Design of DPWS Adaptor for Interoperability between Web Services and DPWS in Web Services on Universal Networks

Hyung-Jun Yim, Il-Jin Oh, Yun-Young Hwang, Kyu-Chul Lee
Department of Computer Engineering, Chungnam National University
{hyungjun25, victory25, yyhwang, kclee}@cnu.ac.kr

Kangchan Lee, Seungyun Lee
Electronics and Telecommunications Research Institute
{chan, syl}@etri.re.kr

Abstract

In ubiquitous computing, clients should be able to use any kinds of services whenever and wherever they are. Service has the same quality without concerned with characteristics such as the status and location of services or devices. Because the characteristics are dynamically changed, a Sub-network is difficult to reflect all the changes. So, there are various Sub-networks. It is necessary to integrate communication between Sub-networks with interoperability. For this reason, a system must support data transmission and conversion. It also has to indicate dynamic detection of the Sub-network service by events and notifications. So, we introduced WSUN environment which is composed of US Broker, US Registry and UniA for using universal service. This paper designs the DPWS Adaptor for providing universal service through operational scenario of DPWS.

Keywords-component: Adaptor, Interoperability, Web Services, DPWS

1. Introduction

The various characteristics which are status and location of services or devices continuously change in ubiquitous computing. It is difficult to indicate all the changes, but a service discovery middleware is able to guarantee the problems which are caused by liquidity of the characteristics. We refer to a service discovery middleware as ‘Sub-network’ in this paper. However there are various Sub-networks. For example, Sun Microsystems’s Java Intelligent Network Infrastructure (JINI) [1], Microsoft’s Universal Plug and Play (UPnP) [2] and Home Audio Video Interoperability (HAVi) [3] and so on. Devices Profile for Web Services (DPWS) [4] is a research which designed for integration of the devices. The Sub-network can reflect the changes of services and devices, but they can’t support interoperability between Sub-networks. Because each one of them uses a different protocol, a specific environment must be constructed. Namely, the client’s device must have every environment for using the services which are supported by all the Sub-networks. For example, the client needs to support Simple Object Access Protocol (SOAP) and IEEE 1394 interface for convergence between UPnP and HAVi.

Recently, to solve the problems, various studies related with Common Layer method appeared. There are Open Services Gateway Initiative (OSGi) [5], Domotics Network (DomoNet) [6] [7] and so on. OSGi technology provides a service-oriented and component-based environment for developers. It also offers standardized ways to manage the software lifecycle. These capabilities greatly increase the value of a wide range of computers and devices that use the Java™ platform. DomoNet, Service-Oriented Architecture (SOA) based, approaches to provide the interoperability between Sub-networks through changing Sub-network services into Web Services.

The client should be able to use any kinds of services, which have the same quality regardless of characteristics. For this reason, we introduced WSUN (Web Services on Universal Networks) environment which is composed of US Broker (Universal Service Broker), US Registry (Universal Service Registry), UniA (Universal Adaptor) and so on. And this paper designs the DPWS Adaptor for the interoperability of
the heterogeneous Sub-networks and the guarantee of universal service discovery.

The following chapter briefly describes an overview of the existing studies for interoperability of the heterogeneous Sub-networks. Chapter 3 introduces WSUN environment and describes its components. Chapter 4 designs DPWS Adaptor through operational process, finally chapter 5 presents the conclusion of this study and further related to this study.

2. Related Work

This chapter gives an overview of the existing studies for interoperability between Sub-networks or devices. The following is the explanation the existing studies.

2.1. Open Services Gateway Initiative

Open Services Gateway Initiative (OSGi) technology provides a service-oriented and component-based environment for developers. It also offers standardized ways to manage the software lifecycle. These capabilities greatly increase the value of a wide range of computers and devices that use the Java™ platform. OSGi supports the connection and control between services, OSGi framework and exterior network through a registry that contains service descriptions published service providers. Services in OSGi, Java classes or interfaces supported, are independent platforms or applications. They also provide various network protocols.

![OSGi Framework's Bundle Architecture](image)

**Figure 1. OSGi Framework's Bundle Architecture**

OSGi framework consists of a System Bundle and Bundle. The Bundle is the smallest component unit of the Bundle framework to use services in OSGi. The System Bundle should be able to constitute a framework on its own and update dynamically. The Bundle consists of the Java ARchive file (JAR) that describes a Bundle and includes the classes of Bundle process or status of the other resources.

A Bundle supports services collaboration and uses services. Clients, however, don’t know the interior process of the Bundle through Java interfaces or control of the other mechanism, as shown in Figure 1.

2.2. Domotics Network

Domotics Network (DomoNet), SOA based framework, approaches to provide the interoperability between Sub-networks. The architectural solution follows a typical approach in computer science contexts and improves the scalability avoiding the one-to-one protocol conversion approach. Like the clients using Web Services, DomoNet supports the clients using Sub-network services through TechManager (TM), called gateway, which takes charge of the conversion that changes the Sub-network services into Web Services. It essentially consists of a network connecting application TMs, and Web Services, each one handling a different device typology. TM exists on each Sub-network. It is efficient to change all services to the Sub-network subordinately.

![DomoNet's TM Architecture](image)

**Figure 2. DomoNet's TM Architecture**

As seen in Figure 2, TM has both real devices and virtual devices. It stores Sub-network service as a form of DeviceWS in the Universal description, discovery and integration (UDDI) registry and generates a virtual device fitting to each Sub-network. Through the virtual device, different Sub-network devices enable the real device to be used as the Sub-network of several environments with a communication device in the DomoNet environment. The clients recognize virtual services as real services, but the clients actually use services which are provided from the Sub-network services.

2.3. Devices Profile for Web Services
Devices Profile for Web Services (DPWS), SOA based framework, is research for the integration of devices. DPWS is an open networking standard that enables industry partners to confidently invest in Web services for devices. DPWS enables network-connected devices to use industry standards-based Web services mechanisms [8]. Devices are connected to the Internet or network which include open interface for independently communication with Web Services and application. When the clients search devices with services that they want through P2P communication, device profiles are used, as shown in Figure 3.

There are various existing studies to make interoperability between Sub-networks. They approach to solve the problems which the existing Sub-network has. However they also have some problems. OSGi technology doesn’t support the way of various service discovery method. Because OSGi platform is Java based, the client has to provide Java Virtual Machine (JVM) environment. If not, the client doesn’t use Sub-network’s services. Besides, the Bundle has a problem in that it always has to reinstall when being changed. DomoNet also has problem in that it doesn’t include service status because it uses only UDDI except the other registry. DomoNet always have to change each service into Web Services for indicating the changes in the status and location of services or devices. Moreover, because it uses DomoML which operates TM, it has a problem which is subordinate to a language. DPWS doesn’t include service status as well as support for general Web Services.

To solve these problems, we propose the WSUN which is composed of US Broker, US Registry, UniA and so on. The next chapter describes the features of these components.

3. Web Services on Universal Networks

Web Services on Universal Networks (WSUN), which is a SOA based, is able to provide the client with universal service through virtual Web Services. Because the WSUN serve virtual Web Services to the client, the client is able to use any kinds of services. The WSUN environment is composed of US Broker, US Registry, UniA and so on, as shown in Figure 4.

The following describes the WSUN components which support these functions.


Universal Service Broker (US Broker) is a core component of WSUN. The US Broker should be able to support the following requirements which are service discovery, registration and management.

- **Client Appearance**: When the client appears at the WSUN, the US Broker must provide the client with service discovery method.
- **Sub-network Hello/Bye**: When the Sub-network Hello/Bye at the WSUN, the US Broker should indicate the changes of services or devices information.
Sub-network Device Hello/Bye: Like Sub-network Hello/Bye, the US Broker indicates the service status by the changes of device location.

WSUN Device Hello/Bye: WSUN device is a WSUN environment supported device. The US Broker should be able to provide WSUN device Hello/Bye.

Service Discovery: When the client wants to discover Sub-network services, the US Broker should be able to deal with client’s inquiry.

The following is the explanation of the US Broker components, as shown in Figure 5.

3.1.1. **Listener.** When the client or the WSUN device appears at the WSUN, they search the US Broker by multicast message. Then, the US Broker offers them basic information for communication each other.

3.1.2. **Publish Agent.** Publish Agent registers device/service information in the Device/Service Registry and Context Registry.

3.1.3. **Query Agent.** Query Agent deals with inquiry of the client’s service discovery. After the Query Agent searches available services using a US Registry search, it sends result lists of search to the client.

3.1.4. **Routing Proxy.** When the client wants to bind the appropriate service, The Routing Proxy connects specific Sub-network service to the client.

### 3.2. Universal Service Registry

Universal Service Registry (US Registry) is composed of Device/Service Registry and Context Registry. They store device/service information and service location for dynamic service discovery and universal service.

3.2.1. **Device/Service Registry.** Device/Service Registry store the WSUN features (device/service information) in Device/Service Registry which are common, static and essential elements. For this reason, we extract the WSUN features from JINI, UPnP and DPWS, as shown in Table 1.

3.2.2. **Context Registry.** Context Registry stores the location of service. In ubiquitous computing, the service location frequently changes. So, the service location is important factor of service discovery.
### Table 1. Device/Service Registry Information

<table>
<thead>
<tr>
<th>Design Metadata</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sub-network Type</td>
<td>Sub-network type for controlling each Sub-network in the US Broker</td>
</tr>
<tr>
<td>Service ID in Sub-network</td>
<td>Service unique ID in the Sub-network</td>
</tr>
<tr>
<td>Service ID in US Broker</td>
<td>Device unique ID for controlling the device of each Sub-network in the US Broker</td>
</tr>
<tr>
<td>Service Name</td>
<td>Service name</td>
</tr>
<tr>
<td>Device Name</td>
<td>Device name</td>
</tr>
<tr>
<td>Category Name</td>
<td>Category information about service</td>
</tr>
<tr>
<td>Service Status</td>
<td>Service status (valid, invalid)</td>
</tr>
<tr>
<td>Device Description</td>
<td>Device Description</td>
</tr>
<tr>
<td>Service Description</td>
<td>Service Description</td>
</tr>
</tbody>
</table>

### 3.3. Universal Adaptor

Universal Adaptor (UniA) connects the US Broker with Sub-networks like broker. And it creates the Sub-network service as the virtual Web Services for supporting universal service discovery.

As seen in Figure 6, UniA components play a role in message transfer, Sub-network services discovery, virtual Web Services creation, event and notifications.

The following is the explanation of UniA components:

- **Message Transfer:** This function takes charge of a message delivery in the Sub-network and US Broker.
- **Discovery Sub-network Services:** Because UniA creates virtual Web Services, it needs to get the device/service information.
- **Creation Virtual Web Services:** To support the client with universal service, UniA should change Sub-network service into virtual Web Services.
- **Events and Notifications:** The UniA indicates the change of the service status for dynamic service discovery.
4. DPWS Adaptor

This chapter, we describe the DPWS Adaptor which is one of the UniA. First of all, we consider necessary of the UniA through real scenario. And we design the DPWS Adaptor using operational scenario of DPWS. Finally, we recognize features as a result of the DPWS Adaptor process.

4.1. Scenario

The client enters the conference room for a meeting. The client uses the services which are necessary for a meeting. The client should be able to search and use services like Web Services because the US Broker supports virtual Web Services. The client searches for an available printer in the conference room and provides materials to the people in attendance. The client receives lists of results on the available projector. Then, the client selects the best result from the lists and makes a progress presentation. If the client wants to show horizontal distribution (HD) videos for a presentation, the client deliver information by means of an LCD TV. The US Broker searches available services using a US Registry search. Although the client doesn’t recognize the US Broker, the client only knows about the Web Services discovery method. Because of that, the client is able to use services without restricting the client’s environment, as shown in Figure 7.

4.2. Operational Process of the DPWS Adaptor

The client should be able to use any kinds of services. For this reason, The DPWS Adaptor plays a role in registry device/service information, creation virtual Web Services, events and notifications.

4.2.1. DPWS Adaptor Hook-up. As seen in Figure 8, the Hook-up process of the DPWS Adaptor.

This sequence diagram describes the sequential process of the DPWS Adaptor Hook-up using 10 steps. From Step 1 to Step 4, the DPWS Adaptor receives the devices and services information of DPWS and it creates virtual Web Services in Step 5. At step 6, the DPWS Adaptor transfers virtual Web Services information to Publish Agent with devices and services of DPWS. Publish Agent registers device/service information to a US Registry in Step 6 and 7, the service location to the Context Registry in Step 8 and 9. Finally in Step 10, the DPWS Adaptor confirms the result message about each registry.

When the DPWS Adaptor Hook-up process is started, the DPWS Adaptor sends multicast Hello message of WS-Discovery to search the DPWS devices. Then, the DPWS devices transfer response message that include UUID, type, scope, and metadata version to the DPWS Adaptor. The DPWS Adaptor obtains the service name, device name, WSDL, and service location for storing it in the Device/Service Registry. After it creates virtual Web Services, it sends data which includes UUID, service name, device name, and service location with WSDL pointer to the Publish Agent. Publish Agent registers common information such as Sub-network ID, Sub-network type, service ID, Universal Networks (UN) ID, device ID, service name, device name, and category name to the Device/Service Registry. It also sends the service location to the Context Registry. When the process of the chain is completed, the Hook-up processes of the DPWS Adaptor ends.

4.2.2. DPWS Device Connection Started (Hello Message). There are two cases which are appearance of the new DPWS device and the previously existed DPWS device, as shown in Figure 9.
This sequence diagram describes the sequential process of the DPWS device Hello with Step 19. From Step 1 to 3, the DPWS device informs appearance through a Hello message of WS-Discovery. Then, the DPWS Adaptor obtains device/service information of DPWS with UUID. The Query Agent confirms the DPWS device in the Device/Service Registry with UUID in Steps 5 to 7. If it doesn’t contain the DPWS device information, the DPWS Adaptor operates from Step 8 to 13 like the processes of the DPWS Adaptor Hook-up. If not, the DPWS Adaptor operates from Step 14 to 19. It is the DPWS Adaptor which changes the service status at the Device/Service Registry and updates the service location in the Context Registry concurrently with UUID.

4.2.3. DPWS Device Connection Finished (Bye Message). As seen in Figure 10, the DPWS device finishes the connection using a Bye message of WS-Discovery. The Bye message contains only UUID. If the DPWS Adaptor receives a Bye message from the DPWS device, it sends the service status of the DPWS device to the Publish Agent. Through these processes, the Device/Service Registry is updated and the service status is invalid.

4.2.4. DPWS Device Connection Finished (System Error). The DPWS Adaptor periodically confirms DPWS service duration. If it doesn’t receive the response message about requesting the service duration, it then changes the service status in the Device/Service Registry through the Publish Agent, as shown in Figure 11:

4.2.5. Change Service Status of the DPWS Device. As seen in Figure 12, this sequence diagram shows the change in service status of the DPWS device. If the status of service changed, the DPWS Adaptor updates the Device/Service Registry information by an event message.

4.3. DPWS Adaptor Functions

The DPWS Adaptor is one of the UniA. So, the DPWS Adaptor has basic UniA functions.

The following is the explanation of DPWS Adaptor specific functions.

- Message Transfer: The DPWS Adaptor operates with UniA is similar.
- Discovery DPWS Services: The DPWS Adaptor sends multicast message for searching DPWS device/service information with UUID.
- Creation Virtual Web Services: Because DPWS is SOAP based communication so that Web Services operate similarly as device level, the DPWS Adaptor doesn’t create WSDL for virtual Web Services. The
DPWS Adaptor just uses the WSDL of DPWS.

- Events and Notifications: The DPWS Adaptor operates with UniA is similar.

5. Conclusions

The researches of the existing studies are OSGi, DomoNet and DPWS like we proposed concept in ubiquitous computing. OSGi and DomoNet are studies on interoperability between Sub-networks, and DPWS is a research for the integration of devices. However they don’t provide the various service discovery methods and include service status. And they have the problems which are subordinate to a platform or a language.

To solve the problems, we introduced WSUN environment which is composed of US Broker, US Registry, UniA and so on. The US Broker provides interoperability between heterogeneous Sub-networks to discover universal services. And we design the DPWS Adaptor which is a core component of the US Broker with Scenario. The DPWS Adaptor is able to support message transfer, DPWS services discovery, virtual Web Services creation, events and notifications.

The current US Broker is designed for the integration of JINI and DPWS. However upon further studies, other Sub-network environments will be included. Finally, after the clients’ inquiry is optimized and their contextual information is considered, a more improved WSUN discovery protocol will be defined.

Acknowledgment

This research was supported by the Ministry of Information and Communication, Korea, under the College Information Technology Research Center Support Program, Grant Number IITA-2006-C1090-0603-0031.

References