의 UPnP 초음파 센서는 서비스에 구독을 신청한 모든 UPnP 컨트롤 포인트에 이를 알린다. 그림 14c는 UPnP 컨트롤 포인트에서 통보 받은 Non-IP 초음파 센서의 초음파 센싱 값이다.

위의 시나리오에서 보듯이 UPnP 컨트롤 포인트는 일반적인 UPnP 장치를 제어하듯 투명성 있게 Non-IP 기반의 장치들을 제어한다.

4.2 평가

기존의 브리지는 사용자가 개발한 바이너리 메시지 변환기를 통해 일대일 바이너리 형태의 메시지 변환을 제공한다. 따라서 추가적으로 새로운 프로토콜 혹은 메시지가 추가될 때 마다 새로운 바이너리 형태의 메시지 변환기를 개발하여 추가하여야 하고, 이러한 변환기를 설치할 때마다 브리지의 동작을 중지해야 하는 단점이 있다. 그러나 제안하는 브리지는 바이너리 메시지 변환기를 새롭게 개발할 필요가 없이 메시지 변환을 위한 XML기반 메시지 변환 문서를 제공하면 제안된 구조에서 해당 메시지 변환 문서를 참조하여 알맞게 메시지를 변환하여 주기 때문에 프로그램을 개발할 필요도 없고 동작 시에도 브리지를 중지할 필요도 없다. 제안하는 새로운 형태의 메시지 변환 방식은 기존의 방식보다 시간이 약간 더 소요된다는 단점이 있지만, 이는 빠른 CPU를 사용하고 변환 방식을 좀 더 효율적으로 하면 차이는 줄어들 것으로 예상된다.

제안된 브리지의 동작 검증은 그림 14에 의해 보여 주었다. 그림 11과 그림 12와 같은 XML 기반의 메시지 변환 관련 문서를 작성하여 동작시킴으로써 그림 13의 초음파 센서 패킷 관련된 메시지 변환이 이루어질 수 있음을 그림 14c에서 보여 주고 있다.

V. 결 론

다양한 네트워크 인터페이스를 요구하는 분산 컴퓨팅 환경에서 UPnP의 기능을 증대시키기 위해서는 Non-IP 장치와 상호연동은 필수적이다.

본 논문은 RS232C, CAN 인터페이스들과 연결된 다양한 메시지 프로토콜을 가진 일반적인 Non-IP 장치들과 UPnP 장치간의 상호 연동을 위한 새로운 UPnP 브리지 구조를 제안하였다. Non-IP 장치의 정보를 표현한 Non-IP 장치 기술(記述)과 다양한 메시지 프로토콜을 표현하는 메시지 필드 기술(記述) 그리고 UPnP와 Non-IP 네트워크간 제어 명령 및 상태 정보를 매핑 시키기 위한 확장된 UPnP 서비스 기술(記述)을 정의하였다. 위의 세 기술(記述)들을 이용하여 자동으로 Non-IP 장치 메시지를 직렬화/역직렬화 하고 UPnP 메시지와 상호 변환이 가능하도록 하는 텍스트 메시지 포맷기와 텍스트 메시지 변환기를 제공함으로써 브리지 내에 추가적인 실행 프로그램이 없이 쉽게 동적으로 UPnP와 상호 연동 할 수 있음을 보여주었다. 실제

로 RS232C와 CAN 인터페이스 장치를 위한 브리지 모듈과 텍스트 메시지 포맷기와 변환기를 구현하여 UPnP 네트워크상의 장치간 여러 동작을 수행함으로써 제안한 방법의 유효성을 보여 주었다.

앞으로 텍스트 및 바이너리 메시지 변환기에 대한 메시지 변환 시간 등의 성능을 분석할 필요가 있다. 또한 현재는 RS232C, CAN 상에서 각 한 종류의 메시지 형식에 대해 검증을 하였지만 다양한 네트워크 인터페이스와 메시지 형식에 대해 검증할 필요가 있다.

참 고 문 헌

〔1〕 UPnP Forum, “UPnP Device Archtecture Version 1.0”, 08 Jun 2000, www.upnp.org

〔2〕 Sun-Mi Jun, Nam-Hoo Park, “Controlling non IP Bluetooth devices in UPnP home network”, The 6th Internaltional Conference on Advanced Communication Technology, Vol.2, pp.714-718, 2004

〔3〕 Donghee Kim, Jun Hee Park, Poltavets Yevgen, KyeongDeok Moon, YoungHee Lee, “IEEE1394/UPnP Sortware Bridge”, IEEE Transactions on Consumer Electronics, Vol.51, Issue 1, pp.319-323, Feb 2005

〔4〕 S. Chemishkian, J.Lund, “Experimental Bridge LonWork/UPnP”, CCNC Conference, pp.400-405, Jan 2004

〔5〕 Miller.B.A, Nixon.T, Tia.C, Wood.M.D, “Home networking with Universal Plug and Play”, Communications Magazine IEEE, Vol.39, Issue 12, pp.104-109, Dec 2001

〔6〕 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”, IEEE Transactions on Consumer Electronics, Vol.48, Issue 4, pp.963-972, Nov 2002

〔7〕 Sang Chul Ahn, Jin Hak Kim, Kiwoong Lim, Heedong Ko, Yong-Moo Kwon, Hyoung-Gon Kim, “UPnP Approach for Robot Middleware”, Robotics and Automation, ICRA 2005, pp.1959-1963, April 2005

〔8〕 BOSCH, “CAN Specification Version 2.0”, April 5 1995, www.can-cia.org

〔9〕 IEEE1394 TA, “IEEE std 1394a-2000”, 30 March 2000, www.1394ta.org,

〔10〕 USB Forum, “Universal Serial Bus Specification Version 2.0”, April 27 2000, www.usb.org

〔11〕 www.cybergarage.org/net/upnp/cc/index.html

〔12〕 J. Allard, V.Chinta, S.Gundala, G.G.Richard III, “Jini Meets UPnP: An Architecture for Jini/UPnP Interoperability”, Application and the Internet 2003, pp.268-275, Jan 2003

강 정 석 (Jeong-Seok Kang) 준회원

2006년 8월 강원대학교 전기전자정보통신공학부 졸업

2006년 9월~현재 강원대학교 전자통신공학 전공(석사과정)

<관심분야> 미들웨어, 분산 컴퓨팅

최 용 순 (Yong-Soon Choi) 준회원

2006년 2월 강원대학교 전기전자정보통신공학부 졸업

2006년 3월~현재 강원대학교 전자통신공학 전공(석사과정)

<관심분야> UPnP 네트워크, 웹 개발 프레임워크, 무선 센서 네트워크

박 홍 성 (Hong-Seong Park) 정회원

1983년 2월 서울대학교 제어계측공학과 졸업

1986년 2월 서울대학교 제어계측 전공(석사)

1992년 2월 서울대학교 제어계측 전공(박사)

1992년~현재 강원대학교 전기전자정보통신공 학부 교수

<관심분야> 실시간 통신, 무선데이터통신, 미들웨어