이종 에이전트 플랫폼 간의 상호운용을 위한 서비스 디스커버리 미들웨어 설계 및 구현*

오세정, 백주련, 고혁진, 신동렬, 김응모
성균관대학교 컴퓨터공학과
e-mail : ohsejung@skku.edu

Implementation of Service Discovery Middleware for Heterogeneous Agent Platforms

Se-Jung Oh, Ju-Ryun Paik, Hyuk-Jin Ko, Dong-Ryeol Shin, Ung-mo Kim
Dept. of Computer Engineering, Sung-Kyun-Kwan University

요 약

유비쿼터스 컴퓨팅 환경에서 분산되어 존재하는 서비스나 디바이스의 효과적인 사용을 위해서는 서비스의 종류나 서비스의 위치 등의 기본적인 정보가 사전에 제공되어야 한다. 이러한 기본 정보를 사용자의 요구에 맞게 정확히 탐색해서 찾아주는 일련의 과정을 서비스 디스커버리(Service Discovery)라 칭하며, 유비쿼터스 같은 분산 이동 환경에서 가장 먼저 구축되어야만 하는 시스템이다. 최근에는 발달된 에이전트 기술을 서비스 디스커버리와 접목시켜 분산되어 있는 서비스와 디바이스 그리고 사용자까지 효율적이고 유용성 있게 관리하는 추세이다. 각국의 기업이나 학교에서 수많은 에이전트 플랫폼이 개발되어 왔지만, 각기 다른 개발 기준의 적용으로 인하여 에이전트 플랫폼 간의 상호운용(Interoperability)에 어려움이 존재한다. FIPA 에서는 이종 에이전트 플랫폼 간의 제공 서비스 상호운용을 위해서 디스커버리 미들웨어(Discovery Middleware) 모듈을 제안하였다. 본 논문에서는 FIPA-OS 와 JADE 그리고 SLP 플랫폼 들이 서로의 서비스를 공유할 수 있도록, FIPA 에서 제안한 디스커버리 미들웨어를 실질적으로 개발하여 상호간의 서비스 디스커버리가 이루어질 수 있도록 하였다.

1. 서론

유비쿼터스 환경이란 물이나 공기처럼 시공을 초월해 '언제 어디에나 존재한다'는 뜻의 라틴어로, 사용자가 컴퓨터나 네트워크를 의식하지 않고 장소에 상관없이 자유롭게 네트워크에 접속할 수 있는 환경을 말한다[3]. 이러한 환경에서는 중앙 집중식 컴퓨팅 환경보다는 분산 컴퓨팅 환경이 적합하다. 네트워크의 연결상태 및 서비스 정보가 항상 동적으로 변화하기 때문에 변화가 발생할 때마다 중앙 시스템이 제어하게 된다면 무수히 많은 네트워크 부하가 발생되기 때문이다. 분산 환경에서는 모든 서비스나 장치들의 상호운용(Interoperability)을 위해서 반드시 해당 서비스의 종류와 위치 또는 서비스에 대한 개략적인 정보 등의 제공이 있어야 한다. 이러한 정보를 제공해 주는 시스템을 서비스 디스커버리(Service Discovery)라 칭한다[4]. 서비스 디스커버리 기술과 더불어 개별적인 서비스의 관리를 담당하는 시스템의 필요성이 높아졌으며 이를 위해 에이전트 플랫폼(Agent Platform)이 개발되었다[2]. 에이전트 플랫폼은 각각의 서비스를 에이전트화 하며, 에이전트가 필요로 하는 모든 자원과 환경을 제공한다. 뿐만 아니라, 서비스 환경 탐색과 정보 획득을 위해서 서비스 디스커버리 매커니즘 역시 제공한다. 가장 널리 알려진 에이전트 플랫폼으로는 JxTA, JADE 그리고 FIPA-OS 가 있으며, 에이전트 플랫폼은 아니지만 서비스 디스커버리의 기본적인 체계를 제공하는 SLP 가 있다[9,5,7,1].

유비쿼터스 환경에서 에이전트 플랫폼 간에는 필연

* 본 연구는 유비쿼터스 컴퓨팅 및 네트워크 원천기반기술과 과학재단 특정기초연구사업(R01-2004-000-10755-0)의 연구 결과로 수행되었음.

적으로 도메인 중첩이 발생한다. 따라서 이종 플랫폼 간의 호환을 위한 표준이나 매개체가 필요하다. FIPA 에서는 이러한 문제를 해결하기 위해 디스커버리 미들웨어(Discovery Middleware)를 제안하였다[6].

본 논문에서는 유비쿼터스 컴퓨팅 환경에서 FIPA-OS 와 JADE 등과 같은 이종 에이전트 플랫폼 간의 상호운용 서비스 디스커버리를 위한 미들웨어를 제안한다. 제안된 디스커버리 미들웨어를 통해 단일 에이전트 플랫폼 상의 서비스는 다른 에이전트 플랫폼 상에 서비스의 등록 및 삭제 등의 갱신이 이루어질 수 있으며, 에이전트 플랫폼 간의 서비스 공유 역시 가능하다.

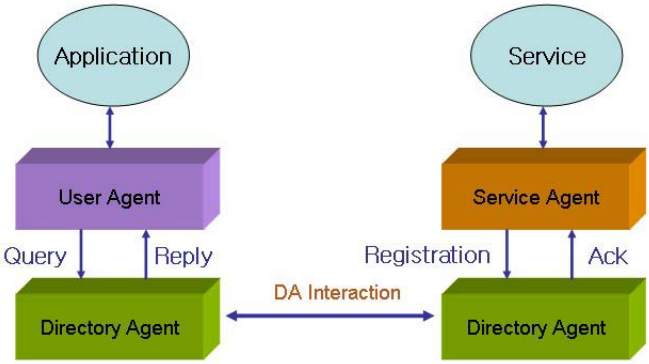
본 논문의 구성은 다음과 같다. 2 장에서는 SLP, FIPA-OS, 그리고 JADE 에 대해 개괄적으로 기술하며, 이들 에이전트 플랫폼에 대해 알아본다. 3 장에서는 FIPA-OS 와 SLP 의 서비스 디스커버리 매커니즘 상호운용을 위해 필요한 디스커버리 미들웨어를 소개한다. 마지막으로 결론 및 향후 연구 방향을 논하며 본 논문을 맺는다.

2. 관련 연구

2.1 SLP

SLP(Service Location Protocol)는 이름 자체가 의미하듯이 서비스 위치 획득 프로토콜로서 SLP 도메인에 존재하는 서비스에 대하여 등록, 삭제, 수정 등을 가능하게 한다[1]. SLP 의 주 목적은 같은 도메인 내에 존재하는 서비스들에 대하여 네트워크 레벨에서 서비스에 대한 위치를 자동으로 찾는 것이다. 때문에 실제 서비스를 제공하는 어플리케이션 레벨에서는 다루지 않고 있다. SLP 의 구조는 크게 DA(Directory Agent), UA(User Agent), 그리고 SA(Service Agent)로 구성되어 있다. SA 는 특정 서비스를 제공하는 Provider 의 역할을 수행하는 모든 장치 또는 어플리케이션을 대표하며, UA 는 특정 서비스 사용을 요청하는 Consumer 라 할 수 있다. DA 는 SA 와 UA 사이에서 중재자 역할을 하며 서비스에 대한 모든 정보를 저장하고 있다. 이들은 에이전트라기 보다는 데몬에 가깝지만 특정 역할을 수행한다는 측면에서 에이전트라 명한다. [그림 1]은 SLP 의 시스템 구조를 나타낸다. SA 와 UA 는 멀티캐스팅을 통해서 자신이 속한 도메인 내의 모든 DA 를 찾는다. SA 는 DA 에 서비스를 등록, UA 는 DA 를 통해 특정 서비스를 제공하는 SA 에 대한 정보를 얻는다.

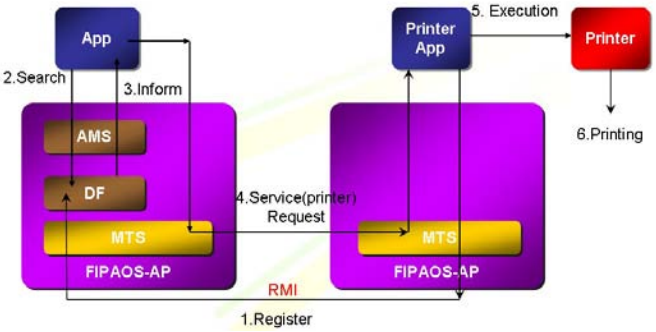
앞서 언급했듯이 SLP 는 서비스에 대한 위치 획득을 목적으로 한다. 때문에 어플리케이션에 대한 구현은 제공하지 않는다. 따라서 이종 에이전트 시스템과의 상호운용을 위해 어플리케이션 레벨의 확장이 필요하다.



[그림 1] SLP 시스템 구조

2.2 FIPA-OS

FIPA(Foundation for Intelligent Physical Agents)는 에이전트들과 에이전트 기반의 응용프로그램 간의 개발 설계서를 공개함으로써, 지능형 에이전트 연구 분야의 표준을 만드는 국제 기구이다[8]. FIPA-OS 는 FIPA 에서 정한 국제 표준을 기반으로 하여 개발된 에이전트 플랫폼이다. FIPA-OS 에이전트 플랫폼은 크게 AMS(Agent Management Service), DF(Directory Facilitator), MTS(Message Transport Service)로 이루어져 있으며, 서비스 디스커버리 매커니즘은 SLP 와 거의 유사하다[7]. 먼저 특정 서비스를 제공하는 에이전트가 자신의 서비스를 DF 에 등록시키면, 해당 서비스의 사용을 원하는 다른 에이전트가 동일 DF 를 검색한 후 서비스의 정보를 얻는다. 일단 원하는 서비스의 정보를 얻은 에이전트는 MTS 를 통하여 해당 서비스를 제공하는 에이전트와 통신하게 된다. 통신에 이용되는 프로토콜은 HTTP(HyperText Transfer Protocol), IIOP(Internet Inter-ORB Protocol), RMI(Remote Method Invocation)등이 있다. [그림 2]는 FIPA-OS 에이전트 플랫폼의 시스템 구조와 앞서 언급한 일련의 작업 과정을 나타낸다.

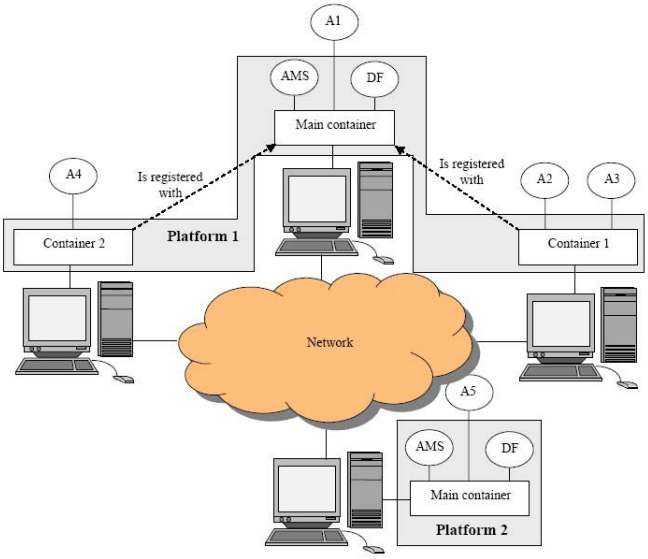


[그림 2] FIPA-OS 시스템 구조

FIPA-OS 에이전트 플랫폼의 현재 버전에서는 디스커버리 미들웨어에 대한 구현이 이루어지지 않았다. 본 논문에서는 FIPA-OS 이전트 플랫폼에 디스커버리 미들웨어 부분을 추가하여 이종 에이전트 플랫폼간의 상호 운용을 제공하고자 한다.

2.3 JADE

JADE(Java Agent DEvelopment Framework)는 FIPA 표준안을 기반으로 만들어진 에이전트 플랫폼으로서 기본 구조는 FIPA-OS 와 유사하다. FIPA-OS 와 같이 AMS, DF 가 존재하고, 컨테이너라고 하는 에이전트의 런타임 환경이 존재한다. 컨테이너는 MTS 와 ACC(Agent Communication Channel) 같은 통신 기능을 포함하며, 다수의 에이전트가 존재할 수 있다. 하나의 에이전트 플랫폼 상에는 반드시 하나의 메인 컨테이너가 있어야 하고, 메인 컨테이너의 하위에 여러 개의 컨테이너가 존재할 수 있다[5]. 이때 AMS 와 DF 는 반드시 메인 컨테이너에 존재해야 한다. JADE 의 서비스 디스커버리 매커니즘은 FIPA-OS 와 매우 유사하다. DF 를 통해서 서비스의 등록, 삭제 및 수정이 가능하고, 검색을 통해서 원하는 서비스를 찾을 수 있다. [그림 3]은 이와 같은 JADE 의 시스템 구조를 나타내고 있다.

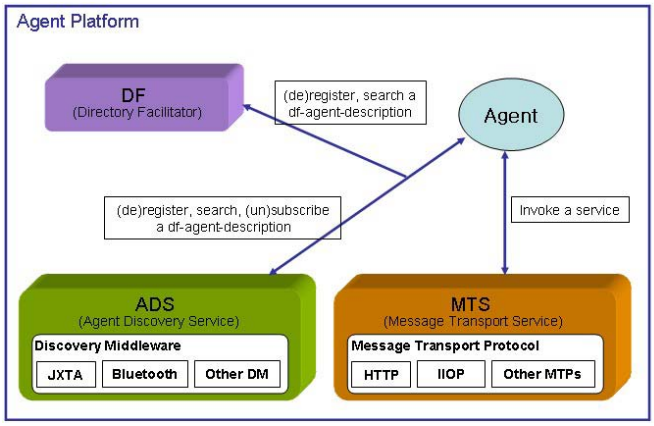


[그림 3] JADE 의 시스템 구조

JADE 에이전트 플랫폼 역시 FIPA-OS 와 마찬가지로 디스커버리 미들웨어 부분에 대한 구현은 현재 버전까지는 이루어지지 않고 있다. 따라서 이 부분에 대한 추가적인 확장이 필요하다.

3. 디스커버리 미들웨어

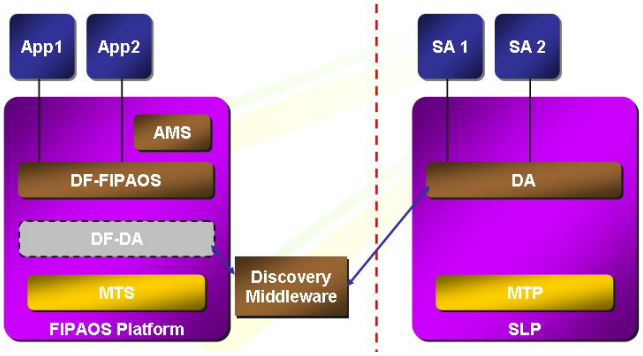
[그림 4]는 FIPA 에서 제안하는 디스커버리 미들웨어(Discovery Middleware)를 통한 서비스 디스커버리 매커니즘을 나타낸다[6]. 다른 에이전트 플랫폼에 등록된 서비스 이용을 위해서 디스커버리 미들웨어가 사용된다. 즉, DM 모듈은 에이전트 플랫폼의 호환 여부를 고려하지 않고도 상대방의 서비스를 어떠한 제약없이도 사용 가능하게 한다. 하지만 에이전트 플랫폼 상에서 디스커버리 미들웨어 부분이 실제로 구현되기 까지는 여러 가지 고려사항이 존재한다. 예를 들면, 이종 에이전트 플랫폼 간의 통신을 위한 프로토콜이 필요하고, 각각의 서비스를 등록하거나 갱신하는 작업에서 데이터의 일관성을 유지시키는 방법 등이다. 따라서, 특정 사항만을 고려하여 구현하고 있는 것이 일반적인 에이전트 플랫폼의 개발 추세이며, 본 논문에서는 이종 에이전트 플랫폼간 실질적인 서비스의 호출에 중점을 두어서 연구가 진행 되었다.



[그림 4] 디스커버리 미들웨어를 통한 서비스 디스커버리 매커니즘

3.1 FIPA-OS 와 SLP 그리고 JADE 의 상호운용을 위한 DM 의 개발

본 논문에서 제안하는 FIPA-OS 와 SLP 를 이용한 디스커버리 미들웨어의 구조는 [그림 5]와 같다.



[그림 5] SLP 를 이용한 DM 의 구조

FIPA 에서는 서비스에 대한 정보를 각자의 에이전트 플랫폼 상의 데이터베이스에 담고 있다. 따라서 이종 에이전트 플랫폼에 존재하는 서비스를 디스커버리 하기 위해서는 여러 단계의 커뮤니케이션 과정을 거쳐야 한다. 에이전트가 특정 서비스에 대한 요청을 ADS 에게 보내면, ADS 는 DM 에게 해당 서비스를 요청하게 된다. 다시 DM 이 이종 에이전트 플랫폼에 해당 서비스를 요청하게 되고, 이후의 응답과정은 위의 역순으로 이루어지게 된다. 즉, Agent ↔ ADS ↔ DM ↔ AP 형식의 통신이 이루어진다.

본 논문에서 제안하는 DM 의 구조는 기존 과정에 비해 디스커버리 절차를 단순화 함으로써, 쿼리에 대한 응답 시간을 단축 시킬 수 있다. 그림 5 와 같이

SLP 플랫폼 상의 서비스를 FIPA-OS 에이전트 플랫폼 상에 DF 의 형태로 미리 저장시켜 놓음으로써, 매번 요청이 이루어질 때 마다 발생하는 플랫폼 간의 커뮤니케이션 과정을 단순화 시킨다. 이렇게 되면 에이전트는 일단 자신의 DF 에서 서비스에 대한 검색을 하고, 매치되는 서비스가 없을 경우, 바로 자신의 에이전트 플랫폼에 저장된 이종 에이전트 플랫폼이 제공하는 서비스를 검색하게 된다. 이때 SLP 가 제공하는 서비스에 대한 정보는 FIPA-OS 에이전트 플랫폼이 처음 실행할 때 가져오게 되고, 이후 주기적으로 동기화 시켜준다.

3.1.1 시스템 설계

DM 의 설계 시 고려해야 할 사항은 다음과 같다.

첫째, 데이터베이스의 일관성을 유지해야 한다. 본 논문에서 제안한 DM 사용시 서비스의 갱신이나 추가, 삭제 시에 필연적으로 서비스 정보의 불일치 문제가 발생한다. 이 부분은 현재 연구가 진행되고 있으며, 해결 방안으로 타임 스탬프(Time stamp)를 이용하는 방법과 실시간 데이터베이스 동기화 방식을 고려하고 있다. 타임 스탬프를 이용하는 방법은 서비스가 등록될 때 서비스마다 타임 스탬프를 두어서 유효기간을 명시하는 방법이다. 유효기간이 지난 서비스는 더 이상 사용할 수 없게 되고, 일정 시간이 지나면 두 에이전트 플랫폼간의 데이터베이스를 동기화 시킨다. 실시간 데이터베이스 동기화 방식은 한쪽의 데이터베이스에 갱신이 발생할 경우 즉각 이를 다른 쪽의 데이터베이스에 반영하는 방법이다.

둘째, 서비스에 대한 표준화된 정의가 필요하다. FIPA-OS 에이전트 플랫폼에서 사용하는 서비스에 대한 정의와 SLP 에서 사용하는 서비스에 대한 정의가 각각 다르기 때문에 이를 매핑시킬 필요가 있다. 따라서 본 논문에서는 [표 1]과 같은 서비스 디스크립션 온톨로지(Ontology)를 정의한다.

[표 1] 서비스 디스크립션 온톨로지

Parameter	Description	Value
Print_Serv1	This is the printing service. Type of printer is laser printer.	Name : printer
		Vendor : HP
		Model Name : 2300dtn
		Type : laser
		Address : 203.252.53.107
Print_Serv2	This is the printing service. Type of printer is color printer.	Name :printer
		Vendor : HP
		Model Name : 1180C
		Type : color
		Address : 203.252.53.110

마지막으로 FIPA-OS 와 SLP 간 서로의 서비스를 이용하기 위해 서비스 호출 방식의 정의가 필요하다. 에이전트 플랫폼은 각기 다른 서비스 호출 방식을 사용하며, SLP 의 경우 응용 프로그램에 대한 지원을 해주지 않기 때문에 이를 위한 추가적인 작업이 필요하다. 현재 이 부분에 대한 연구가 진행되고 있으며, 소켓을 이용한 서비스의 호출 방식과 HTTP 를 이용한 웹 어플리케이션 형식의 서비스 방식을 고려하고 있다.

4. 결론 및 향후 연구 방향

유비쿼터스 컴퓨팅 환경에서 분산된 서비스나 장치들을 사용하기 위해서는 반드시 해당 서비스의 위치를 알 수 있게 해주는 서비스 디스커버리 기능이 필요하다. 본 논문에서 제안한 디스커버리 미들웨어를 적용함으로써 이종 에이전트 플랫폼상의 서비스를 동일 에이전트 플랫폼 상의 서비스처럼 사용이 가능하게 된다. 때문에 보다 폭넓은 멀티 에이전트 플랫폼 시스템을 구성할 수 있다. 또한 FIPA 에서 제안하는 디스커버리 미들웨어에 비해 디스커버리 절차를 단순화 함으로써, 쿼리에 대한 응답 시간을 단축 시키는 장점을 갖는다. 그러나 완전한 서비스를 위해서는 갱신이 발생하였을 경우, 플랫폼 간의 서비스 정보 불일치 문제나 적절한 SLP 어플리케이션의 개발 문제 등을 해결해야 한다.

이후의 연구는 앞서 설명한 플랫폼 간의 서비스 정보 불일치 문제의 처리 및 실제 서비스를 제공하는 SLP 어플리케이션의 개발에 초점을 맞춰서 진행될 것이다.

참고문헌

[1] James Kempf, Pete St. Pierre “Service Location Protocol for Enterprise Networks” Wiley Computer Publishing.

[2] N. Davies, A. Friday, S. P. Wade and G. S. Blair, L2imbo : A distributed systems platform for mobile computing, Mobile Networks and Applications, pp. 143-156, August 1998.

[3] M. Weiser, “Some Computer science issues in ubiquitous computing” Communications of the ACM, pp. 75-84, July 1993.

[4] N. Gibbins, W. Hall, “Scalability Issues for Query Routing Service Discovery”, Proceedings of the Second Workshop on infrastructure for Agents, MAS and Scalable MAS, 209-217, May 2001.

[5] JADE Tutorial Giovanni Caire.
http://jade.tilab.com/

[6] FIPA Agent Discovery Service Specification. Foundation for Intelligent Physical Agents, 2003.
http://www.fipa.org/specs/fipa00095

[7] FIPA-OS V2.1.0 Distribution Notes.
http://fipa-os.sourceforge.net

[8] Foundations for Intelligent Physical Agent.
http://www.fipa.org/

[9] Project JXTA.
http://www.jxta.org/