Smart Fog: 실시간 인터랙션 및 에너지 효율을 위한 스마트 게이트웨이 중심 사물 추상화 프레임워크

홍경환○, 박은수, 최시훈, 신동군
성균관대학교 정보통신공학부
redc7328@skku.edu pes9488@skku.edu, beswan@skku.edu, dongkun@skku.edu

Smart Fog: Smart Gateway-centric Thing Abstraction Framework for Real-time Interaction and Energy-efficiency

Gyeonghwan Hong, Eunsoo Park, Sihoon Choi, Dongkun Shin
School of Information and Communication Engineering, Sungkyunkwan University

요 약

스마트 홈, 스마트 시티 등 IoT 서비스를 구현하기 위해 사물 추상화에 대한 여러 연구가 제시된 바 있다. 기존의 서버 중심 사물 추상화는 네트워크 지연 시간이 길기 때문에 실시간 인터랙션을 할 수 없고, 분산형 사물 추상화는 IoT 서비스 중복 문제로 인해 모바일 장치의 전력 소모가 증가하는 문제가 있다. 본 논문에서는 이 문제를 해결하기 위해 스마트 게이트웨이 중심 사물 추상화 구조를 제안하고, 실제 장치에서 동작하는 프레임워크인 Smart Fog를 설계하고 구현하였다. Smart Fog는 스마트 게이트웨이를 운용하기 위한 여러 인터페이스와 소프트웨어 모듈로 구성되며, 이를 통해 IoT 서비스를 프로그래밍할 수 있다. 실험을 통해, Smart Fog가 실시간 인터랙션이 가능할 정도로 네트워크 지연 시간이 짧고, 네트워크 트래픽과 모바일 장치의 전력 소모가 감소함을 확인하였다.

1. 서 론

최근 사물 인터넷(IoT; internet of things)은 센서(sensor)와 액츄에이터(actuator)가 부착된 사물 장치(thing device)들이 상호 운용하는 플랫폼으로 부상하고, 이를 기반으로 각종 IoT 서비스가 등장하고 있다. IoT 서비스는 사물 장치의 센서 데이터와 액츄에이터 동작을 새로운 형태의 센서 데이터와 액츄에이터 동작으로 가공하여 사용자에게 제공하는 서비스를 의미한다. 예를 들어, 스마트 홈에서는 슬립 센서를 통해 거주자가 깨어났는지를 확인하여 오디오 재생기와 커피 포트를 켜주는 것과 같은, 여러 시나리오의 IoT 서비스를 제공한다.

IoT 서비스에서 사용되는 사물 장치들은 처리 능력이 낮고 입출력 장치가 미비한 임베디드 시스템이기 때문에, IoT 서비스들은 클라우드 서버의 처리 능력과 스마트폰의 입출력 장치로 구현되고 있다. 따라서, 사물 장치가 서버나 스마트폰과 데이터를 교환할 수 있어야 IoT 서비스를 구현할 수 있다.

그러나, 사물 장치마다 위치와 센서/액츄에이터 구성이 다르기 때문에, 모바일 응용 프로그램이 사물 장치의 위치나 구성에 관계 없이 사물 장치와 통신하고 IoT 서비스를 제공하는 사물 추상화(thing abstraction)에 대한 많은 연구가 제시되었다. 기존에는 사물 추상화의 구조로서 서버 중심 구조와 분산형 구조가 제시된 바가 있다.

본 논문에서는 기존의 사물 추상화인 서버 중심 구조와 분산형 구조의 문제점을 지적하고, 이를 해결하기 위해 스마트 게이트웨이 중심 구조의 사물 추상화를 제시한다. 이를 기반으로, 스마트 게이트웨이 중심 사물 추상화 프레임워크인 Smart Fog를 개발하였다. 본 논문에서는 Smart Fog가 실시간 인터랙션이 가능할 정도로 네트워크 지연 시간이 짧고, 기존 사물 추상화 구조에 비해 모바일 장치의 전력 소모와 네트워크 트래픽이 감소하였음을 실제 임베디드 보드 기반으로 실험하여 보였다.

본 연구는 미래창조과학부 및 정보통신기술진흥센터의 서울어코드활성화지원사업의 연구결과로 수행되었음 (IITP-2015-R0613-15-1062)

본 연구는 미래창조과학부 및 정보통신기술진흥센터의 SW중심대학-ICT/SW창의연구과정 지원사업의 연구결과로 수행되었음 (IITP-2015-R2215-15-1005)

2. 관련 연구

사물 추상화는 사물 장치의 위치와 구성에 관계 없는 통신 추상화 인터페이스와, 프로그래밍 가능한 IoT 서비스 인터페이스를 제공하는 것을 의미한다. 그림 1과 같이, 기존의 사물 추상화는 서버 중심 구조와 분산형 구조를 사용하였다.

초기에는 사물 장치가 IPv6 네트워크 스택조차 동작할 수 없을 정도로 처리 성능이 낮아서, 게이트웨이가 사물 장치 대신 네트워크 연결 기능을 제공해주는 서버 중심 구조를 사용하였다. 서버 중심 구조에서는 그림 1의 (a)와 같이, 클라우드 서버가 게이트웨이에 수집된 센서 데이터를 가져오고, 이를 가공하는 IoT 서비스를 제공한다. 모바일 응용 프로그램은 RESTful 인터페이스를 통해 IoT 서비스를 사용할 수 있다.

기존의 서버 중심 구조 중 하나인 사물 통신(M2M; machine to machine)에서는 게이트웨이를 통해 단순히 센서 데이터나 액츄에이터 명령을 전달하는 구조가 제시되었다[1]. 무선 센서 네트워크에서는 클라우드 서버 IoT 서비스가 센서 데이터를 가져오는 방법에 대한 연구가 있다[2]. IoT 서비스를 제공하는 서버 중심 사물 추상화 프레임워크인 Actinium이 제안된 바가 있으며[3], 무선 센서 네트워크(WSN; wireless sensor network)에서는 네트워크를 CEB(Cloud-Edge-Beneath)의 3단계 구조로 구성하고, 센서 데이터 취득을 최적화하는 연구도 있다[4].

그러나, 서버 중심 구조에서는 실시간 인터랙션(real-time interaction)이 불가능하다는 문제가 있다. 실시간 인터랙션은 실시간으로 사물 장치와 통신하여 센서 데이터를 제공받거나 액츄에이터를 제어하는 것을 의미한다. 예를 들어, 스마트 홈 사물 장치들로부터 취득한 센서 데이터를 바탕으로, 현재 거주자의 행동을 분석하여 실시간으로 가전 기기를 조작할 수도

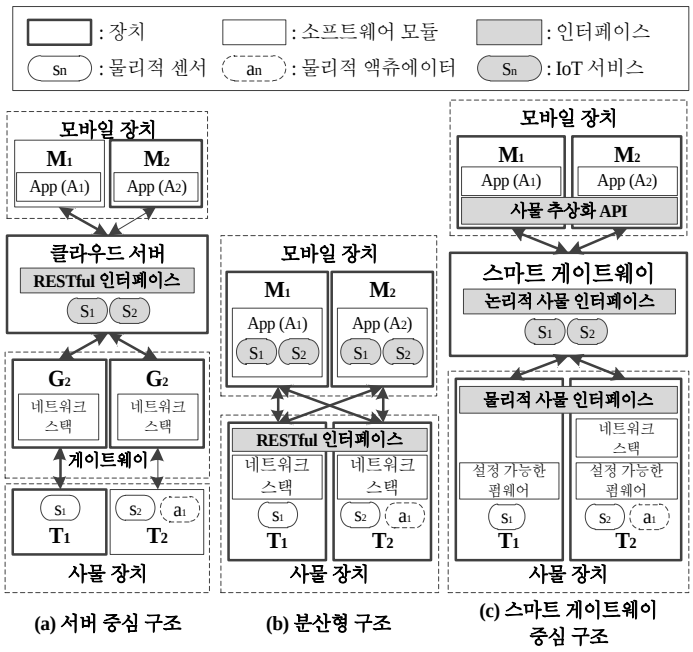

그림 1 기존의 사물 추상화 구조와의 비교

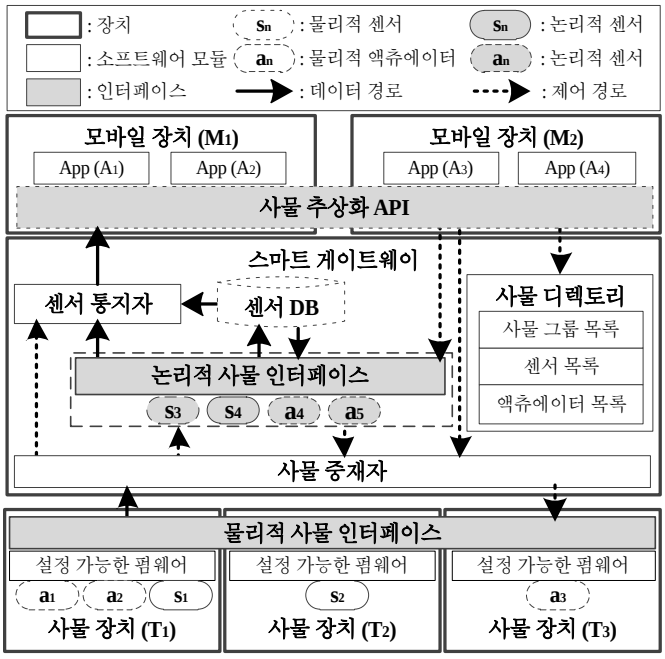

그림 2 Smart Fog의 구조

있다[4]. 서버 중심 구조에서는 클라우드 서버를 접근할 때 소요되는 네트워크 지연 시간이 길기 때문에, 사물 장치와의 실시간 인터랙션이 필요한 응용 프로그램을 구현하기 어렵다.

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 (b)와 같이, 스마트폰 응용 프로그램의 형태로 IoT 서비스를 제공하는 분산형 구조가 제기되었다. 분산형 구조에서는 사물 장치가 RESTful 인터페이스를 통해 모바일 장치에 직접 센서 데이터와 액츄에이터 제어 기능을 제공한다.

분산형 구조 중 하나인 AllJoyn은 퀄컴이 주도하는 분산형 IoT 프레임워크이며, 네트워크 스택을 추상화하는 코어 API와 음성 스트리밍 등 상위 단계 서비스도 제공한다[5]. IoTivity는 리눅스 파운데이션이 관리하는 분산형 IoT 프레임워크로, OIC 표준을 바탕으로 IoT 장치 간의 통신을 추상화한 베이스 API와 소프트 센서 등 상위 단계 서비스도 제공한다[6].

그러나, 이 구조에서는 여러 모바일 장치 상에서 각자 IoT 서비스가 동작하는 IoT 서비스 중복 문제가 발생한다. IoT 서비스는 센서 데이터를 수집과 센서 데이터 관리, 센서 데이터 결합, 액츄에이터 제어를 수행하고, 이는 각 모바일 장치에서 중복 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.

3. 스마트 게이트웨이 중심 사물 추상화

기존의 서버 중심 구조에서는 실시간 인터랙션 불가 문제가 있으며, 분산형 구조에서는 IoT 서비스 중복 문제가 있다. 클라우드 서버보다 네트워크 지연 시간이 더 짧으며, IoT 서비스를 공유할 수 있는 포그 서버(fog server)[8]가 로컬 네트워크에서 모바일 장치와 사물 장치 간의 통신을 중계하면, 기존 사물 추상화의 문제를 모두 해결할 수 있다.

본 논문에서는 포그 서버 역할을 하는 스마트 게이트웨이(smart gateway) 중심 구조의 사물 추상화를 제안한다. 스마트 게이트웨이는 기존의 게이트웨이가 수행하였던 사물 장치와의 통신 기능뿐만 아니라, IoT 서비스까지 제공한다는 점이 다르다. 스마트 게이트웨이 중심 구조는 그림 1의 (c)와 같이, 사물 장치와 스마트 게이트웨이, 모바일 장치로 구성된다.

스마트 게이트웨이 중심 구조에서는 스마트 게이트웨이가 사물 장치와 통신하여 센서 데이터를 접근하거나 액츄에이터 제어를 수행하고, 모바일 응용 프로그램에는 사물 추상화 인터페이스를 통해 IoT 서비스를 제공한다. 스마트 게이트웨이가 사물 장치와 통신하기 위해 물리적 사물 인터페이스를 제공하며, IoT 서비스를 스마트 게이트웨이에서 동작시키기 위해 사물 추상화 인터페이스(thing abstraction interface)를 제공한다.

스마트 게이트웨이 중심 구조는 기존 서버 중심 구조와 완전히 다르다. 서버 중심 구조에서 사용하는 클라우드 서버에 비해 스마트 게이트웨이는 물리적으로 사물 장치와 가까운 곳에 존재하는 포그 서버이기 때문에, 네트워크 지연 시간이 짧고 상황 인식(context-awareness)을 고려한 IoT 서비스를 만들 수 있다[7]. 네트워크 지연 시간이 짧기 때문에, 사물 장치와의 실시간 인터랙션을 구현하는 데 적합하다. 또한, 상황 인식은 신체 네트워크(body area network)나 설치 센서 네트워크(emplaced sensor network)에서 사용될 수 있다[8].

스마트 게이트웨이 중심 구조는 기존 분산형 구조에 비해서도 강점이 있다. 분산형 구조에서 모바일 장치가 각자 수행하던 센서 데이터 수집, 센서 데이터 관리, 센서 데이터 결합, 액츄에이터 제어 동작을, 스마트 게이트웨이 중심 구조에서는 스마트 게이트웨이가 대신 수행한다. 이를 바탕으로, 스마트 게이트웨이는 IoT 서비스를 제공하며, 모바일 장치는 이를 공유하여 사용할 수 있다. 따라서, 기존의 분산형 구조에서 발생하는 IoT 서비스 중복 문제가 해결된다.

4. 프레임워크 설계

본 논문에서는 스마트 게이트웨이 중심 사물 추상화 프레임워크인 Smart Fog를 구현하였다. Smart Fog를 그림 2과 같이, 모바일 장치, 사물 장치, 스마트 게이트웨이 상에서 동작해야 하는 인터페이스와 소프트웨어 모듈로 구성된다.

Smart Fog는 세 가지의 인터페이스로 구현된다. 사물 추상화 API(thing abstraction API)는 모바일 응용 프로그램이 스마트 게이트웨이를 통해 해당 네트워크의 사물들을 접근할 수 있게 한다. 물리적 사물 인터페이스(physical thing interface)는 사물 장치의 펌웨어가 스마트 게이트웨이와 통신하기 위해 필요한 최소한의 기능을 정의한다. 사물 추상화 인터페이스와 물리적

```
var arlib = require('activity-recognition-library');
var mWin = new MotionDataWindow(size = 10);
var sleepsensor = new LogicalSensorPlugin(
  inputAttr = ["motion"],
  onInputSensorData = function(var sData) {
    rData.sleeping = arlib.isSleeping(sData.motion);
    return (malib.isSleeping(mWin));
  });
var morningMode = new LogicalActuatorPlugin(
  onAction = function(var action) {
    makeAction("coffeepot.switch", "on");
    makeAction("audioplayer.switch", "on");
    makeAction("audioplayer.play", "brahms.mp3");
  });
```

그림 3 논리적 센서 및 액츄에이터 플러그인 구현 예시

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[6]를 사용하여 구현하였다.

IoT 서비스는 Smart Fog에서 논리적 사물 인터페이스(logical thing interface)를 통해, 논리적 센서 플러그인(logical sensor plugin)과 논리적 액츄에이터 플러그인(logical actuator plugin)의 형태로 구현될 수 있다. 모바일 장치가 이 플러그인들을 스마트 게이트웨이에 원격 설치할 수 있어야 하기 때문에, 스크립트 언어인 자바스크립트를 사용한다. 그림 3은 스마트 홈의 슬립 센서와 아침 모드 액츄에이터를 구현한 예시다. 슬립 센서는 실시간으로 모션 센서로부터 전달된 데이터를 행동 인식 알고리즘[4]으로 현재 거주자의 수면 상태를 분석하고, 아침 모드 액츄에이터는 커피 포트와 오디오 재생기를 켜고, 오디오 재생기가 "brahms.mp3"를 재생하도록 한다.

스마트 게이트웨이에서는 여러 소프트웨어 모듈이 동작한다. 사물 디렉토리(thing directory)는 OIC 자원 모델(resource model)[9]에 기반하여 표현된 사물 그룹과 센서, 액츄에이터의 목록을 관리한다. 사물 중재자(thing arbiter)는 사물 장치를 멀티캐스팅 메시지로 발견하고, 센서 데이터를 요청하거나 액츄에이터 제어 명령을 전달하는 역할을 담당한다. 센서 데이터베이스(sensor database)는 사물 장치로부터 수집한 센서 데이터를 저장하고 관리한다. 센서 통지자(sensor notifier)는 네트워크를 통해 센서 데이터를 모바일 장치로 전달한다.

사물 장치는 물리적 사물 인터페이스에 따라 펌웨어를 제작하여, 스마트 게이트웨이가 센서 데이터를 가져오거나 액츄에이터 명령을 내릴 수 있다. 또한, 사물 장치의 펌웨어가 물리적 사물 인터페이스를 사용하기 때문에, 스마트 게이트웨이가 사물 장치의 센서 데이터 샘플링 방법이나 전송 방법 등을 설정을 통해 변경할 수도 있다.

5. 실 험

본 논문에서 제안하는 Smart Fog의 효과를 실험으로 검증하였다. 이를 위해, 모바일 장치와 사물 장치, 스마트 게이트웨이로서 Exynos 5422 기반 임베디드 보드 ODROID-XU3를 사용하였다. 각 장치는 이더넷으로 하나의 LAN에 연결하였다.

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. 인간의 관심을 피할 수 있는 최대 지연 시간은 100ms라고 밝힌 실험[10]에 따르면, 센서 데이터를 취득하는 실제 사용 상황에서는 실시간 인터랙션을 수행할 수 있다.

기존 사물 추상화 구조와의 비교 실험 시나리오는 스마트 홈 환경을 가정하였다. 사물 장치상에서는 32개의 센서가 동작하며, 스마트 게이트웨이에서는 그림 3과 같은 논리적 센서가 9개 동작하도록 하였다. 이 시나리오를 Smart Fog와, 분산형 구조의 사물 추상화 프레임워크에서 모두 실험하였다. 분산형 구조를 구현하기 위해, 스마트 게이트웨이 상에서 동작하는 모든 소프트웨어 모듈을 모바일 장치에서 실행하고, 사물 장치와 P2P로 통신하도록 하였다.

표 1 기존의 사물 추상화 구조와의 비교 실험

모바일 장치 상의 측정치	분산형	Smart Fog
코어 부하 (%)	20.69	6.76
코어 전력 소모 (mW)	476.33	393.64
받은 패킷 (Bytes)	164,950	24,444
보낸 패킷 (Bytes)	66,706	35,472

두 가지 구조에서 실험하여, 표 2과 같은 결과를 얻었다. CPU 코어의 평균 부하가 분산형 구조에서 20.69%인 반면, Smart Fog에서는 6.76%로 줄어든다. 이에 따라, 전력 소모는 21% 줄어든다. 또한, 네트워크를 통해 받은 패킷은 약 85%, 보낸 패킷은 47% 줄어든다. 이는 스마트 게이트웨이 중심 구조에서 IoT 서비스 중복 문제가 해결되기 때문이다.

6. 결 론

본 논문에서는 기존의 사물 추상화인 서버 중심 구조와 분산형 구조의 문제점을 지적하였다. 서버 중심 구조에서는 실시간 인터랙션 불가 문제가 있고, 분산형 구조에서는 IoT 서비스 중복 문제가 있다. 이를 해결하기 위해, 본 논문에서는 스마트 게이트웨이 중심 사물 추상화를 제안하고, 프레임워크인 Smart Fog를 구현하였다. 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 패킷이 85%, 보낸 패킷이 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.

참고 문헌

[1] S. K. Datta, et al. “An iot gateway centric architecture to provide novel m2m services.” *World Forum on Internet of Things (WF-IoT)*, IEEE, 2014.

[2] Y. Xu, et al. “Optimizing push/pull envelopes for energy-efficient cloud-sensor systems.” *Proceedings of the 14th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems (MSWiM)*, ACM, 2011.

[3] M. Kovatsch, et al. "Actinium: A restful runtime container for scriptable internet of things applications." *Internet of Things (IOT)*, 2012 3rd International Conference on the. IEEE, 2012.

[4] N.C. Krishnan et al. “Activity recognition on streaming sensor data.” *Pervasive and mobile computing*, vol. 10, pp. 138-154, 2014.

[5] Allseen Alliance. “AllJoyn”, 2015, http://allseenalliance.org.

[6] Linux Foundation. “IoTivity: Architecture Overview.” 2015, https://www.iotivity.org/documentation/architecture-overview.

[7] F. Bonomi, et al. “Fog computing and its role in the internet of things.” *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, ACM, 2012.

[8] A. D. Wood, et al. “Context-aware wireless sensor networks for assisted living and residential monitoring.” *in Network*, IEEE, vol. 22, no. 4, pp. 26-33, 2008.

[9] Open Interconnect Consortium. “OIC Candidate Specification 1.0.” 2015, http://openinterconnect.org/developer-resources/specs.

[10] J. Nielsen. “Usability engineering.” Elsevier, 1994.