제25회 한국정보처리학회 춘계학술발표대회 논문집 제13권 제1호 (2006. 5)

에이전트를 이용한 온톨로지 기반의 서비스 디스커버리

신 준*, 김규일*, 황현식*, 김응모*
*성균관대학교 컴퓨터공학과
e-mail : crashjun@ece.skku.ac.kr

Ontology Based Service Discovery using Agents in the Service Block

Jun Shin*, Kyuil Kim*, Hyunsik Hwang* , Ungmo Kim*
*Department of Computer Engineering, Sungkyunkwan University,

요 약

유비쿼터스 컴퓨팅환경에서는 다양한 서비스가 존재한다. 사용자들은 수많은 서비스들을 사용하기에 앞서 이들이 어디에 있는지, 어떻게 사용해야 하는지를 알아야 한다. SLP, Jini, 그리고 UPnP 같은 대부분의 현존하는 서비스 디스커버리 (Service Discovery) 기술들은 키워드 매치 방법으로 서비스를 검색한다. 이 같은 방식에서는 사용자가 자신이 이용하고자 하는 서비스에 대한 정확한 명칭을 알고 있어야 서비스를 검색할 수 있다. 그러나 유비쿼터스 환경에서는 매우 다양한 서비스가 존재하고, 사용자들 대부분은 해당 서비스의 전문가가 아니기 때문에 정확한 키워드를 선택하여, 원하는 서비스를 찾는데 어려움이 있다. 시맨틱 웹 분야에서 새롭게 떠오른 온톨로지는 특정 분야에서 사용되는 정보의 구조 혹은 용어에 관한 내용을 사용자들이 공유할 수 있도록 정리해놓은 것이다. 본 논문에서는 이러한 온톨로지의 특성을 이용하여, 사용자에게 이용 가능한 서비스에 대한 정보들을 제공하여 사용자가 다양한 서비스를 쉽게 찾아 쓸 수 있는 방법을 제안하였다. 이 방식은 에이전트간의 온톨로지 교환을 통해서 이루어지며, 서비스 블록(Service Block)이라고 불리는 공간 안에서 수행된다. 또한 서비스 컨트롤러(Service Controller)를 이용하여 서비스를 쉽게 사용할 수 있도록 UI를 제공한다. 본 시스템을 위하여 JENA2, JADE, Aglet, OWL, 그리고 RDQL 이 사용되었다.

1. 서론

컴퓨팅 환경이 점점 유비쿼터화 되어감에 따라서 점점 많은 네트워크 기반의 서비스들이 생겨나게 된다. 사용자가 어떤 서비스를 이용하기 위해서는 그 서비스를 검색하는데 있어서 충분한 지식이 있어야만 한다. 그러나 실제적으로 사용자들은 전문가가 아니기 때문에 자신이 원하는 서비스에 대한 충분한 지식을 가지고 있지 못하다. 그렇기 때문에 더 많은 서비스가 존재하면 할수록, 사용자는 원하는 서비스를 콕 집어 찾기가 어려워진다. 그러므로 사용자로 하여금 해당 도메인상에 존재하는 서비스에 대해 학습하도록 할 필요가 있다. 온톨로지는 특정 분야에서 사용되는 정보의 구조 혹은 용어에 관한 내용을 사용자들이 공유할 수 있도록 정리해놓은 것이다. 사용자에게 온톨로지를 제공함으로써, 사용자는 서비스들에 대한 정보를 얻고 이를 기반으로 자신이 원하는 서비스를 쉽게 찾아 쓸 수 있다. 한편, 사용자에게 너무 많은 정보를 제공하면 사용자가 이를 감당할 수 없게 되므로 그 정보의 사이즈가 중요하다. 본 논문에서는 사용자의 입장을 고려한 서비스 디스커버리에 초점을 두고 있으며, '서비스 블록' 이라는 개념을 도입하였다. 또한 에이전트를 이용한 온톨로지의 교환 및 서비스 전달 (Service Delivery) 방법을 제시한다. 논문이 제안한 시스템을 위해서 JADE[1]와 Aglet[2]이 에이전트간 통신을 위해서 이용되었으며, 온톨로지를 다루기 위해서 JENA[3]와 OWL[4]이 사용되었다. 에이전트들은 FIPA에서 정의한 ACL[11] 메시지를 이용하여 통신한다.

2. 관련연구

전통적으로 사용자는 자신이 원하는 서비스를 이용하기 위해서 해당 서비스의 네트워크 주소 혹은 호스트 이름을 알고 있어야 했다. 주요 서비스 디스커버리 프로토콜 중의 하나인 SLP[5]는 사용자로 하여금 원하는 서비스를 쉽게 찾기 위해서 도와준다. 더 자세하게는 원하는 서비스의 네트워크 주소를 찾기 쉽게 도와준다. 그러나 SLP 는 단순한 카테고리만을 지원하며, 모든 사용자가 자신이 찾고자 하는 서비스의 정확한 이름, 정확한 키워드를 알고 있어야 한다. 한편 UPnP[6]는 특정한 XML 포맷을 이용하여 장비들을 묘사(describe)하는데 있어서 더 많은 정보를 표현할 수 있게 했다. 그러나 이 디스크립션들은 반드시 현존하는 템플릿에 이반해야 하며 UPnP 포럼에서 동의를 얻어야 사용할 수 있다. Jini[7]는 서비스를 단지프로그래밍 관점에서만 기술하였다. Jini 에서의 서비스 디스크립션은 Java 인터페이스들로 나타낸다. OCTOPUS[8]는 Jini 의 서비스 디스커버리 모듈을 확장시켰다.

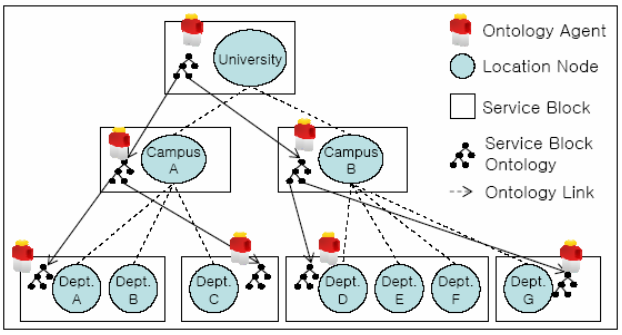
결국 기존의 서비스 디스커버리 프로토콜들은 사용자 측면이 아닌 프로그래밍 혹은 다른 측면들이 강조되었으며, 사용자에게 서비스를 쉽게 찾기 위한 정보를 제공하지 않았다고 말할 수 있다. 그러나 유비쿼터스 환경에서는 매우 다양한 서비스들이 존재 하기 때문에 사용자는 그만큼 서비스들에 대한 정보가 부족하다. 이를 해결하기 위해서 온톨로지를 사용할 수 있는데, 현재는 온톨로지를 이용한 서비스 디스커버리에 관한 연구가 부족한 실정이다. [9]에는 온톨로지가 왜 사용되는지 그 이유에 관해서 설명한다. 하나는 특정 분야에서 사용되는 정보의 구조 혹은 용어에 관한 내용을 소프트웨어 에이전트나 사용자들이 공유하기 위함이고, 또 다른 이유는 특정 분야에서 암묵적인 동의하에 공유되는 지식들을 명시적으로 기술하기 위함이다. 본 논문에서는 온톨로지를 이용하여 사용자에게 서비스를 검색하기 쉽도록 적당한 정보를 제공하고, 또한 검색된 서비스를 이용하기 쉽도록 이동 에이전트 기술을 도입하였다.

3. 서비스 디스커버리를 위한 온톨로지 교환 에이전트 시스템

제안된 시스템을 이해하기 위해서 5 가지의 주요 개념을 먼저 정의한다. 그림 1 에서는 Location Hierarchy 와 서비스 블록에 대한 개념을 나타내며 아래 정의 1,2 에서 각각을 정의한다.

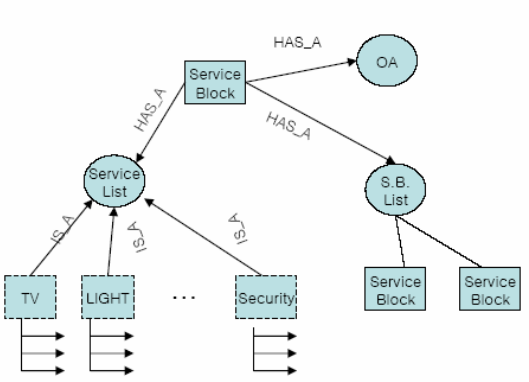
정의 1 *(Location Hierarchy). Location Hierarchy 는 여러 개의 Location 노드로 구성되며 각각의 노드들은, 그 부모 노드와 IS-PART-OF 관계를 가진다. 아래에 정의된 서비스 블록은 하나 혹은 그 이상의 노드들로 구성될 수 있다.*

정의 2 *(서비스 블록). 서비스 블록(Service Block)은 온톨로지 에이전트(Ontology Agent, OA)에 의해서 관리되는 하나의 공간을 의미한다. 각각의 서비스 블록은 하나의 온톨로지 에이전트 그리고 하나의 서비스 블록 온톨로지를 가진다*



(그림 1) Location Hierarchy 와 Service Block

. **정의 3** *(온톨로지 에이전트). 온톨로지 에이전트(Ontology Agent)는 하나의 서비스 블록 온톨로지를 관리한다. 서비스 블록 온톨로지는 특정 서비스 블록에서 이용 가능한 서비스들에 대한 정보를 포함하고 있다.*



(그림 2) 서비스 블록 온톨로지

서비스 블록은 서비스의 관리를 위한 공간이며, 한 개의 방, 한 개의 부서, 혹은 건물 하나등과 비슷한 개념이다. 서비스 블록들은 트리 구조를 형성하며, 서비스 블록을 관리하는 온톨로지 에이전트(OA)역시 트리 구조로 되어있다. 서비스 블록 온톨로지는 (그림 1)에서 보여지는 것같이 연결되어 있다. 전체 서비스에 관한 온톨로지는 이들 서비스 블록 온톨로지의 링크들을 모두 연결함으로써 얻어질 수 있다. 전체의 서비스 온톨로지의 나눠진 한 조각을 서비스 블록 온톨로지라 정의하고 그 하나의 서비스 블록 온톨로지를 OA 가 관리를 하며, 하나의 OA 가 관리하는 구역을 서비스 블록이라고 한다. 이러한 트리 구조는 높은 범위성(Scalability)을 가진다. 정의에 의하면, 온톨로지는 특정 도메인상에서 암묵적인 동의 하에서 사용되는 용어들 혹은 지식들을 명시적으로 열거한 것이다. 이는 특정 서비스 블록에 들어온 새로운 사용자로 하여금 해당 서비스 블록내의 지식들 즉, 서비스에 대한 정보들을 배우게 함으로써 필요한 서비스를 손쉽게 찾을 수 있도록 하는데 온톨로지가 이용될 수 있다는 것이

다.

(그림 2)는 온톨로지 에이전트에 의해서 관리되는 서비스 블록 온톨로지의 예이다. 각각의 온톨로지 에이전트는 서비스 블록 온톨로지를 가지고 있으며, 특정 서비스 블록에서 새로운 서비스가 생성 혹은 삭제될 때 이에 대응되는 서비스 블록 온톨로지를 업데이트한다. 서비스 블록 온톨로지에는 서비스들에 대한 정보뿐만 아니라, 다른 서비스 블록과의 계층관계도 포함되어 있다.

정의 4 *(서비스 블록 온톨로지). 서비스 블록 온톨로지(Service Block Ontology)는 어떤 서비스 블록 안에서 이용 가능한 서비스들에 대한 정보 및 다른 서비스 블록과의 관계를 포함하고 있다*

정의 5 *(서비스 컨트롤러). 서비스 컨트롤러(Service Controller)는 이동 에이전트로써 특정 서비스의 인터페이스 역할을 한다. 사용자는 서비스 컨트롤러를 통해서 실제 서비스를 이용 혹은 조작할 수 있다.*

서비스 컨트롤러(SC)는 이동 에이전트이며, 이것은 텔레비전이나 라디오 등의 리모컨과 비슷한 역할을 한다. 이동에이전트는 네트워크를 통해서 호스트와 호스트간을 이동하며, 그 코드(Code)와 데이터가 함께 이동한다. 그러므로 사용자가 서비스 디스커버리 단계에서 서비스를 찾은 다음, 이동에이전트가 사용자의 장비(device)로 이주한 뒤, 사용자로부터 실행에 필요한 정보들을 얻어 실제 서비스가 존재하는 호스트로 돌아가 서비스를 수행하도록 한다. 이러한 이동 에이전트의 특성을 이용하여 서비스에 사용성 (usability)를 향상 시킬 수 있다.

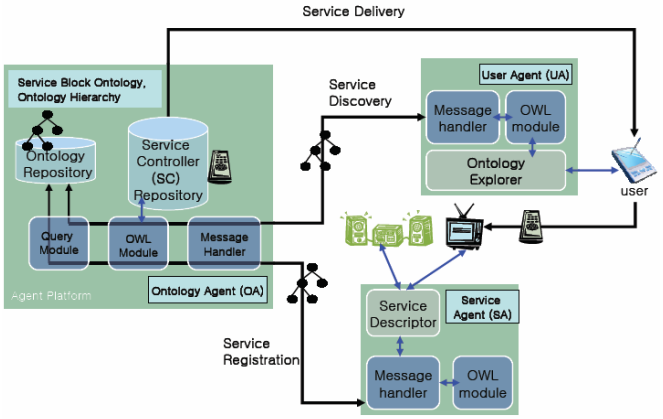
가정 1 *본 논문에서 이용된 모든 객체들은 자신의 위치에 대한 x,y,z 좌표 값을 가지고 있다. 이는 온톨로지 에이전트가 IPv6 에서 Anycast 주소를 가지는 노드처럼 수행하기 위함이다.*

(그림 3)은 논문에서 제안하는 시스템을 도식화 한 것으로써, 온톨로지 에이전트(OA), 서비스 에이전트(SA), 사용자 에이전트(UA), 그리고 서비스 컨트롤러(SC)를 포함한다. 이들 에이전트들은 FIPA 에서 정의한 ACL 메시지를 이용하여 통신한다.

OA 는 사용자 혹은 다른 에이전트들(SA 와 UA)과 온톨로지 정보를 공유하기 위해서 사용된다. 특정 서비스 블록의 서비스 정보를 얻기 위해서 OA 에게 쿼리 메시지를 보내면 OA 가 이에 응답한다. 쿼리 메시지는 ACL 메시지의 내용(Content)필드에 포함되며, RDQL[12]을 사용한다. 응답 역시 ACL 메시지에 포함되며 이는 RDF 혹은 OWL 의 형태를 가진다.

SA 는 서비스의 등록을 수행하는 에이전트이다. 새로운 서비스를 OA 를 통해서 온톨로지 저장소에 저장하기 위해서 SA 는 서비스 디스크립션 폼을 OA 로부터 얻어온다.

서비스 디스크립션 폼(Service Description form)은 특정 서비스의 네트워크 주소, 이름, 용도, 위치 등의 정보를 담기 위한 미리 정의된 양식이다. SA 는 자신이 등록하려는 서비스의 정보들을 서비스 디스크립션 폼에 적어서 서비스 디스크립션을 작성한다. 그리고 완성된 서비스 디스크립션을 OA 를 통해서 등록한다. 서비스를 등록하는 시점에서 SA 는 SC 를 UA 의 SC 저장소로 이주시킨다.



(그림 3) 서비스 디스커버리 아키텍처

UA 는 사용자에게 특정 서비스 블록의 정보를 제공하기 위해서 사용된다. UA 는 RDQL 을 이용한 쿼리 메시지를 OA 에세 전달하고 이에 대한 응답으로 오는 OWL 정보로부터 서비스에 대한 정보를 얻는다. 사용자가 UA 를 이용하여 원하는 OA 로부터 서비스 블록 온톨로지를 받는 메커니즘은 DNS[10]에서의 재귀적 방법(Recursive Resolution)과 같다. 그러므로 사용자는 오직 자신이 현재 위치한 서비스 블록을 담당하는 OA 와 통신을 한다. (그림 3)에서 온톨로지 익스플로러(Ontology explorer)는 OWL 로 쓰여진 서비스 블록 온톨로지를 사용자에게 시각적으로 보여준다. UA 를 통해서 현재 사용자의 위치에서 사용 가능한 서비스에 대한 정보를 얻은 뒤, 사용자가 원하는 서비스를 선택해서 OA 에게 이를 알리면, OA 가 SC 저장소에 이주되어있는 해당 SC 를 사용자의 장비로 이주시킨다. 최종적으로 사용자는 이주된 SC 에게 서비스의 실행에 필요한 정보들을 입력하고, 이 SC 가 자신의 원래 호스트 즉, 실제 서비스가 있는 곳으로 데이터를 가지고 이주하여 서비스를 실행하게 된다.

4. 프로토타입과 예제

논문에서 제안된 시스템은 CALM[13](Component-based Autonomic Layered Middleware) 프로젝트의 일부로서 현재 개발 중에 있으며, 프로토타입을 위해서 JADE 와 Aglet 를 이용하였다. UA, SA 그리고 OA 는 JADE 기반의 에이전트 이며, JADE 가 현재까지는 이동에이전트를 지원하지 않기 때문에, Aglet 을 이용하여 SC 를 구현하였다. 온톨로지를 다루기 위해서 JENA2 를 사용하였으며, 쿼리 메시지로는 RDQL 을 이용하였다.

(그림 4)의 윗부분은 UA 가 OA 에게 보내는 ACL 메시지이며, 아랫부분은 이에 해당하는 응답 메시지의

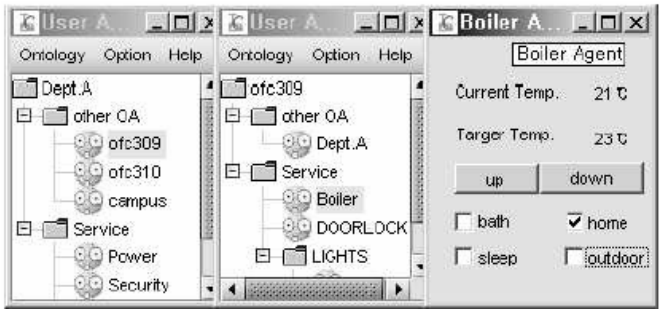
예이다. (그림 5)는 UA 가 OA 로부터 얻은 서비스 블록 온톨로지를 화면으로 나타내는 것과 최종적으로 SC 가 사용자의 장비로 이주되어 서비스의 실행에 필요한 정보를 얻어가는 모습이다.

(그림 4)와 (그림 5)를 위한 시나리오는 다음과 같다. 갑돌이는 현재 그의 사무실에서 퇴근하면서, 사무실내의 온도 조절기를 끄는 것을 깜빡 잊었다. 그는 사무실로 돌아가는 대신, 자신의 UA 를 실행시킨다 (UA 는 PDA 에서 동작한다). UA 는 가까운 OA 를 찾아서 현재 갑돌이가 있는 서비스 블록의 서비스 블록 온톨로지를 가져온다. 첫 번째로 가져온 서비스 블록 온톨로지에는 갑돌이 사무실의 서비스들이 나오지 않으므로, 갑돌이는 다른 서비스 블록을 담당하는 OA 를 선택하고, 자신의 사무실을 담당하는 OA 의 서비스 블록 온톨로지를 얻어온다. 이제 여기서 온도조절을 담당하는 서비스를 선택하고, 이 서비스의 SC 가 갑돌이의 PDA 로 이주한다. 갑돌이는 이 SC 를 이용하여 온도조절기를 끄고 가던 길을 간다.

```
(QUERY-REF
  :sender (agent-identifier :name UserAgent@icarus:1099/JADE
            :addresses (sequence http://icarus:7778/acc ))
  :receiver (set ( agent-identifier :name OntologyAgent@icarus:1099/JADE ) )
  :content
     "SELECT ?x WHERE (?x rdf:type ont:SERVICE)
     USING
        ont FOR <http://dblab.skku.ac.kr#>
        rdf FOR ...   rdfs FOR ...
  :language RDQL :ontology http://dblab.skku.ac.kr )
:ontology http://dblab.skku.ac.kr )
```

```
(INFORM
:sender ... :receiver ...
:content "<rdf:RDF
xmlns:rdf=\http://www.w3.org/1999/02/22-rdf-syntax-ns#\
  ...
<owl:Class rdf:about=\"http://dblab.skku.ac.kr#Thermostat\">
<rdfs:subClassOf rdf:resource=\"http://dblab.skku.ac.kr#SERVICE\"/>
 </owl:Class>
<ofc309:Boiler rdf:about=\"http://dblab.skku.ac.kr#Boiler\">
<ofc309:cmt>thermostat of office 309 </ofc309:cmt>
<ofc309:agentpath>ofc309.service.Boiler</ofc309:agentpath>
 ...
 </ofc309:LIGHT>
...
```

(그림 4) 온톨로지 교환을 위한 ACL 메시지의 예



(그림 5) UA 와 SA 의 스크린 샷

5. 결론

본 논문에서는 온톨로지를 이용한 서비스 디스커버가 다루어졌다. 또한 온톨로지의 교환을 위해서 에이전트를 이용하였으며, 다른 연구에서는 잘 다루지 않는 서비스 전달(Service Delivery)을 고려하였다. 처음에 언급했듯이, 기존의 서비스 디스커버리에서는 사용자가 반드시 자신이 원하는 서비스에 대한 정확한 키워드를 알고 있어야 검색이 가능했다. 본 논문에서는 대화식의 온톨로지 교환방식을 통해, 사용자에게 서비스들의 정보를 제공함으로써, 사용자가 원하는 서비스를 찾을 수 있도록 유도하였다. 또한 서비스 컨트롤러라는 이동에이전트를 이용하여 서비스 이용을 편리하게 사용하도록 하였다. 다만, 사용자가 원하는 서비스를 찾을 때까지, 몇 번이고 쿼리와 응답의 단계를 거쳐야 하는 단점이 있다. 그러나 이는 캐쉬를 기용하거나, 상황인지 기술을 이용하여 단계를 대폭 축소할 수 있다. 현재는 온톨로지 기반의 상황 인식 서비스 디스커버리가 연구되고 있다.

참고문헌

[1] Java Agent DEvelopment Framework(JENA), http://jade.tilab.com/
[2] Aglets, http://aglets.sourceforge.net/
[3] JENA, http://jena.sourceforge.net/
[4] World Wide Web Consortium. "OWL Web Ontology Language Reference", W3C Recommendation 10 Feb, 2004.
[5] E. Guttman, C.Perkins, J.Veizades, "Service Location Protocol, version 2",RFC 2608.
[6] "UPnP White Paper", http://upnp.org/resources.htm/,June 2000.
[7] K. Arnold, R. Schei°er, J. Waldo, B. O'Sullivan, and A.Wollrath, "The Jini Specifi-cation", V1.1, 1999.
[8] J. K. Yao, T. Gu, H. K. Pung. "A Jini-based Service Location Manager in OCTOPUS", In Proceedings of the 7th IASTED International Conference on Inter-net and Multimedia Systems and Applications (IMSA 2003), pp. 751-756. Honolulu, Hawaii, August 2003.
[9] Natalya F. Noy and Deborah L. McGuinness,"Ontology Development 101: A Guide to Creating Your First Ontology",Stanford Knowledge Systems Laboratory Technical Report KSL-01-05. March 2001.
[10] P. V. Mockapetris. "Domain names: Concepts and facilities", RFC 882, Internet Engineering Task Force, Nov. 1983.
[11] FIPA, "FIPA ACL Message Structure Specification", Foundation for Intelligent Physical Agents, 2002. http://www.fipa.org/specs/fipa00061/
[12] A. R. L. Miller, A. Seaborne, "Three implementations of SquishQL, a simple RDF query language", Technical Report HPL-2002-110, Hewlett Packard Laboratories, Palo Alto, California, 2002.
[13]. Ubiquitous computing Technology Research Institute. CALM (Component-based Autonomic Layered Middleware), http://utri.skku.ac.kr/.