An Intelligent Middleware Architecture for Context-Aware Service Discovery*

Kyu Min Lee, Hyung-Jun Kim, Kee-Hyun Choi, and Dong-Ryeol Shin

School of Information and Communication Engineering,
Sungkyunkwan University,
300 Cheoncheon-dong, Jangan-gu, Suwon, Gyeonggi-do 440-746, Korea
{kmlee, mairi, gyunee, drshin}@ece.skku.ac.kr

Abstract. There is a high interest on service discovery techniques, which minimize the cost of detecting services and provide users with convenience. For more dynamic and useful service discovery, middleware for context-aware service discovery is required. This paper proposes an intelligent middleware architecture for context-aware service discovery that is based on JADE, which is a FIPA-compliant agent platform. The proposed middleware provides not only service scalability among heterogeneous domains, but also modules processing context information. When a service is detected, context information relating to the user and environment is used. As a policy-based system, the proposed middleware also use predefined policy. In other words, user preference can be considered. Near the conclusion of this paper, we discuss future works.

1 Introduction

Both service providers and users of these services require a middleware system to integrate heterogeneous environments, employing various techniques to reduce complexity. There are many existing middleware system to solve that problems. Although they try to solve integration problems, another problem emerges. That is to use context information such as temperature, position and body information because service users can find more appropriate service if the middleware reflects context information, so recent middleware approach is to support context-awareness.

In this paper, we propose a middleware architecture based on Java Agent Development Framework (JADE) [1], a popular agent platform, for context-aware service discovery. In the proposed middleware, raw context information is discarded and useful raw data is represented to high-level data. Context information can be combined, inferred by Context Manger which is comprised of several modules in our middleware. Because our middleware also have Policy Module, it can consider users' preference.

The remainder of the paper is organized as follows. Section 2 presents our proposed middleware for context-aware service discovery in detail. Finally, the conclusion is discussed in section 3.

* This research is supported by the ubiquitous Autonomic Computing and Network Project, the Ministry of Information and Communication (MIC) 21st Century Frontier R&D Program in Korea.

Fig. 1. The Architecture of Middleware

2 Proposed Middleware System

2.1 Architecture

Novel middleware is designed for context-aware service discovery. As demonstrated in Figure 1, this middleware consists of various modules such as parser, composer, service repository, and so on. In this paper, context-awareness in middleware is focused on, therefore other modules are discussed only briefly.

The Message Monitor has each discovery agent as a monitoring module for each service discovery protocol or network domain that can detect all messages from them because each network protocol or domain uses a well-known IP and Port number, i.e. 239.255.255.250:1900 on UPnP and 239.255.255.253:2427 on mSLP. The Discovery Adapter consists of a Parser, which parses messages, and Composer, which composes indispensable data parsed from messages. It means the Composer represents the parsed data to the appropriate format, DAD (DF-Agent-Description) [1] format. The Composer then registers it with the Service Repository such as the Directory Facilitator (DF) in an agent platform [1]. The agent-based service provider can also register services directly via an Agent Communication Language (ACL) message [1]. When a service requester on behalf of a user requests a service, a request message is transmitted to the Matchmaker, which calculates which service is the most appropriate for the service request. The Context Provider such as light, camera and temperature, serves raw context information to the Context Manager. The Context Manager consists of modules which guarantee context-awareness. Useful context information is stored in

the Context Repository. If you want to see detail information about the middleware, you can see [2].

2.2 Context Awareness

By introducing modules in Context Manager, people can know how Context Manager is comprised of. The Context Manager consists of six modules and one database as shown in Figure 2. Each of them can be described as the following:

Fig. 2. Context Modules

- **Context Device Manager:** In ubiquitous environments, there are many diverse context devices and context sources. They frequently enter or go out in the network, so we design the Context Device Manager, which manages them. When they register or deregister, a message is sent to the Context Device Manager. Then, it recognizes the context sources composed in the middleware network.
- **Context Filtering Module:** The next module is the Context Filtering Module. Its main purpose is to protect the Context Repository from being flooded with excessive information. When context information (e.g., time) from context sources is continuously delivered, users or user agents may be interested in receiving a few values periodically. This is especially important for both the Matchmaker and users because the Matchmaker can search requested context information in the Context Repository as soon as possible and users reduce the response time of the Matchmaker.
- **Context Representation Module:** The Context Representation Module converts raw context information from context sources such as sensors into normalized forms. Information from various context sources, in general, exist in ubiquitous computing environments and there are many different types of data, e.g., binary, integer, real number, etc. For the more, the representation may be different from each other. This will result in difficulties not only in usability, but also in production of higher-level contexts from lower-level contexts. This module represents raw data in normalized forms.

- **Context Aggregation Module:** The Context Aggregation Module has the ability to combine the context information from the Context Representation Module. This module produces combined contexts. The combined context can be used to provide better information with users because they can understand easily what kind of situation. For example, when Context Representation Module generates Location (“713”), Patient (“Chris”), Sensor (“Bodily Temperature”), Temperature (“34”), Context Aggregation Module creates Room(Number, “713”, Patient, “Chris”, Sensor, “Bodily Temperature”, Temperature, “34”).

- **Context Inference Module:** Context Inference Module can contain one or more context reasoning modules, based on the users’ requirements. The need for a Context Inference Module arises because not all information can be gathered from context sources. It is used to derive higher-level context information from lower-level information. Therefore, the module infers new context information from the current context. This makes the middleware system increasingly intelligent. For example, in a room a man lies down on a bed, he turns off lights and closes his eyes. This means he is sleeping.

- **Context Policy Module:** When a situation or a environment is configured, users might want to know the moment. At that time, middleware invokes actions already described. This is possible due to the Context Policy Module. User can send policy message as an ACL message format to the Context Agent, which has Context Policy Module. In other words, this invokes actions if a pre-defined context is detected in the current situations or surroundings.

3 Conclusion

This paper proposes middleware architecture based on JADE for context-aware service discovery. We are currently implementing the middleware in accordance with a hospital scenario because the best way to test this is to create a scenario.

References

1. JADE, <http://jade.tilab.com>, Telecom Italia Lab, 2005.
2. Hyung-Jun Kim, Kyu Min Lee, Kee-Hyun Choi and Dong-Ryeol Shin, “Service Discovery using FIPA-Compliant AP to Support Scalability in Ubiquitous Environments”, Proc. of the IEEE on ICIS, Pages: 647-652, July 2005.