UPnP IPv4/IPv6 Bridge for Home Networking Environment

Chung-Sheng Li, Yueh-Min Huang, Member, IEEE and Han-Chieh Chao, Senior Member, IEEE

Abstract — The purposes of Universal Plug and Play (UPnP) are to allow consumer electronics, intelligent appliances (IA) and mobile devices from different vendors to be internet connected seamlessly and installed easily at home. Also, as the IPv6 environment becomes inevitable, IPv6 ready IA are more and more popular. However, the fact is that IPv4 UPnP control points cannot communicate with IPv6 UPnP devices or vice versa. In this paper, we proposed an UPnP IPv4/IPv6 bridge for control points and devices can interoperate between IPv4 and IPv6 protocols; that is, they can communicate with each other although some of them support only IPv4 and others support IPv6.

Index Terms — UPnP, IPv4, IPv6, Bridge

I. INTRODUCTION

IP technology has already been prevailing even at home; consequently, more and more applications for home network via Internet continue being studied and implemented for making a better life for human being. However, how to integrate those applications from different vendors into a whole system needs a common protocol for devices to interact with each other. Subsequently, some service discovery protocol such as UPnP, Jini, SLP, Salutation and Ninja, etc. have already been proposed to cope with this problem. In specific, these protocols are designed for devices to talk with each other in its own homogeneous domain. In comparison with other protocols, UPnP emphases on using existed technologies such as IP, TCP, UDP, HTTP, and XML; that is, a vendor can implement an UPnP device by using any programming language, and on any operation system. The advantage of UPnP is no need to install any special driver. Furthermore, OS vendors can produce APIs that accommodate the requirements which are from various electronic appliance manufactures. Accordingly, customers can operate these devices by using UPnP controllers, although these devices are from different vendors.

Basically, UPnP network is a service discovery framework with a collection of protocols for dynamic client/server applications. With advertisement of availability from a service, clients could search, fetch and use needed services. In a service discovery-enabled network, devices that are plugged in become part of the community and may be discovered. Users could access those services with minimum manual configuration. To be more exact, they are designed to support zero-configuration, invisible network connectivity, and automatic discovery. Some researches for UPnP innerrability are available in the literatures. [3] focuses on the interoperability between JINI and UPnP. [4] presents the mechanism of interoperability of IEEE 1394 and UPnP. However, if both IPv4 and IPv6 UPnP are supported as shown in Fig. 1, it results that an IPv4 control point cannot communicate with an IPv6 device or vice versa. If customers have IPv4 UPnP devices and IPv6 controllers, they will consider that these devices are useless. For solving the problem, we proposed UPnP IPv4/IPv6 bridge to create a real Universal Plug and Play environment no matter the customers’ devices are in IPv4 or IPv6 environment.

This paper is organized as follows: section 2 presents UPnP networking, section 3 describes the basic functions of the bridge, section 4 describes the design of the bridge and the message flows, section 5 gives the experiment results, and finally section 6 makes conclusions of this paper.

![Fig. 1 UPnP home network scenario](image)

II. UPnP NETWORKING

The UPnP architecture offers pervasive peer-to-peer network connectivity of PCs, intelligent appliances, and wireless devices. It is well defined protocol targeting to home networks, and small businesses. Following, we briefly illustrate UPnP behavior.
UPnP aims to enable the resource discovery, service control, and consumer electronics. In UPnP, a device can dynamically join a network, obtain an IP address, convey its capabilities on request, and get the presence and capabilities of other devices. A device can also leave a network with zero configurations.

UPnP leverages current networking technologies such as HTTP, SSDP, SOAP and, GENA over TCP/UDP and IP. It uses the protocol stack as shown in Fig. 2 for service discovery, advertisement, description, and event.

![UPnP protocol stack](image)

Basically, the UPnP framework is operated as the following steps:

- **Discovery**
  In this first step, control points search for devices and services by multicasting M-SEARCH; therefore, devices multicast NOTIFY of services they offer. In IPv4, the multicast address is 239.255.255.250 port 1900; meanwhile in IPv6, the multicast address is FF02::C port 1900. The sample messages for IPv4 and IPv6 are shown in Fig. 3 and Fig. 4 respectively.

- **Description**
  After a control point has received NOTIFY of an UPnP device, it can fetch the description document through the URL in LOCATION header as shown in Fig. 4. It is device’s XML-format description (APPENDIX I) which contains data items for a device such as friendly name, manufacturer, and model name, etc. At this moment, the control point still has to get service XMLs (APPENDIX II) from SCDPURL in service list, because they contain data definitions such as action names, arguments and service stable variables. The data definitions in services can be used to send commands to controlURL to control the device or get current state values in the device.

- **Control**
  The controlURL of a service can receive action Invoke or query Invoke which are defined in Simple Object Access Protocol (SOAP) protocol from a control point. After processed, the device will send the results (action Response, query Response) to the control point.

- **Event Notification**
  The eventSubURL of a service can receive SUBSCRIBE event as shown in Fig. 5 from a control point. If the device accepts the subscription, it will return a 200 OK to the control point as shown in Fig. 6. In this message, SID is a unique code to identify the subscription. Thus, when any change in the subscription, the device will send NOTIFY event back to the CALLBACK url of the control point. After the control point parsing the event, it will have the up-to-date data.

- **Presentation**
  After all, if a device has an URL for presentation, it is a HTML-based user interface for control points to retrieve a page from this URL into a browser. Depending on the capabilities of the page, a user can control the device and/or view device status.

  Basically, a device is a simple web server with a directory as shown in Fig. 7. After the device multicasts baseURL, the control can retrieve device/service descriptions to parse them for the correspondent URLs to interact with the device.
III. THE BASIC FUNCTIONS OF THE BRIDGE

For relaying messages between IPv4 and IPv6 UPnP networks, the bridge must double as an IPv4 control points and bridged IPv6 devices or an IPv6 control points and bridged IPv4 devices. In specific, when the bridge acts as an IPv4 control point, it can learn IPv4 devices which can be converted bridged IPv6 devices to support these three basic processes, which are discovery, control and event. For convenient describing, we presume the bridge performing an IPv4 control point and bridged IPv6 devices because the process is similar while the bridge is an IPv6 control point and bridged IPv4 devices.

In discovery process, the bridge can learn IPv4 devices and services and expose them to IPv6 network. So the UPnP control point in IPv6 network can be aware of those bridged devices. Next, IPv6 UPnP control points can send SOAP messages to those bridged UPnP devices. Then, the bridge can forward them to real IPv4 UPnP devices. After execution, responses can be sent back to the IPv6 control point. Furthermore, IPv6 UPnP control points are supposed to send subscribe messages and wait for events from bridged devices. The bridge must receive these subscriptions and send events back to IPv6 UPnP control points whenever the states of IPv4 UPnP devices have changed.

Finally, IPv4 UPnP control points may request the content of manufacture, model or presentation. The bridge should have the capability to proxy these data items from IPv6 UPnP devices.

IV. DESIGN OF THE UPnP IPv4/IPv6 BRIDGE

A. Structure and Components of the bridge

The goal of an UPnP IPv4/IPv6 bridge is for an IPv4 control points to be able to operate the IPv6 device or vice versa. Therefore, we design an IPv4/IPv6 UPnP bridge which can communicate with the IPv4 UPnP network and IPv6 network. The bridge must satisfy the requirements described in section 3 as shown in Fig. 8(a)(b). It has two directions; that is, Fig. 8 (a) is for IPv4-to-IPv6 and Fig. 8 (b) is for IPv6-to-IPv4. Also, control point layer and UPnP API layer is used by data modules to communicate with IPv4 network and IPv6 network. Device module is a composite module which includes UPnP device and service, embedded device module, and URL adapter. This module is instantiated for a correspondent real UPnP device. On the other hand, while the device manager detects a new real UPnP device, it creates a related device module to expose in the other IP domain.

The following is the detailed descriptions of bridge in Fig. 8(a)(b).

IPv4/IPv6 control point layer is a general control point to collect real IPv4/IPv6 UPnP devices for bridging. It has a device manager to save the definition of UPnP devices and instantiate a data module for bridging.

IPv4/IPv6 UPnP API Layer is an API for UPnP stack. We developed our own UPnP API layer which can support IPv4 or IPv6 network.

UPnP Device is the interface for control points to retrieve device/service definitions; that is, they are descriptions in discovery step. It also provides the presentation of each device module.

UPnP Services provides the interface for control points to send control and event commands. When one of them got a command, it could send to URL adapter and then down to the control point to operate real UPnP device. The response also can be returned following the opposite direction.

Embedded Device Module is the interface for embedded devices. It is linked to the root device module and provides the functions of embedded devices.
**IPv4-to-IPv6/IPv6-to-IPv4 URL Adapter** – is the adapter to convert IPv4 address to IPv6 address or IPv6 address to IPv4 address and keep the relation between a device module and a real device. For example, when it gets an IPv4 control message, it can find the correspondent IPv6 service for control points to send this control message to the real device.

**B. Working Scenarios and Message Flows**

We took an IPv6-to-IPv4 bridge as an example and the message flows for initialization of a device module as shown in Fig. 9. Before creating a new data module, the control point learns UPnP devices in IPv6 networks. The next step forwards the device/service descriptions to URL adapter. The URL adapter will convert URL for new HTTP server and keep link to the original device/service. Then, UPnP device/service will create an entry for a new device. Finally, the description is multicast in IPv4 network. Once an IPv4 UPnP control point detects the message, it can complete UPnP networking steps to control an IPv6 device through a bridged IPv4 device.

**IV. EXPERIMENT RESULT**

We used Intel Device Spy, Intel AV Media Controller, Intel AV Renderer, MediaGate [2] to verify our implementation of IPv4/IPv6 UPnP bridge. Intel Device Spy is a universal UPnP control device, but it only supports IPv4 network and is used to verify functionalities of bridged IPv6 devices. Inter AV Controller is an explicit control point for UPnP Media Server and Renderer. CyberMediaGate is an UPnP A/V media server which supports IPv6 networking.

If without bridge, Intel AV Controller can only reach Intel AV Renderer. After the bridge finished the initialization, Intel AV controller will add bridged CyberMediaGate into its Media Server device list.

**APPENDIX**

I. Device XML description sample

```xml
<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <URLBase>base URL for all relative URLs</URLBase>
  <device>
    <deviceType>urn:schemas-upnp-org:device:deviceType:v</deviceType>
    <friendlyName>short user-friendly title</friendlyName>
    <manufacturer>manufacturer name</manufacturer>
    <manufacturerURL>URL to manufacturer site</manufacturerURL>
    <modelDescription>long user-friendly title</modelDescription>
    <modelName>model name</modelName>
    <modelNumber>model number</modelNumber>
    <modelURL>URL to model site</modelURL>
    <serialNumber>manufacturer's serial number</serialNumber>
    <UDN>uuid:UUID</UDN>
    <UPC>Universal Product Code</UPC>
  </device>
</root>
```

The AV controller can invoke an action to AV Media Renderer and the Renderer interoperates with AV Media Server through the IPv4/IPv6 bridge. In sum, the whole message flows is shown in Fig. 10.

**IX. CONCLUSION**

UPnP is designed for home networking, so its goal is easy to operate for a common user. Since UPnP standard can support both of IPv4 and IPv6 networking protocols, it is not convenient when a common user has equipments with two kinds of IP protocol. In the paper, we design an IPv4/IPv6 UPnP bridge to let control points and devices be able to interoperate between IPv4 network and IPv6 network easily. Finally, a true UPnP inerrability networking environment can be created for home network.
II. Service XML description sample

<?xml version="1.0"?>
<scdp xmlns="urn:schemas-upnp-org:service-1-0"/>
<specVersion>
    <major>1</major>
    <minor>0</minor>
</specVersion>
<actionList>
    <action>
        <name>serviceName</name>
        <argument>
            <name>formalParameterName</name>
            <direction>in xor out</direction>
            <retval />
            <relatedStateVariable>stateVariableName</relatedStateVariable>
        </argument>
        <relatedStateVariable>stateVariableName</relatedStateVariable>
    </action>
</serviceStateTable>
</scdp>

Acknowledgment

Authors would like to express their gratitude to Dr. Wei-Ming Chen, Mr. Andrew Chang and Mr. Louk Chi for their kind assistance in preparation of this work.

REFERENCES


AUTHORS

Chung-Sheng Li graduated in June, 1994 from the Institute of Information Engineering National Cheng-Kung University, Taiwan. He is currently a PhD student of Department of Electrical Engineering National Dong-Hwa University. His research interest includes IPv6, Wireless Network, Mobile Computing and Network Modeling and Simulation etc.

Yueh-Min Huang is a Chair Professor and Chair of the Department of Engineering Science, National Cheng-Kung University, Taiwan, R.O.C. His research interests include Multimedia Communications, Wireless Networks, Embedded Systems, and Artificial Intelligence. He received his MS and Ph.D. degrees in Electrical Engineering from the University of Arizona in 1988 and 1991 respectively. Dr. Huang is in the editorial board of the Journal of Internet Technology, International Journal of Internet Protocol Technology, International Journal of Ad Hoc and Ubiquitous Computing. Dr. Huang is a member of the IEEE and IEEE communication and computational intelligence societies

Han-Chieh Chao is a jointly appointed Professor of the Department of Electronic Engineering and Institute of Computer Science & Information Engineering, National Ilan University, I-Lan, Taiwan. He also holds a joint professorship of the Department of Electrical Engineering, National Dong Hwa University, Hualien, Taiwan. His research interests include High Speed Networks, Wireless Networks and IPv6 based Networks and Applications. He received his MS and Ph.D. degrees in Electrical Engineering from Purdue University in 1989 and 1993 respectively. Dr. Chao is also serving as an IPv6 Steering Committee member and Deputy Director of R&D division of the NICI Taiwan, Co-chair of the Technical Area for IPv6 Forum Taiwan. Dr. Chao is an IEEE senior member, IET and BCS Fellows.