Ubiquitous system provides web services which clients want irrespective of specific places and time. However, current services and devices supported by them are operated by different communication systems. To solve this problem, a system which can guarantee interoperability of services and devices is required. This paper introduces the UWS Broker, which is a middleware to discover services dynamically in ubiquitous environments. The UWS Broker, which is SOA oriented, is devised for different network based devices and services.

1. Introduction

Service discovery middleware, used in distributed network environment and most commonly applied is Jini[6], UPnP[1], HAVi[4]. These are sub-networks. Sub-network for each device must use a different communication protocol and for actual application of clients, a specific environment must be constructed. Clients of Jini, Java-oriented environment must know how to process SOAP binding to use UPnP sub-network service and understand service technology information expressed with XML. This results in clients’ limitation of service use and search environment. So far, to solve this problem and to make interoperability between sub-network possible, although much research such as Open Services Gateway initiative (OSGi)[9], Dominic Network (DomoNet) [10][11], and Device Profile for Web Services (DPWS)[2][3] have been done, they have not provided dynamic search and independent environments needed in the ubiquitous environments. Also, these researches did not consider the existing web service clients.

This paper introduces the SOA-oriented UWS Broker which makes it possible for existing clients to use ubiquitous web services in a ubiquitous environment. The UWS Broker, which operates as an intermediate passage between sub-networks, makes interoperability possible and helps web service clients use sub-network services. The UWS Broker generates sub-network service as a virtual web service, registers UDDI for web service clients and puts gateways in every sub-network.

The second chapter of this study reviews the existing research on service and device interoperability, the third introduces how the UWS Broker operates, the fourth describes methods for sub-network interoperability and the fifth presents the conclusions of this study and further research related to this study.

2. Related Work

The study on service and device interoperability is divided into a study to solve the problems of interoperability between sub-networks and between devices. OSGi and DomoNet are studies on interoperability between sub-network, and DPWS is a study to solve the problems of interoperability between devices.

2.1. Open Services Gateway Initiative

The goal of the OSGi, SOA-oriented structure, is to make it possible to install, update, and delete software without rebooting devices.

As seen in Figure 1, OSGi which is located between the exterior network and the home network, controls the devices and service of the home network outside. OSGi is Java VM based and possesses Bundle and OSGi service to use services within the home network. Bundle provides functions necessary for interface and service control for access to the home network services. OSGi services are basic functions for Bundle, stored in a registry within the OSGi framework (Figure 2).
As seen previously, the OSGi framework controls the home network, does not provide the interoperability between sub-networks, and which enables clients to use services of every sub-network. Also, it has another problem, as the environment is subordinated to Java language.

2.2. Domotic Network

DomoNet, SOA-based framework, helps to solve the interoperability between the home networks.

Figure 3 demonstrates the framework architecture of DomoNet. DomoNet has Tech Manager (TM) for sub-network interoperability. It stores sub-network service as a form of DeviceWS in the UDDI registry and generates a virtual device fitting to each sub-network. By the virtual device, different sub-network devices enable the real device to be used as sub-network of several environments with a communication device in the DomoNet environment.

However, the main study focus of DomoNet is not on service discovery but on service control. And it fails in dynamic service discovery because it cannot reflect status information of service and device.

2.3. Device Profile for Web Services

DPWS, SOA based framework, generates device profiles according to devices for integration of devices and they have the profiles. When clients search devices with services that they want through P2P communication, device profiles are used (Figure 4).

Since DPWS supports P2P communication and operates multicast discovery to acquire necessary information whenever a service search is performed, it unfortunately takes a long time for a service search. It also excludes service interoperability and focuses only on device interoperability.

Therefore, we introduce the UWS Broker, which makes interoperability of both services and devices possible and which can perform a dynamic service search method that reflects the service status information.

3. Ubiquitous Web Service Broker

The UWS Broker, which is an SOA-oriented framework, supports not only web services but also ubiquitous web services search and binding.

The UWS Broker is located between the web service clients and web service providers (figure 5). Web service clients can discover both the existing web services and ubiquitous web services through a UDDI registry search.
However, ubiquitous web services provide additional service location information and service status information except the general features of web services. Therefore, web service clients must perform additional inquiries using ubiquitous web services by searching UDDI service through the UWS Broker. Also, the UWS Broker provides clients with currently feasible ubiquitous web service and virtual web service for sub-network services in order to use ubiquitous web service without the clients’ additional work.

3.1. UWS Broker Components

The following is the explanation of the UWS Broker Components.

- **UWS Registry**: It stores specific metadata that the ubiquitous web service possesses. Although the UDDI registry can represent features of web service well, it cannot store and manage device information that provides service status information and location information. Meanwhile, the UWS Broker stores and manages ubiquitous web service features in the UWS registry.

- **Context Registry**: It stores location information of ubiquitous web service and information about client preferences. In a ubiquitous environment, the service location changes frequently. Service location is an important factor for service discovery. For example, clients who are close to the Eiffel Tower are provided with service which is supported nearby.

- **UUID Generator**: It generates unique UUID of service and devices which are used in the UWS Broker. If the service and device ID used in the sub-network are not unique within the UWS Broker, it will have an inconsistent form. Therefore, after unique UUID of service and device is created, they are registered in the UDDI and UWS registry.

- **Query Agent**: It processes the clients’ service discovery inquiry. Query Agent searches UDDI and UWS registry after it receives the clients’ service discovery inquiry. It delivers the results to the clients.

- **Publish Agent**: It stores ubiquitous web service information in the UDDI, UWS, and context registry. After it creates virtual web service, the publish agent stores the ubiquitous web service in the UDDI registry and stores the service/device name, the service ID and category information within the sub-network in the UWS registry. After the storage is completed, the publish agent informs the gateway of this fact.

- **Gateway**: It transfers the sub-network and the UWS Broker. Gateway creates the sub-network service as the virtual web service. At this time, to acquire the information needed, the sub-network protocol is used to communicate and gSOAP is used to communicate within the UWS Broker.

- **Routing Proxy**: It binds the service that the clients want. The routing proxy provides virtual web service within the UWS Broker that the web service clients choose and the existing web service binding.

3.2. Universal Services Discovery Protocol

The service discovery inquiry which the UWS Broker provides is divided into two stages (Figure 6).
The first inquiry, inquiry on the UDDI registry, includes BusinessEntity, BusinessService name, description information and tModel information. The second inquiry is on ubiquitous web service. The second inquiry occurs when the clients want to provide additional information among the first inquiry results and a more detail inquiry. The second inquiry includes service location and category information and the service status information valid after the default.

The first and second query parameter and the results of them are as follows:

- **The First Query Parameter**
  - BusinessService Name: Mandatory
  - BusinessEntity Name: Optional
  - Description: Optional
  - tModel Information: Optional

- **The Results of the First Query**
  - BusinessService ID(UUID) List
  - BusinessEntity Name
  - WSDL URL

- **The Second Query Parameter**
  - Service ID(UUID) List: Mandatory
  - Service Location: Optional
  - Category Name: Optional
  - Service Status: Default

- **The Results of the Second Query**
  - Service ID
  - Service Name
  - Service Location
  - WSDL URL

When the clients select services that they want among the services discovered, routing proxy performances binding.

4. **Interoperability for Sub-Networks**

Since different sub-networks use different communication protocols, clients must change protocols for message delivery whenever they search sub-network service. Also, clients must download several binding information necessary for the specific sub-network environment they want to use. If the UWS Broker repeats this process for each client’s inquiry, it will take a very long time for inquiry processing and a large amount of work is needed. Therefore, this study creates a virtual web service through the gateway and stores the service metadata in the registry. Therefore, when clients use the service, they search the UDDI registry directly and can use the service which they want to use instantly.

4.1. **Gateway Components**

As seen in Figure 7, the gateway components which play a role in transferring the sub-network include Message Transfer, Eventing and Notification, Creation VWS, and Discovery Sub-network Services.

![Figure 7. Gateway Components](image-url)
updates events such as the appearance of new services or the disappearance of the existing services in the registry in real time and provides the clients with feasible ubiquitous web services.

4.2. Creation Virtual Web Services

This section describes the virtual web service generation among the functions of the gateway. The order by which the gateway generates the virtual web service is shown differently in the design stage and run time stage. Also, depending on many cases generated in the sub-network, the generation order can be different.

The cases generated in the sub-network are as follows:
- Sub-network Hello/Bye
- Device Hello/Bye

First of all, the sub-network hookup process in a design stage is started. Figure 8 is a diagram of the sub-network hookup process.

1. The gateway within the UWS Broker must bring the service information within the sub-network.
2. It generates virtual web service and stores the information in the UDDI registry.
3. After the information is stored in the UUID registry and UUID generated in the UDDI, the device and service information in the UDDI registry are registered in the UWS Registry.
4. After completion of the storage of the UWS Registry, the device location is stored in the context registry.

**Figure 8. Sub-Network Hookup in Design Time**

**Figure 9. Sub-Network/Device Hello in Run Time**
The next case which occurs in runtime is appearance and disappearance (hello and bye) in the sub-network or device. In a case of appearance of new sub-network or device, the generation process of virtual web service for the request service is shown in Figure 9.

This process adds a process to check existing service stored in Figure 8 and the remaining processes are the same. This process helps change the existing service condition as valid and reduces overload of the UWS Broker gateway without generating new virtual service when a service is stored in a UWS registry.

In the UWS Broker Run Time, the case of Sub-network/Device Bye is divided when the connection is cut off normally and abnormally. When the connection is cut off normally, the UWS Broker gateway makes the status information of the service which the real device supports invalid because the sub-network or device sends a bye message. However, when the connection is cut off unpredictably, through a renew process of the sub-network service, service for the device of no response is updated in an invalid status.

5. Conclusion

This study explains the SOA-based UWS Broker. The UWS Broker supports interoperability between sub-networks, and helps clients to use ubiquitous web service without additional work by generating virtual web service for the clients’ convenience. The current UWS Broker is designed for the integration of Jini, UPnP, and DPWS but through further study, other sub-network environments will be included.

The further study will verify this with several scenarios by realizing the UWS Broker. Also, after clients’ inquiry will be maximized and their context information will be considered, more improved ubiquitous web service discovery protocol will be defined.

6. Acknowledgement

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.

7. References