유비쿼터스 웹서비스 환경의 DPWS와 웹서비스의 상호운용성을 위한 DPWS 어댑터 설계*

임형준[O1] 오일진[1] 황윤영[1] 이강찬[2] 이승윤[2] 이규철[1†]
[1]충남대학교 컴퓨터공학과
[2]한국전자통신연구원
{hyungjun25[O1], victory25[1], yyhwang[1], kclee[1]}@cnu.ac.kr
{chan[2], syl[2]}@etri.re.kr

Design of DPWS Adaptor for Interoperability between Web Services and DPWS in Web Services on Universal Networks

Hyung-Jun Yim[O1] Il-Jin Oh[1] Yun-Young Hwang[1] Kangchan Lee[2]
Seungyun Lee[2] Kyu-Chul Lee[1]
[1]Department of Computer Engineering, Chungnam National University
[2]Electronic and Telecommunications Research Institute

요 약

유비쿼터스 환경에서는 서비스 사용자가 언제, 어디서나 모든 서비스를 사용할 수 있어야 한다. 즉, 디바이스나 서비스의 상태, 위치 정보 등 특성에 관계 없이 동질의 서비스를 제공해야 한다. 이를 위해, 산재한 이질적인 서브네트워크의 상호운용성을 위한 다양한 분산 시스템의 네트워크 프로토콜과 디바이스 간의 통신을 통합하기 위한 연구가 필요하다. 서브네트워크의 상호운용성을 위해서는 메시지 전송 및 변환 기능이 요구되며, 서브네트워크의 변화를 동적으로 감지하여 서비스 사용자에게 알려주어야 한다.

따라서 본 연구에서는 유비쿼터스 웹서비스를 지원하기 위한 Web Services on Universal Networks를 제안하였다. 제안하는 WSUN은 Universal Service Broker의 프레임워크를 통해 SOA 기반으로 접근하고 있으며 요구 사항을 정의하여 필요한 구성 요소에 대한 설계하였다. 또한, 동적인 서비스 검색을 위한 Universal Service Registry를 사용한다. 본 논문에서는 DPWS의 동작 시나리오를 통해 이질적인 서브네트워크의 상호운용성을 지원하기 위한 DPWS 어댑터를 설계한다.

1. 서 론

최근 컴퓨팅과 네트워크 등 연관 기술들의 급격한 발전으로 유비쿼터스 환경에 대한 관심도 증가하고 있다. 하지만, 현재의 컴퓨팅 환경과 다르게 유비쿼터스 환경은 시스템의 운영체제나 네트워크, 프로그래밍 언어 등에 독립적으로 동작하는 환경이 요구된다.

디지털 기술의 진화는 모바일 디바이스를 중심으로 기능 융합이 이루어지고 있으며, 디바이스간의 영역이 모호해지고 있다. 이는 사용자 주변의 서비스를 통합하는 새로운 패러다임으로 기존에 사람과 기계의 통신이 이루어지는 것에 비해 디바이스간에 자동적으로 통신을 할 수 있도록 변화하고 있다. 하지만 각종 디바이스 및 서비스는 서로 다른 프로토콜 및 플랫폼을 사용한다. 따라서 사용자는 이를 지원하기 위한 모든 환경을 포함해야 하거나 서로 다른 프로토콜 및 플랫폼을 기반한 디바이스와 서비스의 위치, 상태, 정보 등의 유동성에서 생기는 문제를 서비스 디스커버리 미들웨어 (Service Discovery Middleware)로 해결할 수 있다. 본 논문에서는 서비스 디스커버리 미들웨어를 서브네트워크로 지칭한다. 대표적인 서브네트워크로는 썬 마이크로시스템즈의 Java Intelligent Network Infrastructure (JINI) [1], 마이크로소프트의 Universal Plug and Play (UPnP) [2]와 Home Audio/Video Interoperability (HAVi) [3] 등이 있으며, 디바이스간의 통합을 위한 연구로는 Devices Profile for Web Services (DPWS) [4]가 있다.

산재한 이질적인 서브네트워크의 통합을 위한 방법으로 유비쿼터스 환경에 웹서비스가 융합된 형태로 접근할 수 있다. 유비쿼터스 웹서비스 (Ubiquitous Web Services)는 어떠한 단말/네트워크 환경에서도 다양한 응용 서비스를 연계하여 이용할 수 있도록 하는 미래형 웹서비스이다. 이는 계속 동적으로 변하는 디바이스에 대한 능동적인 서비스의 검색과 이용이 요구되고, ad-hoc한 환경에서의 검색 방법 등을 지원해야 한다. 현재는 트랜잭션, 보안, QoS, 시멘틱과 웹서비스 조합 등 여러 분야로 연구되고 있다 [5].

본 연구에서는 이러한 유비쿼터스 웹서비스를 연구하기 위해 Web Services on Universal Networks (WSUN)을 제안하였다. 이를 위해, WSUN의 가장 중요한 구성 요소인 Universal Service Broker (US Broker)를 설계하여 서브네트워크 및 디바이스, 서비스의

* 본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT 연구센터 육성지원사업 (IITA-2005-C1090-0502-0016)의 연구비 지원으로 수행하였습니다.
† 교신 저자

상호운용성을 지원하도록 하였으며, Universal Service Registry (US Registry)를 통해 동적인 서비스 검색을 지원하였다. 따라서, 본 논문에서는 US Broker가 이질적인 서브네트워크의 상호운용성을 제공하기 위한 DPWS 어댑터를 설계한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대해 소개하고, 3장에서는 DPWS 시나리오를 통해 DPWS 어댑터의 필요성을 도출한다. 4장에서 DPWS 어댑터의 동작 과정을 설계하고, 5장에서는 DPWS 어댑터의 구조 및 기능에 대해 정의한다. 마지막으로 6장에서는 결론과 향후 연구방향에 대하여 논한다.

2. 관련 연구

이번 장에서는 이질적인 서브네트워크의 상호운용을 위한 기존의 연구에 대해 살펴본다.

표 1은 서브네트워크간의 통합을 위한 연구와 디바이스간의 통합을 위한 연구 및 서브네트워크의 특징을 나타낸다.

표 1 서브네트워크와 디바이스간의 통합을 위한 연구 [6]

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

가장 대표적인 서브네트워크인 JINI와 UPnP는 이질적인 특성을 가장 잘 반영하기 때문에 서브네트워크간의 통합을 위한 연구인 Open Service Gateway Initiative (OSGi) [7]와 Domotics Network (DomoNet) [8][9] 등에서 사용한다. JINI는 자바 기반으로 자바 가상 머신 (Java Virtual Machine)이 필요한 단점이 있지만 플러그 앤 플레이 (Plug-and-Play)와 보안 등 여러 특성을 포함하고 있다. 반면에 UPnP는 프로그래밍 언어에 독립적이지만 광대역의 네트워크 환경을 지원하지 못하며 보안에 취약한 문제를 지니고 있다. 이질적인 환경의 상호운용성을 제공하기 위해 웹서비스는 다수의 특징을 지원하지만 플러그 앤 플레이를 지원하지 못하기 때문에 동적인 서비스 검색이 불가능하다. 또한, 디바이스 관점의 웹서비스를 제공하지 못하기 때문에 유비쿼터스 환경에서 제대로 적용할 수 없다. 웹서비스가 가지는 문제점을 해결하기 위한 일환으로 DPWS는 디바이스 관점의 웹서비스를 지원하여 유비쿼터스 환경의 특징을 반영한 UPnP의 차세대 기술로 각광받고 있다.

기존의 서브네트워크에서 가지는 문제점을 해결하기 위한 OSGi와 DomoNet은 여러 관점으로 연구되고 있지만, OSGi는 자바기반의 플랫폼이기 때문에 자바 환경을 제공하지 않으면 서비스를 제공해 줄 수 없으며 다양한 서비스를 디스커버리하는 방법을 지원하지 못한다. DomoNet은 레지스트리를 사용하지 않기 때문에 서비스의 상태에 대한 정보를 포함하지 못할 뿐만 아니라 서비스를 사용하기 위해 매번 웹서비스화하는 문제점이 있다. 또한, TechManager (TM)이 동작하기 위해 DomoML의 사용해야 하므로 언어에 종속적으로 작용한다. DPWS도 역시 디바이스 레벨의 웹서비스를 제공하지만 서비스의 상태 정보를 포함하지 않는다.

2.1 Web Services on Universal Networks [10]

앞서 살펴본 기존 연구의 문제점을 해결하기 위해 유비쿼터스 웹서비스 프레임워크를 위한 WSUN을 제안하였다. 서브네트워크 및 디바이스, 서비스의 상호운용을 위한 WSUN의 가장 중요한 구성 요소인 US Broker를 설계를 완료하였다. 또한, US Registry를 통해 동적인 서비스 검색이 가능하다.

WSUN은 그림 1과 같이 사용자가 US Broker를 통해 서비스를 검색할 수 있는 범위를 나타낸다. US Broker는 산재한 이질적인 서브네트워크의 상호운용성을 지원하는 SOA 기반의 미들웨어이다. US Broker는 사용자와 서비스 제공자 사이에 위치하여 서브네트워크의 서비스 및 WSUN 디바이스가 제공하는 서비스 즉, 유니버설 서비스 (Universal Service)를 검색 및 사용할 수 있도록 한다.

US Broker를 위한 구성 요소는 다음과 같다.

- Listener: 사용자 및 WSUN 디바이스가 US Broker를 이용하기 위한 기본 정보를 제공한다.
- Query Agent: 사용자의 질의를 처리한다.
- Publish Agent: WSUN 환경의 디바이스 및 서비스의 명세 정보의 저장을 담당한다.
- Routing Proxy: 사용자가 원하는 서브네트워크의 서비스를 연결한다.
- US Registry: 디바이스 및 서비스의 명세 정보를 저장하는 Device/Service Registry와 디바이스의 위치 정보를 저장하는 Context Registry로 구성된다.
- Universal Adaptor (UniA): 서브네트워크와 US Broker 사이의 중계 역할을 담당한다. 서브네트워크의 서비스를 가상의 웹서비스로 변환하는 기능을 한다.

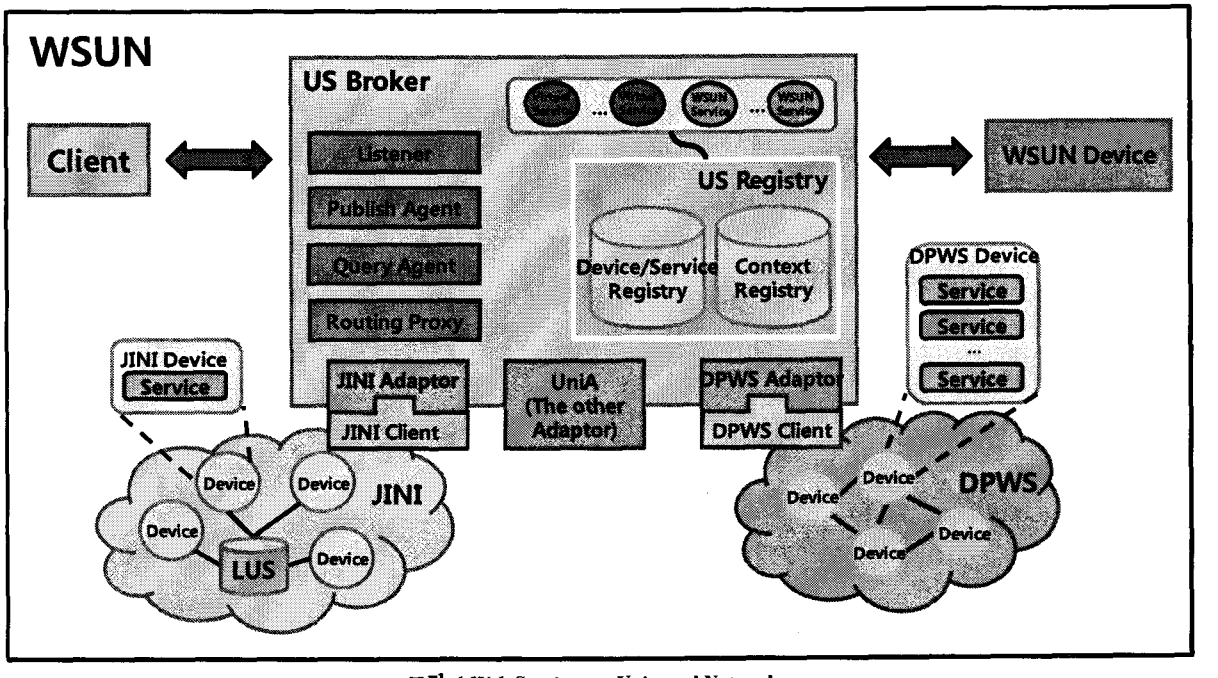

그림 1 Web Services on Universal Networks

3. DPWS 시나리오

이번 장에서는 DPWS 어댑터를 설계하기 위해 시나리오를 통한 필요성을 기술한다. 다음의 시나리오는 회의에 필요한 서비스를 이용하는 경우이다.

3.1 공통 시나리오

사용자는 그림 2와 같이 회의 참석을 위해 회의 장소에 들어간다. 회의 참석자들에게 회의 자료를 나누어주기 위해 회의 장소 내에서 사용 가능한 프린터를 검색한다. 그리고 나서, 사용자는 프리젠테이션을 진행하기 위한 프로젝터를 검색하여 회의를 진행한다. 회의 진행 중, 추가 자료를 설명하기 위해 LCD TV를 검색하여 HD 영상을 보여준다.

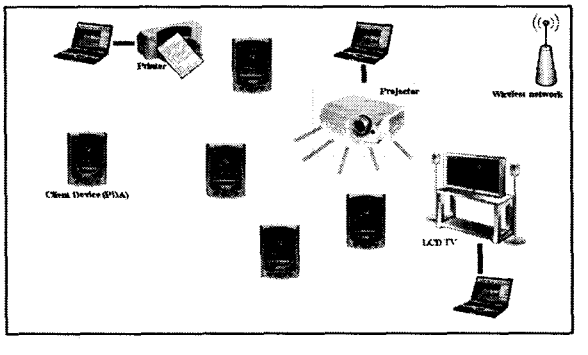

그림 2 공통 시나리오

3.2 US Broker가 존재하지 않는 경우의 시나리오

사용자는 회의에 필요한 서비스들을 검색하여 사용한다. 즉, 프린터, 프로젝터, LCD TV에서 제공하는 서비스들을 이용해야 하는데 회의 장소 내에 존재하는 서브네트워크는 다양하다. 사용자는 어떤 서브네트워크에 어떤 서비스가 존재하는지 알 수 없기 때문에 각 서브네트워크에 사용 가능한 서비스들을 질의해야 한다. 또한, 그림 3과 같이 사용자는 서브네트워크에 의존적인 질의 방법과 통신을 모두 지원해야 원하는 서비스를 이용할 수 있다. 예를 들어, DPWS를 지원하는 디바이스에서 프로젝터를 검색할 경우 WS-Discovery를 통해 이루어진다. 프로젝터가 이미 다른 사용자가 사용하고 있다면 다시 검색 과정을 거쳐 다른 프로젝터를 이용해야 한다. 이와 같이 DPWS는 서비스의 상태 정보를 반영하지 못한다.

3.3 US Broker가 존재할 경우의 시나리오

사용자는 회의에 필요한 서비스들을 검색하여 사용한다. 그림 4와 같이 회의 장소 내에 사용 가능한 서비스들은 모두 가상의 웹서비스로 되어 있기 때문에 사용자는 웹서비스를 사용하는 것과 같이 실제 서비스들을 검색하고 사용한다. 이와 같은 원리는 US Broker가 존재하기 때문에 가능하다. US Broker는 사용자가 원하는 서비스에 대한 질의가 오면 US Registry의 정보를 이용하여 사용 가능한 서비스만을 검색한다. 실제 존재하는 서브네트워크의 서비스를

이용하기 위해 어댑터는 서브네트워크에 맞는 메시지 변환을 담당하기 때문에 서브네트워크의 환경에 제약을 받지 않고 서비스를 제공할 수 있다.

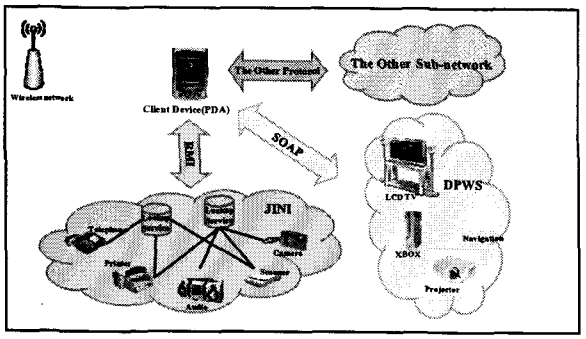

그림 3 US Broker가 존재하지 않는 경우

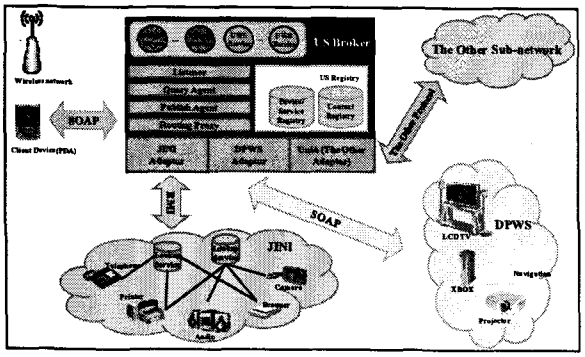

그림 4 US Broker가 존재하는 경우

4. DPWS 어댑터의 동작 과정

앞에서 살펴 본 것과 같이 US Broker를 통해 사용자의 환경에 제약을 받지 않고 서비스들을 이용할 수 있다. 이러한 기능은 DPWS 어댑터가 제공한다. DPWS 어댑터는 DPWS가 디바이스 레벨에서 웹서비스를 사용할 수 있도록 제공하고 있기 때문에 Web Services Description Language (WSDL)를 이용하여 US Broker의 가상의 웹서비스로 노출시킨다. 또한, US Registry에 필요한 디바이스와 서비스 정보를 검색하고 서비스의 상태 정보를 이벤트로 동적인 처리가 가능하다. 이러한 기능들을 제공하기 위해 DPWS 어댑터는 다음의 일련의 과정이 동작한다.

4.1 Design Time - DPWS 어댑터의 Hook-up 과정

그림 5는 DPWS 어댑터가 Hook-up되는 과정이다.

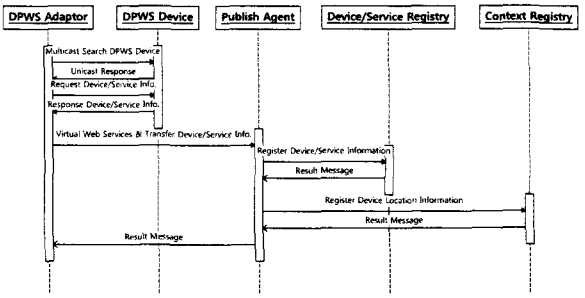

그림 5 DPWS 어댑터의 Hook-up 과정

위의 그림은 DPWS 어댑터가 10단계로 동작하는 과정을 순차적으로 나타낸다. 1단계에서 4단계는 DPWS의 어댑터가 DPWS의 디바이스와 서비스의 정보를 얻어 오는 과정이고, 5단계는 가상의 웹서비스를 생성하여 DPWS의 디바이스와 서비스의 정보와 함께 Publish Agent로 전송하는 과정이다. Publish Agent는 6, 7단계에서 가상의 웹서비스를 Device/Service Registry에 등록을 하고 8, 9단계에서는 Context Registry에 각각의 정보를 등록하는 기능을 한다. 마지막으로 10단계에서는 위의 일련의 과정이 제대로 동작하였는지 결과 메시지를 DPWS의 어댑터가 확인을 한다.

DPWS의 어댑터는 Hook-up을 할 때, DPWS의 디바이스를 찾기 위해 WS-Discovery의 Hello 메시지를 멀티캐스트로 전송을 한다. DPWS의 디바이스는 응답 메시지로 UUID, Type, Scope, 메타데이터 버전을 포함한 정보를 보내게 된다. DPWS의 어댑터는 US Registry에 등록하기 위한 DPWS의 디바이스와 서비스의 정보를 요청하여 서비스 이름, 디바이스 이름, WSDL과 DPWS의 디바이스의 위치 정보를 얻게 된다. US Registry에 저장할 정보를 모두 얻게 되면 DPWS의 어댑터는 WSDL 정보를 이용하여 가상의 웹서비스를 생성한 후, 서비스 ID, 서비스 이름, 디바이스 이름, DPWS의 디바이스의 위치 정보와 함께 WSDL를 가리키는 포인터를 Publish Agent로 전송한다. Publish Agent는 Device/Service Registry에 서브네트워크 ID, 서브네트워크 타입, 서비스 ID, US Broker에서의 서비스 ID, 디바이스 ID, 서비스 이름, 디바이스 이름, 카테고리 타입의 공통 정보를 등록한다. 또한, Context Registry에 DPWS의 디바이스 위치 정보를 등록하여 상황 정보로 이용하게 된다. 일련의 과정이 완료되면, DPWS의 어댑터의 Hook-up 과정이 종료된다.

4.2 Run Time - DPWS 디바이스의 등장

DPWS 디바이스의 등장은 새로운 DPWS 디바이스와 기존에 존재했던 DPWS 디바이스가 등장하는 경우로 나눌 수 있다.

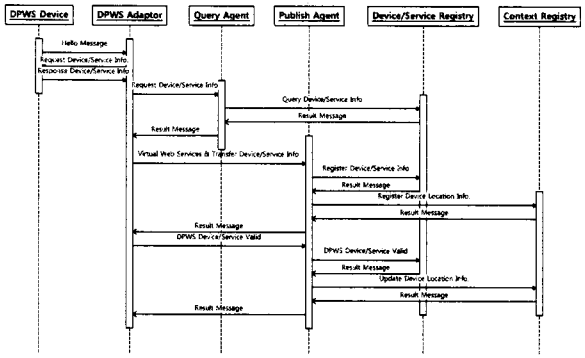

그림 6 DPWS 디바이스의 등장

위의 그림 6의 DPWS 디바이스가 등장하는 순차적인 도식은 DPWS의 어댑터가 동작하는 과정을 19단계로 나타낸다. 1단계에서 3단계의 과정은 DPWS의 디바이스가 WS-Discovery의 Hello 메시지를 통해 등장을 알리게 되고 DPWS의 어댑터는 UUID를 통해 DPWS의 디바이스와 서비스의 정보를 얻는다. 4, 5, 6, 7단계는 Query Agent가 UUID를 통해 US Registry에 등장한 DPWS의 디바이스가 존재하는지 확인하는 과정이다. US Registry에 정보가 존재하지 않으면 8단계에서 13단계에 도식한 것과 같이 DPWS 어댑터의 Hook-up 과정과 동일하게 구성된다. 반면에 서비스가 Invalid 상태로 존재하면 14단계에서 19단계의 과정이 수행된다. 이 과정은 DPWS의 어댑터가 DPWS의 디바이스의 UUID를 통해 US Registry의 서비스의 상태 정보를 valid로 변경하도록 한다. 동시에 Context Registry의 디바이스 위치 정보를 변경하게 된다.

4.3 Run Time - DPWS 디바이스의 퇴장 (Bye Message)

그림 7은 DPWS의 디바이스가 WS-Discovery의 Bye 메시지를 전송하고 연결이 끊어지는 경우이다. DPWS의 디바이스는 연결을 종료할 경우, Bye 메시지를 보내게 되는데 Bye 메시지는 UUID 정보만을 포함하고 있다. DPWS의 어댑터는 Bye 메시지를 받으면 Publish Agent에 DPWS의 디바이스와 서비스의 상태가 Invalid하다는 정보를 전송한다. 이로 인해, US Registry의 서비스의 정보가 업데이트된다.

4.4 Run Time - DPWS 디바이스의 퇴장 (System Error)

그림 8은 DPWS의 어댑터가 DPWS의 서비스에 존속 시간을 지속적으로 확인하는 기능을 담당한다. 서비스 존속 시간의 확인 메시지에 대한 응답 메시지가 오지 않는 경우, DPWS의 어댑터는 Publish Agent를 통해 US Registry의 서비스의 정보를 Invalid로 업데이트한다.

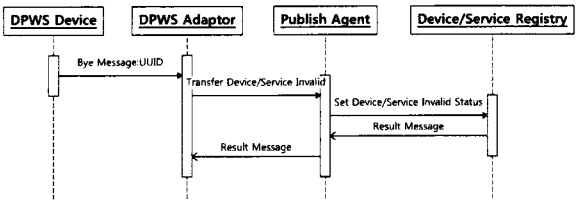

그림 7 DPWS 디바이스의 퇴장 (Bye Message)

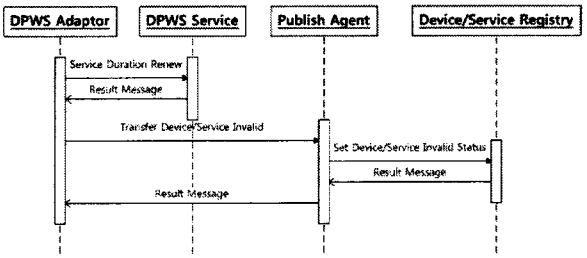

그림 8 DPWS 디바이스의 퇴장 (System Error)

4.5 Run Time - DPWS 디바이스의 상태 정보 변경

그림 9는 DPWS의 디바이스의 서비스의 정보가 변경이 된 경우이다. DPWS의 디바이스에서 제공하는 서비스의 상태가 변경이 되면 DPWS의 어댑터에 이벤트 메시지를 통해 US Registry의 상태 정보를 변경한다.

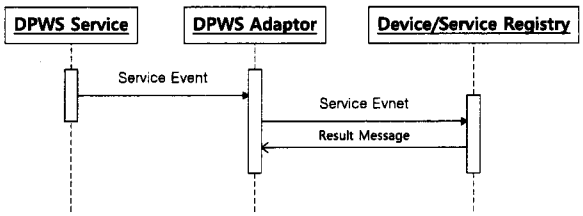

그림 9 DPWS 디바이스의 상태 정보 변경

5. DPWS 어댑터의 구조 및 기능

DPWS 어댑터는 위에서 살펴 본 것과 같이 다양한 동작과 기능을 담당한다. 이를 위해, DPWS 어댑터는 그림 10과 같이 메시지 전달, 가상 웹서비스 생성, DPWS 서비스 발견, event와 notification으로 구성된다.

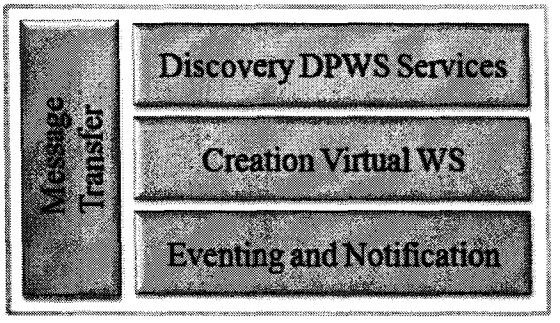

그림 10 DPWS 어댑터의 구조 및 기능

5.1 메시지 전달

DPWS와 US Broker에 필요한 정보들은 전송하는 기능을 담당한다. 이는 US Broker와 DPWS의 호환을 위해 DPWS 어댑터를 통해 중재된다.

5.2 DPWS 서비스 발견

DPWS의 서비스를 가상의 웹서비스로 변환하고 US Registry에 저장할 정보들을 검색한다. 검색한 정보들은 가상의 웹서비스 생성 후 Publish Agent로 보내 US Registry에 저장한다. 또한, 디바이스 등장 및 연결을 종료할 경우도 바로 처리를 하기 때문에 플러그 앤 플레이를 지원한다.

5.3 가상의 웹서비스 생성

사용자는 서브네트워크의 존재를 인식하지 않고 서비스를 검색하고 사용하기 때문에 서브네트워크의 서비스를 웹서비스 형태로 변환한다. 변환된 서비스가 외부에 가상의 웹서비스로 사용자에게 노출된다. 이를 통해, 사용자는 웹서비스를 사용하는 것으로 보이지만 실제로 DPWS 어댑터를 통해 DPWS의 서비스가 제공된다. DPWS는 디바이스와 서비스의 메타데이터를 WSDL로 제공하기 때문에 WSDL을 생성할 필요는 없다. 하지만, WSDL을 이용해 가상의 웹서비스로 클라이언트에게 제공하는 기능을 한다.

5.4 Event and Notification

DPWS의 디바이스의 등장과 퇴장이 발생하였을 경우 이를 반영하여 플러그 앤 플레이를 지원한다. 또한, 사용자가 서비스의 사용을 종료하였을 경우, 서비스의 상태 정보를 변경하여 서비스가 사용 가능하도록 한다. 서비스의 상태 정보를 반영하여 클라이언트가 사용 가능한 서비스만 검색하도록 한다. 즉, 동적 웹서비스 검색을 지원한다.

6. 결 론

본 연구를 통해 제안한 유비쿼터스 웹서비스의 Web Services on Universal Networks 환경과 구성 요소인 Universal Service Broker를 설계하였다. WSUN의 환경에서 사용자는 서브네트워크의 환경에 제약을 받지 않고 유니버설 서비스 (Universal Service)를 검색 및 사용할 수 있으며, US Broker를 통해 이질적인 서브네트워크의 상호운용성을 지원하였다. US Broker의 이런 기능은 유니버설 어댑터 (Universal Adaptor)를 통해 가능하며 유니버설 어댑터는 각 서브네트워크마다 존재하여 특성에 맞도록 설계되었다. 본 논문의 DPWS 어댑터는 DPWS 디바이스가 제공하는 서비스의 상호운용성을 위해 시나리오를 통해 DPWS 어댑터의 필요성을 도출하였으며, DPWS의 동작에 따른 기능을 정의하였다.

현재 US Broker는 JINI와 DPWS의 특성을 고려하여 설계되었지만, 추후 UPnP, HAVi 등의 서브네트워크로 확장하여 설계 및 구현을 할 것이다. 또한, 사용자 질의의 최적화와 프로파일 등의 상황 정보를 통해 향상된 서비스 검색뿐만 아니라 트랜잭션, 보안, QoS, 웹서비스의 조합 등의 연구로 유비쿼터스 웹서비스의 최적의 프레임워크를 제공할 것으로 기대된다.

참고문헌

[1] Sun Microsystems. JINI Specifications Archive v2.1, 2005

[2] UPnP Forum. UPnP Device Architecture v1.0.1, 2 December 2003

[3] Havi Consortium. HAVi Specification v1.1, 15 May 2001

[4] Shannon Chan et. Al, "Devices Profile for Web Services", February 2006

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

[6] Hendrik Bohn, Andreas Bobek, Frank Golatowski, "SIRENA - Service Infrastructure for Real-time Embedded Networked Devices: A service oriented framework for different domains", April 2006

[7] OSGi Alliance. About the OSGi Service Platform, Technical Whitepaper Revision 4.1, 7 June 2007

[8] Vittorio Miori, Luca Tarrini, Maurizio Manca, "An Open Standard Solution for Domotic Interoperability," IEEE Transactions on Consumer Electronics, vol. 52, NUMB 1, pp. 97-103, February 2006

[9] Vittorio Miori, Luca Tarrini, Maurizio Manca, "DomoNet: A Framework and a Prototype for Interoperability of Domotic Middlewares Based on XML and Web Services," ICCE 2006 on Consumer Electronics, pp. 117-118, January 2006

[10] 오일진, 임형준, 황윤영, 이규철, "유비쿼터스 웹서비스에서 동적 웹서비스 디스커버리를 위한 UWS Broker의 설계", 한국전자거래학회, 춘계 컨퍼런스, pp.165-174, 2007