UPnP 브리지를 위한 범용 MFD 구조

최용순, 강정석, 박홍성
강원대학교 IT 특성화대학 전자통신공학과

General MFD Structure for UPnP Bridge

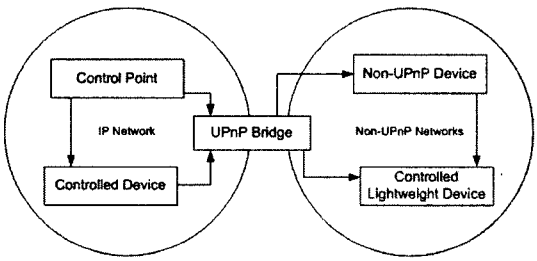
Choi Yong-soon, Kang Jung-seok, Park Hong-seong
Kangwon National University Dept. of Electronic and Communication Engineering

Abstract - UPnP Bridge supporting diverse network interface has to meet standard requirements in order to be connected with legacy devices. It is able to provide or bridge a service description and device description according to a specification because IEEE1394 and USB have this standard requirements. But it is difficult to know whether it RS232C supports only serial communication and packet transfer. It need a document for the standard definition of communication protocol on UPnP device having such interface. By doing so, this document can understand device and packet type. This paper defines MFD(Message Field Description) and makes UPnP message converter. So it will be base to standardize supporting variable legacy device.

1. 서 론

현재의 홈 네트워크 미들웨어의 표준화 방향은 가전 및 장비 제조업체의 공통적인 제어 및 통신 프로토콜의 정의와 상호 운용성의 확대이다. 이를 위하여 JINI, OSGi 진영, 그리고 MS와 인텔, 3COM 등이 주축이 된 UPnP 등 다양한 표준이 제시되고 있으며 이 중 UPnP(Universal Plug and Play)는 수많은 회원사를 거느리고 있으며 DLNA(Digital Living Network Alliance)에서도 기본으로 채택되어 정보 가전 통신 미들웨어의 표준으로 각광받고 있다. UPnP는 분산, 개방형 네트워킹 구조로서 TCP/IP 기반의 HTTP 프로토콜을 사용한다. 이는 인터넷 환경에서 상호 운용성과 확장성에서 강점을 지니고 있으며 IP기반 하에서 플랫폼 독립성과 사용편의성을 지니고 있다.[2]

UPnP는 TCP/IP, HTTP, XML과 같은 개방형 프로토콜을 사용하므로 편리하게 다른 네트워크 및 장치와 연결된다.[3] 하지만 IP 네트워크를 사용하지 않는 네트워크에서는 사용할 수 없다. 따라서 브리지 또는 프록시를 통해야만 이기종 네트워크와 연결될 수 있다. <그림 1>



<그림 1> 브리지를 사용한 UPnP 네트워크의 예

UPnP 브리지는 UPnP 네트워크를 Non-UPnP 네트워크까지 확장시키며 이러한 노력은 모든 장치를 하나의 프로토콜을 통해서 통신할 수 있는 기반을 마련해 준다. 하지만 기존의 장치(Legacy Device)는 대부분 표준화된 프로토콜에 의하여 제작되어 있지 않아 현존하는 미들웨어와 순조로운 연결이 어렵다. 따라서 프로토콜을 이해하기 위한 XML(eXtensible Markup Language) 포맷을 정의한다면 브리지가 이를 해석하여 UPnP 네트워크에 서비스함으로써 기존의 장치를 UPnP 네트워크에 참여시킬 수 있는 기반이 될 것이다.

본 논문에서는 이를 위하여 MFD(Message Format Description)을 XML로 정의하고 레거시 디바이스의 프로토콜과 종류 및 유형, 그리고 데이터의 타입 등을 기술하여 다양한 레거시 디바이스를 지원하고 브리지와의 연결을 표준화할 수 있는 방법을 제시하였다. 특히, CAN 또는 RS232C와 같이 디바이스 모델이 존재하지 않는 경우에 유용하게 쓰일 수 있다.

논문의 내용은 다음과 같이 구성되어 있다. 2장에서 UPnP 미들웨어 기술에 관하여 살펴보고 3장에서 UPnP 브리지의 기능과 구조에 관해 알아볼 것이다. 그리고 4장에서 MFD(Message Field Description)의 설계와 파서에 설명하고 앞으로의 개선 방향에 대해 논의할 것이다.

2. UPnP 미들웨어

UPnP는 MS에서 1999년에 제안하였으며 UPnP Device Architecture Ver 1.0(08 Jun 2000)에서는 이에 관하여 각종 가전기기 및 무선 장치, 그리고 PC 또는 미디어 장치 등을 사용하기 쉽고 유연하게 연결하는데 필요한 표준이다. 이를 위하여 인터넷 표준프로토콜인 TCP/IP를 기반으로 하여 IP, TCP, UDP, HTTP, and XML 등의 인터넷 프로토콜과 이를 이용한 SSDP (Simple Service Discovery Protocol), GENA (Generic Event Notification Architecture), 그리고 SOAP (Simple Object Access Protocol) 을 통하여 장치의 Addressing, Discovery, Description, Control, Eventing, Presentation 등의 기능을 제공한다. 이에 따른 개방형 분산 네트워크의 구조는 특정 운 Device Description에 의해 필요한 기능들이 정해지므로 각 벤더는 자유롭게 구현 및 개발할 수 있다.

UPnP의 기본적인 구성요소는 Device(Controlled Point)와 그에 따른 Service, 그리고 CP(Control Point)이다.

Device는 Service 및 Embedded Device로 구성될 수 있으며 각 Embedded Device는 Device와 동일한 기능을 가질 수 있다. Service는 UPnP 네트워크의 최소 제어 단위로서 하나의 동작(Action)에 대해 관련된 여러 개의 상태 변수(State Variable)를 가진다.

CP는 Device를 제어하는 주체로서 UPnP 네트워크의 장치를 검색하고 서비스의 목록을 가져올 수 있다. 또한 이벤트를 등록하여 서비스의 상태가 변경될 때마다 정보를 수신할 수 있다.

UPnP의 네트워킹은 각 디바이스의 주소를 검색 또는 가져오는 Addressing, 디바이스의 서비스와 정보를 가져오는 Discovery, SOAP메시지를 통하여 실제 디바이스를 제어하는 Control, 상태변수를 모니터링할 수 있는 Eventing과 외부 HTML브라우저를 통한 인터페이스를 제공하는 Presentation의 5단계로 이루어진다.

이렇게 UPnP는 광범위한 범용성과 유연성을 바탕으로 업계의 많은 지원을 받고 있고 다양한 플랫폼에서 사용 가능하기 때문에 지원하는 장치가 늘어나고 있다. 따라서 이러한 UPnP 네트워크를 이기종 네트워크와 통합하고 Small Form Factor 장치와 연결하기 위한 UPnP브리지의 개발이 이루어지고 있다.

3. UPnP 브리지

UPnP 브리지는 기본적으로 UPnP 네트워크를 포함한다. 그리고 다른 미들웨어(JINI, HAVi, OSGi 등)와의 연동을 하거나 이기종 네트워크(RS232C, CAN, IEEE1394 등)로 연결된 장치들에 대한 UPnP 브릿징 기능을 제공한다.

이를 위해서 브리지는 다양한 모듈로 이루어져 있으며 이러한 모듈은 Device Manager에 의해 동적으로 생성되는 **Virtual UPnP Device**과 디스크립션 파일과 함께 Message Converter에 대한 기본 설정을 하여주는 **Device Manager**, 그리고 네트워크 추상화 계층을 구현하는 **NIAL(Network Interface Abstraction Layer)**과 실제로 MFD를 파싱하여 필요한 메시지 타입으로 번역하는 **Message Converter**로 구성된다. <그림 2>

현재 구현된 브리지의 프로토타입은 UPnP Library로 CyberLink Open source SDK를 사용하고 있다. [8] CyberLink 라이브러리는 UPnP 개발자를 위한 유틸리티 클래스와 함께 UPnP 프로토콜 패키지 및 XML Wrapper를 담고 있으며 빠르게 UPnP 관련 애플리케이션을 만들 수 있게 하여 준다. 본 논문에서는 이 라이브러리의 Device 클래스를 상속하여 브리지 구조에 따른 모듈을 포함한 Bridge Device를 구현하였다. 또한 XML라이브러리로는 Expat 2.0을 사용하였다.

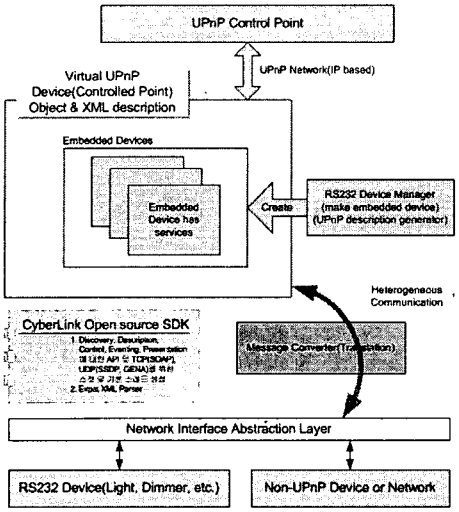
4. MFD(Message Field Description)

이기종의 네트워크에 연결된 장치는 일반적으로 특정할 수가 없다. IEEE 1394 또는 USB의 경우 디바이스 모델에 따라서 구분할 수 있지만 RS232C 또는 CAN의 경우 레거시 디바이스에 대한 지원이 필수적이다. 이를 지원하기 위해서는 장치의 통신 프로토콜에 따른 일종의 정의 파일이 필요하다.

본 논문은 이러한 파일을 XML 스키마로 정의하고 몇 가지 장치에 대해서 예제 MFD를 만들고 CP와 효율적으로 상호작용하는 모습을 보였다.

4.1 MFD의 정의

MFD는 Non-UPnP Device와 브리지가 효율적으로 통신하기 위한 필드 정의 XML 파일로 프로토콜 규약서라고 할 수 있다. MFD의 요소는 다음



<그림 2> UPnP Bridge Prototype

과 표 1과 같다.

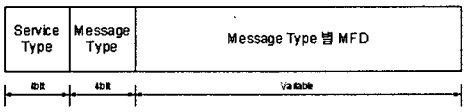
<표 1> MFD 요소

요소 설명	속 성	비 고
메시지 필드 정의	Mandatory	타입, 순서, 길이 포함, 필수 여부
메시지 타입 정의	Mandatory	Action, Query, Event, etc.
메시지 흐름 정의	Mandatory	Request, Response
데이터 타입 정의	Mandatory	ui1, ui2, ui4, boolean, string 등의 데이터 타입 정의
메시지 에러 체크	Optional	

4.2 MFD General Packet Header Type

Message Converter가 레거시 디바이스를 지원하기 위해서는 일정한 형식의 패킷 형식을 파악하는 것이 중요하다. 하지만 이에 따른 임의의 패킷 형식과 페이로드의 의미를 알기는 어렵다. 그러므로 본 논문에서는 최소한의 일반적인 패킷 헤더 타입을 정의하고 이에 따른 메시지 타입별 MFD를 구성하였다.

그림 3와 같은 기본적인 패킷 헤더 타입을 따르지 않는다면 브리지가 장치를 이해하지 못할 수도 있다. 만일 이러한 형식을 지키지 않는 레거시 디바이스를 지원하려면 Message Converter와 장치 사이에 브로커(Broker)가 필요하다.



<그림 3> MFD General Packet Header Type

4.3 MFD(Message Field Description) Architecture

MFD는 크게 패킷 형식에 대한 정보와 필드 구조 정의로 구성되어 있다.

4.3.1 Packet Information

패킷 정보는 가변 길이 또는 고정 길이 패킷의 패킷 타입, 고정 길이의 경우 전체 패킷의 크기, 그리고 이기종 플랫폼의 바이트 배열 정보(Little Endian, Big Endian)를 포함한다. UPnP 브리지에서 기본적으로 사용되어지는 메시지의 타입을 정의하며 예제 MFD에서는 13개의 메시지 타입으로 구분하였다. 또한 메시지의 페이로드에 들어갈 값의 Argument Type도 정의한다.

4.3.2 Packet Structure

Device Description의 서비스 ID에 따른 서비스 타입과 UPnP 네트워크에서 필요한 정보에 따라 13개의 메시지 타입으로 구분하였다. 이에 따른 세부적인 정의는 MFD Example(그림 4)에서 볼 수 있다. 서비스 타입과 액션을 위한 ID는 기존의 디스크립션에 ID를 추가하는 확장 디스크립션 파일을 정의하여 매핑 된다.

4.5.3 Non-UPnP Device Example

예제로 정의한 MFD 파일을 테스트하기 위하여 시리얼 통신 프로그램을 만들고 디바이스 디스크립션과 서비스 디스크립션, 그리고 패킷 포맷을 정의하였다.

```
<Packet>
  <Name>ActionRequest</Name>
  <!-- Packet Type is "Fixed" or "Variable" -->
  <PacketType>Fixed</PacketType>
- <PayloadList>
- <Payload>
    <Name>MessageType</Name>
    <PayloadType>TYPE_FIELD</PayloadType>
    <SizeUnit>bit</SizeUnit>
    <Size>4</Size>
  </Payload>
- <Payload>
    <Name>ServiceID</Name>
    <PayloadType>FIXED</PayloadType>
    <Value>0x01</Value>
    <SizeUnit>bit</SizeUnit>
    <Size>4</Size>
  </Payload>
- <Payload>
    <Name>ActionID</Name>
    <PayloadType>UPnP_ACTIONID</PayloadType>
    <SizeUnit>bit</SizeUnit>
    <Size>4</Size>
  </Payload>
- <Payload>
    <Name>Reserved</Name>
    <PayloadType>RESERVED</PayloadType>
    <SizeUnit>bit</SizeUnit>
    <Size>4</Size>
  </Payload>
- <Payload>
    <Name>Argument Data</Name>
    <PayloadType>UPnP_PAYLOADDATA_UI1</PayloadType>
    <SizeUnit>byte</SizeUnit>
    <Size>1</Size>
  </Payload>
 </PayloadList>
</Packet>
```

<그림 4> Message Field Structure

MFD는 벤더(Vendor)가 디바이스를 제작할 때 프로토콜의 표준을 지켜야 하는 부담을 덜어준다. 브리지는 벤더의 Device Description과 Service Description을 MFD에 따라 번역하고 UPnP 네트워크와 통신하게 하여 준다.이는 기존의 레거시 디바이스에 대한 연동성을 부여하는 기반이 될 수 있다.

5. 결 론

지금까지 UPnP 네트워크에서 이기종 네트워크와의 연동을 위한 브리지 구조와 레거시 디바이스를 통합하기 위한 MFD 구조에 대해 기술하였다.

앞으로 홈 네트워크를 포함한 다양한 분야의 장치를 만드는 벤더는 여러 미들웨어의 표준과 통신 매체 사이에서 상호 운용성을 확보해야 하는 어려움이 있다. 이를 위해서는 브리지를 통한 이기종 네트워크 인터페이스의 연동과 장치의 통합이 필수적인 조건이 될 것이다.

추후, MFD의 설계는 다양한 레거시 디바이스의 지원이라는 방향으로 수정되고 보완될 것이다. 또한 브리지 구조도 다양한 미들웨어를 통합할 수 있도록 확장되어질 수 있다.

[참 고 문 헌]

[1] Donghee Kim, Jun Hee Park, Yevgen, P. KyeongDeok Moon, YoungHee Lee "IEEE 1394/UPnP software bridge", Consumer Electronics, IEEE Transactions on. Feb 2005

[2] Microsoft Corporation, "Understanding Universal Plug and Play", http://www.upnp.org/download/UPnP_UnderstandingUPnP.doc,, Jun. 2000

[3] Microsoft Corporation, "Understanding Universal Plug and Play Device Architecture", http://www.upnp.org/download/UPnPDA10_20000613.htm, Jun. 8, 2000

[4] Microsoft Corporation, "Understanding Universal white paper, Microsoft Corp., 2000.

[5] Windows XP Professional 기술정보, "Windows XP의 범용 플러그 앤 플레이 기능"http://www.microsoft.com/korea/windowsxp/pro/techinfo/planning/upnp/protocol.asp

[6] "UPnP(Universal Plug and Play) 기술 분석", 전자부품 연구원 전자정보센터(www.eic.re.kr)

[7] "차세대 Network 접속 규격 UPnP", 전태수, 미래기술팀, 삼성 SDS IT Review, 2000-09

[8] Cyber Link UPnP Libraryhttp://www.cybergarage.org/net/upnp/cc/index.htmlCyberLink for C++

[9] Allard, J., Chinta, V., Gundala, S., Richard, G.G. III, "Jini meets UPnP: an architecture for Jini/UPnP interoperability" , Applications and the Internet, 2003. Proceedings. 2003 Symposium on, 2003

[10] Aitor Almeida Escondrillas, David Sainz Gonzalez "Enabling Service Orchestration, Transactionality and Security in UPnP" , IADIS International Conference on Applied Computing. San Sebastian, Spain. Febrauary, 2006

[11] Juan Ignacio Vazquez and Diego Lopez de Ipina. "Empowering Wireless UPnP Devices with WebProfiles", 10th IFIP International Conference on Personal Wireless Communications. Colmar, France. August 2005.

[12] T.M. Tran, P.J.F. Peters, J.J. Lukkien, P.H.F.M. Verhoeven; Controlling networked devices: a validation of two middleware architectures; Proceedings of the 3rd workshop on Embedded Systems, STW, Oct. 2002