A Mechanism on OSGi Agent for Dynamic Monitoring Service Discovery Protocols in Local Network

Pitak Sawetsunthorn

Abstract—Although there are several standard services to discover protocol in the network such as UPnP, SLP, Jini, and DPWS (minimal set of web service) on computer and electronic devices such as mobile devices, these protocols cannot search the same standards across services. Most research was often modified by customizing the protocol. To solve this problem, in this work, we propose how to find services in the different standards used by agents. This solution does not need to modify the original protocol and dynamic monitoring to reduce unnecessary processes. We propose a prototype system to allow users to discover the service in a local area network by using mechanism-based services oriented architecture, namely OSGi Framework.

Index Terms—Service Discovery Protocols, Agent, Service Oriented Architecture, OSGi Framework.

I. INTRODUCTION

Group communications of computers and electronic devices such as mobile devices have the service discovery protocols to communicate with each other on a standard protocol. However, these industrial standardizations provide the same basic service discovery. They differ significantly in architecture, message exchange pattern and service description because service discovery protocol has been developed by many industrial standardizations; for example, Universal Plug and Play (UPnP) and Devices Profile for Web Services (DPWS) developed by Microsoft, Service Location Protocol (SLP) developed by IETF and also Jini developed by Sun Microsystems. Moreover, these protocols have developed on the basis of objective and different environments (See in Fig. 1).

Fig. 1. Service discovery protocols on local network

Even though each service discovery protocol has base structure similarly, it cannot be used together. Since it has chosen different language; for instant, XML description format for UPnP and DPWS, SLP is attribute-value pair

based (text-based), Jini used Java class hierarchy and assistant service attributed objects.

In order to get fully benefit from all the networked services on the environment is required interoperability of existing discovery protocols. There are various researches which use of adapted or modified the original protocol support the work of a standard search services. For example, the trend of research approach of matching service between the two protocols such as Jini and UPnP [1]. Although this method has many advantages in terms of lightweight discovery in view of the users, it has workload on the provider in network because all parts of the framework are not ready if they need to activate all in the same time. Furthermore, they bring to the restrict scope of developing.

This paper suggests how service oriented architecture (SOA) concept can be achieved on the fly in the case of pervasive network. In addition, it describes a prototype mechanism implementation on framework by taking advantage of the open standard OSGi for an interoperable service discovery at application layer which do not need to modify the original protocol and supports loose coupling each processes of providers. A prototype has been developed to evaluate the feasibility of the framework

II. RELATED WORK

A. Interoperate Service Discovery Protocols Approaches

Interoperating of difference service discovery protocols architecture is difficult because they have a few similar things. Popular approach is focused on a modifying protocol and the mapping between the different message exchange pattern, such as research [2] developed to convert from UPnP format to Jini format through bridging the agent network. (Only one way to UPnP can be transformed to the native Jini protocol is to deal with the format service description of service such as attribute and control URL.

Another approach is to create a framework such as research [3] UPnP and Jini by creating a specific framework to manage a unit operation within the framework which is used to map them. However, the result of mapping shows troubles from the basic protocol structure. UPnP is described by XML language which is not completely supported with Jini because Jini uses look up service with a table hash table as well as the search service. This approach is similar as the research [4] which has developed a program named INDISS (Interoperable Discovery System for Networked Services) from Project Amigo proposed mechanism by using observations with the parser and the composer modified at network layer.

Manuscript received May 9, 2012, revised June 27, 2012. The work is supported by Faculty of Science, Prince of Songkla University, Thailand.

Pitak Sawetsunthorn is with Prince of Songkla University, Hatyai, Songkhla, 90110 Thailand (e-mail: pitak.s@psu.ac.th).

The framework has close relationship with the service discovery protocols is an Open Service Gateway Initiative (OSGi) framework, and Multi-protocols Service Discovery (MSDA) [5] is a framework to collaborate between protocols. MSDA platform is a platform component on the application layer which acts to extend service discovery protocol. However, we offer related framework only OSGi standard because it is a popular standard of group communication resulted in the four briefly researches.

- **UPnP Environment - UPnP Based Driver [6]** is a bundle on the OSGi framework which provides an environment for serving protocols including SSDP and GENA protocol which are working with UPnP Device as operate service in the OSGi aspects. The discovery, controlling and event phases of the UPnP protocol are naturally mapped to the OSGi service layer which allows to publish, find, bind and notify events.
- **SLP Environment - jSLP (java OSGi) [7]** is a bundle on the OSGi framework by using the Locator and Advertiser instances to search the other services. SLP Service is managed by SLPSERVICELOCATIONMANAGER and ServiceFactory component is responsible for registration with the OSGi registry.
- **Jini Environment -Jini Driver [8], [10]**, Jini standard protocol is put into a standard OSGi release 3. When evolution of OSGi developed to release 4, a Jini standard protocol is not put into standard the OSGi R4 version. The popularity of Jini protocol dropped because complex interactions between class loaders are made Jini complicated and security implications. However, Jini is still active. This research led Jini protocols to join in the prototype system.
- **DPWS Environment - DPWS Based Driver [9]** is a bundle on the OSGi framework which serves to provide an environment for Web services providing for resource-constrained devices. Plug-n-play protocol middleware built on the top of a set of Web Services specifications. This protocol is a middleware which is used to address, to discover description and to control of devices and services on local networks such as home network.

Therefore, we will introduce key features on OSGi standard platforms In view of services module on the next section.

B. OSGi Framework

The collaboration services on the OSGi framework are flexible to environment which gains a significant advantage by applying this research. Each service is a function within a bundle on the OSGi framework called "Bundle" is a unique service for each bundle. They can be called from any other bundle by running through OSGi framework. Fig. 2 shows the OSGi architecture.

Fig. 2. OSGi architecture.

The OSGi framework consists of three main parts, as shown in Fig.2.

- **Part 1** is a bundle developed with Java language on the OSGi framework which is responsible for services. Within the bundle manifest file is used to determine the characteristics of bundles such as specific import or export services package of services from other bundles.
- **Part 2** is the life cycle of the bundle consists of status as the **STARTING, RESOLVED, INSTALLED, ACTIVE, STOPPING AND UNINSTALLED**. This feature enables control bundle to work flexible manner as well as reducing the workload of processing. The users can control them manually independently.
- **Part 3** focuses on the service registry is a collection Services run through the Application Programming Interface (API) will be associated with ServiceRegistration, ServiceTracker and ServiceReference on OSGi framework.

The concept of this purposed work is to create a service management unit with a flexible application of existing projects such as UPnP based driver, SLP, Jini, DPWS. Each service discovery protocols changes to a bundle with the OSGi framework on Java Virtual Machine (JVM). When bundle stopped and uninstalled, this bundle can be restored memory from processing environments without turning off the OSGi framework. Moreover, inactive bundle is not loaded into memory.

III. PROPOSED MECHANISM AND SYSTEM ARCHITECTURE

Computer Agent is responsible for detection of network services in the different protocols and services in Fig. 3. From the mentioned project, we will create a bundle of agent in the OSGi framework to collaborate with other bundles following below picture.

Fig. 3. Proposed agent at pervasive environment

Agent bundle divided into 3 parts which is worked together. 1. Monitoring service discovery protocols as an OSGi services (See in Fig. 4).

Fig. 4. Concept of agent bundle on OSGi framework

In the standard OSGi Release 4.2, we start to implement the interface *ServiceListener* of the class and the program to detect *ServiceEvent* object which are delivered to *ServiceListeners* and *AllServiceListeners*. When a change occurs in this service's lifecycle, service description comes with the services of a registered class *org.osgi.framework*.

ServiceEvent objects are sent to *ServiceChanged* function to receive notification when *ServiceEvent* change every time.

We purpose and implement enhanced data collection services in data store such as simple text file for searching services in the service registry.

Moreover, we develop *BundleContext* as a function of the object services registered on the service registry. The service registry uses the WhiteBoard pattern and *BundleContext* object will be created and provided to the relevant bundle with this context.

Then we implement *BundleActivator* interface to start and stop functions which is used to control the opening and closing operation of the bundle. In the bundle, *Service* element is used to define a service representing exported OSGi service. You must specify the service to be exported and the interface service before advertising the service.

Monitoring Service: monitoring service collects a detailed operation of the service. Then it adds it to the service registry data store by developing of a servlet [11] in a function on the bundle of agent used for automatically provide in the list via a web browser which based on in DPWS based driver.

In experimental, our development is associated with many functions. However, we specify only important function on monitor of list of relevant services.

- 1) Import package *javax.servlet.** and *org.osgi.service.**; and create classe extend *HttpServlet* class and develop on the bundles.
- 2) Create object *StringBuffer* class for retrieve information stored (text based on Web Based Programming) in the data store to object *stringbuffer* with *append* function.
- 3) Send this object to *getWriter* function to handle the display. To extends *HttpServlet* class and developed on the bundle.

The program is developed to be jar file contains manifest files such as image and execution. This library allows us to avoid unnecessary processing on the agent but it must managed by the appropriate functions which is used in the consideration in the next section.

A. Consider the Function

In fact, we cannot know the sequence of the service discovery protocols, as shown in Fig.5. Therefore, it is necessity to have consideration of the functional prototype program which is developed on the bundle.

Fig. 5. Sequence of the service discovery protocols

Objects registered with the Service Registry are called *services* which registered with an interface name and a set of properties. For agent bundle, the OSGi Monitoring Service would be registered with the interface name and service descriptions.

The concept of consideration the function began the first by detecting availability of each Service discovery protocol, as shown in Fig.6. (Type of variable is possible UPnP SLP Jini and DPWS only). If it found the service discovery protocol, it would be update the status of the service and be stored in a data store and started specific monitoring function.

If it could not find service, (Range of *count_variable* = 0, 1, 2) the next round does not repeat until the search this service run four round completely. *Count_round* variable reduce workload of system to update these protocols.

Stored data in the data store service is a type of service and service description which will be applied in providing

services through a servlet on the OSGi framework by running on a web browser.

```

Set SDPs is type of array variable;
Set i,count round = 0;
Do while loop
{
// Monitor SDPs is available on environment.
Turn on monitor each SDPs;
If (found SDPs)
UPDATE status of SDPs and keep value in data store;
Else if (not found SDPs)
{
If count_round in data store < 4 round;
{
Turn off monitor this SDP;
Count increment +1;
}
}
Else if count_round in data store > 4 round ;
{
Turn on monitor this SDP;
Reset Count = 0;
}
}
UPDATE status of SDPs and keep value in data store;
}

// Display for SDPs

Display list of services;
Delay/sleep time;
Shift to next SDPs still not last SDPs
if last SDPs set begin;
}

```

SDPs = Service Discovery Protocols

Fig. 6. Pseudo code of consider the function.

Part of delay/ Sleep time is used to avoid system loading too much memory. We recommend that this should be an important part in the program before shifting to next service discovery protocol. When it reaches to the last protocol, the round will reset to the first protocol.

B. Display List of Services

This will be part of the monitoring service work with the value stored will be run through a bundle of the servlet as a OSGi Service.

IV. EXPERIMENT AND RESULT

We assume that all services have already registered in their respective. Four service discovery protocols of project have been running on local network, which is shown in Fig.7.

Computer agent installed bundle agent on Apache Felix [9] which is OSGi framework with dynamic aware of services in the network to find all four services following below. The snapshot of results has been shown in Fig 8.

Fig. 7. Scenarios for experimental.

Fig. 8. List of service on web browser.

V. CONCLUSION AND FUTURE WORK

We propose a prototype system with agent to assist dynamic monitoring services in various networks with support for discovering services without need to modify any service discovery protocols by using an OSGi framework. This framework can control the life-cycle of applications without requiring the JVM to be restarted. It also supports monitoring of client service through a Web browser.

Moreover, Issue of the reducing workload on the agent can be managed independently that we have already presented control bundles on the application level. However, we use only four types of service discovery protocols in the experimental environment. In fact it may be several services (service descriptions) and devices may be led to create problems of workload processing and require more data store on the computer agent.

The future work is to analyze and evaluate the factors which are affecting the increase in finding service for increasing information as well as details of the service, URL of the service and other features.

REFERENCES

- [1] J. Allard, V. Chinta, S. Gundala, and G. Richard III, "Jini Meets UPnP: An Architecture for Jini/UPnP Interoperability" *Proc. of the 2003 Symposium on Applications and the Internet (SAINT '03)*, pp. 268, doi: <http://doi.ieeeecomputersociety.org/10.1109/SAINT.2003.1183059>, January 2003.
- [2] A. Sameh, and R. El-Kharboutly, "Modeling Jini-UPnP Bridge using Rapide ADL" *Proc. IEEE of the IEEE/ACS International Conference on Pervasive Services (ICPS'04)*, pp. 237, doi:10.1109/PERSER.2004.1356809, July 2004.
- [3] J. Newmarch, "A Custom Lookup Service for UPnP Services and Jini Clients", <http://jan.newmarch.name/java/jini/upnpLUS/paper.pdf>
- [4] Y. Bromberg, V. Issarny, and P. Raverdy, "Interoperability of Service Discovery Protocols: Transparent versus Explicit Approaches" *Proc. IST Mobile and Wireless Summit 2006*. 2006.
- [5] P. Raverdy, V. Issarny, R. Chibout, A. Chapelle, and Agnes, "A Multi-Protocol Approach to Service Discovery and Access in Pervasive Environments" *Proc. IEEE of the 3rd Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services*, July 2006, pp. 1-9, doi:10.1109/MOBIQ.2006.340448.
- [6] "Apache Felix UPnP", [Online] Available: <http://felix.apache.org/site/upnp-driver-architecture.html>, January 2011.
- [7] "jSLP-OSGi", [Online]. Available: <http://jslp.sourceforge.net/jSLP-OSGi/index.html>, January 2011.
- [8] "Jini Driver", [Online]. Available: <http://www.osgi.org/javadoc/r3/org/osgi/service/jini/JiniDriver.html>, January 2011.
- [9] "INRIAGforge", [Online]. Available: https://gforge.inria.fr/frs/?group_id=160&release_id=1804, January 2011.
- [10] C. Lee, J.Yi and W. Lee, "Bridging OSGi Islands Through SLP Protocol", *Proc. of the 4th International Conference on Ubiquitous Intelligence and Computing (UIC'2007)*, pp. 787-797, July. 2007.
- [11] K. Lee, D. Kim, K. Choi and S. Dong-Ryeol, "Web-based Integrated Service Discovery Using Agent Platform for Pervasive Computing Environments" *Proc. of the 7th international conference on Computational Science (ICCS 2007)*, pp. 952-955.

Pitak Sawetsunthorn received his BEng (2007) and MEng degrees (2009) in Computer Engineering both from Prince of Songkla University, Thailand. He worked as a Lecturer at Prince of Songkla University, Thailand, since 2010 at Information and Communication Technology Programme, Faculty of Science. His research interests are: mobile computing, mobile networking, SOA and ubiquitous computing.