논문 13-38C-01-03

한국통신학회논문지 '13-01 Vol.38C No.01
http://dx.doi.org/10.7840/kics.2013.38C.1.19

IoT 서비스 지능화를 위한 디바이스 오브젝트화 및 오케스트레이션 메커니즘

김 영 준◆, 전 용 근°, 정 일 영*

Device Objectification and Orchestration Mechanism for IoT Intelligent Service

Youngjun Kim◆, Yongkeun Jeon°, Ilyoung Chong*

요 약

본 논문은 IoT 서비스 지능화 환경에서 디바이스 웹 서비스로 제공할 수 있도록 물리적인 디바이스를 오브젝트화 하고, 오브젝트화 된 디바이스들을 사용자가 이용할 수 있는 웹 서비스 형태로 제공할 수 있는 스마트 게이트웨이 구조와 기능을 제안한다. 또한 서비스 오버레이 네트워크의 개념을 이용하여 다양한 디바이스 웹 서비스들을 오케스트레이션 하여 사용자에게 제공할 수 있는 메커니즘을 제안한다. 그리고 스마트 게이트웨이와 웹 서비스 플랫폼 사이의 디바이스 웹 서비스 및 서비스 오케스트레이션을 위한 인터페이스를 정의한다. 본 논문을 통해서 오브젝트화된 디바이스는 서비스 오버레이 네트워크를 이용하여 웹 서비스 오브젝트와 다른 디바이스의 콘텍스트 등과 결합하여 IoT 환경에서 지능적이고 능동적인 서비스를 제공할 수 있다.

Key Words : IoT Intelligent Service, M2M Service, Smart Gateway, Device Objectification, Service Overlay Network

ABSTRACT

This paper suggests smart gateway structure and functionalities that provide device web service which can be used by user using objectified devices. The smart gateway objectify a physical device in other to provide the device web service in IoT intelligent service. And the paper suggests a mechanism that provide orchestrated service of various devices to user using a concept of service overlay network. and also suggests a interface that provide device web services and the orchestrated service between the smart gateway and a web service platform. In the paper, the objectified devices can be combined with web service objects and device contexts based on the service overlay network, furthermore the objectified devices can provide intelligence and dynamic service in the IoT environment.

Ⅰ. 서 론

2012년 가트너의 10대 전략 기술 중 하나가 IoT(Internet of Things)이다. IoT는 기존의 USN (Ubiquitous Sensor Network)과 유사하지만 전혀 새로운 기술로, 현재 국내외의 많은 연구 단체, 산

※ This work was partially supported by the EU ITEA-2 project 10028 "Web-of-Objects" (WoO) funded by MKE(The Ministry of Knowledge Economy) and supervised by KIAT
◆ 주저자 : 한국외국어대학교 정보통신공학과 차세대네트워크연구실, ddanggae@hufs.ac.kr, 정회원
° 교신저자 : 한국외국어대학교 정보통신공학과 차세대네트워크연구실, portions15@gmail.com, 학생회원
* 한국외국어대학교 정보통신공학과 차세대네트워크연구실, iychong@hufs.ac.kr, 종신회원
논문번호 : KICS2012-10-517, 접수일자 : 2012년 10월 31일, 최종논문접수일자 : 2013년 1월 4일

업체 등에서 활발한 연구를 진행 중에 있다. IoT 연구 중 하나인 WoO(Web of Objects)[1] 연구는 유럽의 EUREKA[2] 프로젝트에서 진행 중인 IoT 기술이며, WoO는 IoT에서의 사물을 오브젝트화 하여 가상의 사물을 만들고, 이 오브젝트화된 사물을 다양한 콘텍스트 정보와 결합하여 새로운 서비스를 생성, 제공하는 IoT 환경을 구축하기 위한 연구를 진행 중이다. 그리고 IoT와 관련하여 다양한 표준화 기구에서 표준화를 진행 중에 있는데, 현재 ETSI의 M2M, 3GPP의 MTC, ITU-T의 IoT 등 여러 표준화 기구에서 다양한 이름으로 진행 중에 있다.

IoT 환경이란 다양한 IoT를 구성하는 디바이스들이 사용자의 인지 또는 간섭 없이 서로 간의 연결 및 통신을 통해서 사용자에게 서비스를 제공하는 환경을 말한다. 그리고 이러한 디바이스들이 인터넷으로 연결되어 사용자와 센싱에 의한 주변 환경 정보, 그리고 서비스 등과 결합하여 새로운 서비스를 창출하여 사용자에게 제공할 수 있다. 그러나 이와 같은 IoT 환경을 구성하기 위하여 디바이스들을 단순히 인터넷으로 연결할 경우, 특정 서비스 구조만을 만족하는 IoT 환경 또는 시스템이 될 것이다. IoT에서 고려하는 사물은 인터넷 및 웹 프로토콜이 디바이스에 탑재되어 사물들을 이용한 웹 서비스를 제공하는 구조이다. 그러나 오늘날의 대부분의 사물들은 IP 환경 및 HTTP 프로토콜을 사용하는 것이 아니라, Zigbee, 블루투스 등 가벼운 통신 프로토콜을 사용하고 있다. 기존의 사물들을 IoT 환경에 포함시키기 위해서는 사물들과 웹 서비스 환경 사이에서 IoT 서비스 맵핑이 요구되며, 기존의 홈 네트워크 및 유무선 환경에서 이루어지는 디바이스의 협업 서비스 제공방법에 대한 확장이 필요하다[3].

또한 IoT 환경을 구성하기 위해 요구되는 상호운용성 및 관리 기능은 지능적인 디바이스와 고수준의 인터페이스를 포함하고 있지 못하며, 이는 기존의 Zigbee, 블루투스 등의 인터페이스를 포함하는 디바이스들에 대한 IoT 서비스 제공에 한계가 있다. 따라서 다양한 IoT 환경에서의 디바이스들을 연결하여 서비스를 제공하기 위해서는, 실제 세계의 물리적인 사물들을 다양한 콘텍스트 정보를 기반으로 가상화하고, 각각의 식별자를 통해 서로 구분 짓고 상호작용할 수 있는 디바이스 형태를 제공해야 하며, 정보를 저장하고 일련의 처리과정을 거쳐 변형되어 제공하기 위한 디바이스 프로파일링이 반드시 요구된다. 이와 같은 과정을 본 논문에서는 디바이스 오브젝트화라 하며, 본 논문은 웹 서비스 환경에서 디바이스를 오브젝트화하고, 오브젝트화한 디바이스의 정보를 이용하여 웹 서비스를 제공하는 IoT 서비스 지능화 환경에 대한 논문이다. 또한 서비스 오버레이 네트워크 구조를 이용해서 사용자의 환경에 따른 다양한 사용자 중심의 통합된 디바이스 웹 서비스를 제공하고자 한다.

본 논문의 2장은 본 논문에서 참고하는 연구들에 대한 관련 연구이며, 3장은 서비스 오버레이 네트워크 개념을 이용한 IoT 서비스 지능화에 대해서 기술한다. 4장은 디바이스를 오브젝트화하기 위해서 제안하는 본 논문의 스마트 게이트웨이 구조, 디바이스 오브젝트화 메커니즘, 디바이스 웹 서비스 절차 등을 기술하며, 5장에서 IoT 서비스 지능화를 위한 디바이스 웹 서비스 오케스트레이션 메커니즘을 기술한다. 끝으로 본 논문의 6장에서 결론 및 향후 연구로 끝을 맺는다.

Ⅱ. 관련 연구

IoT 환경은 기존의 M2M, USN을 기반으로 통신하는 사물의 범위를 넓혀 인간과 사물, 서비스 세 가지 분산된 환경 요소들이 인터넷 범위로 확장되어 센싱, 네트워킹, 정보처리 등의 상호작용을 통해 지능적 관계를 갖는 개념이다. IoT에서 사물들은 언제, 어디서든, 어떠한 사물도 서로 연결 가능하도록 하는 것을 지향하며, IoT 서비스는 사물들의 상호작용을 통해 기존의 서비스보다 향상된 서비스를 제공한다. 본 논문에서의 관련연구는 IoT와 관련하여 DIYSE 프로젝트의 내용을 간략히 기술하며, 웹 서비스 표준의 일부분으로 디바이스의 프로파일 정보를 이용한 서비스를 제공할 수 있도록 제공하는 DPWS 기술과 임베디드 시스템, 컴퓨팅, 네트워킹 분야에서 제공되는 다양한 디바이스들의 협업을 위한 연결 방법을 제시하는 SOCRADES 프로젝트를 기술한다.

2.1. DIYSE 프로젝트[4]

DIYSE(Do It Yourself Smart Experiences) 프로젝트는 ITEA2 프로젝트의 일환으로, IoT 환경을 구축하기 위한 다양한 비즈니스 모델을 제시하며 컴퓨팅 능력을 가진 다양한 디바이스들이 지능적인 인터페이스를 통해 사용자와 연결하고 상호작용할 수 있는 환경을 제공한다. 또한 사용자 스스로 환경을 구성함으로써 사용자의 환경에 따라 다른 어플리케이션을 제공하고 제어할 수 있도록 하여 IoT

환경에서 개인화를 추구하고 또 다른 개인, 사회와 상호작용할 수 있는 기능을 제공한다. DIYSE 프로젝트에서는 DIY 패러다임을 추구하며, 가구 및 집안의 장식품들이 네트워크 환경에서 인식되고 사용될 수 있도록 정보를 갖도록 하며, 개방된 IoT 환경에서 사용자 주변의 방대한 사물들을 인식하고 사물들을 연결하여 사용자로 하여금 쉽게 서비스를 생성하고 결합시킬 수 있도록 한다. 그러나 DIYSE 프로젝트는 IoT 서비스의 초기화 단계로 각각의 정의된 서비스에 따른 사물들을 연결하여 서비스를 제공하는 것으로, 지능적인 디바이스를 요구하며, 다양한 사물을 연결하여 새로운 서비스를 창출해 내는데 한계가 있다.

2.2. DPWS

IoT 서비스 환경에서 디바이스는 직접 인터넷에 연결된다. 이를 연결하기 위해서는 디바이스의 관리 및 제어가 필요하며, 디바이스 관리를 위해서 디바이스의 프로파일은 매우 중요한 정보가 된다. 현재 디바이스 프로파일 관리를 위한 많은 연구들이 진행 중에 있다. 그러한 연구들 중에 DPWS(Device Profile for Web Services)[5]는 UPnP(Universal Plug and Play)의 차세대 버전으로 네트워크에 연결된 디바이스에 적합한 경량의 웹 서비스 프로토콜을 부분적으로 명시하며, 임베디드 디바이스와 서비스의 상호작용을 제공하기 위한 웹 서비스 표준이다. 임베디드 디바이스에서 웹 서비스를 사용하기 위해 필요한 디바이스 프로파일 정보를 명시하며, DPWS 프로토콜을 통한 메시징, 서비스 발견, 보안, 명세, 자원을 필요로 하는 디바이스의 이벤팅 기능을 제공하기 위해 사용된다. DPWS 프로토콜 스택은 웹 서비스 검색 및 웹 서비스 이벤팅에 대한 표준을 제공하며, SOAP(Simple Object Access Protocol)과 WS-Addressing에 기반 하여 다양한 웹 서비스를 통합할 수 있다. DPWS 디바이스에 제공되는 서비스 구조는 두 가지 형태로, 호스팅 서비스와 호스티드 서비스로 구분된다. 호스팅 서비스는 네트워크에 연결되어 있는 디바이스의 정보를 스스로 알리고 호스팅되는 디바이스의 정보를 찾아 연결하고 제어한다. 호스티드 서비스는 호스팅 서비스에 의존하여 관련 기능을 제공받는 형태로 구분된다. DPWS의 서비스 구조는 네트워크상의 사용자와 디바이스, 그리고 연결된 서비스 사이에 주고받는 메시지 배치 구조를 나타내며, 메시지는 디스커버리, 명세, 제어, 이벤팅과 관련된 콘텍스트 정보를 포함하여 XML 형태의 데이터가 포함된 SOAP 메시지를 주고받게 된다. DPWS 표준을 이용하여 유럽의 SIRENA(Service Infrastructure for Real-time Embedded Networked Applications) 프로젝트[6]는 임베디드 디바이스를 위한 다양한 개발 툴을 제공하여 산업 자동화, 홈오토메이션, 자동차 및 통신 분야에 적용 가능한 어플리케이션 통합을 목표로 연구를 완료하였으며, SIRENA 프레임워크를 기반으로 DPWS를 이용한 웹 서비스 개발을 위해 WS4D (Web Service for Devices), SODA (Service- Oriented Device Architecture), OSAMI (Open Source AMbient Intelligence) 프로젝트로 확장되어 실질적인 구현 및 개발을 목표로 연구 중이다. DPWS는 SOAP 기반 표준을 제공하기 때문에 이미 많이 상용화되어 있고, 메시지를 교환하는 과정에서 표준을 따르기 때문에 다른 표준과 통합이 쉬우며, 방화벽 외부와의 자원 공유가 가능한 장점이 있다. 그러나 HTTP를 이용하지만 별개의 프로토콜로 인코딩 및 디코딩 과정이 필요하며, 프로토콜 자체가 무겁고 복잡하기 때문에 오버헤드가 발생하게 되고 표준에 맞춰 다양한 툴을 사용하여 개발하는 어려움이 있다.

2.3. SOCRADES 프로젝트[7]

SOCRADES(Service-Oriented Cross-layer infrastructure for Distributed smart Embedded device) 패러다임과 IoT 서비스 환경을 반영하여 디바이스와 연결된 웹 서비스를 구성하여 차세대 산업 자동화 시스템을 위한 설계, 실행 및 관리에 대한 플랫폼을 연구하고 있다. SOCRADES 프로젝트에서 제안하는 SIA(SOCRADES Integration Architecture) 구조는 임베디드 디바이스를 이용하여 실질적이고 통합된 서비스를 제공하기 위해 제안되었으며, 하드웨어 및 소프트웨어 시스템의 모델링, 디자인, 동작을 위한 방법을 제시한다. SIA 구조는 산업 자동화 시스템에서 디바이스들에 대한 실시간 모니터링 기능 및 관리기능을 제공하며, 디바이스의 프로파일 및 정보 전달을 위해서 DPWS, REST(Representation State Transfer)를 지원하는 IP 디바이스와 Non-IP 디바이스에 대한 동적 디스커버리 방법을 제공한다. 그리고 SIA 구조는 다양한 디바이스를 엮은 매쉬업 서비스를 제공하여 웹 2.0에서 제공하는 서비스들을 통합하고, 비즈니스 프로세스 과정을 효과적이고 효율적으로 제공한다. SIA 구조는 어플리케이션 인터페이스, 서비스 관리, 디바이스 관리, 보안, 추상화 플랫폼, 디아비스 레이어로 구성된다. 그리고 SIA

구조에서는 DPWS 기반의 WS-Discovery 방법과 능동적 RESTful, 수동적 RESTful의 3가지 디바이스 디스커버리 방법을 사용하여 유비쿼터스 및 분산 컴퓨팅 환경에서 비스를 통합하고 디바이스를 탐색하는 중요한 기능을 제공한다. SOCRADES 프로젝트에서 제안된 SIA 구조는 웹 서비스 기반 기업의 산업 자동화 시스템과 같은 전사적 관리 시스템에서 유용하게 쓰일 수 있으나, 큰 규모에서 사용하기에 아직까지는 초기 개발 단계이며 다양한 디바이스에 대한 로컬 시스템에서의 동작 및 외부에서의 접근, 제어에 대한 확장성에 한계가 있다.

Ⅲ. IoT 서비스 지능화 구조를 위한 아키텍처

3.1. IoT 서비스 지능화를 위한 웹 서비스 플랫폼 아키텍처

다양한 장치와 사물을 이용해서 IoT 서비스 환경을 구성하기 위해서 IoT 서비스를 제공하는 플랫폼은 상호운용성, 범용성, 통신효율성, 이동성, 지능화 및 능동화 기능 등의 요구사항을 가진다. 또한 IoT 서비스를 제어 및 관리하기 위하여 오류관리, 컴포지션 및 서비스 관리, 위치 관리, 콘텍스트 관리, 트래픽 관리, 보안 및 프라이버시 등의 관리 기능도 요구된다[8].

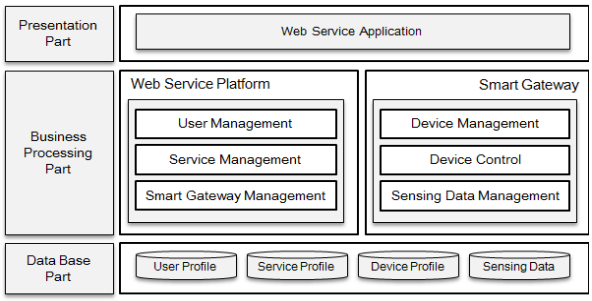

그림 1. 웹 서비스 플랫폼 구조를 이용한 서비스 지능화 아키텍처
Fig. 1. IoT intelligent service architecture using web service platform

본 논문에서 제안하는 IoT 서비스 지능화 아키텍처는 기본적으로 웹 서비스 플랫폼 구조를 이용한다. 웹 서비스 플랫폼을 이용하므로, 여러 사용자나 기존의 환경에서 쉽게 적용할 수 있는 범용성을 가진다. 그림 1은 본 논문에서 제안하는 웹 서비스 플랫폼 구조를 이용한 IoT 서비스 지능화 아키텍처이다. 그림 1에서 프레젠테이션 파트는 웹 서비스를 사용자에게 제공하는 어플리케이션을 담당하고, 비즈니스 처리 파트는 웹 서비스 플랫폼의 기능과 스마트 게이트웨이 기능으로 나눠진다. 웹 서비스 플랫폼은 사용자, 서비스, 그리고 분산된 스마트 게이트웨이를 관리하는 기능을 포함하며, 스마트 게이트웨이는 디바이스 관리, 디바이스 제어, 그리고 디바이스 센싱 정보를 관리한다.

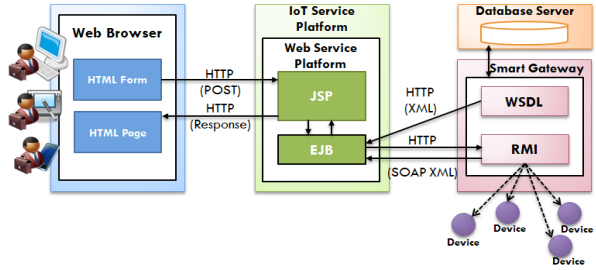

그림 2. IoT 서비스 지능화 아키텍처에서 각 시스템 간의 인터페이스 환경
Fig. 2. Interface environment between each systems in IoT intelligent service architecture

그림 2는 본 논문에서 제안하는 IoT 서비스 지능화 아키텍처에서 각 시스템 사이의 인터페이스 환경을 나타낸다. 사용자에게 제공하는 서비스는 브라우저를 이용한 웹 서비스 형태로 제공하며, 사용자는 웹 페이지의 폼 등을 이용하여 쉽게 서비스를 제공받을 수 있는 구조이다. 그리고 웹 서비스 플랫폼은 각각의 디바이스를 관리하는 스마트 게이트웨이를 통해서 디바이스 웹 서비스를 사용자에게 제공할 수 있다.

스마트 게이트웨이는 스마트 홈, 스마트 빌딩, 그 밖의 센서들 주변에 위치하여 Non-IP 디바이스에게 웹 서비스를 제공할 수 있게 한다. IoT 서비스 플랫폼의 요구사항 중, 지능화 및 능동화에 대한 요구사항을 만족하기 위해서는 단순 웹 서비스 플랫폼으로써 부족한 측면이 있다. 본 논문에서는 지능화 및 능동화를 만족시키기 위해서, 더 나아가 서비스 컴포지션, 서비스 오케스트레이션 등을 통한 새로운 서비스를 제공하기 위해서 서비스 오버레이 네트워크 개념을 이용하였다. 그리고 다양한 디바이스 웹 서비스의 관리 기능, 즉 오류 관리, 컴포지션 및 디바이스의 서비스 관리, 콘텍스트 관리 등을 만족하기 위해서 스마트 게이트웨이를 이용한 디바이스의 프로파일 관리와 센싱 정보 관리를 제공할 수 있다.

3.2. 서비스 오버레이 네트워크를 이용한 IoT 지능화 서비스 모델

본 논문에서 IoT 서비스 환경에서 능동화 및 지

능화 기능을 추가하기 위해서 서비스 오버레이 네트워크 개념을 이용하였다. 그림 3은 디바이스를 오브젝트화 하여 서비스 오버레이 네트워크로 구성하는 개념을 설명한 것이다.

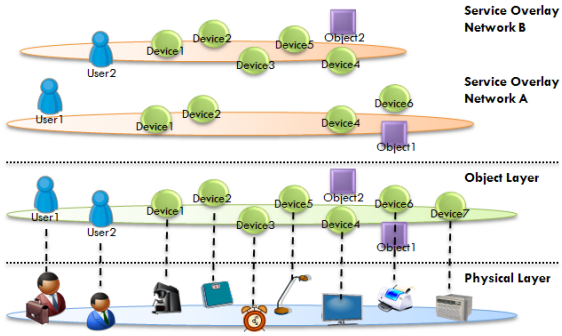

그림 3. 디바이스의 오브젝트화 및 서비스 오버레이 네트워크 구성
Fig. 3. Service overlay network configuration and device objectification

본 논문에서 의미하는 디바이스의 오브젝트화는 웹에서 프로그램들이 이용할 수 있도록 디바이스를 객체화한다는 것이다. 그림 3에서 각각의 User와 Device 등은 Physical world에 존재하는 사물과 사람을 오브젝트화 하여 웹 서비스를 제공하는 요소로 쓰인다. 그리고 각각의 object는 웹에 존재하고 있는 요소를 의미한다. IoT 서비스 환경은 단순히 디바이스를 오브젝트로 표현하여 서비스를 제공하는 것이 아니라, 웹의 다양한 오브젝트와 컴포넌트를 오브젝트화 된 디바이스와 결합하여 새로운 IoT 서비스를 제공하기 때문에, 본 논문에서는 오브젝트들을 결합하기 위해서 서비스 오버레이 네트워크의 개념을 사용하였다.

Ⅳ. 스마트 게이트웨이 기능 구조 및 디바이스 오브젝트화 메커니즘

4.1. 스마트 게이트웨이 기능 모델

4.1.1. 스마트 게이트웨이의 Use-Case 모델링

IoT 환경에서 스마트 게이트웨이를 이용한 디바이스 웹 서비스에 대한 Use-Case를 모델링하여, 이에 필요한 기능들을 포함한 스마트 게이트웨이 구조를 디자인하였다. 스마트 게이트웨이에서 제공해야 하는 디바이스 웹 서비스의 요소는 다음과 같다.

- 디바이스 등록
- 디바이스 프로파일 정보 저장 및 전달
- 디바이스 센싱 정보 저장 및 전달
- 디바이스 제어
- 디바이스 웹 서비스의 통합 및 관리

제공되는 디바이스 웹 서비스의 요소는 단일 디바이스 서비스에 관련된 것으로, 웹 서비스 플랫폼을 통한 요청에 의해서 전달된다. 그러나 디바이스 웹 서비스의 통합 및 이를 이용한 서비스 관리는 단일 서비스가 아니라, 디바이스의 통합된 서비스를 제공 및 관리하기 위한 것이다.

그림 4는 스마트 게이트웨이에서 제공하는 기능을 Use-Case 다이어그램으로 모델링 한 것으로, 각각의 시스템은 웹 서비스 플랫폼, 게이트웨이, 그리고 디바이스 및 데이터베이스 서버 등으로 구성된다.

4.1.2. 스마트 게이트웨이의 기능 구조

그림 4의 Use-Case 모델에서 스마트 게이트웨이

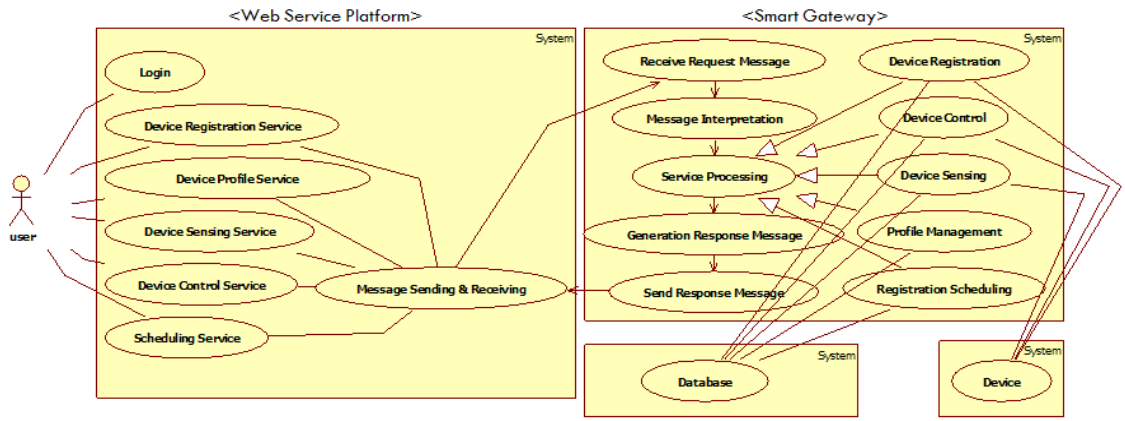

그림 4. 스마트 게이트웨이를 이용한 디바이스 웹 서비스의 Use-Case 다이어그램
Fig. 4. Use-case diagram for device web service using smart gateway

는 자신의 도메인 내에서 디바이스와 관련하여 디바이스 등록, 디바이스 프로파일 관리, 센싱 정보 관리, 디바이스 제어, 디바이스 웹 서비스들의 스케줄링 등의 서비스 처리를 담당한다. 또한 스마트 게이트웨이는 각각의 서비스 처리를 제공하기 위해서 서비스 요청 메시지 수신 및 분석, 서비스 처리 후 응답 메시지 생성 및 전송 등의 기능을 포함한다.

Use-Case 모델에서 각각의 디바이스 웹 서비스를 처리하는 기능들과 처리 과정에 의해서 완성한 스마트 게이트웨이의 기능구조는 그림 5와 같다. 스마트 게이트웨이는 웹 서비스 플랫폼과 디바이스 사이에 위치하며, 스마트 게이트웨이 안의 디바이스들 및 스마트 게이트웨이 자신의 정보를 저장 및 관리하기 위하여 데이터베이스 서버와 연결된다. 그리고 각 기능들 사이의 실선은 디바이스와 관련된 메시지 전달을 의미하며, 점선은 데이터베이스와 관련된 메시지 전달을 의미한다. 그리고 스마트 게이트웨이 구조에서 각각의 기능의 설명은 표 1과 같다.

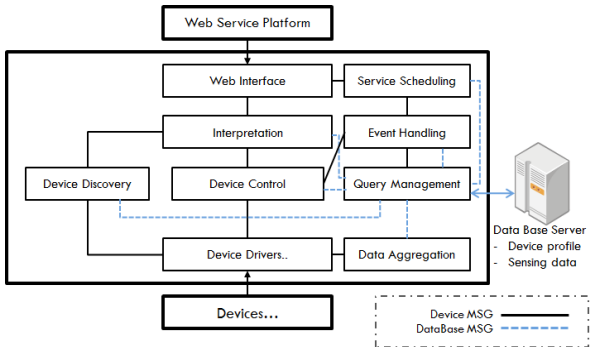

그림 5. IoT 서비스 지능화를 위한 스마트 게이트웨이의 기능 구조
Fig. 5. Functional architecture of smart gateway for IoT intelligent service

4.2. 디바이스 오브젝트화 메커니즘

디바이스의 오브젝트화는 다양한 사물, 디바이스 등을 IoT 환경의 웹 서비스를 제공할 수 있도록 구성하기 위해서 디바이스 또는 사물의 정보, 즉 프로파일을 이용하여 서비스 연동이 가능한 객체로 만드는 것이다. 많은 웹 서비스들이 서비스 통합, 협업 등의 과정을 거쳐서 새로운 서비스를 생성하고 있는데, 기존의 디바이스를 연동할 경우 더 많은 사용자 기반의 서비스가 생성된다. 따라서 디바이스 오브젝트화는 현재 제공되는 다양한 웹 서비스와 디바이스의 서비스를 연결하여 새로운 IoT 서비스를 생성하기 위한 시작점이다.

본 논문의 디바이스 오브젝트화는 다양한 사물, 즉 Non-IP 기반의 디바이스들을 스마트 게이트웨이를 통해서 오브젝트화하고, 오브젝트화된 디바이스를 이용하여 다양한 웹 서비스, IoT 서비스와 연동하는 서비스 콜라보레이션을 제공할 수 있는 장점을 가진다. 본 논문에서는 디바이스의 오브젝트화를 위해서 DPWS[5] 기반의 디바이스 프로파일을 본 논문에 맞게 수정하여 프로파일링 하였다. 모든 디바이스는 디바이스에 해당하는 디바이스 프로파일, 즉 디바이스의 정보를 포함하며, IoT 서비스를 이용하기 위한 메소드, 즉 디바이스 제어 정보를 포함한다. 따라서 디바이스는 디바이스 오브젝트화를 통한 디바이스 명세를 통해서 쉽게 제어할 수 있다. 본 논문에서 IoT 환경의 웹 서비스를 제공하기 위하여 디바이스에 대한 명세는 SOAP 프로토콜 기반의 WSDL(Web Service Description Language) 문서를 제공함으로 이루어진다.

표 1. 스마트 게이트웨이의 각 기능에 대한 설명
Table 1. Description of smart gateway functions

Web Interface	웹 서비스 플랫폼과의 연결 및 전송을 담당하는 웹 프로토콜 인터페이스
Interpretation	수신 메시지 해석, 서비스 처리를 위한 서비스 요청 전달, 응답 메시지 생성 등을 담당
Device Discovery	디바이스 등록 시 그에 대한 처리를 담당, 디바이스 프로파일 정보, 디바이스 제어 정보를 데이터베이스의 디바이스 리스트 테이블에 저장, 센싱 가능한 디바이스의 경우 디바이스 센싱 테이블을 생성
Device Control	RMI와 디바이스 제어 메소드 간의 맵핑 정보를 이용하여 웹 서비스 플랫폼으로 부터의 RMI에 따른 실제 디바이스 제어 메소드 호출, 디바이스 제어의 결과를 데이터베이스의 디바이스 프로파일에 저장
Query Management	스마트 게이트웨이와 데이터베이스 간의 연결을 담당, 스마트 게이트웨이를 통한 데이터베이스에 정보를 저장할 때 쿼리를 작성하여 데이터베이스에 전달
Data Aggregation	스마트 게이트웨이 내 등록된 디바이스 중에 센싱이 가능한 디바이스에게 센싱 정보를 수집하여 데이터베이스의 각 디바이스 센싱 테이블에 저장
Service Scheduling	디바이스 웹 서비스들을 오케스트레이션 하여 서비스 스케줄링을 구성, 트리거에 해당하는 조건을 이벤트 핸들링에게 전달
Event Handling	서비스 스케줄링에 의해 작성된 오케스트레이션 서비스 조건을 저장, 해당 조건에 맞는 이벤트가 발생 시 서비스에 따라서 스마트 게이트웨이의 각 기능에게 서비스 요청
Device Driver	스마트 게이트웨이 영역 내의 디바이스와 연결을 담당 (Zigbee, 블루투스, Wi-Fi 등의 드라이버)

4.2.1. 디바이스 오브젝트화를 위한 디바이스 클래스 구조 정의

디바이스의 프로파일, 디바이스의 제어 메소드

등의 정보를 기반으로 오브젝트화를 하기 위해서 스마트 게이트웨이 내에 그림 6과 같은 클래스를 정의하였다. 디바이스 오브젝트화를 위한 클래스 구조는 DeviceModels 클래스와 DeviceModels 클래스에서 상속받은 각각의 디바이스 제품에 대한 클래스로 정의된다. Device Models 클래스는 스마트 게이트웨이의 영역 내의 모든 디바이스의 프로파일 정보와 메소드를 정의하기 위한 최상위 클래스이며, 모든 디바이스의 클래스가 가지고 있는 일반적인 프로파일에 관련된 변수와 기본 메소드를 포함한다.

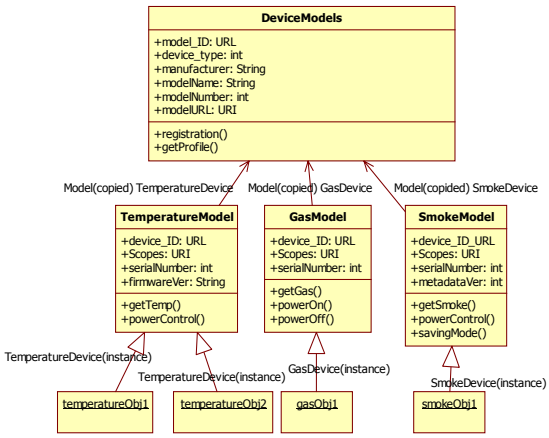

그림 6. 디바이스 프로파일을 이용한 디바이스 클래스 구조
Fig. 6. Device class structure using device profile

DeviceModels 클래스에서 상속받은 각각의 디바이스 클래스는 해당 디바이스만의 기능 및 정보들을 포함한다. TemperatureModel, GasModel, 그리고 SmokeModel은 DeviceModels에서 상속받은 클래스로, 각각의 클래스의 멤버 변수는 고유한 디바이스에 대한 프로파일과 인스턴스화 후 사용될 객체의 실제 디바이스 ID 등을 포함한다. 그리고 멤버 메소드는 각각의 디바이스를 제어하는 메소드들이다. 따라서 인스턴스화 한 디바이스 객체는 디바이스의 등록, 프로파일 요청, 센싱정보 요청, 그리고 디바이스 제어 등 RMI와 맵핑되는 메소드의 정보를 포함하게 된다. temperatureObj1, gasObj1 등의 객체들은 디바이스 클래스를 인스턴스화 한 객체로 실제 스마트 게이트웨이에서 이용하는 객체이다.

4.2.2. 스마트 게이트웨이와 웹 서비스 플랫폼 간의 XML 인터페이스

스마트 게이트웨이는 디바이스의 프로파일과 메소드를 이용한 원격 프로시저를 생성한 후, 웹 서비스 플랫폼을 통한 사용자의 디바이스 웹 서비스 요청에 의해서 디바이스 관련 서비스를 제공하며, RMI 호출을 위하여 스마트 게이트웨이와 웹 서비스 플랫폼 간의 인터페이스 방식은 HTTP 프로토콜 위에서 SOAP XML 메시지 전달 방식을 이용하였다. 그리고 각각의 디바이스 모델은 WSDL 문서를 통해서 디바이스에 대한 프로파일을 명세하고, 관련 RMI를 기술하였다. 웹 서비스 플랫폼은 해당 디바이스 웹 서비스를 이용하기 위해서 WSDL 문서를 이용하여 스텁을 생성하여 디바이스 웹 서비스를 제공받는다.

```
<?xml version="1.0" encoding="UTF-8"?>

<definitions>
   <types>
         ……
   </types>
   <message>
         ……
   </message>
   <portType name="model_ID">
      <operation name="registration" parameterOrder="… …">
         <input message="tns:deviceID_registration"/>
         <output message="tns:deviceID_registrationResponse"/>
      </operation>
      <operation name="getProfile" parameterOrder="… …">
         <input message="tns:deviceID_getProfile"/>
         <output message="tns:deviceID_getProfileResponse"/>
      </operation>
      <operation name="getSensingData" parameterOrder="… …">
         <input message="tns:deviceID_getSensingData"/>
         <output message="tns:deviceID_getSensingDataResponse"/>
      </operation>
      <operation name="deviceControl" parameterOrder="… …">
         <input message="tns:deviceID_deviceControl"/>
         <output message="tns:deviceID_deviceControlResponse"/>
      </operation>
   </portType>
   <binding>
         ……
   </binding>
   <service>
      <port>
         ……
      </port>
   </service>
</definitions>
```

그림 7. 디바이스에 따른 WSDL 문서
Fig. 7. WSDL document of device

그림 7은 디바이스에 대한 WSDL 문서 구조로, 스마트 게이트웨이는 자신의 도메인 안의 디바이스들과 연결하기 전에 WSDL 문서를 생성한다. 동일한 디바이스들은 같은 RMI로 제어하며, 각각의 실제 디바이스에 해당하는 ID를 인자로 갖는다. 그러므로 동일한 WSDL 문서를 통해서 동일한 여러 디바이스의 구분이 가능하다. RMI와 관련된 엘리먼트는 portType 엘리먼트로 디바이스 모델에 대한 ID를 인터페이스로 갖는다. 그리고 각각의 RMI에 대한 정의는 operation 엘리먼트를 통해서 정의하며, input 및 output 엘리먼트는 인자 및 리턴값에 대한 정보를 나타낸다. 인자 및 리턴값에 대한 자세한 변수 타입은 message 엘리먼트에서 정의하며, 복합타입, 즉 배열 및 구조체 등의 정의된 인자에 대해서는 types 엘리먼트에서 정의하였다.

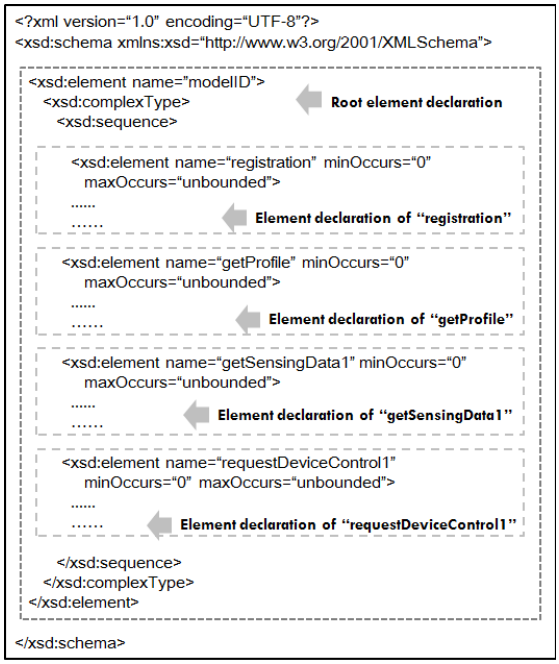

그림 8. 디바이스의 XML 스키마
Fig. 8. XML schema of device

WSDL 문서를 정의한 후, 정의된 WSDL 문서를 웹 서비스 플랫폼에 전달하여 스텁을 생성하면 사용자의 요청에 따라서 웹 서비스 플랫폼은 SOAP XML 메시지를 GET 또는 POST 방식으로 전달이 가능하다. 본 논문에서는 보안 문제나 GET 메시지의 길이 등의 제한에 대한 문제 때문에 POST 방식을 사용한다. 사용자가 웹 페이지의 폼을 이용한 POST 방식의 메시지 전달 시, 웹 서비스 플랫폼은 이를 SOAP XML 메시지 형식으로 제공하는데, 이때 사용하는 디바이스의 XML 스키마는 그림 8과 같다. 디바이스의 XML 스키마는 각각의 디바이스 모델마다 정의되며 디바이스의 WSDL 문서의 RMI 제공 형식과 유사한 구조를 가진다. 디바이스의 XML 스키마는 최상위 엘리먼트로 디바이스 모델 ID를 가진다. 그리고 그 하위의 엘리먼트는 각각 디바이스 등록, 디바이스 프로파일 요청, 디바이스 센싱 정보 요청, 디바이스 제어에 대한 엘리먼트이다. 이를 통해서 서비스 이용 시의 XML 메시지 형식은 모두 해당 메소드에 따른 엘리먼트의 XML 메시지 형식을 따른다. 그리고 디바이스의 WSDL 문서의 operation 엘리먼트, 그리고 XML 스키마에서 정의한 RMI에 해당하는 엘리먼트들은 디바이스에서 제공하는 RMI의 내용과 수에 의해서 정해진다. 디바이스 등록과 디바이스 프로파일 정보 요청을 위한 RMI는 모든 디바이스가 공통으로 이용하는 원격 프로시저이나, 디바이스 센싱과 디바이스 제어에 관련된 RMI는 해당 디바이스 모델에 의해서 결정된다.

4.3. 스마트 게이트웨이를 이용한 디바이스 웹 서비스 제공 방법

4.3.1. 디바이스 등록

디바이스 등록 절차는 스마트 게이트웨이와 디바이스 사이의 물리적인 연결이 된 후, 사용자의 요청에 의해서 스마트 게이트웨이에 디바이스의 등록을 완료하는 절차이다. 그리고 사전에 정의된 디바이스의 WSDL 문서 및 이에 따른 XML 스키마 구조를 이용하여 그림 9와 같은 절차를 진행한다. 사용자의 요청을 받은 웹 서비스 플랫폼은 HTTP 프로토콜을 이용하여 디바이스 등록에 대한 RMI를 SOAP XML 메시지로 전송한다. 스마트 게이트웨이는 SOAP XML 메시지를 수신하여 분석한 후, 디바이스 등록 기능을 통해서 디바이스에게 디바이스의 등록에 대한 연결 확인을 요청한다.

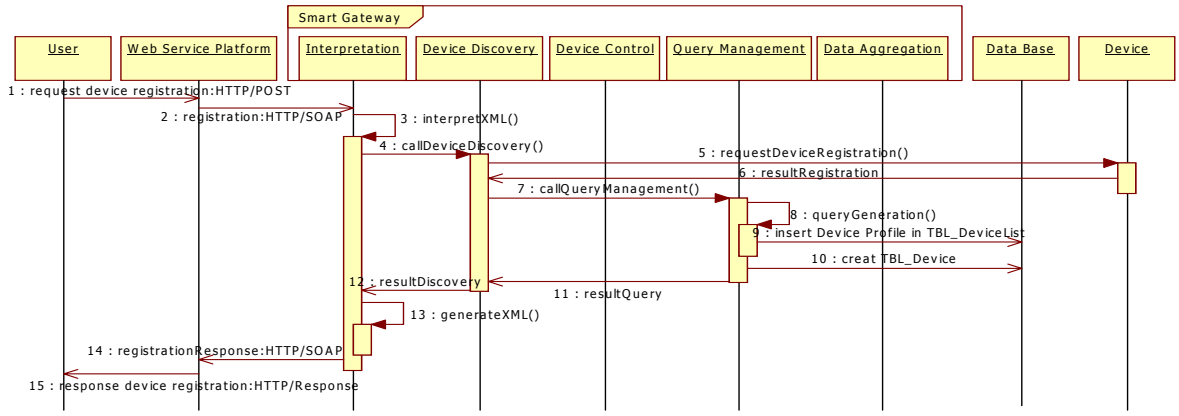

그림 9. 디바이스 등록 절차
Fig. 9. Procedure of device registration

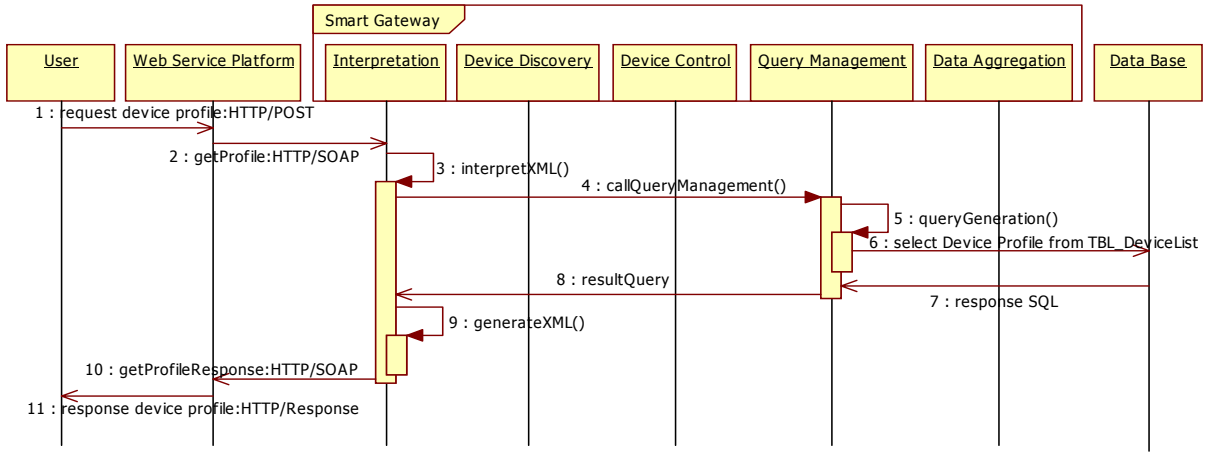

그림 10. 디바이스의 프로파일 제공 절차
Fig. 10. Procedure of device profile provision

이후, 스마트 게이트웨이는 데이터베이스의 디바이스 리스트 테이블에 등록하는 디바이스의 프로파일 정보를 요청하고, 만약 디바이스가 센싱 기능이 있을 경우, 센싱 데이터 수집을 위한 디바이스 테이블을 생성한다. 그리고 이에 대한 결과를 SOAP XML 메시지 형태로 생성하여 웹 서비스 플랫폼에 전달한다.

4.3.2. 디바이스 프로파일 제공

디바이스 프로파일 제공 절차는 사용자가 디바이스의 정보를 확인하거나, 또는 스마트 게이트웨이의 영역 내의 많은 디바이스의 정보들을 확인하고자 할 때 이용하는 절차로 그림 10과 같다. 스마트 게이트웨이는 SOAP XML 메시지를 해석한 후, 디바이스 프로파일을 확인하기 위해서 쿼리 매니지먼트 기능을 통해서 데이터베이스에 접근한다. 디바이스 등록 과정에서 디바이스의 프로파일 정보를 입력했기 때문에 데이터베이스의 디바이스 리스트 테이블에게 디바이스의 ID를 이용하여 정보를 요청, 웹 서비스 플랫폼에게 전달한다. 영역 내의 많은 디바이스의 정보들을 확인하고자 할 때 이용하는 절차로 그림 10과 같다. 스마트 게이트웨이는 SOAP XML 메시지를 해석한 후, 디바이스 프로파일을 확인하기 위해서 쿼리 매니지먼트 기능을 통해서 데이터베이스에 접근한다. 디바이스 등록 과정에서 디바이스의 프로파일 정보를 입력했기 때문에 데이터베이스의 디바이스 리스트 테이블에게 디바이스의 ID를 이용하여 정보를 요청, 웹 서비스 플랫폼에게 전달한다.

4.3.3. 디바이스 센싱 정보 제공

디바이스 센싱 정보 제공 절차는 두 가지의 과정을 통해서 웹 서비스 플랫폼에게 전달된다. 그림 11은 디바이스의 센싱 정보 제공 절차이다. 디바이스 등록 시 센싱 가능한 센서가 동작 가능한 상태에서는 주기적으로 센싱 정보를 보내는데, 스마트 게이트웨이의 데이터 어그리게이션 기능이 이 정보를 수집한다.

데이터 어그리게이션 기능은 쿼리 매니지먼트 기능을 호출하여 데이터를 데이터베이스의 디바이스 테이블에 센싱 정보를 계속 저장한다. 그리고 사용자의 디바이스 센싱 정보 요청이 있을 경우, 스마트 게이트웨이는 웹 서비스 플랫폼을 통해서 SOAP XML 메시지를 수신 및 분석하여 쿼리 매니지먼트 기능을 호출한다. 쿼리 매니지먼트 기능은 데이터베이스의 디바이스 테이블을 통해서 센싱 정보를 받고, 이 결과를 스마트 게이트웨이의 인터프리테이션기능에 전달한다. 이후 인터프리테이션 기능은 SOAP XML 메시지를 생성하여 웹 서비스 플랫폼에 전달한다.

4.3.4. 디바이스 제어

스마트 게이트웨이를 이용한 디바이스 제어 절차는 스마트 게이트웨이 내의 단일 디바이스를 제어하는 과정으로 그림 12와 같다. 디바이스 제어 절차에서 스마트 게이트웨이는 디바이스 제어 기능을 이용하여 RMI와 디바이스 메소드 사이의 맵핑 정보를 확인한다. 이를 통해서 디바이스의 제어 메소드를 호출하고, 그 결과를 다시 데이터베이스의 디바이스 리스트 테

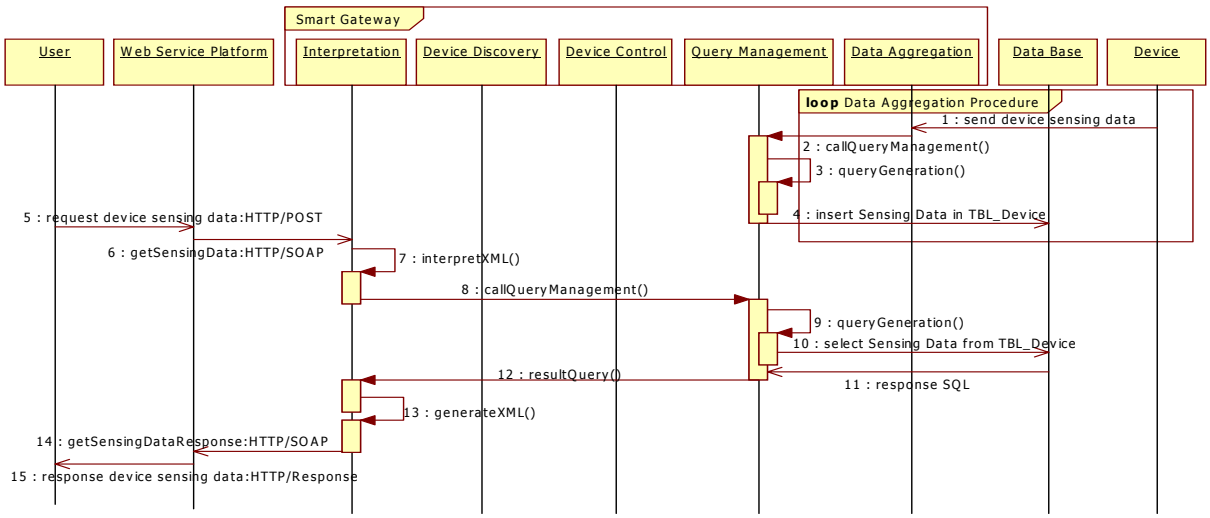

그림 11. 디바이스의 센싱 데이터 제공 절차
Fig. 11. Procedure of sensing data from device

이블에 저장되어 있는 디바이스 프로파일의 상태 정보를 변경하고, 그 결과를 인터프리테이션 기능을 거쳐서 웹 서비스 플랫폼에 SOAP XML 메시지 형식으로 전달한다.

V. 디바이스 웹 서비스의 오케스트레이션 제공 메커니즘

5.1. 오케스트레이션 제공 메커니즘

SOA 기반의 웹 서비스는 각 서비스의 재사용성을 높이기 위해서 서비스를 모듈화 하고, 모듈화 된 서비스들을 조립하여 서비스 오케스트레이션을 제공한다. 서비스 오케스트레이션이란 다양한 서비스를 연합, 협업하여 하나의 새로운 서비스를 제공하는 기능 및 방법을 의미한다. 그러나 현재 IoT 환경의 다양한 디바이스를 이용한 서비스를 기존의 웹 서비스와 통합하여 제공하기 위한 방법이 필요하다. 따라서 디바이스 웹 서비스의 오케스트레이션 제공 메커니즘은 IoT 서비스를 제공하기 위한 디바이스를 이용한 디바이스 웹 서비스들을 순차적으로 제공하는 기술이다. 그림 13은 오케스트레이션 제공 메커니즘을 이용하여 제공되는 디바이스 웹 서비스들을 이용한 서비스 콜라보레이션을 표현한 것이다.

본 논문에서는 스마트 게이트웨이의 스케줄링 기능이 오케스트레이션 메커니즘을 통해서 IoT 서비스의 스케줄링을 제공한다.

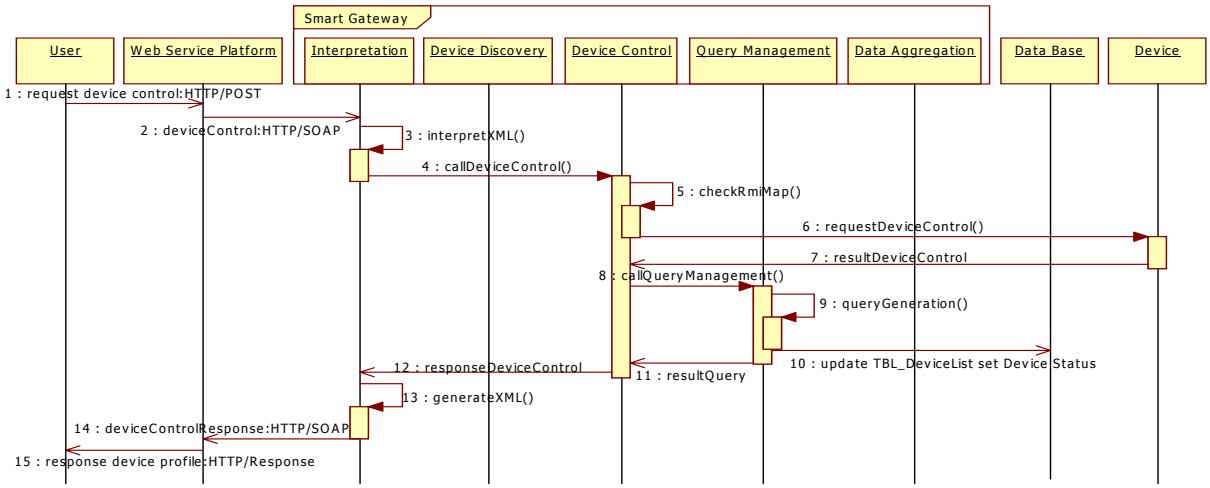

그림 12. 디바이스 제어 절차
Fig. 12. Procedure of device control

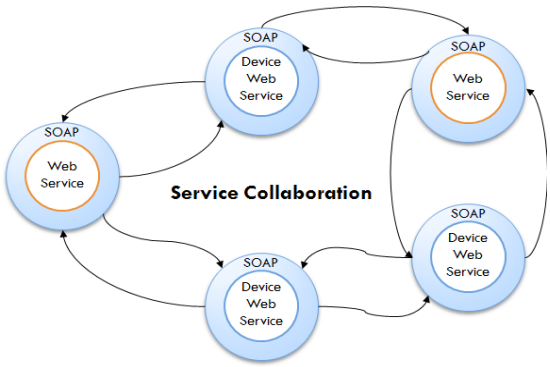

그림 13. 디바이스 웹 서비스의 서비스 콜라보레이션
Fig. 13. Service collaboration of device web service

본 논문에서 오케스트레이션 제공 메커니즘은 디바이스 오브젝트화를 통한 디바이스의 정보를 기반으로 구성한다. 모든 오브젝트화 된 디바이스들 중에 사용자가 선택한 디바이스들은 서비스 레벨의 네트워크로 서로 연결한다. 그리고 이 연결에는 서비스 시작(트리거), 다음 서비스의 시작을 위한 조건(컨디션), 그리고 다음 서비스를 실행하기 위한 시간(타이밍)을 가지고 있다. 사용자 또는 관리자는 스마트 게이트웨이와 연결된 디바이스의 리스트에서 원하는 디바이스를 선택하고, 선택한 디바이스들을 트리거, 컨디션, 타이밍으로 연결한다. 이후, IoT 환경의 디바이스 웹 서비스는 해당 트리거가 발생할 때, 순차적으로 각각의 디바이스 서비스를 실행한다.

5.2. IoT 서비스 지능화를 위한 디바이스 웹 서비스의 시스템 아키텍처

본 논문에서 제안하는 디바이스 오브젝트화와 오브젝트화 된 디바이스들을 이용하여 사용자에게 디바이스 웹 서비스를 제공하기 위해서는 웹 서비스 환경의 구축이 필요하다.

그림 14는 본 논문에서 제안하는 디바이스 웹 서비스 제공을 위한 IoT 개발 환경을 나타낸다. 그림 14에서 사용자의 단말은 웹 프로토콜을 탑재하고, 웹 브라우저를 이용해서 웹 서비스 플랫폼까지 HTTP/POST 방식으로 메시지를 전송한다. 그리고 웹 서비스 플랫폼이 접근하는 데이터베이스는 사용자 프로파일, 서비스 프로파일, 그리고 스마트 게이트웨이의 정보 등을 저장한다. 스마트 게이트웨이가 이용하는 데이터베이스는 디바이스의 프로파일, 센싱 데이터들, 그리고 스마트 게이트웨이에 연결되어 있는 디바이스들의 리스트 등을 저장한다. 그리고 디바이스와 통신을 위해서 스마트 게이트웨이는 디바이스가 이용하는 통신 드라이버를 포함하고 있다. Non-IP의 디바이스일 경우, 스마트 게이트웨이부터 웹 서비스 플랫폼까지 DPWS 메시지 형태를 이용한다. 그러나 IP 디바이스의 경우, 디바이스부터 웹 서비스 플랫폼까지 DPWS 메시지 형태로 확장이 가능하다.

5.3. 디바이스 웹 서비스의 오케스트레이션 제공 방법

디바이스 웹 서비스의 오케스트레이션은 스마트 게이트웨이의 서비스 스케줄링 기능을 통해서 제공하며, 오케스트레이션의 시작은 이벤트 핸들링 기능을 통해서 시작된다. 스마트 게이트웨이의 서비스 오케스트레이션은 서비스 오버레이 네트워크의 개념을 이용한다.

그림 15는 서비스 오버레이 네트워크 개념을 이용하여 스마트 게이트웨이의 서비스 스케줄링 기능을 통한 디바이스 웹 서비스의 구성 예이다. 기존의 서비스 오버레이 네트워크의 구성과 다르게, 디바이스 웹 서비스의 오케스트레이션은 각각의 오브젝트의 연결에 Condition과 Interval의 값과 방향성을 가

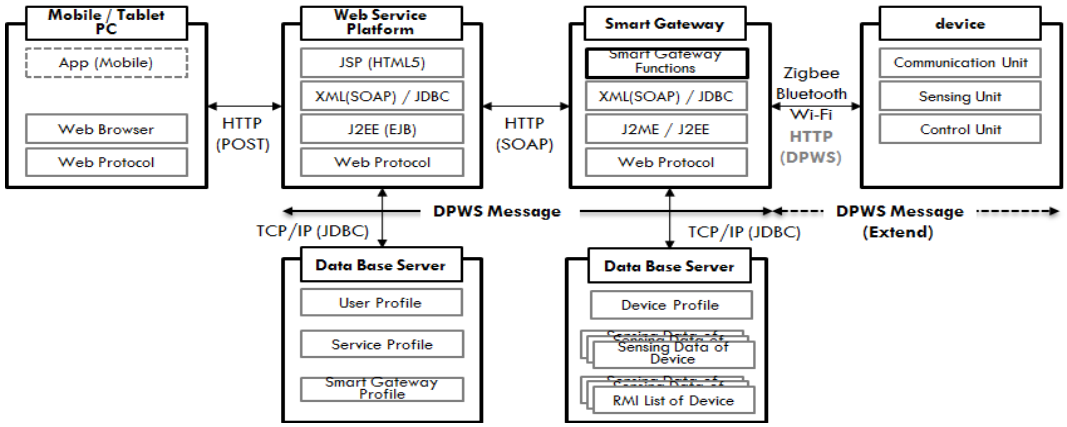

그림 14. 디바이스 웹 서비스의 시스템 아키텍처
Fig. 14. System architecture of device web service

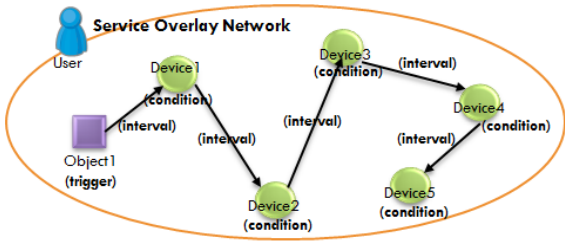

그림 15. 디바이스 웹 서비스의 오케스트레이션 예
Fig. 15. Example of orchestration for device web service

진다. 각 사용자가 지정한 스케줄링 서비스에 따라서 다양한 디바이스를 엮어서 오케스트레이션 서비스를 제공한다. Object1은 트리거로 동작하는데, 이벤트 핸들링에 등록되는 첫 오브젝트의 Condi- tion과 Interval을 포함한다. 그리고 트리거는 특정 조건(Condition)에 의해서 Interval의 시간 후에 다음 오브젝트를 호출한다. 스마트 게이트웨이의 이벤트 핸들링은 트리거에 의해서 연결된 오브젝트가 실행된 후, 오브젝트의 Condition과 Inter- val에 의해서 다음 오브젝트를 호출하여 서비스 스케줄링에 등록된 디바이스들을 순차적으로 실행한다.

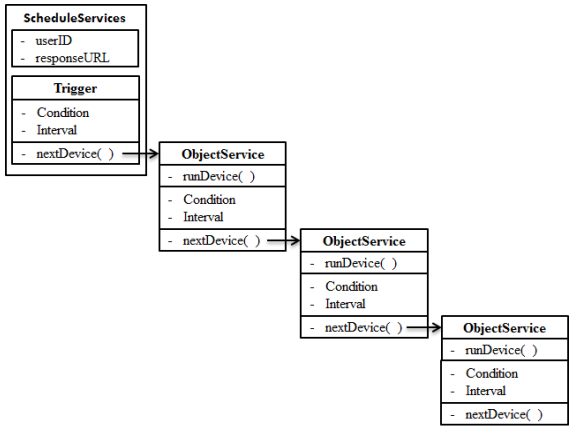

그림 16. 서비스 스케줄링 기능의 서비스 오케스트레이션 구조
Fig. 16. Service orchestration structure of service scheduling function

그림 16은 서비스 스케줄링 기능의 서비스 오케스트레이션 구조를 나타내며, ScheduleServices 클래스를 통해서 스케줄링 서비스를 구성한다. Sche- duleServices 클래스는 userID와 스케줄링 서비스가 동작 한 후의 응답 메시지를 전송할 수 있는 responseURL을 포함한다. 그리고 Schedule- Services 클래스는 Trigger 클래스를 포함하는데, Trigger 클래스는 Condition과 Interval을 변수로 가지며, nextDevice( )를 오퍼레이션으로 가진다. nextDevice( )를 통해 연결된 ObjectServ- ice는 Trigger와 동일한 클래스 구조에 추가로 runDevice( )를 오퍼레이션으로 포함한다. 각 변수와 오퍼레이션에 대한 설명은 표 2와 같다.

표 2. 서비스 스케줄링 구조에서의 변수 및 오퍼레이션 설명
Table 2. Description of variable and operation in service scheduling structure

	Variable/Operation Explanation	Default Value
userID	디바이스 웹 서비스를 이용하는 사용자 ID [int]	0
responseURL	스케줄링 서비스의 실행 후, 리턴되는 메시지의 destination URL [URL]	NULL
Condition	디바이스 웹 서비스 실행을 위해서 (오브젝트.값, 비교연산자, 대상값) 으로 구성하는 조건식 [Boolean]	FALSE
Interval	디바이스를 실행하기 전의 시간 간격으로, 초 단위로 동작 [unsigned int]	0
nextDevice( )	ObjectService 클래스를 호출하는 오퍼레이션 [argument: reference point]	NULL
runDevice( )	디바이스의 RMI를 호출하는 오퍼레이션 [argument: reference point]	NULL

표 2에서 Condition의 값이 디폴트인 FALSE일 경우, nextDevice( ) 함수는 호출되지 않는다. 만약 Condition을 조건식으로 설정하지 않고, TRUE로 셋팅할 경우, 하나의 디바이스가 실행된 후, nextDevice( )는 무조건 실행된다. Interval의 값이 디폴트인 0일 경우, runDevice( )가 실행된 후 nextDevice( )는 곧바로 실행된다. 그리고 runDevice( )는 디폴트가 NULL인데, 마지막 ObjectService 클래스의 runDevice( )의 값이 NULL로 설정된다. 그리고 스마트 게이트웨이의 이벤트 핸들링은 Trigger 클래스의 Condition의 조건식과 해당 ScheduleServices 클래스 인스턴스의 레퍼런스로 구성된 맵핑 테이블을 가지고 있으며, '오브젝트.값'에 대한 데이터를 주기적으로 확인하여 조건식이 TRUE가 될 경우, Schedule- Services 클래스 인스턴스를 동작시킨다. 이후 nextDevice( )에 대한 ObjectService 클래스의 Condition 조건식과 ObjectService 클래스 인스턴스의 레퍼런스를 이벤트 핸들링 기능에 저장하여 다음 디바이스 웹 서비스에 대한 이벤트를 계속 확인이 가능하다.

5.4. 서비스 스케줄링을 위한 스마트 게이트웨이의 XML 형식

```
<?xml version="1.0" encoding="UTF-8"?>

<definitions name="IoT"
   targetNamespace="http://220.67.124.148:8080/wsdl/IoT"
   xmlns:tns="http://220.67.124.148:8080/wsdl/IoT"
   xmlns:utype="http://220.67.124.148:8080/type/IoT"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
   xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime">
~
   <message name="smartGatewayIF_schedulingService">
      <part name="Int_uID" type="xsd:int"/>
      <part name="String_URL" type="xsd:string"/>
      <part name="Struct_Trigger" type="utype:Trigger"/>
      <part name="ArrayStruct_ObjService" type="utype:ArrayofObjService"/>
   </message>

   <message name="smartGatewayIF_schedulingServiceResponse">
      <part name="Result" type="xsd:string"/>
   </message>

   <portType name="smartGatewayIF">
      <operation name="schedulingService"
         parameterOrder="Int_uID String_URL Struct_Trigger ArrayStruct_ObjService">
         <input message="tns:smartGatewayIF_schedulingService"/>
         <output message="tns:smartGatewayIF_schedulingServiceResponse"/>
      </operation>
   </portType>
~
</definitions>
```

그림 17. 스마트 게이트웨이의 WSDL 문서
Fig. 17. WSDL document of smart gateway

서비스 스케줄링을 제공하기 위해서는 스마트 게이트웨이는 이에 대한 RMI를 웹 서비스 플랫폼에게 제공해야 한다. 따라서 스마트 게이트웨이 역시 WSDL 문서와 XML 스키마의 형식이 요구된다.

그림 17은 스마트 게이트웨이의 WSDL 문서의 형식으로, 스마트 게이트웨이에서 제공하는 RMI에 대해서 기술하였다. WSDL 문서에서 portType을 통해서 스마트 게이트웨이와 연결할 수 있는 인터페이스를 정의하였고, operation으로 서비스 스케줄링을 제공하는 원격 프로시저를 정의하였다. 이 원격 프로시저는 스마트 게이트웨이에서 스케줄링 서비스를 구성할 때 필요한 인자들을 나타내며, int 형태의 사용자 ID, URL 형태의 응답 URL, 구조체 형태의 Trigger와 구조체 배열 형태로 오브젝트 서비스들을 인자로 제공한다. 그리고 스케줄링 서비스에 대한 응답 메시지는 스트링 형태로 응답에 대한 결과를 리턴한다.

그림 18은 스마트 게이트웨이의 WSDL 문서를 활용하여 사용자가 웹 서비스 플랫폼을 통해서 스케줄링 서비스를 구성하기 위해서 전송하는 XML SOAP 메시지의 예이다. 사용자의 ID와 응답서비스를 위한 URL, 스케줄링 서비스 시작을 위한 Trigger 구조체, 오브젝트 서비스를 정의한 ObjService 구조체의 배열 등을 인자로 전송한다. ObjService 구조체는 디바이스의 ID, 디바이스에서 사용가능한 RMI, 그리고 Trigger 구조체와 마찬가지로 다음 디바이스를 실행시킬 조건에 해당하는 Condition과 Interval을 가진다. 그림 18은 Trigger를 통해서 현재의 시간이 아침 6시일 때, 시간 간격 없이 온도센서를 통해서 센싱된 현재의 온도를 요청한다. 그리고 1분의 간격으로 가스센서를 통해서 현재의 대기 중 가스 농도, 연기센서를 통한 현재 대기 중의 연기 농도 등을 요청하는 서비스이다.

```
<?xml version="1.0" encoding="UTF-8"?>

<env:Envelop>
  <env:Body>
    <tns:schedulingService>
      <Int_uID xsi:type="xsd:int">12345</Int_uID>
      <String_URL xsi:type="xsd:string">http://220.67.124.147:8080/IoT/pushService</String_URL>
      <Struct_Trigger href="#Trigger1"/>
      <ArrayStruct_ObjService href="#Array_ObjService1"/>
    </tns:schedulingService>
    <Struct_Trigger id="Trigger1" xsi:type="utype:Trigger">
      <condition1 xsi:type="xsd:string">Date.currentHour</condition1>
      <condition2 xsi:type="xsd:string">==</condition1>
      <condition3 xsi:type="xsd:string">06</condition1>
      <interval xsi:type="xsd:int">0</interval>
    </Struct_Trigger>
    <ArrayStruct_ObjService id="Array_ObjService1" xsi:type="utype:Array_ObjService[3]">
      <item href="#ObjService1"/>
      <item href="#ObjService2"/>
      <item href="#ObjService3"/>
    </ArrayStruct_ObjService>
    <item id="ObjService1" xsi:type="utype:ObjService">
      <deviceID xsi:type="xsd:string">220.67.124.148/smartgateway1/tempObj1</deviceID>
      <methodName xsi:type="xsd:string">getTemp</methodName>
~
      <interval xsi:type="xsd:int">60</interval>
    </item>
    <item id="ObjService2" xsi:type="utype:ObjService">
      <deviceID xsi:type="xsd:string">220.67.124.148/smartgateway1/gasObj1</deviceID>
      <methodName xsi:type="xsd:string">getGas</methodName>
~
      <interval xsi:type="xsd:int">60</interval>
    </item>
    <item id="ObjService3" xsi:type="utype:ObjService">
      <deviceID xsi:type="xsd:string">220.67.124.148/smartgateway1/smokeObj1</deviceID>
      <methodName xsi:type="xsd:string">getSmoke</methodName>
~
      <interval xsi:type="xsd:int">60</interval>
    </item>
  </env:Body>
</env:Envelop>
```

그림 18. SOAP XML 메시지의 예
Fig. 18. Example of SOAP XML message

Ⅵ. 결 론

기존의 IoT 연구를 확장하여 다양한 디바이스를 서비스와 연결하는지의 관점에서 디바이스의 오브젝트화는 매우 중요한 이슈이다. 디바이스를 다양한 IoT 서비스, 사용자, 그리고 콘텍스트 등과 같은 동일한 레벨로 만드는 것이 디바이스의 오브젝트화이며, 어떻게 디바이스의 프로파일을 오브젝트화 하는가에 따라서 IoT 서비스의 제공과 확장성에 큰 영향을 준다. 본 논문은 웹 서비스 플랫폼으로부터 디바이스까지 IoT 서비스를 제공할 수 있는 스마트 게이트웨이의 기능과 구조, 디바이스의 오브젝트화 방법, 그리고 디바이스의 서비스를 오케스트레이션 하는 서비스 스케줄링 메커니즘을 정의하였다.

본 논문에서 제공하는 서비스 인터페이스 방식으로 SOAP 프로토콜을 사용하였으며, 스마트 게이트웨이를 통해 디바이스들을 오브젝트화 하고, IoT 환경으로 포함시킴으로써 기존의 웹 서비스가 제공하

는 다양한 오브젝트들과 결합하여 새로운 서비스를 만들 수 있다.

스마트 게이트웨이는 구조적 측면에서 기존의 서비스에 추가적으로 새로운 서비스를 수용할 수 있는 확장성을 가지며 이에 대한 성능 분석을 향후 연구할 예정이며, 디바이스의 서비스를 지능화, 능동화하여 사용자로 하여금 안정적인 서비스를 제공받기 위한 응답시간 및 QoS/QoE 보장에 대한 성능 분석도 진행할 예정이다. 그리고 스마트 게이트웨이의 영역 내의 디바이스에서만 서비스 스케줄링을 제공하는 문제를 확장하여, 분산 구조로 연결된 스마트 게이트웨이를 통해서 사용자가 여러 스마트 게이트웨이의 다양한 디바이스를 이용한 서비스 오케스트레이션을 제공할 수 있도록 확장하는 연구를 진행하고자 한다.

References

[1] http://www.web-of-objects.com/wiki

[2] http://www.eurekanetwork.org

[3] D. C. Kim and K. S. Chung, "Design of system architecture for device collaboration service in home network environments," in *Proc. KICS Winter Conf. 2011(KICS WC 2011)*, pp. 297, PyeongChang, Korea, Jan. 2011.

[4] http://www.dyse.org

[5] http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01

[6] H. Bohn, A. Bobek, F. Golatowski, "SIRENA – service infrastructure for real-time embedded networked devices: a service-oriented framework for different domains," in *Proc. Int. Conf. on Netw. 2006 (ICN 2006)*, pp. 43-48, Apr. 2006.

[7] http://www.socrades.eu

[8] Y. J. Kim, Y. K. Jeon, and I. Y. Chong, "Service overlay network platform and service composition method for M2M service," in *Proc. Korea Computer Congress 2012 (KCC 2012)*, pp. 230-232, Jeju Island, Korea, Jun. 2012.

김 영 준 (Youngjun Kim)

2007년 한국외국어대학교 컴퓨터 및 정보통신공학과 석사

2007년~현재 한국외국어대학교 컴퓨터 및 정보통신공학과 박사 과정

<관심분야> Service Overlay Network, Web of Objects

전 용 근 (Yongkeun Jeon)

2012년 한국외국어대학교 정보통신공학과 학사

2012년~현재 한국외국어대학교 컴퓨터 및 정보통신공학과 석사 과정

<관심분야> Service Overlay Network, Web of Objects

정 일 영 (Ilyoung Chong)

1992년 Univ. of Massachusetts 전산학전공(공학박사)

1996년~현재 한국외국어대학교 정보통신공학과 교수

<관심분야> Service Overlay Network, Home Network, NGN, Web of Objects 등