유비쿼터스 웹서비스 환경의 동적 서비스 검색을 위한 레지스트리 구성 요소 설계*

임형준○ 오일진 황윤영 이규철
충남대학교 컴퓨터공학과
{hyungjun25○, victory25, yyhwang, kclee}@cnu.ac.kr

Design Ubiquitous Web Services Registry For Dynamic Services Discovery in Ubiquitous Environment

Hyung-Jun Yim○ Il-Jin Oh Yun-Young Hwang Kyu-Chul Lee
Dept. Computer Engineering, Chungnam National University

요 약

유비쿼터스 웹서비스 환경에서는 이질적인 환경에 대한 서비스의 상호운용성을 위해 다양한 분산 시스템의 네트워크 프로토콜과 디바이스간의 통신을 통합하기 위한 연구가 요구된다. 이질적인 네트워크 프로토콜의 연동과 디바이스간의 통합을 위해 이들의 공통적인 명세 요소를 추출하여 레지스트리에 등록함으로써 동적 서비스 검색을 지원할 필요가 있다. 유비쿼터스 웹서비스 환경의 동적 서비스 검색은 이질적인 네트워크 프로토콜과 디바이스의 공통적인 명세 요소, 정적인 명세 요소, 필수적인 명세 요소를 포함하는 레지스트리의 검색과 네트워크 프로토콜이나 디바이스의 직접검색을 제공한다. 이를 통해, 본 논문에서는 유비쿼터스 웹서비스 환경에서의 동적 서비스 검색을 위한 레지스트리의 구성 요소를 설계함으로써 이질적인 네트워크 프로토콜과 디바이스를 효과적으로 통합하는 방법을 제안하고자 한다.

1. 서 론

컴퓨팅 기술과 네트워크 기술의 급격한 발전으로 인해 유비쿼터스 컴퓨팅 환경에 대한 연구가 활발하다. 이에 따라 유비쿼터스의 이질적인 환경에 대한 상호호환성을 위해 웹서비스를 적용하고 있다. 웹서비스는 서비스 공급자, 수요자, 비즈니스 등에 인터넷을 통해 분산된 컴포넌트를 접근하고 사용할 수 있도록 한다. 유비쿼터스 컴퓨팅 환경과 웹서비스의 특징을 혼합한 유비쿼터스 웹서비스 환경에서는 트랜잭션, 보안, QoS, 시멘틱, 웹서비스 구성 등 여러 분야로 연구되고 있다[1].

이질적인 환경의 상호 호환성을 위해 웹서비스가 해결책으로 제시되고 있지만, 네트워크 프로토콜과 디바이스간의 통합에 대한 해결책은 아직 연구가 이루어지고 있다. 이질적인 환경의 네트워크에서 하나의 프로토콜이나 특정한 물리 계층만으로 통신을 하기는 매우 어렵다. 따라서 다양한 네트워크 프로토콜과 물리 계층에 대한 특성을 반영하여 서로 다른 기기들끼리 연결을 위한 미들웨어에 대한 요구가 증가하고 있다. 이러한 기술의 미들웨어 형태인 네트워크 프로토콜은 UPnP(Universal Plug-and-Play), JINI(Java intelligent network infra-structure), Havi(Home Audio/Video Interoperability) 등이 대표적이고, 디바이스간의 통합을 위한 연구로는 DPWS(Devices Profile for Web Services)가 있다(본 논문에서는 네트워크 프로토콜을 Sub-network로 지칭한다). 그러나 하나의 Sub-network가 다양한 물리 계층에 대한 특성을 반영하고 네트워크를 관리할 수 없기 때문에 다수의 Sub-network를 사용하는 것이 산재한 문제들을 쉽게 해결해 나갈 수 있는 방법이다[2].

최근 다양한 네트워크 표준과 Sub-network 표준을 통합 관리하기 위해 OSGi(Open Service Gateway Initiative) 플랫폼이 등장하게 되었으며, 그 역할은 서비스를 로컬 네트워크나 디바이스에게 전달하고 전달된 서비스를 운용하는 개방적 표준을 만드는데 있다[3]. 플랫폼 독립적이며 이기종간 호환성이 가능한 OSGi 서비스 플랫폼은 다양한 Sub-network 표준과 디바이스간의 통합에 대한 해결책으로 제시되고 있지만, Sub-network마다 이를 처리해 주는 번들(Bundle)이 필요하다는 문제점을 가진다. 또한, 자바 기반으로 구성이 되어있기 때문에 유비쿼터스의 경량화된 컴퓨팅 환경에 적합하지 않다. 이를 해결하기 위해, 본 논문에서는 Sub-network와 디바이스의 공통적인 명세 요소, 정적인 명세 요소, 필수적인 명세 요소를 추출하여 레지스트리의 구성 요소를 설계 한다. 이를 기반으로 유비쿼터스 웹서비스 환경에서의 이질적인 Sub-network와 디바이스간의 통합된 동

* 본 논문은 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성·지원 사업의 지원을 받아 수행된 연구임 (IITA-2005-C1090-0502-0016)

적 서비스 검색을 제안한다.

본 논문은 2장에서 유비쿼터스 웹서비스 환경에서 사용하는 이질적인 Sub-network와 디바이스간의 통합을 위한 JINI, UPnP, DPWS에 대한 연구들을 설명하고, 3장에서 본 논문에서 제시하는 레지스트리의 구성 요소를 설계하기 위한 Sub-network와 디바이스의 명세 항목을 설명한다. 4장에서는 유비쿼터스 웹서비스 환경의 동적 서비스 검색 과정에 대해 기술하고, 5장에서는 유비쿼터스 웹서비스 환경의 동적 서비스 검색을 위한 레지스트리의 구성 요소를 설계한다.

2. 관련 연구

본 논문에서 제안된 유비쿼터스 웹서비스 환경의 동적 서비스 검색을 위해 레지스트리를 구성하기 위한 공통 요소를 추출할 수 있는 Sub-network와 디바이스는 많다. 하지만, 그 중 이질적이고 상호운용성의 특성을 반영하여 JINI, UPnP, DPWS만을 고려한다[표1][4]. 그 외의 Sub-network와 디바이스는 추후 추가할 계획이다.

표 1 디바이스 통합을 위한 관련 기술의 특징

	OSGi	HAVi	JINI	UPnP	WS	DPWS
Plug and Play	-	○	○	○	-	○
Device support	○	○	○	○	-	○
Programming Language independent	-	○	-	○	○	○
Network media independent	-	-	○	○	○	○
Large scalability	○	-	○	-	○	○
Security	○	○	○	-	○	○
High market acceptance	○	-	○	○	○	○

2.1 JINI

JINI(Java intelligent network infra-structure)는 썬에서 자바 기반으로 네트워크에 접속된 지능형 디지털 디바이스들이나 소프트웨어들이 동적으로 상호 작용을 할 수 있는 네트워크 프로토콜이다. JINI의 구조는 서비스 제공자, 관리자, 사용자로 관계성을 나타내며, 자바 클래스 통신인 RMI(Remote Method Invocation)를 통한 통신이 이루어진다[그림1].

- 서비스 제공자(Service Provider): 서비스 관리자(Lookup Service)에 서비스를 제공하여 사용자들이 서비스를 이용
- 서비스 관리자(Lookup Service): JINI 시스템을 전체적으로 관리, 운용, 제어하며 서비스 이용자와 서비스 제공자를 서로 연결시켜주는 역할을 수행하는 JINI 연합체
- 서비스 사용자(Client): 자신이 원하는 서비스가 있는 서비스 관리자(Lookup Service)에게 등록되어 있는 서비스를 검색해 원하는 서비스를 획득

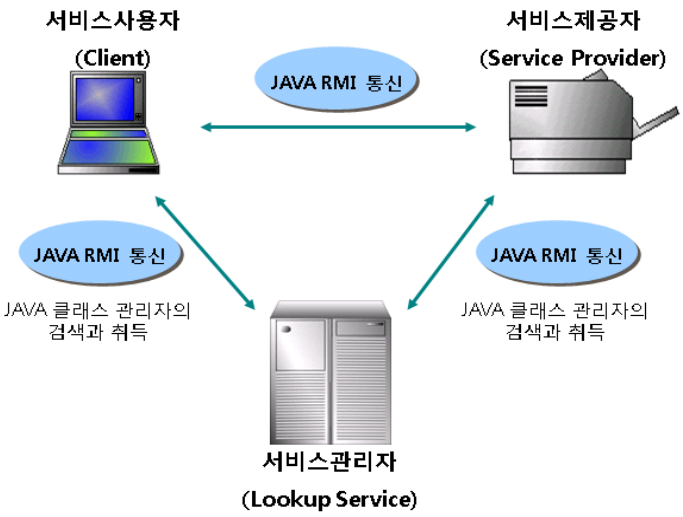

그림 1 JINI(Java intelligent network infra-structure)의 구조

디지털 디바이스들은 하드웨어의 종류에 상관없이 네트워크에 접속할 수 있으며, 객체와 에이전트 기반의 분산객체 컴퓨팅 환경으로 변화되었다[5].

2.2 UPnP

UPnP(Universal Plug-and-Play)는 마이크로소프트에서 표준화작업을 하였으며, ad-hoc 환경의 네트워크에 연결된 디바이스, 서비스 등이 P2P(Peer-to-Peer)방식의 프로토콜 스택을 통해 통신, 검색, 관리할 수 있도록 한다[그림2]. 고유한 IP 주소를 기반으로 인터넷 프로토콜을 이용한 디바이스들 간의 통신이 가능하며, 운영체제, 프로그래밍 언어, 물리적 디바이스에 독립적으로 구성할 수 있다[6].

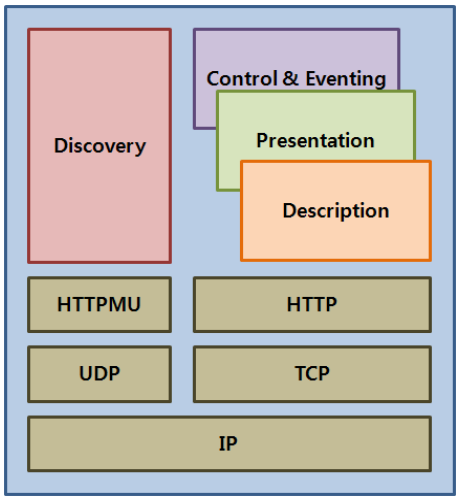

그림 2 UPnP(Universal Plug-and-Play) 프로토콜 스택

2.3 DPWS

DPWS(Devices Profile for Web Services)는 UPnP의 차세대 버전으로 네트워크에 연결된 디바이스에 적합한

경량의 웹서비스 프로토콜을 부분적으로 명시하여 사용한다[그림3].

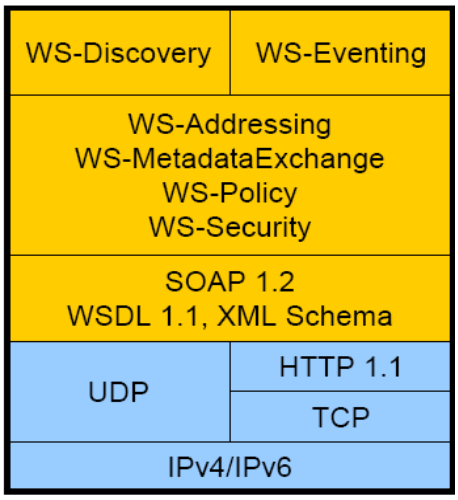

그림 3 DPWS(Devices Profile for Web Services) 프로토콜 스택

디바이스 프로파일은 다음의 네트워크 기능들을 디바이스에서 사용할 수 있는 웹서비스 표준에 대해 규정하고 있다.

- 동적 웹서비스 발견
- 웹서비스의 Send/Response 메시지 보안
- 웹서비스 명세
- WSDL(Web Services Description Language)를 이용한 웹서비스 디바이스간의 통신
- 웹서비스 이벤트 처리

또한, 인터넷과 네트워크에 연결된 디바이스들이 웹서비스와 애플리케이션 등 독립적으로 통신할 수 있도록 개방된 인터페이스를 제공하고 있다[7].

3. Sub-network의 디바이스와 서비스 명세 요소

하나의 Sub-network가 다양한 물리 계층에 대한 특성을 반영하고 네트워크를 관리할 수 없기 때문에 다수의 Sub-network를 사용하는 것이 산재한 문제들을 쉽게 해결해 나갈 수 있는 방법이다. 유비쿼터스 웹서비스 환경에서 다수의 이질적인 Sub-network를 사용하기 위해 각각의 Sub-network가 디바이스와 서비스를 명세 하는 항목에 대해 살펴본다.

3.1 JINI의 디바이스와 서비스 명세 요소

JINI는 클래스를 이용하여 서비스와 디바이스에 대하여 명세를 한다. 서비스 제공자가 LUS(Lookup Service)에 서비스 객체와 애트리뷰트를 등록하게 되는데 서비스 객체는 서비스를 사용하기 위한 프록시 정보들로 이루어져 있으며, 애트리뷰트는 서비스에 대한 명세를 담게 된다[표2].

- 애트리뷰트의 구성
  - 기본 서비스 정보(Basic Service Information): 서비스에 대한 기본적인 정보를 제공하기 위한 클래스를 제공한다.
  - 자세한 서비스 정보(More Specific Information): 서비스에 대한 일반적인 정보보다 특별한 인스턴스에 대해 명세를 한다.
  - 서비스 이름(Naming a Service): 서비스의 이름을 명세를 한다.
  - 서비스 기술(Adding a Comment to a Service): 서비스에 대한 간단한 설명을 기술한다.
  - 물리적인 위치 정보(Physical Location): 서비스의 물리적 주소 정보를 제공한다.
  - 상태 정보(Status Information): 서비스에 따라 기술되는 정보가 다르며, 서비스에 대한 상태 정보를 제공한다.

표 2 JINI의 디바이스와 서비스 명세 요소 분류

	명세 요소 이름	설 명
필수 명세 요소	<manufacturer>	제조회사 이름
	<model>	제품 모델 이름
	<serialNumber>	제품 번호
	<serviceType>	서비스 이름
	<serviceID>	서비스 아이디
	<statusInformation>	서비스 상태 정보
	<version>	제품 버전
	<name>	제품 이름
	<vendor>	판매회사 이름
추천 명세 요소	<name>	사용하기 편한 이름
	<Location>	서비스의 물리적 상세 주소
	<Address>	서비스의 물리적 주소
선택 명세 요소	<comment>	서비스에 대한 간단한 설명

3.2 UPnP의 디바이스와 서비스 명세 요소

UPnP는 물리적 컨테이너와 논리적 컨테이너를 명세하는 디바이스 명세와 하나 또는 그 이상의 서비스에 대한 명세를 하는 부분으로 나눌 수 있다.

하나의 물리적 디바이스는 다수의 논리적인 디바이스를 포함할 수 있다. 다수의 논리적인 디바이스는 하나의 최상위 물리적 디바이스와 같은 형태로 물리적 디바이스에 포함되거나 논리적 디바이스가 내장되어 있지 않은 다수의 최상위 물리적 디바이스로 존재한다. 하나의 물리적인 디바이스에는 내장된 모든 논리적 디바이스 명세 정보가 포함된다. 논리적 디바이스가 내장되어 있지 않은 디바이스는 최상위 디바이스로 자신에 대한 명세 정보만 다룬다[그림4].

UPnP의 디바이스 명세는 제조회사에 의해 제공된다. UPnP 포럼에서 제공하는 템플릿을 기반으로 제품명, 제품번호, 제조회사, 판매회사의 웹 사이트 URL 정보 등을

XML으로 표현한다. 디바이스에 포함된 각각의 서비스에 대한 명세도 UPnP 포럼에서 제공하는 템플릿을 기반으로 서비스 유형, 서비스명, 관리와 이벤팅을 위한 URL 정보를 포함한다[표3].

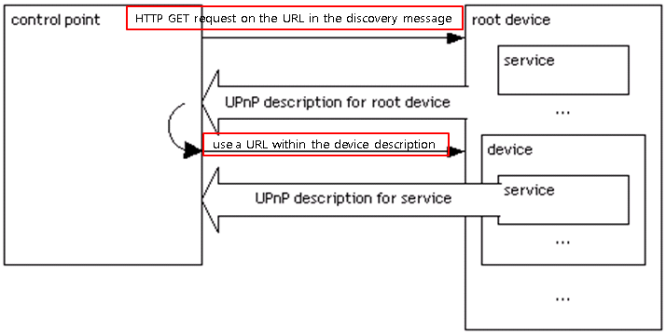

그림 4 물리적 디바이스와 논리적 디바이스와의 관계

표 3 UPnP의 디바이스와 서비스 명세 요소 분류

	명세 요소 이름	설 명
필수 명세 요소	<friendlyName>	사용하기 편한 이름
	<manufacturer>	제조회사 이름
	<modelName>	제품 모델 이름
	<deviceType>	디바이스 이름
	<UDN>	UUID
	<SCPDURL>	서비스 명세 URL
	<controlURL>	컨트롤 URL
	<eventSubURL>	이벤트 URL
	<serviceType>	서비스 이름
	<serviceID>	서비스 아이디
	<serviceStateTable>	서비스 상태 정보 테이블
추천 명세 요소	<modelNumber>	모델 번호
	<serialNumber>	제품 번호
	<modelDescription>	모델 명세
	<PresentationURL>	사용자 UI 페이지 URL
선택 명세 요소	<manufacturerURL>	제조회사 URL
	<modelURL>	모델 URL
	<UPC>	제품 코드
	<icon>	아이콘

3.3 DPWS의 디바이스와 서비스 명세 요소

서비스 사용자는 디바이스의 확장성과 기능에 대한 관리를 위해 디바이스와 디바이스에서 제공하는 서비스를 검색할 필요가 있다. 일반적인 웹서비스와 같이, DPWS는 XML 스키마, WSDL, WS-Policy로 기술하며 디바이스와 서비스에 대한 특징과 호스팅으로 분류하여 명세한다. 디바이스의 특징에 대한 명세는 제조회사에 의해 고정된 정보를 담고 있는 ThisModel과 하나의 디바이스에서 여러 정보를 담을 수 있는 ThisDevice로 나눌 수 있다. 또한, 디바이스와 디바이스에서 제공하는 서비스의 관계에 대한 기술과 WS-Policy를 통한 프로파일의 제약사항을 명시하고 있다[표4].

표 4 DPWS의 디바이스와 서비스 명세 요소 분류

	명세 요소 이름	설 명
필수 명세 요소	<friendlyName>	사용하기 편한 이름
	<manufacturer>	제조회사 이름
	<modelName>	제품 모델 이름
	<serviceType>	서비스 이름
	<serviceID>	서비스 아이디
추천 명세 요소	<modelNumber>	모델 번호
	<serialNumber>	제품 번호
	<PresentationURL>	사용자 UI 페이지 URL
	<FirmwareVersion>	펌 웨어 버전
선택 명세 요소	<manufacturerURL>	제조회사 URL
	<modelURL>	모델 URL

4. 유비쿼터스 웹서비스 환경의 동적 서비스 검색

유비쿼터스 웹서비스 환경에서는 시스템 및 플랫폼, 서비스에 관계없이 다수의 Sub-network와 디바이스를 사용할 수 있어야 한다. 이질적인 환경의 상호운용성을 위해 기존의 웹서비스 구조에서 확장된 유비쿼터스 웹서비스의 구조가 형성된다[그림5]. 유비쿼터스 웹서비스 구조는 다음과 같이 구성된다.

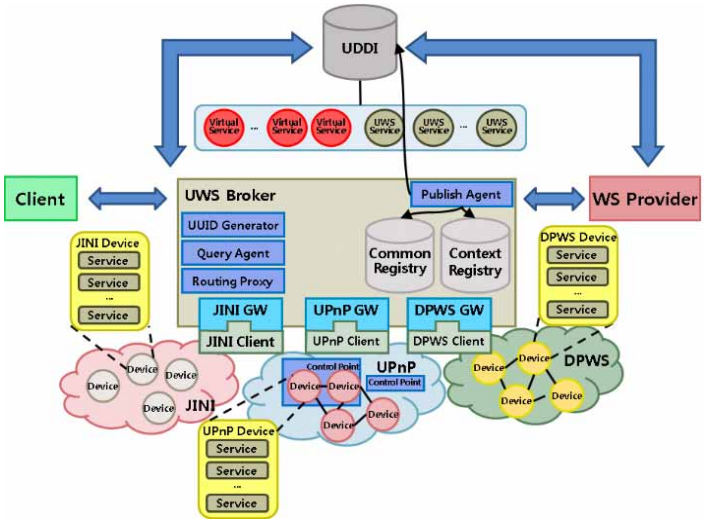

그림 5 유비쿼터스 웹서비스 환경의 동적 서비스 검색 구조

유비쿼터스 웹서비스 환경의 동적 서비스 검색 과정에서 각각의 컴포넌트는 다음과 같은 기능을 수행한다.

- 유비쿼터스 웹서비스 브로커(Ubiquitous Web Services Broker): 유비쿼터스 웹서비스 브로커는 유비쿼터스 웹서비스 환경에서 가장 중요한 부분으로 Publish Agent, Query Agent, Routing Proxy, UUID Generator로 분류된다.
  - Publish Agent: Sub-network 게이트웨이에서 생성된 가상 서비스(Virtual Service)를 UDDI에 등록하는 부분과 유비쿼터스 웹서비스 레지스트리에 저장 요소를 등록시키는 부분으로 구성된다.

- Query Agent: 서비스 사용자로부터 요청된 질의를 중재한다. 질의가 들어오면 UDDI(universal description, discovery, and integration)의 검색을 한 결과가 웹서비스 형태의 서비스이면 서비스 사용자에게 연결시켜주고, 유비쿼터스 웹서비스 형태의 서비스이면 서비스 사용자에게 2차 질의를 요구한다. 서비스 사용자에 의해 2차 질의가 요청되면 유비쿼터스 웹서비스 레지스트리를 검색한 결과를 보내준다.
- Routing Proxy: 유비쿼터스 웹서비스 환경의 서비스를 이용하는 서비스 사용자와 Sub-network로의 경로를 결정하는 기능을 한다.
- UUID Generator: 유비쿼터스 웹서비스 환경의 디바이스 아이디를 생성하여 유비쿼터스 웹서비스 레지스트리에 저장 요소로 사용된다.

• 유비쿼터스 웹서비스 레지스트리(Ubiquitous Web Services Registry): 다수의 Sub-network에 존재하는 디바이스와 서비스의 명세에 대한 공통 정보를 관리한다. 유비쿼터스 웹서비스 환경에 질의가 요청되면 유비쿼터스 웹서비스 레지스트리에 의해 검색한 결과를 알려주게 된다.

• 상황 정보 레지스트리(Context Registry): 유비쿼터스 웹서비스 환경에서 서비스 사용자의 질의에 대한 결과를 QoS하기 위한 정보를 담고 있다. 디바이스, 서비스, 서비스 사용자에 대한 위치 정보와 상태 정보 등을 포함한다.

• Sub-network 게이트웨이(Sub-network Gateway): 다수의 Sub-network와 디바이스마다 이질적인 특성을 가지고 있기 때문에 이를 관리하는 모듈이 필요하다. 이를 위해, Sub-network를 확장하여 게이트웨이 형태를 띠게 된다. Sub-network인 JINI의 LUS나 UPnP의 CP(Control Point)를 확장하여 사용한다. 각 Sub-network 게이트웨이는 서비스를 가상 서비스로 변환하여 서비스 사용자는 웹서비스를 사용하는 것과 같은 기능을 한다.

본 논문의 유비쿼터스 웹서비스 구조를 통해 동적 서비스 검색을 하는 과정은 다음과 같다[그림6].

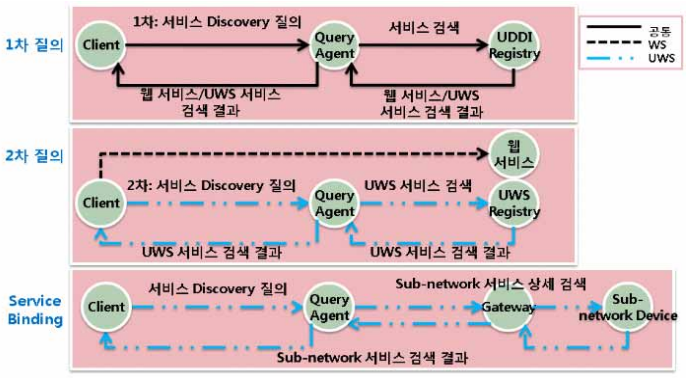

그림 6 유비쿼터스 웹서비스 환경의 동적 서비스 검색 과정

유비쿼터스 웹서비스 환경의 동적 서비스 검색 과정은 서비스 사용자는 서비스 형태에 상관없이 필요로 하는 서비스를 질의한다. 서비스 사용자의 모든 질의는 질의 에이전트에 의해 중재되어 UDDI와 유비쿼터스 웹서비스 레지스트리를 검색하게 된다. 질의 에이전트는 모든 서비스가 등록되어 있는 UDDI를 먼저 검색하여 웹서비스 형태이면 서비스 사용자에게 결과를 알려주고 서비스를 연결한다. 반면, UDDI 검색 결과가 유비쿼터스 웹서비스 환경의 서비스이면 유비쿼터스 웹서비스 레지스트리를 검색한 결과에 따른 추가적인 질의를 서비스 사용자에게 요청하게 된다. 서비스 사용자는 상세한 2차 질의를 요청하여 서비스를 검색 결과를 응답받을 수 있다. 모든 서비스가 가상의 웹서비스처럼 서비스 사용자에게 제공되기 때문에 Sub-network나 서비스를 제공하는 디바이스에 상관없이 서비스를 사용할 수 있다.

5. 유비쿼터스 웹서비스 환경의 동적 서비스 검색을 위한 공통 레지스트리의 구성 요소 설계

본 논문의 4장에서 유비쿼터스 웹서비스 환경의 이질적인 Sub-network와 디바이스들이 제공하는 서비스를 동적으로 검색하는 구조와 과정에 대해 살펴보았다. 이번 장에서는 유비쿼터스 웹서비스 환경의 레지스트리가 필요한 이유, 레지스트리의 구성 요소와 각각의 구성 요소가 필요한 이유에 대해 설명하도록 한다.

산재해 있는 Sub-network나 디바이스의 서비스를 사용하기 위해서는 각각의 프로토콜 특성에 맞는 검색 조건이 필요하다. 서비스 사용자는 원하는 서비스에 대한 Sub-network나 디바이스에 대한 정보를 알고 있어야 하기 때문에 동적인 서비스 검색이 불가능하다. 예를 들어, JINI의 서비스를 이용한다고 보면 JINI의 서비스가 동작할 수 있도록 자바 가상 머신과 서비스 연결을 위한 자바 RMI 통신을 지원해야 한다. 물론, 서비스 사용자가 각각의 Sub-network나 디바이스에 질의를 통해 원하는 서비스를 검색할 수 있다. 이는 중개자(Mediator)방식으로 서비스 사용자가 원하는 서비스를 검색할 때마다 모든 Sub-network나 디바이스에 질의를 통해 검색 결과를 얻는 방법이다. 각각의 Sub-network나 디바이스가 제공하는 서비스에 대한 관리 모듈이 필요 없기 때문에 동적인 검색이 가능한 장점이 있지만, 서비스 사용자가 원하는 서비스를 검색할 때마다 런타임이 증가되며 모든 Sub-network나 디바이스의 서비스 질의 형식을 알 수 없다. 또한, 서비스 사용자와 제공되는 서비스간의 연결이 끊어지면 서비스 사용자는 다시 Sub-network나 디바이스를 검색해야 하고 Sub-network나 디바이스의 이벤트 정보를 동적으로 처리하지 못하기 때문에 유비쿼터스 웹서비스 환경에서 중개자방식은 적절하지 못하다.

각각의 Sub-network나 디바이스의 명세 정보를 레지스트리에 등록함으로써 중개자방식의 문제점을 해결할 수 있다. 하지만, 모든 Sub-network나 디바이스의 명세 정보를 레지스트리에 등록하게 될 경우 중개자방식보다 서비스 검색 시간을 줄어들게 되지만, 모든 Sub-network나 디바이스의 명세 정보에 대한 등록, 삭제 또는 수정에 대한 부담이 가중된다.

각각의 Sub-network나 디바이스의 이질적이고 상호운

용성인 특성을 반영하여 각각의 Sub-network나 디바이스의 공통, 필수, 정적인 명세 항목을 유비쿼터스 웹서비스 레지스트리에 구성해야 한다. 서비스를 검색하기 위한 최소의 명세 항목을 선정하여 레지스트리에 등록함으로써 유비쿼터스 웹서비스 환경에 적합한 모델을 형성할 수 있다. 이로 인해, 유비쿼터스 웹서비스 환경의 서비스를 중개자방식과 데이터 웨어하우스방식을 혼합한 하이브리드 방식을 사용한다. 공통, 필수, 정적인 명세 요소를 통해 각각의 Sub-network나 디바이스의 서비스 검색이 이루어지고, 서비스 사용자가 더 많은 정보를 요구하면 각각의 Sub-network나 디바이스를 직접 검색함에 따라 유비쿼터스 웹서비스 환경에 맞는 동적 서비스 검색이 가능해진다.

아래의 표에서 제시하는 각각의 Sub-network나 디바이스가 공통적으로 디바이스와 서비스를 명세 하는 항목은 디바이스 이름, 제조회사, 모델명, 제품번호, 서비스 이름, 서비스 ID이다[표5].

표 5 Sub-network의 공통 명세 요소

	JINI	UPnP	DPWS
공통 요소	<name>	<friendlyName>	<friendlyName>
	<manufacturer>	<manufacturer>	<manufacturer>
	<model>	<modelName>	<modelName>
		<modelNumber>	<modelNumber>
	<serialNumber>	<serialNumber>	<serialNumber>
	<serviceType>	<serviceType>	<serviceType>
	<serviceID>	<serviceID>	<serviceID>

산재해 있는 Sub-network나 디바이스의 명세 항목을 위와 같이 추출하였다. 각각의 Sub-network나 디바이스 명세 항목의 공통적인 명세 요소이지만, 모든 요소가 서비스를 검색하기 위한 필수적인 명세 요소는 아니다.

본 논문에서 다루고 있는 각각의 Sub-network나 디바이스에서 명세 하는 디바이스와 서비스 항목은 필수 요소가 다르기 때문에 동적 서비스 검색을 할 경우 제대로 된 결과를 제시할 수 없다. 이를 위해, 각각의 Sub-network나 디바이스가 명세 하는 공통적인 명세 요소, 필수적인 명세 요소와 정적인 명세 요소만을 공통 레지스트리에 구성한다[표6].

표 6 레지스트리의 공통적인 명세 요소, 필수적인 명세 요소, 정적인 명세 요소 설계

설계 요소	설 명
categoryType	- 서비스에 대한 카테고리 정보 관리
deviceID	- 유비쿼터스 웹서비스 브로커에 의해 생성된 디바이스 ID
serviceID	- Sub-network에서의 서비스 ID
deviceType	- 디바이스 이름
serviceType	- 서비스 이름
serviceStatus	- 서비스 상태 정보 - 서비스 사용가능한 여부 확인
UWSID	- 유비쿼터스 웹서비스 ID
	- Sub-network 게이트웨이에서 생성한 가상 서비스를 UDDI에 등록한 UUID

유비쿼터스 웹서비스 환경의 동적 서비스 검색을 위한 레지스트리 구성으로 산재해 있는 Sub-network나 디바이스에 대한 통합적인 서비스 검색이 가능하다.

6. 결론

본 논문에서 우리는 유비쿼터스 웹서비스 환경의 동적 서비스 검색을 위한 레지스트리 구성 요소에 대한 설계를 하였다. 이를 위해, 이질적인 Sub-network의 JINI, UPnP와 디바이스간의 통합을 위한 DPWS의 명세 항목에 대해 연구를 진행하였다. 유비쿼터스 웹서비스 환경의 동적인 서비스 검색은 Publish Agent, Query Agent, Routing Proxy와 UUID Generator로 구성된 유비쿼터스 웹서비스 브로커와 Sub-network 게이트웨이, 유비쿼터스 웹서비스 레지스트리, 상황 정보 레지스트리를 이용하여 이질적인 Sub-network나 디바이스에 관계없이 통합적인 서비스 검색이 가능했다. 하지만, 이질적인 Sub-network나 디바이스가 명세 하는 디바이스와 서비스의 공통 명세 요소는 필수 명세 요소가 다르기 때문에 동적 서비스 검색이 불가능했다. 이는 공통적인 명세 요소, 필수적인 명세 요소, 정적인 명세 요소를 고려하여 유비쿼터스 웹서비스 레지스트리에 등록함으로써 해결 방법을 제시하였다.

우리는 본 논문에서 제시한 유비쿼터스 웹서비스 환경의 동적 서비스 검색 구조 및 과정을 통해 유비쿼터스 웹서비스 레지스트리의 필요성을 강조하였고, 이를 온톨로지 기반의 상황 인식 서비스와 유비쿼터스 웹서비스 컴포지션에 확장하여 유비쿼터스 웹서비스 환경의 실제 시스템 구현에 많은 도움을 줄 것이라 생각한다.

참고문헌

[1] Malcolm Attard, “Ubiquitous Web Services”, http://www.cs.um.edu.mt/~csaw/CSAW03/Proceedings/UbiqWebServices.pdf COMPUTER SCIENCE ANNUAL RESEARCH WORKSHOP, 2003

[2] 안명환, “유비쿼터스 환경을 위한 OSGi 기반 상황 인식 서비스 아키텍쳐의 설계 및 구현”, 한국정보과학회, 한국컴퓨터종합학술대회, pp 16-18, 2006

[3] OSGi Alliance. OSGi Service Platform Release 4 CORE, 2005.

[4] Hendrik Bohn, Andreas Bobek, Frank Golatowski, “SIRENA - Service Infrastructure for Real-time Embedded Networked Devices: A service oriented framework for different domains”

[5] Sun Microsystems. Jini Architecture Specification Version 1.2, 2001.

[6] UPnP Forum. UPnP Device Architecture v.1.0.1, 2003.

[7] Microsoft. Devices Profile for Web Services, 2006.