논문 14-39C-02-06

한국통신학회논문지(J-KICS) '14-02 Vol.39C No.02
http://dx.doi.org/10.7840/kics.2014.39C.2.164

무선 인터넷 공유기에서 사물 인터넷 장치 제어를 위한 가상 오브젝트 호스팅 기술 연구

양 진 홍◆, 박 효 진*, 김 용 록**, 최 준 균°

A Virtual Object Hosting Technology for IoT Device Controlling on Wireless AP's

Jinhong Yang◆, Hyojin Park*, Yongrok Kim**, Jun Kyun Choi°

요 약

IoT(Internet of Things)로 대표되는 사물 인터넷 환경에서 최근 개인 건강 및 홈 자동화 관련 센서류 장치들이 급격히 보급되고 있다. 설치된 장치의 수가 증가함에 따라 사용자 입장에서는 이용하고자 하는 장치마다 개별 서비스 또는 어플리케이션을 찾아 실행해야 하는 번거로움이 생기고, 서비스 공급자 입장에서는 장치간의 정보 공유나 연동 및 통합 제어 기능을 제공하기 어려운 문제가 심화된다. 본 논문에서는 이러한 문제점을 해결하기 위해 댁내 인터넷 서비스의 거점이 되는 무선 인터넷 공유기를 이용, 다양한 IoT 장치들을 연동하고 통합 제어하는 방법을 제시 한다. 이를 위하여, 먼저 IoT 장치를 가상 오브젝트(Virtual Object)화하는 방법을 제시하고, 생성된 가상 오브젝트들을 무선 인터넷 공유기상에서 실행, 연동 및 관리하기 위한 서비스 구조를 제안한다. 또한 이를 직접 홈 환경의 무선 인터넷 공유기상에 구현해 다수의 IoT 장치들을 연결한 상황에서 안정적인 서비스 제공 여부와 무선 라우팅 기능과 병행 동작시의 성능에 대해 시험함으로써, 실제 환경에서의 이용 가능성을 검증한다.

Key Words : Internet of Things, Virtual Object, Web of Object, Instance Hosting, Instance Manager

ABSTRACT

Recently, as the number of IoT (Internet of Things) devices for personal or home intelligence increases, the need for unified control and cooperative utilization is required. This paper presents a novel idea, proposes methods for virtualizing Internet of Things (IoT) devices and hosting them on the home AP instead of relying on a cloud service or purchasing a new device to do so. For this, the process and profile of the IoT gadgets need to be virtualized into JavaScript-based objects. Then, to execute and control the instances of the virtualized IoT objects on the wireless AP, a novel instance management method and their interfaces are designed. The implementation and performance section demonstrates the proposed system's stability and operability by showing the stress test results while the wireless AP is running for its wireless routing.

※ 본 연구는 EU ITEA-2 프로젝트(10028, Web-of-Objects) 지원 및 한국산업기술진흥원 관리로 수행되었습니다.
※ 본 연구는 미래창조과학부 '범부처 Giga KOREA 사업' (GK13P0100, Giga Media 기반 Tele-experience 서비스 SW플랫폼 기술 개발)의 일환으로 수행되었습니다.
※ 본 연구는 미래창조과학부 및 한국산업기술평가관리원의 산업융합원천기술개발사업(정보통신)(10044454, 기기 정보뿐 아니라 사용자의 환경/감성/인지 정보에 적응적으로 반응하는 정보기기용 원격 UI 기술 개발)의 일환으로 수행되었습니다.
◆ First Author : 한국과학기술원 정보통신공학과 미디어네트워크 연구실, sunupnet@kaist.ac.kr, 정회원
° Corresponding Author : 한국과학기술원 정보통신공학과 미디어네트워크 연구실, jkchoi59@kaist.edu, 종신회원
* 한국과학기술원 정보통신공학과 미디어네트워크 연구실, gaiaphj@kaist.ac.kr
** 한국과학기술원 IT융합연구소 미래디바이스팀, yrkim@itc.kaist.ac.kr
논문번호 : KICS2013-10-478, 접수일자 : 2013년 10월 30일, 심사일자 : 2013년 12월 19일, 최종논문접수일자 : 2014년 1월 27일

Ⅰ. 서 론

IoT(Internet of Things)으로 대표되는 사물 인터넷 환경에서 최근 개인 건강 및 홈 자동화와 관련된 다양한 장치(Sensory Devices, Gadgets)들이 급격히 보급되고 있다. 기존의 센서류 시장이 산업체 또는 서비스 공급자를 대상으로 하던 것과 달리, 개인 및 홈 환경의 개인 사용자를 대상으로 하는 센서류 서비스 시장이 활성화되고 있는 것이다. 이는 차차 개인이 보유하는 기기의 범위가 센서류로 확대되고, 그 개수도 증가하게 됨을 의미한다.

현재 사용자가 직접 구입 및 설정하여 이용 가능한 IoT 장치들은 앱 인에이블드(App-Enabled) 액세서리[1] 또는 스마트 홈용 가젯 및 센서 등이 있으며 각각은 전용 서비스 프로그램, 즉 모바일 어플리케이션이나 웹 서비스 또는 임베디드 시스템 내장형 서비스를 통해 제어 가능한 형태를 띠고 있다. 이러한 서비스 제공 방식은 장치의 수가 증가함에 따라 사용자 입장에서는 이용하고자 하는 장치마다 개별 서비스 또는 어플리케이션을 찾아 실행해야 하는 번거로움이 생기고, 서비스 공급자 입장에서는 장치간의 정보 공유나 연동 및 통합 제어 기능을 제공하기 어려운 문제가 심화된다. 뿐만 아니라 개별적인 장치들의 수직화된 솔루션 구성은 IoT 에코 시스템 조성에도 걸림돌로 작용되고 있다[2,3].

이러한 문제점을 해결하기 위해 개별 서비스(모바일 또는 웹)상에서 기기 등록을 통해 다른 IoT 장치들간의 연동 기능을 지원하거나 클라우드 기반 IoT 장치 연동 및 제어 서비스들이[1)2)] 제공되고 있다[4]. 사용자 입장에서 보다 능동적인 형태인 매쉬업 서비스 생성 기능 까지 제공하는 형태로[5] 등의 서비스가 존재하나 제한된 채널에 연결된 장치들만이 연동 가능하다는 한계점을 가지고 있다. 따라서 사용자 입장에서 자신이 원하는 다양한 장치간의 연동을 통한 스마트 서비스를 생성하고 이용하기에는 제한적이다.

본 논문에서는 이러한 문제점을 해결하기 위해 댁내 인터넷 서비스의 거점이 되는 무선 인터넷 공유기를 이용, 다양한 IoT 장치들을 연동하고 통합 제어하는 방법을 제시 한다. 무선 인터넷 공유기를 IoT 장치 제어에 활용함으로써 얻는 이점은, 첫째, 별도의 기기를 구매하지 않고 기존의 장비를 활용할 수 있다는 점, 둘째, IoT 장치에서 생성되는 개인적인 데이터들을 안전하게 집 안에 두고 관리할 수 있다는 점, 셋째, 클라우드나 서버 기반의 서비스와 달리 네트워크 상황에 따른 지연이나 소실 없이 보다 즉각적인 서비스 응답이 가능하다는 점을 꼽을 수 있다. 이와 같은 이점을 활용하기 위하여 지금의 무선 인터넷 공유기가 가진 제한된 리소스와 임베디드 프로그래밍 환경을 고려한 최적화된 IoT 장치 통합관리 시스템 설계 및 연동 서비스 운영기술을 제안 한다.

이를 위해 먼저 IoT 장치에 대해 기존과 차별화 된 가상 오브젝트(Virtual Object)화 방법을 설명하고, 생성된 가상 오브젝트들을 댁내 무선 인터넷 공유기상에서 실행, 동작 및 관리하기 위한 인터페이스 설계와 서비스 운영 시스템 구조를 제안한다. 또한 이를 직접 홈 환경의 무선 인터넷 공유기상에 구현해 다수의 IoT 장치들을 연결한 상황에서 안정적인 서비스 제공 여부 및 무선 인터넷 공유기의 고유기능인 무선 라우팅 기능의 동작 성능에 대해 시험함으로써 실제 환경에서의 이용 가능성을 검증한다.

Ⅱ. 관련 연구

기존 IoT 관련 연구들은 사물 장치를 인터넷에 연결하는 것에 초점을 맞춘 초기 연구들(예 Cooltown project[6], Tiny Web servers could be embedded in most devices[7])과 인터넷에 연결된 장치를 정보를 효과적으로 활용하기 위한 연구들(예 Pachube[8], SenseWeb[9])로 크게 나누어 볼 수 있다. 이러한 연구들은 다수의 장치 연결 및 연결된 장치의 정보를 통합하고 분석하기 위한 기능에 초점을 두고 있다. 따라서 개인이 실생활에서 사용하는 IoT 장치를 보다 효과적으로 연결해 사용하는 것에 적용하기에는 제한적이다. 그 예로 Pachube(현 Xively)의 경우 IoT 장치로부터 수집된 데이터를 이용해 정규식에 기반한 단순 트리거링 기능만을 제공할 뿐 장치들 간의 메시지 연결 및 별도의 서비스 로직을 통한 매쉬업의 기능은 제공하지 못하고 있다.

사물의 가상화를 위한 연구로는 장치의 정보(device profile) 및 인터페이스 관련 부분과 프로그래밍 모델로 나누어 볼 수 있다. 먼저 장치 정보 표현에 관한 부분의 경우 DPWS(Devices Profile for Web Services)[10]와 같은 표준을 따르거나 각자 서비스 환경에 맞춰 활용하는 경우가 대부분이다. 인터페이스 부분의 경우 메시지 호환성을 위해 SOAP 방식 및 HTTP 기반의 REST 구조를 Web of Things 관련 연구에서 많이 사용하고 있다[11-13]. 프로그래밍 모델의

1) fitbit, Activity Tracker Application, http://www.fitbit.com
2) Withings, Smart Body Analyzer, http://www.withings.com

경우 초기 연구들은 임베디드 환경으로 TinyOS, Contiki 등을 사용한 제한된 시스템을 사용했다. 이에 따라 C 기반의 언어로 장치의 로직을 작성 후 임베디드 환경에 탑재하는 제한적인 서비스 형태만을 지원하였다. 반면 최근의 연구들은 VM (Virtual Machine) 기반의 환경으로 변화하는 추세이다[14,15]. 이를 통해 보다 폭넓은 개발언어의 지원 및 개발자들의 진입장벽을 낮춰 범용적인 서비스 제공 구조의 변화를 꾀하고 있다. 하지만 VM 환경에서의 프로그래밍 모델 또한 각각의 장치에서 직접 수행되는 형태로 탑재되는 장치의 성능 요구사항이 높아 가격 증가 요인이 되고 있다.

마지막으로 가상화된 오브젝트의 실행에 관한 연구는 프로그래밍 모델의 발전과 동일한 형태를 띄고 있다. 초기 임베디드 환경에서의 Native C 형태의 직접 실행하는 구조에서 부터 시작해 VM 기반의 환경으로 발전되었다. 그리고 최근 Web과의 연결을 통한 확장성이 중요한 이슈로 부각됨에 따라 외부 웹 서비스와 연결되어 웹 서비스 상에서 처리되는 형태의 연구들이 이루어지고 있다[16,17]. 이러한 웹 서비스 기반의 실행의 경우 웹 서버가 탑재된 게이트웨이 장비를 통해 장치의 정보를 RESTful 형태로 외부로 노출하는 구조를 가지며 외부 웹 서비스 실행환경에서는 해당 장치의 메시지를 수신해 처리하는 형태를 가진다[18,19]. 이러한 방식의 장점은 다양한 웹용 스크립팅 언어(예 HTML, JavaScript, PHP, Python 등)를 사용해 보다 쉽게 가상화된 오브젝트의 기능을 개발할 수 있는 장점을 제공한다. 또한 이를 웹 상에서 연결 및 브라우징(browsing)할 수 있는 장점을 가진다. 하지만 기존 연구에서는 게이트웨이를 웹 연결을 위한 중간 매개체로써의 역할만으로 사용 하였을 뿐 직접 해당 게이트웨이에서 관련된 서비스 로직을 실행하는 형태의 접근을 취하지 않았다.

본 논문에서는 제한된 컴퓨팅 리소스를 가진 무선 인터넷 공유기 환경에서 IoT 장치를 가상 오브젝트로 정의하고 이를 실행, 구동하며 매쉬업 서비스 기능까지 제공할 수 있는 새로운 시스템 구조를 제시하고자 한다.

Ⅲ. 가상화된 오브젝트의 실행을 위한 인스턴스 매니저 시스템 설계

IoT 장치를 원격, 즉 무선 인터넷 공유기에서 실행 및 제어하기 위해서는 먼저 실세계의 IoT 장치를 사이버 세계에서 활용할 수 있도록 가상 오브젝트화 해야 하고, 이를 실행시키기 위한 구동 환경이 갖춰져야 한다. 본 장에서는 장치 가상화를 위한 방법 및 인터페이스 정의를 통한 오브젝트 간의 연동 방법 그리고 이를 구동하기 위한 인스턴스 매니저의 구조 설계에 대해 설명한다.

3.1 가상 오브젝트 (Virtual Object) 정의를 위한 스키마 구조

가상 오브젝트는 IoT 장치 및 장치가 제공하는 기능을 서술하여 하나의 객체로 정의한 것을 말한다. 기본적인 사물장치의 프로파일 정보와 네트워크 인터페이스에 대한 기술을 포함하며 이를 가상 환경에서 실행하기 위한 오브젝트의 동작 로직을 담고 있다. 본 논문에서는 가상 오브젝트의 활용 용도에 따라 두 가지 가상 오브젝트, 즉, 장치연동형 가상 오브젝트와 매쉬업서비스형 가상 오브젝트로 나누어 정의한다.

장치연동형 가상 오브젝트는 IoT 장치로부터 받은 메시지를 기반으로 원격 환경에서 동일한 기능을 제공할 수 있도록 만들어진 객체를 말한다. 따라서 실제 사물 장치에 대한 기본 정보 (이름, 이미지 정보, 설명 등)와 연동을 위한 네트워크 인터페이스 정보(인터페

표 1. 장치연동형 가상 오브젝트의 스키마
Table 1. Example schema for thing associated virtual object

```
<!-- Omit the schema definition -->
<!-- Use profile as a top element -->
<xs:element name="profile">
  <!-- Virtual Object unique id -->
  <xs:element name="profileId">
  <!-- Virtual Object version -->
  <xs:element name="profileVersion">
  <!-- Virtual Object information -->
  <xs:element name="instanceDetailURL" >
  <xs:element name="instanceDescription" >
  <xs:element name="instanceIconURL" >
  <xs:element name="instanceRunningPeriod" >
  <!-- Thing associated virtual object information -->
  <xs:element name="associatedThingInformation" >
  <xs:element name="configurableValue" >
  <xs:element name="interfaceInformation">
    <xs:element name="oAuth">
    <xs:element name="feeder">
    <xs:element name="subscriber">
    <xs:element name="receiver">
    <xs:element name="controller">
  </xs:element>
  <!-- Describing the virtual object's execution logic in
JavaScript -->
  <xs:element name="instanceScript" >
</xs:element>
```

이스타입, 네트워크 주소, 포트 번호 등)를 XML로 기술한다. 또한 해당 사물 장치가 가지는 로직을 JavaScript 형태로 기술하게 된다. 아래 표 1은 장치연동형 가상 오브젝트 중 주요 엘리먼트만을 요약한 내용이다.

매쉬업서비스형 가상 오브젝트는 장치와의 직접 연결 유무와 관련 없이 독립된 서비스로써의 로직을 가진 오브젝트를 의미한다. 해당 오브젝트는 하나 이상의 장치연동형 가상 오브젝트와 서비스 로직을 결합한 매쉬업 형태의 서비스를 제공할 수 있다. 연결 가능한 가상 오브젝트를 검색하기 위해, 매쉬업서비스형 가상 오브젝트는 별도의 필터 정보를 포함한다. 아래 표 2와 같이 필터 정보는 다른가상 오브젝트의 정보를 검색하기 위해 연결 가능 한 오브젝트의 고유 아이디 값인 'profileID'와 'feederID' 그리고 버전 정보를 기술하고 있다.

표 2. 매쉬업서비스형 가상 오브젝트의 Filter 정보에 대한 스키마
Table 2. Example schema on filter information for mashup service virtual object

```
<xs:element name="subscriber">
  <xs:element name="filter">
  <xs:attribute name="profileId"  />
    <xs:attribute name="feederId"  />
    <xs:attribute name="minVersion"  />
    <xs:attribute name="maxVersion"  />
  </xs:element>
</xs:element>
```

3.2 오브젝트 인스턴스 간 메시지 전달을 위한 인터페이스 설계

가상 오브젝트는 대상 환경에서 구동될 때, 정의된 객체의 실행 프로그램 형태인 '인스턴스'로 동작하게 된다. 즉, 장치연동형 가상 오브젝트는 '장치연동형 인스턴스'로 실행되고, 매쉬업서비스형 가상 오브젝트는 '매쉬업 인스턴스'로 실행된다. 인스턴스간의 상호 연동을 용이하게 하기 위해 본 논문에서는 다섯 가지 인터페이스, 즉 'Feeder', 'Subscriber', 'Controller', 'Receiver', 'OAuth' 를 정의하였다. 그림 1은 무선 인터넷 공유기에서 가상 객체를 실행시키고 관리하기 위해 설계된 인스턴스 매니저 상에서 인스턴스간의 인터페이스 연동구조를 도식적으로 나타낸 것이다. 각각의 장치로부터 데이터를 수신하는 장치연동형 인스턴스로부터 메시지를 받아 매쉬업 인스턴스가 외부 웹 서비스와 OAuth를 통해 연동하여 매쉬업 서비스를 제공하는 관계를 나타내고 있다.

첫 번째로 'Feeder' 인터페이스는 IoT 장치에서 발생되는 데이터나 메시지를 인스턴스 또는 외부 서비스에 제공하기 위하여 정보를 노출하는 인터페이스이다. Feeder 인터페이스는 직접 메시지를 전달하는 것이 아닌 인스턴스의 Message Queue상에 Feeding할 정보를 저장하는 형태로 동작한다.

두 번째로 'Subscriber' 인터페이스의 경우 인스턴스가 외부 사물 또는 타 인스턴스의 Feeder 인터페이스로부터 정보를 수신하는 기능을 제공한다. 매쉬업서비스형 오브젝트의 경우 'filter' 엘리먼트의 조건에 맞

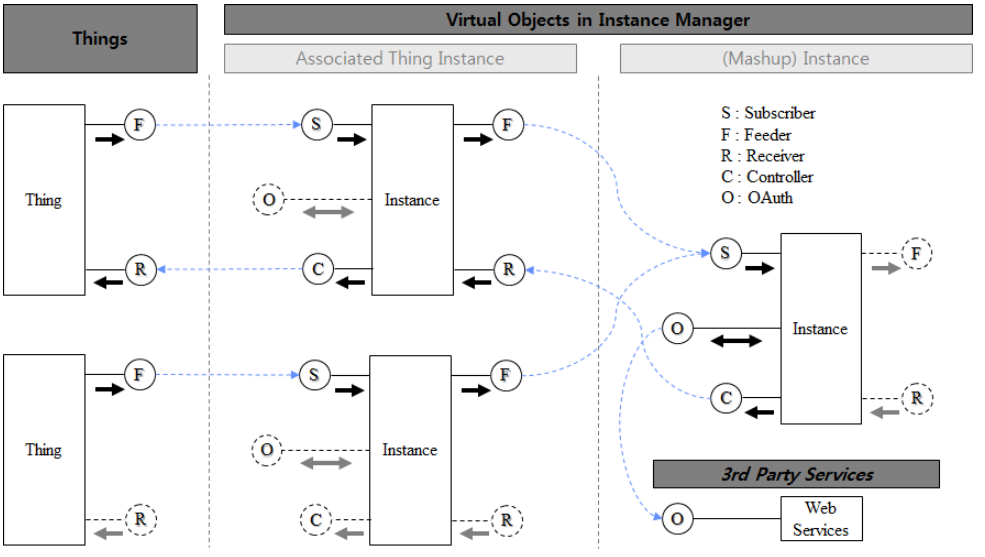

그림 1. 인스턴스 매니저 상에서의 가상 오브젝트 인스턴스 간의 인터페이스 설계
Fig. 1. Interface design for virtual object's instances on the instance manager

는 인터페이스 정보에 대해서만 구독할 수 있다.

세 번째로 'Controller' 인터페이스 경우 인스턴스가 외부로 메시지를 직접 전달하는 기능을 제공한다. 특정 IoT 장치에게 제어 명령에 관한 메시지를 보내는 경우나 타 인스턴스에게 정보가 담긴 메시지를 전달하는 경우에 사용된다.

네 번째, 'Receiver' 인터페이스의 경우 타 인스턴스의 Controller 인터페이스로부터 발신된 제어 명령 또는 정보가 담긴 메시지를 수신하는 인터페이스이다.

마지막 'OAuth' 인터페이스의 경우 외부 3rd Party 서비스와의 연동 기능을 제공한다. 따라서 인스턴스가 3rd Party 서비스로 메시지를 송수신 하는 경우 사용된다.

이와 같이 인스턴스 매니저에 명시적인 인터페이스 호출 구조를 제시하는 이유는 먼저 가상 오브젝트 개발자 측면에서 보다 단순하게 인터페이스를 설계하고, 인스턴스간 호출을 이용할 수 있도록 하기 위함이고, 인스턴스 매니저 측면에서도 인스턴스간 메시지 호출 제어를 통해 오동작을 미연에 차단하기 위함이다.

3.3 인스턴스 매니저

인스턴스 매니저는 가상 오브젝트의 로직을 개별 JavaScript 인스턴스 형태로 실행하는 기능을 제공한다. 인스턴스 매니저는 그림 4와 같은 주요 모듈로 구성되어 있으며, 가상화된 오브젝트 기반의 인스턴스 생성 및 동작과 관련된 주요 부분은 Interface Server (인트턴스 매니저와 외부 모듈간 통신 기능 제공) Message Router (인스턴스간의 입출력 메시지 라우팅 기능 제공), Instance Lifecycle Manager (인스턴스 생애주기 관리 및 스케쥴링 기능 제공), Instance Pool (다수의 인스턴스의 컨텍스트 풀링 기능 제공), V8 (인스턴스 실행 기능 제공)으로[20] 구성되어 있다.

그 이외의 모듈인 IAS(Instance Application Server)는 인스턴스 매니저를 웹 기반 형태로 제어하기 위한 웹 서비스 기능을 제공하며, 가상 오브젝트 파일을 설치하는 deploy 기능을 함께 제공한다. Network Adaptor의 경우 HTTP, Zigbee 인터페이스를 통해 수신되는 장치와의 메시지를 처리하는 기능을 담당하며 External Web Services들의 경우 3rd Party 웹 서비스와의 연계를 의미한다.

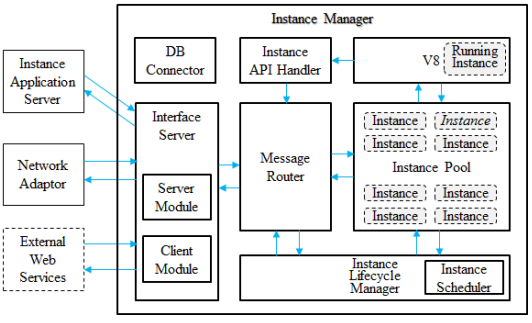

그림 2. 인스턴스 매니저 구조
Fig. 2. Functional architecture of the Instance manager

인스턴스 매니저는 다수의 가상화된 오브젝트를 임베디드 시스템 환경에서 효과적으로 실행하기 위해 인스턴스의 동시 처리량을 조절하고, 개별 인스턴스의 실행에 대해 빠른 응답을 제공할 수 있는 구조를 가져야 한다. 이를 위하여, 최소 인스턴스의 실행 후 해당 인스턴스에 대한 컨텍스트(context) 정보를 인스턴스 풀을 이용해 풀링(Pooling) 함으로써, 이후 인스턴스 호출(invoke)시 컨텍스트의 생성 시간을 최소화함으로써 가능하게 된다. 이를 통해 최초로 인스턴스를 호출 하는 경우 외에는 추가적인 I/O 사용을 피하게 됨으로 성능 향상을 꾀하게 된다.

이러한 구조는 특히 사물 인터넷 환경에서 장치들과의 메시지 수신 및 처리가 빈번하게 일어나는 형태의 서비스 패턴에 적합하다. 즉 주기적으로 잦은 호출이 이루어지는 인스턴스들에 대해 풀링된 컨텍스트 정보를 이용, 직접 JavaScript 엔진에 바인딩 시킴으로써 해당 컨텍스트의 생성 시간을 최소화 할 수 있다.

Ⅳ. 인스턴스 매니저 구현 및 성능 평가

제안된 인스턴스 매니저의 구현은 홈 환경에서 일반적으로 사용하는 무선 인터넷 공유기인 ASUS RT-N16 모델을 이용하였으며 장치의 하드웨어 구성은 아래 표 3과 같다. 일반적인 무선 인터넷 공유기가 Zigbee 인터페이스를 지원하지 않으므로 Zigbee와의 연동을 위해 별도의 내부 시리얼 포트상에 TI사의 Zigbee receiver를 연동하여 Zigbee 통신 기능을 탑재하였다.

인스턴스 매니저가 동작하는 소프트웨어 사양은 다음과 같다. 먼저 무선 인터넷 공유기상의 임베디드 OS 기능을 제공하기 위해 DebWRT 기반의 GNU/Linux를 사용한다. 인스턴스 매니저는 C++ 기반으로 작성되었다. 웹 서비스 형태로 가상 오브젝트의 설치 및 관리를 위한 기능을 제공하는 IAS는 오픈 소스 기반의 Apache-Rewrite Module 및 PHP5, MySQL, Python을 사용해 구현하였다. 실제 가상화된 오브젝트의 실행을 담당하는 V8은 3.19.15버전을 사용하였다.

표 3. 인스턴스 매니저가 구현된 무선 인터넷 공유기의 하드웨어 구성
Table 3. Hardware specifications of the implemented wireless AP

components	specification
Architecture	MIPSEL
Bootloader	CFE
System-on-Chip	Broadcom BCM4718 (MIPS 74KTM)
CPU	Broadcom BCM4716 chip rev1 (MIPS 74k V4.0 480MHz)
Flash-Size	32MB
RAM	128MB
Network Standard	IEEE 802.11b, IEEE 802.11g, IEEE 802.11n
Operating Frequency	2.4GHz
Ports	1 x RJ45 for 10/100/1000 BaseT for WAN 4 x RJ45 for 10/100/1000 BaseT for LAN Support Ethernet and 802.3 with max. bit rate 10/100/1000 Mbps and auto cross-over function(MDI-X) USB 2.0 x 2 Serial JTAG
*Zigbee	TI Zigbee Receiver (Connect with Serial Port)

4.1 가상 오브젝트의 디플로이(Deploy)

가상 오브젝트는 장치의 정보 및 실행 코드만을 가지고 있으므로 실제 서비스에서 구동되기 위해 인스턴스 매니저 상에 설치되는 디플로이(deploy) 과정을 거친다. 이 과정을 통해 인스턴스 매니저는 가상 오브젝트와 연결된 장치 정보 및 서비스 로직을 로드하고 서비스를 제공할 준비를 하게 된다. 매쉬업서비스형 오브젝트의 경우 디플로이 과정에서 해당 오브젝트가 연결하고자 하는 Subscriber 및 Controller 인터페이스를 가진 오브젝트의 정보를 검색한 뒤 사용자의 선택에 따른 정보로 연결하게 된다. 아래 표 4는 매쉬업 인스턴스의 필터 정보 설정에 대한 예제이다.

표 4. 매쉬업 오브젝트에서 조도 센서를 검색하기 위한 필터 예제
Table 4. Filter example for illuminance sensor search on mashup instance

```
<subscriber id="100" name="Illuminance">
  <filter profileId="IlluminanceSensor" feederId="2"
minVersion="1.1" maxVersion="1.1" />
</subscriber>
```

4.2 인스턴스 실행 및 풀링

오브젝트의 실행 로직은 JavaScript 형태로 제공되며 이를 JavaScript Engine인 V8을 통해 처리하게 된다. 오브젝트의 실행 로직은 최소 실행 시 디스크 상에서 읽어 들이게 되며 이후의 사용은 Instance Pool 상에서 호출되어 진다. V8의 경우 Isolate를 독립적으로 이용해 JavaScript 실행 환경을 완전히 독립시키는 방법과 Isolate를 공유하되, 컨텍스트를 독립적으로 적용해 개별 memory를 분리하는 방식을 제공한다. 인스턴스 매니저 구현에 있어 대상 하드웨어 환경이 멀티코어가 아닌 싱글 코어 하나의 Isolate 당 5MB 수준의 메모리를 요구하는 점을 감안해 하나의 Isolate 상에서 개별 인스턴스의 컨텍스트를 풀링하는 방법을 이용해 구현하였다. 해당 컨텍스트에는 V8에서 사용하는 모든 변수와 정보가 저장되며, 다음 호출 시 읽어 들여 동일한 환경을 이용할 수 있게 한다. 인스턴스 매니저는 각각의 Instance 별로 필요한 State를 컨텍스트에 업데이트 해 줌으로써 Instance Logic (JavaScript)이 실행될 때 현재의 Instance 상황을 알고 그에 따라 동작할 수 있다. 이를 통해 현재 수신한 메시지 값과 이전 값과의 비교 연산 등을 수행하는 경우와 같이 IoT 환경에서 빈번하게 발생하는 상황들에 대해 디스크 I/O 접근 없이 빠르게 동작할 수 있다.

4.3 성능

구현 모델의 실험 성능 평가를 위해 먼저 인스턴스 매니저의 벤치마크를 통해 동시에 처리 가능한 인스턴스의 수 및 CPU, Memory의 사용량을 확인한다. 그리고 실제 환경에서의 성능 평가를 위해 무선 인터넷 공유기상의 오리지널 펌웨어와 실험 환경인 DebWRT 상에서의 TCP Performance를 측정하여 그 차이를 확인한다. 또한 인스턴스 매니저가 동작중인 환경에서의 무선 인터넷 공유기의 TCP 처리 성능을 평가함으로써 실제 홈 환경에서 다수의 인스턴스를 실행하는 인스턴스 매니저의 구동이 무선 인터넷 공유기의 성능을 저하시키지 않고 실행 가능한지를 확인하였다.

우선 인스턴스 매니저의 메모리 사용량과 관련해 호스팅하는 오브젝트에 대한 Instance(장치연동형 가상오브젝트로 Ti사의 Zigbee 기반 온도계를 사용)의 개수를 1~40개까지 늘려가면서 Memory 및 CPU 사용량을 측정한 결과는 그림 3과 같다. 왼쪽 축은 Resident Set Size를 의미하며, 오른쪽 축은 Virtual Size를 나타낸 것이다. 그림 5는 테스트 인스턴스의 개수를 40개에서 30분 동안 지속적으로 인스턴스 호

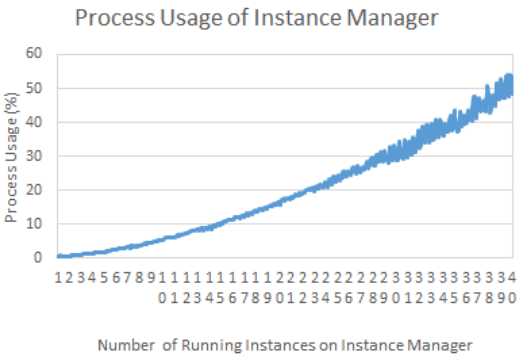

그림 3. 인스턴스 증가에 따른 메모리 사용량
Fig. 3. Memory consumption by the deployed number of instance increases

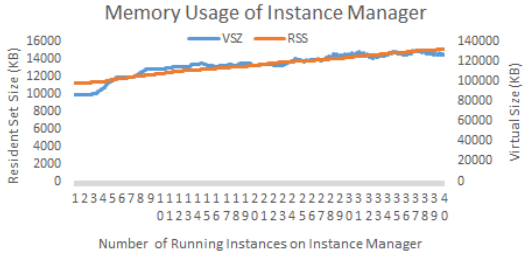

그림 4. 인스턴스 증가에 따른 CPU 사용량
Fig. 4. CPU consumption by the deployed number of instance increases

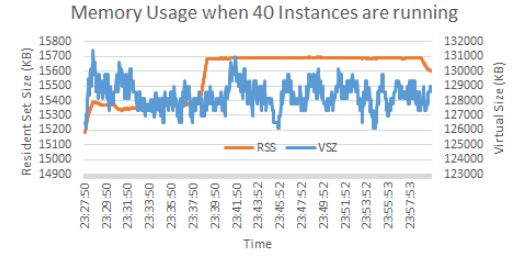

그림 5. 초당 4회의 인스턴스 호출시 메모리 사용량 변화
Fig. 5. Memory consumption as the deployed 40 instances are invoked every 10 seconds

출시(각 인스턴스는 10초 간격의 호출 주기를 가짐)에도 메모리 사용량의 큰 변화 없이 안정적으로 동작함을 확인할 수 있다. 40개 인스턴스 활용 수준에서 평균 15520KB를 사용해 일반적인 무선 인터넷 공유기 상의 메모리 제공수준인 128MB의 메모리 환경에서 충분히 동작 가능함을 확인할 수 있다.

두 번째 실험은 위와 동일 실험에서 CPU 사용량증가에 관한 부분이다. 그림 4에서 보는 바와 같이 실험용 인스턴스 1~40개를 deploy 후 동작시 안정적인 CPU 사용량을 나태남을 확인할 수 있으며, 전체적인 CPU 사용량은 약 51.98% 수준을 나타냄을 확인할 수 있다.

실제 무선 인터넷 공유기 상에서 서비스를 할 경우 무선 인터넷 공유기의 네트워크 처리에 따른 CPU 및 Memory 사용에 따라 인스턴스 매니저가 사용할 수 있는 자원에 제약이 발생하게 된다. 이에 대한 실험으로 오리지널 펌웨어와 DebWRT 기반의 환경에서 iperf를 이용해 LAN, WLAN, WAN 간의 TCP 성능을 비교하고, IDLE 상태의 DebWRT 기반 네트워크 처리 성능과 인스턴스 매니저의 동작 상태에서의 처리 성능 비교한 결과가 표 5이다.

우선 오리지널 펌웨어와 DebWRT를 기반으로 성능 비교 시 오리지널 펌웨어의 성능이 더 우수한 것으로 나타나며, 특히 LAN과 WAN 간의 통신 시에 큰 차이가 발생하는 부분은 오리지널 펌웨어 상에서 LAN과 WAN간의 라우팅을 하드웨어에서 담당하여 처리하기 때문인 것으로 해석 된다.

인스턴스 매니저를 설치하지 않은 DebWRT에서의 네트워크 처리 성능과 40개의 인스턴스를 호스팅 중인 성능 결과의 경우, IDLE 상태와 인스턴스 매니저가 동작중인 상태에서의 네트워크 성능 저하가 크게 발생하지 않은 점을 확인할 수 있다. 이때 사용한 Instance는 장치연동형 가상 오브젝트와 매쉬업서비스형 가상 오브젝트를 복합적으로 이용하였다. 장치 연동형의 경우 Ti사의 Zigbee기반 조도 및 온도 센서,

표 5. 무선 인터넷 공유기 상에서 오리지널 펌웨어, DebWRT, 인스턴스 매니저 동작시 네트워크 처리 성능 비교
Table 5. Network performance comparison while the original firmware, DebWRT, instance manager are running on the wireless AP

	LAN			WLAN			WAN		
	Original	DebWRT	DebWRT + IM	Original	DebWRT	DebWRT + IM	Original	DebWRT	DebWRT + IM
LAN	411Mb/s	425Mb/s	425Mb/s	51Mb/s	45Mb/s	28.5Mb/s	560Mb/s	80Mb/s	79Mb/s
WLAN	45Mb/s	58Mb/s	54.8Mb/s	12Mb/s	22Mb/s	22Mb/s	45.6Mb/s	30Mb/s	29Mb/s
WAN	520Mb/s	66Mb/s	64Mb/s	42Mb/s	13Mb/s	12Mb/s	-		

Phillips Hue, WeMo Switch 가젯 및 Humidifier, Radiator를 이용하였으며 매쉬업서비스형은 Smart Humidifier(온도 및 조도에 따른 최적의 습도 제공 서비스), Smart Radiator(일정온도 유지 서비스)로 구성하였다. 이러한 구성은 일반 가정에서 스마트 홈 환경을 갖추고자 할 경우 범용적으로 이용되는 장치들로, 본 논문에서 제안한 인스턴스 매니저가 일반 가정의 무선 인터넷 공유기 환경에서 충분히 동작 가능함을 알 수 있다.

V. 결 론

IoT 장치들, 특히 개인과 가정용 제품들의 등장으로 보다 편리하고 효율적인 생활이 가능해졌다. 그러나 개인이 보유한, 즉 관리해야할 장치 개수의 증가는 필연 또 다른 불편함을 가져온다. 본 논문에서는 모든 IoT 장치가 인터넷에 연결되는 거점이 되는 무선 인터넷 공유기를 이용해서 추가적인 장비의 구매나 설치 없이 통합 관리, 제어, 및 연동형 스마트 서비스를 가능하게 하는 방법을 제시하였다. 또한 직접 구현을 통한 실험을 통하여 제안된 방식을 이용하면 기존의 무선 인터넷 공유기상에서 무선 라우팅 기능을 방해하지 않고 댁내 다수개의 IoT 장치들을 연결하여 안정적인 서비스 제공이 가능함을 보였다. 본 연구팀은 향후 더 다양한 IoT 장치들의 동작 패턴을 분석하여 반영함으로써 진일보된 시스템으로 발전시킬 계획이다.

References

[1] “Appcessory economics: Enabling loosely coupled hardware / software innovation,” Cornell University Library, Oct. 28. 2013, from http://arxiv.org/abs/1209.5901

[2] Andreas Nettsträter, “Deliverable D1.3-updated reference model for IoT v1.5,” *IoT-A*, Jul. 2012.

[3] Rob Chandhok, A fast track to the Internet of Everything, Oct. 18, 2013 from https://www.alljoyn.org/sites/default/files/alljoyn-alliance.pdf

[4] Xively, “Public cloud for internet of things,” https://xively.com/

[5] IFTTT, “IF this than that,” http://www.ifttt.com

[6] T. Kindberg, J. Barton, J. Morgan, G. Becker, D. Caswell, P. Debaty, G. Gopal, M. Frid, V. Krishnan, H. Morris, J. Schettino, B. Serra, and M. Spasojevic, “People, places, things: Web presence for the real world,” *Mob. Netw. Appl.*, vol. 7, no. 5, pp. 365-376, 2002.

[7] G. Borriello and R. Want, “Embedded computation meets the World Wide Web,” *Commun. ACM*, vol. 43, no. 5, pp. 59-66, 2000.

[8] Pachube is currently renew of site ‘Xively’, Oct. 28, 2013, from https://xively.com/

[9] W. I. Grosky, A. Kansal, S. Nath, J. Liu, and F. Zhao, “SenseWeb: An infrastructure for shared sensing,” *IEEE Multimedia*, vol. 14, no. 4, pp. 8-13, Oct.-Dec. 2007.

[10] Devices Profile for Web Services Version 1.1, OASIS, Oct, 28, 2013, from http://docs.oasis-open.org/ws-dd/dpws/1.1/os/wsdd-dpws-1.1-spec-os.html

[11] S. Duquennoy, G. Grimaud, and J.-J. Vandewalle, “The web of things: Interconnecting devices with high usability and performance,” in *Proc. ICESS*, pp. 323-330, HangZhou, Zhejiang, China, May 2009.

[12] D. Guinard, V. Trifa, and E. Wilde. “A Resource Oriented Architecture for the Web of Things,” in *Proc. IoT*, pp. 1-8, Tokyo, Japan, Nov. 2010.

[13] J. Park and N. Kang, “Entity authentication scheme for secure WEB of Things applications,” *J. KICS*, vol. 38B, no. 5, pp. 394-400, May 2013.

[14] P. Levis and D. Culler, “Mate: A tiny virtual machine for sensor networks,” in *Proc. Int’l Conf. Architectural Support for Programming Languages and Operating Systems*, ASPLOS-X, Oct. 2002.

[15] N. Brouwers, K. Langendoen, and P. Corke. “Darjeeling, a feature-rich VM for the resource poor,” in *Proc. SenSys*, pp. 169-182, Berkeley, California, Nov. 2009.

[16] Y. Kim, Y. Jeon, and I. Chong, “Device objectification and orchestration mechanism for IoT intelligent service,” *J. KICS*, vol. 38C, no. 1, pp. 19-32, Jan. 2013.

[17] M. Kovatsch, M. Lanter, and S. Duquennoy, “Actinium: A RESTful runtime container for scriptable Internet of Things applications,” *3rd Int’l Conf. IoT*, pp. 135-142, Oct. 2012.

[18] D. Guinard, V. Trifa, and E. Wilde, "A resource oriented architecture for the web of things," *IoT 2010*, pp. 1-8, Nov. 2010.

[19] V. Trifa and D. Guinard, "Towards the web of things: Web mashups for embedded devices," *Second Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web(MEM 2009)*, Madrid, Apr. 2009.

[20] V8, "Google' open source JavaScript engine," Oct, 30, 2013 from https://code.google.com/p /v8/

양 진 홍 (Jinhong Yang)

2003년 2월 : 인제대학교 컴퓨터공학과 졸업
2005년 2월 : 인제대학교 컴퓨터공학과 석사
2008년 9월~현재 : 한국과학기술원 정보통신공학과 박사과정
<관심분야> 사물인터넷, 웹 오브 오브젝트

박 효 진 (Hyojin Park)

2006년 2월 : 인제대학교 정보컴퓨터공학부 졸업
2007년 8월 : 한국정보통신대학교 정보통신공학과 석사
2007년 8월~현재 : 한국과학기술원 정보통신공학과 박사과정
<관심분야> IPTV, 미래 미디어, 사물 인터넷

김 용 록 (Yongrok Kim)

2009년 2월 : 한국정보통신대학교 컴퓨터공학과 졸업
2011년 8월 : 서울대학교 컴퓨터공학과 석사
2011년 9월~현재 : 한국과학기술원 IT융합연구소 재직
<관심분야> 전자공학, 통신공학, 광통신 공학

최 준 균 (Jun Kyun Choi)

1988년 2월 : 한국과학기술원 전자 공학 박사
1997년 2월 : 한국전자통신 연구원 책임연구원
2009년 3월~현재 : 한국과학기술원 전기 및 전자공학과 정교수
<관심분야> 에너지 절감 통신, 웹 오브 오브젝트 기술