ISSN 2383-630X(Print) / ISSN 2383-6296(Online)
Journal of KIISE, Vol. 43, No. 6, pp. 710-717, 2016. 6
http://dx.doi.org/10.5626/JOK.2016.43.6.710

Smart Fog: 다중 서비스 사물 인터넷 시스템을 위한 포그 서버 중심 사물 추상화 프레임워크

(Smart Fog: Advanced Fog Server-centric Things Abstraction Framework for Multi-service IoT System)

홍 경 환 [†] 박 은 수 [††] 최 시 훈 [†††] 신 동 군 [††††]
(Gyeonghwan Hong) (Eunsoo Park) (Sihoon Choi) (Dongkun Shin)

요 약 최근 여러 사물 인터넷 서비스가 사물 장치를 공유하는 다중 서비스 시스템을 구현하기 위해, 다양한 구조의 사물 추상화 프레임워크들이 제시되었다. 분산형 구조는 사물 인터넷 서비스 중복 문제가 있으며, 클라우드 서버 중심 구조는 실시간 인터랙션을 할 수 없다. 또한, 기존의 포그 서버 중심 구조에서는 불완전한 인터페이스가 사용되었다. 본 논문에서는 기존 구조의 문제를 해결한 사물 추상화 프레임워크인 Smart Fog를 제안하였다. Smart Fog는 스마트 게이트웨이와 3개의 IoT 인터페이스들로 구성된다. Smart Fog는 IoTivity와 OIC 표준을 기반으로 구현되었고, 이를 이용하여 실제 임베디드 장치인 Odroid-XU3에서 프로토타입을 구현하였다. 프로토타입 상에서 실험한 결과, Smart Fog가 실시간 인터랙션이 가능할 정도로 네트워크 지연 시간이 짧고, 분산형 구조에 비해 모바일 장치에서 발생하는 네트워크 트래픽이 74%, 전력 소모가 21% 감소함을 확인하였다.

키워드: 사물 인터넷, 사물 추상화, 게이트웨이, 사물 인터넷 서비스, 다중 서비스 사물 인터넷 시스템, 사물 인터넷 서비스 중복 문제

Abstract Recently, several research studies on things abstraction framework have been proposed in order to implement the multi-service Internet of Things (IoT) system, where various IoT services share the thing devices. Distributed things abstraction has an IoT service duplication problem, which aggravates power consumption of mobile devices and network traffic. On the other hand, cloud server-centric things abstraction cannot cover real-time interactions due to long network delay. Fog server-centric things abstraction has limits in insufficient IoT interfaces. In this paper, we propose Smart Fog which is a fog server-centric things abstraction framework to resolve the problems of the existing things abstraction frameworks. Smart Fog consists of software modules to operate the Smart Gateway and three interfaces. Smart Fog is implemented based on IoTivity framework and OIC

· 본 연구는 미래창조과학부 및 정보통신기술연구진흥센터의 정보통신·방송 연구개발사업의 일환으로 수행하였음. [10041244, 스마트TV 2.0 소프트웨어 플랫폼]
· 이 논문은 제42회 동계학술발표회에서 'Smart Fog: 실시간 인터랙션 및 에너지 효율을 위한 스마트 게이트웨이 중심 사물 추상화 프레임워크'의 제목으로 발표된 논문을 확장한 것임

† 비 회 원 : 성균관대학교 전자전기컴퓨터공학과
redc7328@skku.edu
†† 비 회 원 : 성균관대학교 IT융합학과
pes9488@skku.edu
††† 비 회 원 : 삼성전자 무선사업부
sihoon.choi@samsung.com
†††† 종신회원 : 성균관대학교 컴퓨터공학과 교수
(Sungkyunkwan Univ.)
dongkun@skku.edu
(Corresponding author임)

논문접수 : 2016년 2월 12일
(Received 12 February 2016)
논문수정 : 2016년 4월 8일
(Revised 8 April 2016)
심사완료 : 2016년 4월 13일
(Accepted 13 April 2016)

정보과학회논문지 제43권 제6호(2016. 6)

standard. We construct a smart home prototype on an embedded board Odroid-XU3 using Smart Fog. We evaluate the network performance and energy efficiency of Smart Fog. The experimental results indicate that the Smart Fog shows short network latency, which can perform real-time interaction. The results also show that the proposed framework has reduction in the network traffic of 74% and power consumption of 21% in mobile device, compared to distributed things abstraction.

Keywords: Internet of things, things abstraction, gateway, IoT service, multi-service IoT system, IoT service duplication problem

1. 서 론

지난 몇년 간 사물 인터넷(IoT; internet of things) 시스템은 센서(sensor)와 액츄에이터(actuator)가 부착된 사물 장치(thing device)들이 상호 운용하는 시스템으로 부상하였다. 최근에는 사물 인터넷 시스템에 몇 가지 큰 변화가 발생하였다.

첫째, 사물 장치를 여러 용도로 활용하기 위해, 사물 인터넷 서비스가 등장하였다. 사물 인터넷 서비스는 사물 장치가 제공하는 센서 데이터와 액션을 새로운 형태의 센서 데이터와 액션으로 가공하는 소프트웨어다. 기존의 사물 인터넷 시스템은 센서 데이터와 액션을 다른 사물 장치나 단말 장치에 전달하는 기능만 제공했으나, 최근에는 사물 인터넷 서비스를 통해 센서 데이터에 대한 필터링과 합성 기능, 액션 그룹화 기능도 제공하여, 사물 장치의 활용도를 높였다. 예를 들어, 스마트 홈은 동작 센서와 음성 인식 센서, 커피 포트, 오디오 재생기 같은 사물 장치들로 구성되며, 이들은 각각 센서 데이터와 액션을 제공한다. 스마트 홈의 사물 인터넷 서비스는 동작 센서와 음성 인식 센서가 제공하는 센서 데이터를 합성하고 필터링하여 거주자가 깨어났는지를 확인하는 수면 센서를 제공한다. 또한, 커피 포트와 오디오 재생기가 일제히 동작하도록, 아침 모드 같이 그룹화된 액션도 제공한다.

둘째, 사물 장치의 구성이 복잡해지고 있다. 기존의 사물 인터넷 시스템에서는 사물 장치를 구성하는 센서와 액츄에이터의 종류가 다양하지 않고, 수도 적다. 최근에는 사물 장치의 활용도를 높이기 위해, 다양한 센서와 액츄에이터로 구성된 사물 장치들이 출시되고 있다. 예를 들어, 기존의 스마트 빌딩 시스템의 스마트 전기 플러그는 단순히 전력량계 센서로만 구성되어 있는 반면[1], 최근에는 온습도 센서, 동작 센서, 문 개폐 센서 등을 탑재한 다목적 센서가 출시되고 있다.

셋째, 동시에 여러 사물 인터넷 서비스가 사물 장치를 사용하는, 다중 서비스 사물 인터넷 시스템(multi-service IoT system)이 되고 있다. 기존 시스템은 특정 사물 인터넷 서비스를 위한 전용 인터페이스를 통해 사물 장치를 사용하여, 다른 사물 인터넷 서비스가 사물 장치를 공유하여 사용할 수 없는 단일 서비스 사물 인터넷 시스템이었다. 그러나, 최근 여러 IoT 프레임워크에서 사물 장치에 대한 오픈 API를 제공함에 따라[2,3], 여러 사물 인터넷 서비스가 사물 장치를 공유하여 사용할 수 있게 되었다.

사물 인터넷 시스템에서 발생한 여러 변화로 인해, 사물 추상화 프레임워크(things abstraction framework)가 대두되었다. 사물 추상화 프레임워크는 사물 장치의 구성에 관계 없이, 여러 사물 인터넷 서비스가 공유해서 사용하는 소프트웨어다. 기존에는 사물 인터넷 서비스가 동작하는 장치에 따라, 여러 구조의 사물 추상화 프레임워크가 제시되었다. 그 중에서 포그 서버 중심 구조(fog server-centric architecture)[4-7]는 분산형 구조[2,3]에서 발생하는 사물 인터넷 서비스 중복 문제를 해결하여 모바일 단말 장치의 전력 소모와 네트워크 트래픽을 줄였으며, 클라우드 서버 중심 구조[8-10]에 비해서는 네트워크 지연 시간이 짧아서 사용자 반응성을 요구하는 시스템에 적합하다. 그러나, 기존의 포그 서버 중심 사물 추상화 프레임워크[4-7]에서는 물리적 사물 인터페이스가 정의되지 않아서, 사물 장치와 사물 인터넷 서비스 간의 의존성을 해결하지 못하였다는 문제점이 있다.

따라서, 본 논문에서는 포그 서버 중심 사물 추상화 프레임워크인 Smart Fog를 제시하였다. Smart Fog는 각 장치 사이에서 데이터 전달 기능을 수행할 뿐만 아니라, 여러 사물 인터넷 서비스가 사물 장치를 공유해서 사용하도록 하는 프레임워크다. 본 논문에서는 오픈 소스 IoT 프레임워크인 IoTivity[3]와 OIC 표준[11]을 기반으로 Smart Fog를 구현하였다. Smart Fog을 기반으로 스마트 홈 프로토타입 시스템을 만들어서 실험한 결과, 실시간 인터랙션이 가능할 정도로 네트워크 지연 시간이 짧고, 분산형 구조에 비해 모바일 단말 장치에서의 전력 소모가 21%, 네트워크 트래픽은 74% 줄어듦을 보였다.

본 논문의 2장에서는 기존 사물 추상화 구조의 문제점을 지적하고, 3장에서는 이를 개선하기 위한 포그 서버 중심 사물 추상화 구조인 Smart Fog를 제안하고 설계하였다. 4장에서는 Smart Fog의 구현에 대해 서술하였다. 5장에서는 실제 프로토타입 시스템에서 실험을 통해 Smart Fog의 에너지 효율과 네트워크 성능을 검증하고, 6장에서는 결론을 제시한다.

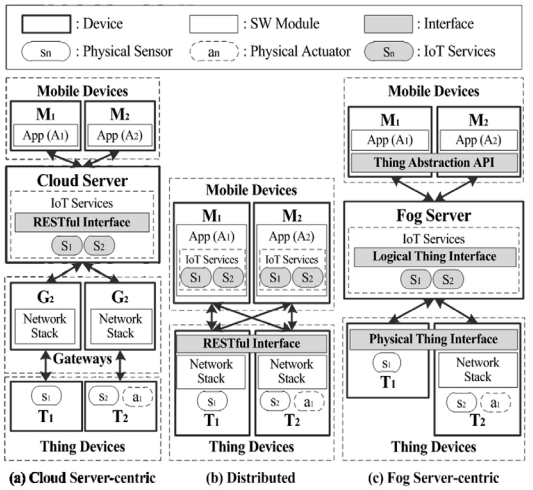

그림 1 사물 추상화 구조 비교

Fig. 1 Comparison of things abstraction architectures

2. 배 경

기존의 사물 추상화 프레임워크들은 사물 인터넷 서비스가 동작하는 장치에 따라서 여러 구조로 구현되었다. 그림 1과 같이, 기존에는 분산형 구조[2,3]와 클라우드 서버 중심 구조[8-10], 포그 서버 중심 구조[4-7]가 제시된 바가 있다.

2.1 분산형 사물 추상화 구조

최근에는 사물 장치의 처리 능력이 증가하여, IPv6 네트워크 스택을 탑재하고 인터넷에 직접 연결할 수 있게 되었다. 또한, 처리 능력이 뛰어난 이동형 장치인 스마트폰이 대중화되면서, 그림 1의 (a)와 같이, 스마트폰 응용 프로그램의 형태로 사물 인터넷 서비스를 제공하는 분산형 구조가 제시되었다. 분산형 구조에서는 사물 장치가 RESTful 인터페이스를 통해 모바일 장치에 직접 센서 데이터와 액션을 제공한다.

분산형 구조 중 하나인 AllJoyn은 퀄컴이 주도하는 분산형 IoT 프레임워크이며, 네트워크 스택을 추상화하는 코어 API와 음성 스트리밍 등 상위 단계 서비스도 제공한다[2]. IoTivity는 리눅스 파운데이션이 관리하는 분산형 IoT 프레임워크로, OIC 표준을 바탕으로 IoT 장치 간의 통신을 추상화한 베이스 API와 소프트 센서 등 상위 단계 서비스도 제공한다[3].

그러나, 이 구조에서는 여러 모바일 장치 상에서 각자 사물 인터넷 서비스가 동작하는 사물 인터넷 서비스 중복 문제가 발생한다. 사물 인터넷 서비스는 센서 데이터를 수집과 센서 데이터 관리, 센서 데이터 결합, 액츄에이터 제어를 수행하는데, 분산형 구조에서는 이러한 동작이 각 모바일 장치에서 중복으로 수행된다. 따라서, 각 모바일 장치의 연산 처리량이 늘어나고, 전력 소모가 증가한다. 또한, 사물 장치와 모바일 장치가 P2P(peer to peer)로 연결되어야 하기 때문에, 네트워크 트래픽도 늘어나서 네트워크 확장성(scalability)이 떨어지게 된다.

2.2 클라우드 서버 중심 사물 추상화 구조

클라우드 서버 중심 구조에서는 그림 1의 (b)와 같이, 클라우드 서버가 게이트웨이에 수집된 센서 데이터를 가져오고, 이를 가공하는 사물 인터넷 서비스를 제공한다. 모바일 응용 프로그램은 웹 서비스를 접근하는 것과 같이, RESTful 인터페이스를 통해 사물 인터넷 서비스를 사용할 수 있다[12]. 클라우드 서버 중심 구조에서는 사물 인터넷 서비스가 클라우드 서버 상에서만 동작하기 때문에, 사물 인터넷 서비스 중복 문제가 발생하지 않는다.

기존의 클라우드 서버 중심 구조 관련 연구로는 센서-클라우드 구조가 있으며, 이 구조에서는 센서 데이터를 클라우드 서버로 전송하는 방법을 최적화하는 연구가 있고[8], 네트워크를 CEB (Cloud-Edge-Beneath)의 3단계 구조로 구성하여 센서 데이터 취득을 최적화하는 연구도 있다[9]. 또한, JavaScript로 프로그래밍 가능한 사물 인터넷 서비스가 서버에서 동작하도록 하는 프레임워크인 Actinium도 제안된 바가 있다[10].

그러나, 클라우드 서버 중심 구조에서는 실시간 인터랙션 (real-time interaction)이 불가능하다는 문제가 있다. 실시간 인터랙션은 실시간으로 사물 장치와 통신하여 센서 데이터를 제공받거나 액션을 수행하는 것을 의미한다. 예를 들어, 사물 장치들로부터 취득한 센서 데이터를 바탕으로, 현재 거주자의 행동을 분석하여 실시간으로 가전 기기를 조작하는 스마트 홈 서비스에서 실시간 인터랙션이 일어난다[13]. 클라우드 서버 중심 구조에서는 사물 장치나 단말 장치가 클라우드 서버를 접근할 때 소요되는 네트워크 지연 시간이 길기 때문에, 사물 장치와의 실시간 인터랙션이 필요한 응용 프로그램을 구현하기 어렵다.

2.3 포그 서버 중심 사물 추상화 구조

클라우드 서버 중심 구조에서 실시간 인터랙션을 할 수 없던 문제를 해결하기 위해, 로컬 네트워크에 위치한 포그 서버(fog server)로 작업을 오프로딩하는 포그 컴퓨팅(fog computing) 모델이 제안된 바가 있다[14]. 이러한 포그 컴퓨팅을 사물 추상화에 적용하여, 그림 1의 (c)와 같이 포그 서버에서 사물 인터넷 서비스가 동작하는 포그 서버 중심 구조의 사물 추상화 프레임워크들이 제시되었다. 포그 서버 중심 구조에서는 사물 인터넷 서비스가 포그 서버에서만 작동하기 때문에, 분산형 구조에서 발생하는 사물 인터넷 서비스 중복 문제도 발생하지 않는다. 또한, 포그 서버 중심 구조는 신체 네트워크 (body area network)나 설치 센서 네트워크(emplaced

sensor network) 같이, 상황 인식(context-awareness)를 고려한 사물 인터넷 서비스에도 적용될 수 있다[15].

다중 서비스 사물 인터넷 시스템을 구현하려면, 각 장치 상에서 동작하는 소프트웨어 간의 의존성을 줄이기 위해 인터페이스를 정의해야 한다. 포그 서버 중심 사물 추상화 구조는 모바일 단말 장치, 포그 서버, 사물 장치로 구성된다. 모바일 단말 장치에는 모바일 응용 프로그램이, 포그 서버에는 사물 인터넷 서비스가, 사물 장치에는 펌웨어가 동작하고, 이들은 장치 간 통신을 담당한다. 그러나, 다중 서비스 사물 인터넷 시스템에서는 다수의 사물 인터넷 서비스가 동작하며, 서비스들을 다수의 모바일 단말 장치로 사용하기 때문에, 각 장치 간 의존성이 최소화되어야 시스템 구현 비용이 줄어들게 된다.

포그 서버 중심 사물 추상화 구조에서는 상호 독립적으로 모바일 응용 프로그램을 구현하기 위해 사물 추상화 API(things abstraction API)가, 사물 인터넷 서비스를 구현하기 위해 논리적 사물 인터페이스(logical thing interface)가, 사물 장치 펌웨어를 구현하기 위해 물리적 사물 인터페이스(physical thing interface)가 정의되어야 한다.

기존의 포그 서버 중심 사물 추상화 프레임워크에서도 IoT 인터페이스를 정의하여 각 장치간의 의존성을 해결하려는 시도가 있었다. 포그 서버에서 작동하는 서비스에 대한 프로그래밍 API인 Mobile Fog[4]와, 다중 사물 인터넷 서비스가 동작하는 무선 게이트웨이 프레임워크인 ParaDrop[5]이 제안된 바 있다. 또한, 사물 인터넷 서비스들이 포그 서버에서 할당받는 자원을 정책 기반으로 관리하는 포그 서비스 편성 기능을 제공하는 플랫폼도 제시되었다[6]. 이 프레임워크들은 공통적으로 논리적 사물 인터페이스와 사용자 측 사물 추상화 인터페이스를 정의하여, 모바일 단말 장치와 사물 인터넷 서비스 간의 의존성을 최소화하였다. 그러나, 물리적 사물 인터페이스는 정의하지 않았기 때문에, 사물 인터넷 서비스와 사물 장치 간의 의존성이 강하다는 문제가 있다.

모든 IoT 인터페이스를 구현한 프레임워크의 경우에는, 논리적 사물 인터페이스가 협소하게 정의되어, 일부 기능만 구현할 수 있다는 문제가 있다. Cloud of Things에서는 센서 데이터를 클라우드 서버로 전달하기 전에 센서 데이터 필터링 작업을 수행하는 IoT 인터페이스가 제안되기도 하였으나[7], 이 프레임워크의 IoT 인터페이스로는 센서 데이터 합성, 액션 그룹화를 수행할 수 없다.

따라서, 본 논문에서는 기존의 분산형 구조와 클라우드 서버 중심 구조, 포그 서버 중심 구조의 문제점을 해결한 사물 추상화 프레임워크인 Smart Fog를 제안한다.

3. 설 계

본 논문에서는 포그 서버 중심 사물 추상화 프레임워크인 Smart Fog를 그림 2와 같이 구현하였다. Smart Fog에서는 스마트 게이트웨이(smart gateway)가 포그 서버 역할을 한다. Smart Fog는 스마트 게이트웨이의 소프트웨어 스택과, 각 장치 상에서 동작하는 소프트웨어 간의 의존성을 최소화하는 IoT 인터페이스들로 구성된다.

스마트 게이트웨이는 각 장치 간의 데이터 전달 기능뿐만 아니라, 사물 인터넷 서비스를 동작하는 기능도 제공한다. 이를 위해, 스마트 게이트웨이 상에서는 여러 소프트웨어 모듈이 동작한다. 스마트 게이트웨이가 사물 장치를 발견하고 데이터를 전달하기 위해 사물 중재자(thing arbiter)가 동작한다. 스마트 게이트웨이가 발견한 사물 장치와 스마트 게이트웨이에서 동작하는 사물 인터넷 서비스를 관리하기 위해 사물 디렉토리(thing directory)가 동작한다. 사물 장치로부터 발생한 센서 데이터와 사물 인터넷 서비스가 제공하는 센서 데이터를 관리하기 위해 센서 데이터베이스(sensor database)가 동작하며, 센서 데이터를 모바일 응용 프로그램에 전달하기 위해 센서 통지자(sensor notifier)가 동작한다.

Smart Fog는 로컬 네트워크 상에서 위치하는 포그 장치인 스마트 게이트웨이에서 사물 인터넷 서비스가 동작한다. 따라서, 기존의 분산형 구조의 사물 추상화에서 발생하는 사물 인터넷 서비스 중복 문제가 해결되어, 불필요하게 발생하였던 모바일 단말 장치의 전력 소모와 네트워크 트래픽이 줄어든다. 또한, 기존의 클라우드

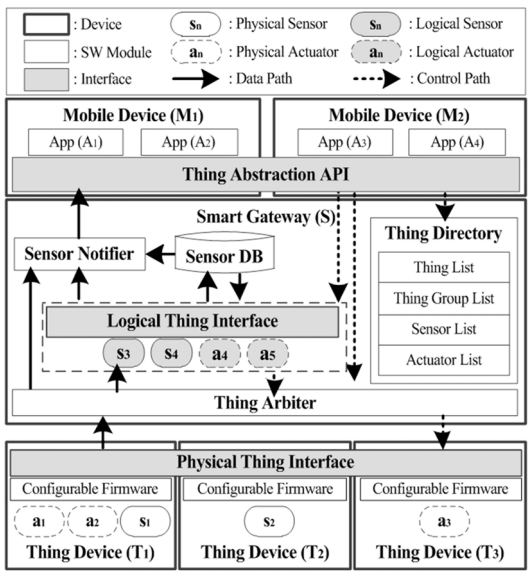

그림 2 Smart Fog의 구조
Fig. 2 The architecture of Smart Fog

서버 중심 구조에 비해서는 네트워크 지연 시간이 짧아서, 실시간 인터랙션을 수행할 수 있다.

Smart Fog에서는 세 가지의 IoT 인터페이스를 정의한다. 사물 추상화 API는 모바일 응용 프로그램이 스마트 게이트웨이와 통신하여, 센서 데이터 및 액션 전달, 사물 인터넷 서비스 설치 기능을 사용할 수 있도록 하는 인터페이스다. 물리적 사물 인터페이스는 스마트 게이트웨이와 통신하기 위해 사물 장치 펌웨어에 필요한 공통 함수에 대한 인터페이스이며, 논리적 사물 인터페이스는 스마트 게이트웨이 상에서 동작하고 다른 장치와 통신하기 위해 사물 인터넷 서비스에 필요한 공통 함수에 대한 인터페이스다.

기존의 포그 중심 사물 추상화 프레임워크에 비해, Smart Fog에서는 각 장치 상에서 동작하는 IoT 인터페이스를 모두 제공하고 있기 때문에, 각 장치 상에서 동작하는 소프트웨어 간의 의존성을 최소화하였다. 또한, 논리적 사물 인터페이스가 센서 데이터 필터링 뿐만 아니라, 센서 데이터 합성, 액션 그룹화까지 범용적으로 사용될 수 있도록 정의되었다.

4. 구 현

4.1 스마트 게이트웨이

Smart Fog의 핵심 장치인 스마트 게이트웨이는, 오픈 소스 IoT 프레임워크인 IoTivity 0.9.1을 기반으로 구현되었다. IoTivity는 IEEE 802.11, 이더넷, 블루투스 등 사물 장치가 사용하는 다양한 네트워크 프로토콜을 RESTful 인터페이스로 추상화하여, 분산형 사물 추상화를 구현하는 IoT 프레임워크다[3]. Smart Fog에서는 IoTivity 기반으로 한 추가 소프트웨어 모듈을 구현하여, 포그 중심 사물 추상화 구조를 구축하였다.

스마트 게이트웨이를 위해 추가로 구현한 소프트웨어 모듈 중, 사물 디렉토리는 사물과 사물 그룹, 센서, 액츄에이터에 대한 목록을 관리한다. 사물 목록(thing list)에서는 사물 장치가 현재 로컬 네트워크에 연결되었는지의 여부를 관리하며, 사물 그룹 목록(thing group list)은 각 사물 장치를 그룹으로 묶어서 관리할 수 있도록 한다. 센서 목록(sensor list)과 액츄에이터 목록(actuator list)에서는 사물 장치에서 제공하는 센서 데이터와 액션에 대한 메타데이터를 관리한다.

사물 중재자는 사물 장치 탐색과 데이터 전달이라는 두 가지 역할을 담당한다. 사물 중재자는 로컬 네트워크에 멀티캐스팅으로 탐색 메시지를 주기적으로 전송하여 사물 장치를 발견한다. 사물 장치가 보낸 응답 메시지를 스마트 게이트웨이가 받으면, 사물 디렉토리의 사물 목록에 네트워크 연결 여부를 업데이트한다. 반면, 응답 메시지가 일정 기간 동안 돌아오지 않는 경우에는, 사물 목록에서 해당 사물이 연결되지 않았음을 명시한다. 또한, 사물 중재자는 사물 장치가 제공하는 센서 데이터와 액션을 직접 주고받는다.

센서 데이터베이스는 사물 장치로부터 수집한 센서 데이터를 저장하고 관리한다. 사물 중재자를 통해 스마트 게이트웨이에 센서 데이터가 들어오게 되면, 가장 먼저 센서 데이터베이스에 저장된다. 모바일 응용 프로그램과 사물 인터넷 서비스들은 센서 데이터베이스를 통해, 현재 시각의 센서 데이터 뿐만 아니라 특정 과거 시간의 센서 데이터도 읽을 수 있다.

센서 통지자는 네트워크를 통해 센서 데이터를 모바일 장치로 전달한다. 모바일 응용 프로그램과 사물 인터넷 서비스들은 특정 센서 데이터를 주기적으로 전달받을 수 있도록 스마트 게이트웨이에 등록한다. 그 후, 센서 통지자는 주기적으로 센서 데이터베이스를 읽어서 해당 모바일 응용 프로그램이나 사물 인터넷 서비스로 센서 데이터를 전달한다.

4.2 인터페이스

Smart Fog에서 제공하는 3가지의 IoT 인터페이스는, IoT 표준 중 하나인 OIC 표준을 바탕으로 구현되었다. OIC 표준은 자원 모델(resource model)을 기반으로 IoT 시스템 내 장치들을 표현하였다. OIC 자원 모델에서는 모든 장치들의 주소를 URI로 기술하고, 장치 간 인터랙션은 RESTful 인터페이스로 표현하며, 센서 데이터와 액션을 자원(resource)이라는 공통 규격으로 표현한다[11].

OIC 표준은 분산형 사물 추상화 구조를 구현하기 위한 표준이기 때문에, 사물 인터넷 서비스가 스마트 게이트웨이에서 구현되는 Smart Fog를 구현하려면 추가 IoT 인터페이스가 필요하다. 따라서, Smart Fog에서는 사물 추상화 API, 물리적 사물 인터페이스, 논리적 사물 인터페이스를 추가로 정의하였다.

사물 추상화 API는 모바일 응용 프로그램이 스마트 게이트웨이를 통해 해당 네트워크의 사물들을 접근할 수 있게 하며, 표 1과 같은 함수로 구성된다. 사물 추상화 API를 통해 모바일 응용 프로그램이 스마트 게이트웨이에 접속하도록 하며, 사물 목록을 전달받는다. 그 후, 해당 API를 통해 사물 장치나 사물 인터넷 서비스가 제공하는 센서 데이터나 액션을 접근할 수 있다. 사물을 편리하게 관리할 수 있도록 그룹으로 묶을 수도 있고, 사물 인터넷 서비스를 설치하고 활성화할 수도 있다.

물리적 사물 인터페이스는 사물 장치의 펌웨어가 스마트 게이트웨이와 통신하기 위해 필요한 최소한의 기능을 정의하는 인터페이스로, 표 2와 같은 함수들로 구성된다. 사물 장치가 초기화하고, 센서 데이터를 샘플링하고, 액션을 적용하는 작업만 물리적 사물 인터페이스에 맞게 구현하면, 사물 장치는 스마트 게이트웨이와 연

표 1 사물 추상화 API의 함수 목록
Table 1 The function list of things abstraction API

Category	Function Name
Initialization	Gateway.{connect(), getThingList()}
Sensor data acquisition	Sensor.{get(), startObserving(), stopObserving()}
Actuator control	Actuator.makeAction()
Thing grouping	ThingGroup.{create(), remove(), include(), exclude()}
IoT service management	Gateway.{installService(), removeService()} Service.{enable(), disable()}

표 2 물리적 사물 인터페이스의 함수 목록
Table 2 The function list of physical thing interface

Category	Function Name
Initialization	onInitialize()
Sensor data sampling	sample()
Action	setAction()

결되어 센서 데이터나 액션을 전달할 수 있다. 이러한 물리적 사물 인터페이스는 센서 데이터 샘플링을 수행하는 함수를 표준화하기 때문에, 스마트 게이트웨이가 사물 장치의 센서 데이터 샘플링 방법과 전송 방법 설정을 변경할 수 있다.

사물 추상화 인터페이스와 물리적 사물 인터페이스는 각각 모바일 응용 프로그램과 사물 장치 펌웨어에서 사용되기 때문에, 네이티브 프로그램을 제작할 수 있는 C++와 C를 사용한다. 장치 간 네트워크 통신은, 내부적으로 IoT 프레임워크인 IoTivity 0.9.1[3]을 사용하여 구현하였다.

사물 인터넷 서비스는 Smart Fog에서 논리적 사물 인터페이스를 통해, 논리적 센서(logical sensor)와 논리적 액츄에이터(logical actuator)의 형태로 구현될 수 있다. 논리적 센서는 다른 센서 데이터를 가공하여 새로운 센서 데이터를 제공하며, 논리적 액츄에이터는 다른 액션들을 이용하여 새로운 액션을 구성한다.

모바일 장치가 스마트 게이트웨이에 사물 인터넷 서비스를 원격 설치할 수 있어야 하기 때문에, 스크립트 언어인 자바스크립트를 사용한다. 그림 3은 스마트 홈의 사물 인터넷 서비스로서 수면 센서와 아침 모드 액츄에이터를 구현한 예시다.

수면 센서에서는 논리적 사물 인터페이스를 사용함으로써, 수면 센서를 구성하기 위해 동작 센서가 필요하다는 점을 inputAttr로 지정할 수 있다. 수면 센서가 스마트 게이트웨이에 설치된 후, 동작 센서가 네트워크에 연결되면 해당 논리적 센서의 입력 센서로서 연결된다. 그 후, 동작 센서로부터 센서 데이터를 전달받으면, 스마트

```
var arlib=require('activity-recognition-library');
var sleepSensor = new LogicalSensor(
   inputAttr = ["motion"],
   onInputSensorData = function(var sData) {
      var rData;
      rData.sleeping =
         arlib.isSleeping(sData.motion);
      return rData;
   });
var morningMode = new LogicalActuator(
   onAction = function(var action) {
      makeAction("coffeepot.switch", "on");
      makeAction("audioplayer.switch", "on");
      makeAction("audioplayer.play", "brahms.mp3");
   });
```

그림 3 논리적 센서 및 논리적 액츄에이터 구현 예시
Fig. 3 An example of logical sensor and logical actuator

게이트웨이에서는 onInputSensorData라는 콜백 함수를 호출한다. onInputSensorData에서는 동작 센서를 가공하여 수면 센서라는 새로운 센서의 데이터를 만들어 반환한다. 이 예제에서는 실시간으로 동작 센서로부터 전달된 데이터를 행동 인식 알고리즘[13] 등으로 현재 거주자의 수면 상태를 분석하도록 구현할 수 있다. 모바일 응용 프로그램들은 스마트 게이트웨이에 접속하여 이러한 논리적 센서 데이터에 접근할 수 있고, 다른 사물 인터넷 서비스도 이 센서 데이터를 가공하여 새로운 센서 데이터를 만들어낼 수 있다.

모바일 응용 프로그램이나 다른 사물 인터넷 서비스가 아침 모드 액츄에이터의 액션을 수행하도록 makeAction 명령을 내리면, 스마트 게이트웨이에서는 해당 논리적 액츄에이터의 onAction이라는 콜백 함수를 실행한다. onAction 함수 내부에서는 다른 액츄에이터에 대해 makeAction 명령을 보내, 액션을 수행하도록 한다. 그림 3과 같은 예시에서는 커피 포트와 오디오 재생기를 켜고, 오디오 재생기가 "brahms.mp3"를 재생하도록 한다. 이 액션은 다른 사물 인터넷 서비스가 사용하여, 새로운 액션을 만들어낼 수도 있다.

5. 실 험

본 논문에서 제안하는 스마트 게이트웨이 중심 사물 추상화의 효과를 실험으로 검증하였다. 이를 위해, 본 논문에서는 스마트 홈 시스템을 가정한 프로토타입을 실제 임베디드 보드 상에 구현하였다. 모바일 장치와 사물 장치, 스마트 게이트웨이로서 Exynos 5422 기반 임베디드 보드 Odroid-XU3를 사용하였고, 각 장치는 이더넷을 통해 하나의 로컬 네트워크에 연결하였다. 스마트 게이트웨이가 멀티캐스팅 메시지를 이용하여 사물 장치를 탐색하기 때문에, 해당 로컬 네트워크의 멀티캐스팅 기능을 활성화하였다. 반복적인 실험을 수행하기 위해, 사물 장치에서는 32개의 시뮬레이션 기반 물리적

센서가 동작하도록 구현하였다. 또한, 스마트 게이트웨이에서는 9개의 사물 인터넷 서비스가 동작하도록 하였다.

Smart Fog의 네트워크 지연 시간은 클라우드 서버 중심 구조보다 짧다. 클라우드 서버인 Amazon EC2와 Smart Fog의 스마트 게이트웨이에서 각각 HTTP 서버가 동작할 때, 10바이트의 작은 웹 페이지에 대한 HTTP 요청에 대한 네트워크 지연 시간을 비교하였다. Amazon EC2는 HTTP 요청에 대해 평균 378ms만큼 소요되는 반면, 스마트 게이트웨이는 평균 15ms만큼 소요된다. 따라서, Smart Fog는 클라우드 서버 중심 구조에 비해 네트워크 지연 시간이 짧으며, 실시간 인터랙션에 더욱 적합하다.

세부적으로는 Smart Fog에서 모바일 장치가 처음 스마트 게이트웨이 연결에 연결할 때 평균 2136.84ms의 지연이 소요되며, 사물 디렉토리를 취득할 때는 평균 87.20ms의 지연이 소요된다. 이후 센서 데이터를 취득할 때는 왕복 시간으로 평균 37.98ms가 소요된다.

분산형 구조와 Smart Fog를 비교 실험하기 위해서, 사물 인터넷 서비스를 비롯하여 스마트 게이트웨이를 구성하는 모든 소프트웨어 모듈을 모바일 단말 장치에서 실행하고, 사물 장치와 P2P로 통신하는 분산형 구조의 프로토타입도 별도로 구현하였다.

프로토타입을 분산형 구조와 Smart Fog에서 모두 실험한 결과는 표 3과 같다. CPU 코어의 평균 부하가 분산형 구조에서 20.69%인 반면, Smart Fog에서는 6.76%로 줄어든다. 이에 따라, 전력 소모는 21% 줄어든다. 분산형 구조를 이용할 때 모바일 장치는 센서 데이터를 가져올 때마다 사물 인터넷 서비스를 실행하기 때문에, 그림 4와 같이 초기 3초 정도의 사물 장치 탐색 시간 이후부터 CPU 코어 부하가 주기적으로 급격하게 증가한다. 반면, Smart Fog 에서는 스마트 게이트웨이가 사물 인터넷 서비스를 수행하고, 모바일 장치는 스마트 게이트웨이와의 통신만 수행한다. 따라서, Smart Fog 에서는 분산형 구조보다 모바일 장치 CPU 코어 부하가 전체적으로 줄어들고, 주기적으로 부하가 급격히 상승하는 현상이 발생하지도 않는다.

표 3 모바일 장치 상 에너지 효율 및 네트워크 성능 비교 실험

Table 3 Experiment for comparison of energy efficiency and network performance in mobile device

Metrics in Mobile Device	Distributed	Smart Fog
CPU core load	20.69%	6.76%
CPU core power consuption	476.33mW	393.64mW
Received network traffic	164,950B	24,444B
Transmitted network traffic	66,706B	35,472B

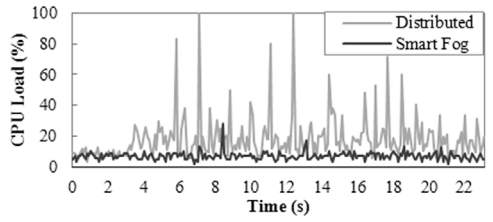

그림 4 분산형 사물 추상화 프레임워크와 Smart Fog의 CPU 코어 부하 비교

Fig. 4 The comparison of the CPU core load in distributed things abstraction framework and Smart Fog

또한, 네트워크를 통해 받은 데이터는 약 85%, 보낸 데이터는 47% 줄어들어, 총 네트워크 트래픽은 약 74% 줄어든다. 이는 스마트 게이트웨이 중심 구조에서 사물 인터넷 서비스 중복 문제가 해결되기 때문이다.

6. 결 론

본 논문에서는 기존의 사물 추상화인 분산형 구조와 클라우드 서버 중심 구조, 포그 서버 중심 구조의 문제점을 지적하였다. 분산형 구조에서는 사물 인터넷 서비스 중복 문제가 있고, 클라우드 서버 중심 구조에서는 실시간 인터랙션 불가 문제가 있다. 또한, 기존의 포그 서버 중심 사물 추상화 프레임워크들은 사물 인터넷 서비스와 사물 장치 간의 의존성이 여전히 남아 있다는 문제가 있다.

이를 해결하기 위해, 본 논문에서는 포그 서버 중심 사물 추상화 프레임워크인 Smart Fog를 설계하였다. Smart Fog는 스마트 게이트웨이와 3개의 IoT 인터페이스들로 구성된다. 스마트 게이트웨이 상에서 동작하는 소프트웨어 스택은 오픈 소스 IoT 프레임워크인 IoTivity를 기반으로 구현되었으며, IoT 인터페이스는 OIC 표준을 기반으로 구현되었다.

실험을 통해, Smart Fog가 서버 중심 구조보다 네트워크 지연 시간이 짧아, 실시간 인터랙션이 가능함을 보였다. 또한, 분산형 구조에 비해 모바일 장치와 전력 소모를 약 21% 줄이고, 네트워크 트래픽은 받는 데이터를 85%, 보낸 데이터를 47% 줄이는 효과가 있음을 증명하였다.

Smart Fog에서 실시간 인터랙션을 하게 되면, 센서 데이터 샘플링 증가로 인해 사물 장치의 전력 소모가 증가하거나, 잦은 센서 데이터 전송으로 인해 네트워크 트래픽이 많아질 수 있다. 향후에는 이 문제를 해결하기 위해, 사물의 센서 데이터 샘플링과 전송 방법을 최적화하는 연구를 진행할 예정이다.

References

[1] P. Bellagente, P. Ferrari, A. Flammini, S. Rinaldi,

"Adopting IoT Framework for Energy Management of Smart Building: A Real Test-case," *Proc. of 1st IEEE International Forum on Research and Technologies for Society and Industry Leveraging a Better Tomorrow (RTSI)*, pp. 138-143, 2015.
[2] Allseen Alliance. AllJoyn Framework [Online]. Available: http://allseenalliance.org/framework (downloaded 2016, Apr. 18)
[3] Linux Foundation. IoTivity: Architecture Overview [Online]. Available https://www.iotivity.org/documentation/architecture-overview (downloaded 2016, Feb. 4)
[4] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwalder, B. Koldehofe, "Mobile Fog: A Programming Model for Large-scale Applications on The Internet of Things," *Proc. of the 2nd ACM SIGCOMM Workshop on Mobile Cloud Computing*, pp. 15-20, 2013.
[5] D. Willis, A. Dasgupta, S. Banerjee, "ParaDrop: A Multi-Tenant Platform to Dynamically Install Third Party Services on Wireless Gateways," *Proc. of the 9th ACM Workshop on Mobility in the Evolving Internet Architecture*, pp. 43-48, 2014.
[6] F. Bonomi, R. Milito, P. Natarajan, J. Zhu, "Fog Computing: A Platform for Internet of Things and Analytics," *Big Data and Internet of Things: A Roadmap for Smart Environments*, Vol. 546, pp. 169-186, 2014.
[7] M. Aazam, E. Huh, "Fog Computing and Smart Gateway based Communication for Cloud of Things," *Proc. of the International Conference on Future Internet of Things and Cloud (FiCloud)*, pp. 464-470, 2014.
[8] D. H. Phan, J. Suzuki, S. Omura, K. Oba, "Toward Sensor-cloud Integration as a Service: Optimizing Three-tier Communication in Cloud-integrated Sensor Networks," *Proc. of the 8th International Conference on Body Area Networks (BodyNets)*, pp. 355-362, 2013.
[9] Y. Xu, S. Helal, M. Thai, M. Scmalz, "Optimizing Push/Pull Envelopes for Energy-efficient Cloud-sensor Systems," *Proc. of the 14th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, pp. 17-26, 2011.
[10] M. Kovatsch, M. Lanter, S. Duquennoy, "Actinium: A RESTful Runtime Container for Scriptable Internet of Things Applications," *Proc. of the 3rd International Conference on the Internet of Things (IOT)*, pp. 135-142, 2012.
[11] Open Connectivity Foundation. (2015, Dec. 23) OIC Specification 1.0 [Online]. Available: http://openconnectivity.org/resources/specifications. (downloaded 2016, Apr. 18)
[12] T. Vlad, S. Wieland, D. Guinard, T. Bohnert, "Design and Implementation of a Gateway for Web-based Interaction and Management of Embedded Devices," *Proc. of the 2nd International Workshop on Sensor Network Engineering (IWSNE)*, pp. 1-14, 2009.
[13] N. C. Krishnan and D. J. Cook, "Activity Recognition on Streaming Sensor Data," *Pervasive and Mobile Computing*, Vol. 10, pp. 138-154, 2014.
[14] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, "Fog Computing and Its Role in the Internet of Things," *Proc. of the First Edition of the MCC Workshop on Mobile Cloud Computing*, pp. 13-15, 2012.
[15] A. D. Wood, J. A. Stankovic, G. Virone, L. Selavo, Z. He, Q. Cao, T. Doan, Y. Wu, L. Fang, R. Stoleru, "Context-Aware Wireless Sensor Networks for Assisted Living and Residential Monitoring," *Network*, Vol. 22, No. 4, pp. 26-33, 2008.

홍 경 환
2013년 성균관대학교 컴퓨터공학과(학사)
2013년~현재 성균관대학교 전기전자컴퓨터공학과 석박통합과정. 관심분야는 임베디드 시스템, 사물 인터넷, 모바일 운영체제

박 은 수
2015년 성균관대학교 컴퓨터공학과(학사)
2015년~현재 성균관대학교 IT융합학과 석사과정. 관심분야는 임베디드 시스템, 사물 인터넷, 모바일 운영체제

최 시 훈
2014년 성균관대학교 컴퓨터공학과(학사)
2016년 성균관대학교 전기전자컴퓨터공학과(석사). 2016년~현재 삼성전자 무선사업부 재직 중. 관심분야는 사물 인터넷, 플래시 메모리, 파일 시스템

신 동 군
1994년 서울대학교 계산통계학과(학사)
2000년 서울대학교 전산과학과(석사). 2004년 서울대학교 컴퓨터공학부(박사). 2004년~2007년 삼성전자 소프트웨어 책임연구원. 2007년~현재 성균관대학교 소프트웨어대학 부교수. 관심분야는 임베디드 시스템, 실시간 시스템, 저전력 시스템