포그 컴퓨팅을 위한 효율적인 IoT 플랫폼

(An Efficient IoT Platform for Fog Computing)

이한솔*, 최정우*, 변기범*, 홍지만**
(Han Sol Lee, Jeong Woo Choi, Gi Beom Byeon, Ji Man Hong)

요 약

IoT 디바이스 기술의 발전으로 디바이스가 주변 환경을 인식하고 동작하게 되면서, 막대한 양의 IoT 디바이스 데이터를 효율적으로 처리하기 위한 방안이 요구되고 있다. 기존에 사용되던 클라우드 컴퓨팅은 부하와 거리에 따른 전송 지연 문제가 발생한다. 이러한 문제를 해결하기 위해 포그 컴퓨팅이 등장하였다. 포그 컴퓨팅은 IoT 디바이스를 제어하기 위한 환경으로, 클라우드의 단점을 해결하기 위해 IoT 디바이스를 가까이 두어 근거리 통신을 수행한다. 그러나 IoT를 위한 포그 컴퓨팅 관련 연구들은 포그컴퓨팅의 구조와 프레임워크에 대한 연구가 주를 이룬다. 따라서 본 논문에서는 포그컴퓨팅을 수행하기 위한 플랫폼을 제안한다. 제안하는 플랫폼은 포그 컴퓨팅 환경에서 IoT 디바이스를 모니터링 및 분석, 제어할 수 있는 통합 플랫폼이다.

■ 중심어 : 사물인터넷 ; 포그 컴퓨팅 ; 플랫폼

Abstract

With IoT device technology developments, such devices now can perceive the surrounding environment and operate upon the condition, but a method for efficiently processing an enormous amount of IoT device data is required. The existing cloud computing has a transmission delay problem due to load and distance. Fog Computing, an environment to control IoT devices, therefore, emerged to solve this problem. In Fog Computing, IoT devices are located close to each other to solve the shortcomings of the cloud system. While many earlier studies on Fog Computing for IoT mainly focus on its structure and framework, we would like to propose an integrated Fog Computing platform that monitors, analyzes, and controls IoT devices.

■ keywords : IoT ; Fog Computing ; Platform

Ⅰ. 서 론

네트워크 인프라(Infrastructure)를 기반으로 저비용의 광대역 무선 연결이 가능해지면서 인터넷은 우리 삶에 막대한 영향을 미치게 되었다[1]. 이러한 인터넷은 사람과 사람 사이를 넘어 사람과 사물 사이, 사물과 사물 사이 상호작용을 실현하는 사물 인터넷(Internet of Things, IoT)의 발전을 이루었다. IoT는 물리적 객체가 정보 네트워크에 원활하게 통합되고 그 객체가 비즈니스 프로세스에 적극적으로 참여할 수 있게 하는 세계이다[2].

IoT 세계에서 활용되는 물리적 객체 중 하나인 IoT 디바이스는 센서와 액추에이터(Actuator), 웨어러블(wearable) 기기 센서 [3] 등을 통해 다양한 종류의 데이터를 대량으로 수집하고 생성한다. .또한 IoT 디바이스는 점점 스마트하게 IoT 환경을 스스로 인지할 수 있고 원격 서버로부터 명령을 받아 특정한작업을 수행할 수 있다. 스마트한 IoT 디바이스들이 상호작용하여 스마트 홈, 스마트 팩토리 등 다양한 분야에 적용되면서 IoT 서비스와 서비스 사용자 수가 계속 증가하고 있다.[4]

IoT 디바이스에 의해 수집되고 생성된 대량의 데이터는 원격 서버로 전달되고 원격 서버는 이 데이터들을 분석하고 처리하여 가공된 형태로 IoT 사용자에게 서비스를 제공한다. 이를 위해 클라우드 컴퓨팅(Cloud Computing)이 스마트해진 IoT를 효율적으로 관리하고 처리하는 컴퓨팅 패러다임이 되었다.[5]

* 비회원, 숭실대학교 컴퓨터학과
** 종신회원, 숭실대학교 컴퓨터학과
이 논문은 2017년도 정부(과학기술정보통신부)의 재원으로 한국연구재단-차세대정보컴퓨팅기술개발사업의 지원을 받아 수행된 연구임(No. NRF-2017M3C4A7069432)..

접수일자 : 2019년 03월 08일
수정일자 : 2019년 03월 20일
게재확정일 : 2019년 03월 20일
교신저자 : 홍지만 e-mail : jiman@ssu.ac.kr

클라우드 컴퓨팅은 메모리, 스토리지(Storage) 등의 자원을 공유하여 구성한 서버를 통해 인프라와 데이터를 중앙 집중적으로 관리하고 서비스를 신속히 제공하기 때문이다[1].

그러나 클라우드 컴퓨팅 환경의 중심에 있는 서버는 IoT 디바이스로부터 물리적으로 먼 곳에 위치하기 때문에 데이터 전송 시 지연이 자주 발생한다[6]. 또한 IoT 서비스와 디바이스, 서비스 사용자의 수가 나날이 증가함에 따라 끊임없이 수집되고 생성되는 대용량의 데이터가 효율적으로 처리되기 힘들다. 이러한 문제를 해결하기 위하여 IoT 디바이스의 응답 시간을 최소화하고 실시간 처리를 가능하게 하는 포그 컴퓨팅(Fog Computing) 패러다임이 등장하였다.

포그 컴퓨팅은 클라우드 서버와 말단 디바이스(Edge Device) 사이에 서버를 위치시켜 데이터 연산과 스토리지, 네트워킹 등의 서비스를 제공하는 패러다임이다[7]. 포그 컴퓨팅은 클라우드와 신속히 연결되고 사용자 가까이에 위치하는 포그 서버를 통해 실시간 처리와 위치 기반 서비스, 이동성 지원을 제공한다[8]. 포그 컴퓨팅은 스마트 홈과 스마트 오피스, 스마트 팩토리 등에 적용되어 다양한 서비스를 제공하고 있으며 포그 컴퓨팅 관련 연구도 활발히 진행되고 있다. 그러나 IoT를 위한 포그 컴퓨팅 관련 연구들은 포그 컴퓨팅 환경에서 포그 서버의 배치, 특정 서비스를 제공하기 위한 프레임워크에 대한 연구가 주를 이룬다.

따라서 본 논문에서 포그 컴퓨팅을 위한 플랫폼을 제안한다. 제안하는 플랫폼은 IoT 디바이스의 상태와 행위를 모니터링하고 IoT 디바이스에 의해 수집되고 생성된 데이터를 저장 및 분석하여 IoT 디바이스를 제어하는 서비스를 제공하는 플랫폼이다. 제안하는 플랫폼은 사용자나 IoT 디바이스의 추가적인 요구사항에 따라 모듈, 프레임워크, IoT 디바이스 등을 내부 구조에 쉽게 이식하여 새로운 서비스를 제공할 수 있어 일관성과 확장성을 지닌다.

Ⅱ. 관련연구

1. 포그 컴퓨팅

포그 컴퓨팅은 클라우드 컴퓨팅을 네트워크 중심부에서 네트워크 가장자리(Edge)까지 확장시킨 패러다임이다. [그림 1]과 같이 중앙 클라우드 서버와 말단 스마트 사물들 사이에 포그 서버들을 배치하여 사용자에게 서비스를 제공하는 구조이다. 클라우드 컴퓨팅은 서버, 스토리지 등 컴퓨팅 자원을 공유하고 사용자가 네트워크를 통해 공유 자원에 접근을 가능하게 하여 요구사항에 맞는 서비스를 신속히 제공한다.

포그 컴퓨팅은 이러한 클라우드 컴퓨팅의 주문형(On-demand) 서비스, 광대역 네트워크 접근, 빠른 탄력성(Elasticity) 등의 특징들을 가지면서 동시에 다음 특징들도 가진다[6][9]. 첫 번째 특징은 가장자리 가까이에 위치하기 때문에 지연 발생 빈도가 낮아진다는 것이다. 이러한 특징은 포그 컴퓨팅이 등장하게 된 첫 배경이며 물리적으로 넓게 분포되어있는 수많은 네트워크 가장자리 노드에 서비스를 원활히 지원하기 위함이다. 두 번째 특징은 실시간 처리를 통한 상호작용과 이동성 지원이 가능하다는 것이다.

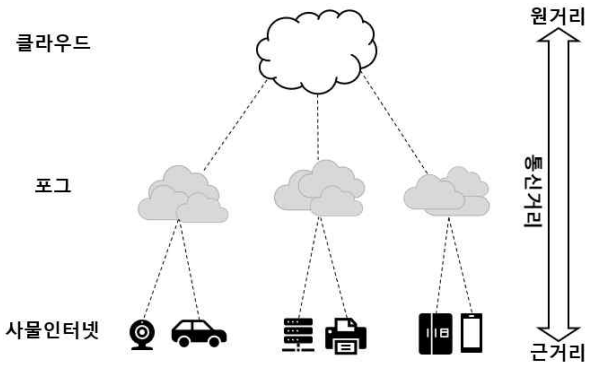


그림 1. 포그 컴퓨팅 구성도

중앙집중화(Centralization) 되어있는 클라우드 컴퓨팅과 다르게 포그 컴퓨팅은 서비스 제공 대상으로 삼는 말단 디바이스들이 매우 광범 위하게 분산되어 있다. 이러한 환경에서 일괄적인 처리보다 실시간 처리를 통한 상호작용이 필요하며 말단 디바이스들 중 모바일 디바이스들도 높은 비율을 차지하고 있어 이동성 지원도 요구된다. 마지막 특징은 이질적인(Heterogeneous) 환경에서 상호 운용이 가능하다는 것이다. 포그 서버는 지리적으로 분산되어있을 뿐만 아니라 서로 다른 환경에 배치되기 때문에 특정 서비스에 대해 지원하기 위해서 업체 간의 협력이 필요하다. 다양한 업체 간에 상호 운용이 가능해야 하고 서로 다른 도메인에서 적용 가능해야 한다.

2. 포그 컴퓨팅 프레임워크에 관한 연구

Dsouza 등은 [10]에서 포그 컴퓨팅의 오케스트레이션 레이어(Orchestration Layer)를 지원하기 위한 정책 관리 모듈을 통해 확장된 프레임워크를 제안하였다. 확장된 프레임워크는 정책 결정 엔진(Policy Decision Engine), 정책 해결자(Policy Resolver), 정책 저장소(Policy Repository), 정책 집행자(Policy Enforcer) 등의 모듈로 구성된다. 이 모듈들은 데이터 기반 의사결정, 사용자 인증 및 확인, 정책 저장 등의 기능을 제공한다.

Cisco의 IOx는 오픈 소스(Open Source) 소프트웨어 Linux와 Cisco의 네트워크 운영체제(Internetwork Operating System, IOS)가 결합된 플랫폼이다[11]. IOx는 Linux 기반의 개방적이고 확장 가능한 환경에서 다른 운영체제나 응용 프로

그램을 호스팅하고 개발한다. IOx는 응용 프로그램을 프로토콜, 디바이스 및 인터페이스와 연결하여 시간과 비용을 절약하고 실시간으로 데이터를 분석하여 클라우드 서버에 저장한다.

Lee 등은 [12]에서 적절한 포그 디바이스 이웃 집합을 찾는 온라인 알고리즘을 접목시킨 온라인 비서(Secretary) 프레임워크를 제안했다. 이 프레임워크는 주어진 포그 디바이스를 중심으로 포그 네트워크를 형성하여 지연을 최소화한다. 이를 통해 이 프레임워크는 컴퓨팅 자원을 원격 클라우드 서버와 포그 네트워크에 적절하게 분배한다.

Nishio 등은 [13]에서 서비스 지향적인 유틸리티 함수 기반의 이기종 자원 공유를 위한 수학적인(Mathematical) 프레임워크를 제안하였다. 포그 컴퓨팅에서 극대화될 수 있는 이 프레임워크는 서로 다른 단위를 정량화(Quantification)하여 모든 수치들을 시간 자원에 동등하게 매핑(Mapping)한다. 이를 통해 이 프레임워크는 서비스 지연을 효과적으로 줄이고 에너지 효율성을 높인다.

Willis 등은 [14]에서 개발자들이 사용자 영역에 있는 컴퓨팅 자원들을 활용할 수 있게 해주는 엣지 컴퓨팅(Edge Computing) 프레임워크 ParaDrop을 제안하였다. 이 프레임워크는 개발자들이 가상화된 컴퓨팅 자원을 설계하여 사용자 가까이에서 서비스를 제공할 수 있게 한다. 사용자들은 개발자 API(Application Programming Interface)를 이용하여 동적 설치(Dynamic Installation)와 관리 정책 설계를 통한 긴밀한 자원 제어 서비스를 제공받는다.

III. MACaaS 플랫폼

1. 포그 컴퓨팅 도메인

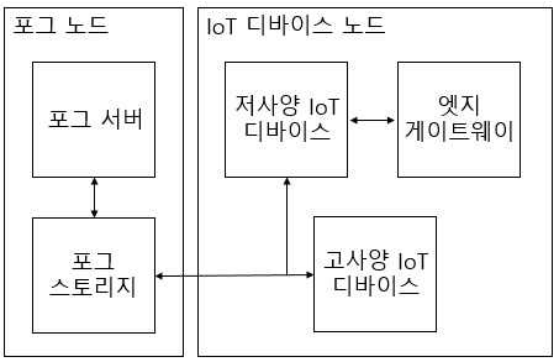


그림 2. 제안하는 플랫폼의 포그 컴퓨팅 도메인

본 논문에서 제안하는 MACaaS 플랫폼이 적용될 영역인 포그 컴퓨팅 도메인은 [그림 2]과 같다. 포그 컴퓨팅 도메인은 포그 공간이라고도 불리며 크게 포그 노드와 IoT 디바이스 노드 두 영역으로 구성된다. 포그 컴퓨팅 도메인은 스마트 홈, 스마트 오피스, 스마트 팩토리 등 다양한 환경으로 간주될 수 있다.

포그 노드는 포그 서버와 포그 스토리지로 구성된다. 포그 서버는 IoT 디바이스 노드를 주기적으로 모니터링하고 IoT 디바이스 노드에 의해 수집되고 생성된 데이터들을 저장하고 분석하여 제어한다. 포그 스토리지는 포그 서버가 사용자 생성 데이터, 시스템 생성 데이터, 그리고 환경 데이터로 분류한 데이터를 저장하고 데이터 백업과 복구도 수행한다.

IoT 디바이스 노드에 위치하는 IoT 디바이스는 크게 고사양 IoT 디바이스, 저사양 IoT 디바이스, 그리고 엣지 게이트웨이 세 유형으로 나누어진다. 이 중 저사양 IoT 디바이스는 단순한 센서와 액추에이터만으로 구성되기 때문에 저조한 연산 능력과 네트워킹 능력을 갖는다. 저사양 IoT 디바이스의 이러한 특징을 보완하기 위하여 엣지 게이트웨이가 사용된다.

엣지 게이트웨이는 저사양 IoT 디바이스가 외부 네트워크에 연결될 수 있도록 저사양 IoT 디바이스에게 내부 인터페이스를 제공한다. 이처럼 저사양 IoT 디바이스는 엣지 게이트웨이와 결합되어 단일 고사양 IoT 디바이스처럼 스마트 IoT 디바이스 역할을 수행할 수 있다.

2. 서비스 프레임워크

본 논문에서 제안하는 플랫폼의 서비스 프레임워크와 데이터 플로우는 [그림 3]과 같다. 포그 서버 안에 구현되는 플랫폼은 모니터링(MaaS) 프레임워크, 분석(AaaS) 프레임워크, 그리고 컨트롤(CaaS) 프레임워크로 구성되며, 각 프레임워크는 세 개의 모듈을 포함한다. 또한 IoT 디바이스 노드와 프레임워크 간, 프레임워크 간, 그리고 IoT 디바이스 간에 사용되는 API들이 정의되며 이는 다음 절에서 설명한다.

가. 모니터링(MaaS) 프레임워크

모니터링 프레임워크는 IoT 디바이스를 연결하고 IoT 디바이스에 의해 수집되고 생성된 데이터를 특성별로 분류하여 저장하고 IoT 디바이스의 상태와 행위를 모니터링한다. 모니터링 프레임워크 내부 모듈들이 수행하는 역할과 기능은 다음과 같다.

(1) 디바이스 커넥터

디바이스 커넥터는 포그 서버에 연결을 요청하는 새로운 IoT 디바이스의 접근을 제어(Access Control)한다. 디바이스 커넥터는 연결 요청 시 전달받은 디바이스 정보를 접근 제어 리스트와 비교하여 인증을 마치고 해당 IoT 디바이스와 포그 서버의 연결을 유지시킨다.

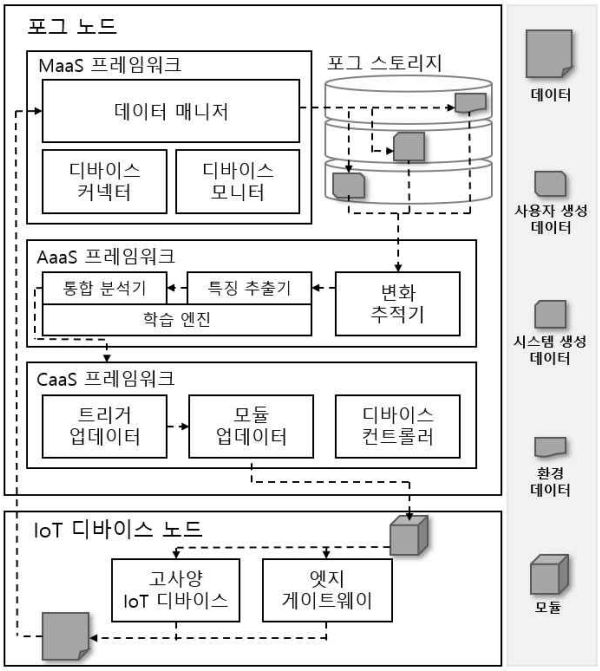


그림 3. 제안하는 플랫폼의 데이터플로우

(2) 데이터 매니저

데이터 매니저는 포그 서버와 연결된 IoT 디바이스에 의해 수집되고 생성된 데이터를 전달받는다. 데이터 매니저는 전달받은 데이터를 [표 1]와 같이 데이터 특성에 따라 분류하여 포그 스토리지에 저장한다.

표 1. 데이터 특성에 따른 데이터 분류

분류	내용	예시
사용자 생성 데이터	사용자에 의해 생성된 개인 데이터	스마트폰의 사진, 동영상, 메모, 일정 등
시스템 생성 데이터	IoT 디바이스에 의해 생성된 데이터	구성파일, 이벤트 트리거, 센싱 데이터 등
환경 데이터	포그 서버와 IoT 디바이스가 위치한 환경 관련 데이터	온도, 습도, 강수량 등

(3) 디바이스 모니터

디바이스 모니터는 포그 서버와 연결된 IoT 디바이스에서 실행되는 프로세스, 네트워크 인프라, I/O(Input/Output) 등의 상태와 행위를 지속적으로 모니터링한다. 디바이스 모니터는 모니터링하는 IoT 디바이스에서 오류(Error/Fault)가 발생하면 오류 정보를 컨트롤 프레임워크의 디바이스 컨트롤러에게 전달한다.

나. 분석(AaaS) 프레임워크

분석 프레임워크는 분류된 데이터의 변화를 추적하여 변화 정보를 만들고 이로부터 학습 엔진(Learning Engine)을 통해 특징(Feature)을 추출하고 새로운 지식(Knowledge)을 생성한다. 분석 프레임워크 내부 모듈들이 수행하는 역할과 기능은 다음과 같다.

(1) 변화 추적기

변화 추적기는 프레임워크 데이터 매니저에 의해 분류되어 저장된 데이터를 포그 스토리지로부터 불러온다. 변화 추적기는 포그 스토리지로부터 불러온 데이터의 변화 추이를 살펴 데이터 변화 정보를 만들고 이를 특징 추출기에 전달한다.

(2) 특징 추출기

특징 추출기는 변화 추적기가 만들어낸 데이터 변화 정보를 변화 추적기로부터 전달받는다. 특징 추출기는 전달받은 데이터 변화 정보를 학습 엔진을 통해 특징으로 가공하여 추출한다.

(3) 통합 분석기

통합 분석기는 특징 추출기가 가공하여 추출한 특징을 특징 추출기로부터 전달받는다. 통합 분석기는 전달받은 특징을 학습 엔진을 통해 새로운 지식으로 가공하여 생성한다. 통합 분석기는 생성한 지식을 컨트롤 프레임워크의 트리거 업데이터에게 전달한다.

다. 컨트롤(CaaS) 프레임워크

컨트롤 프레임워크는 IoT 디바이스의 오류 정보를 확인하여 이벤트 트리거(Trigger)와 모듈을 갱신하고 IoT 디바이스를 제어한다. 컨트롤 프레임워크 내부 모듈들이 수행하는 역할과 기능은 다음과 같다.

(1) 트리거 업데이터

트리거 업데이터는 분석 프레임워크 통합 분석기가 가공하여 생성한 지식을 통합 분석기로부터 전달받는다. 트리거 업데이터는 전달받은 지식을 기반으로 새로이 적용될 이벤트 트리거를 갱신한다. 트리거 업데이터는 갱신한 이벤트 트리거가 모듈에 포함되도록 이를 모듈 업데이터에 전달한다.

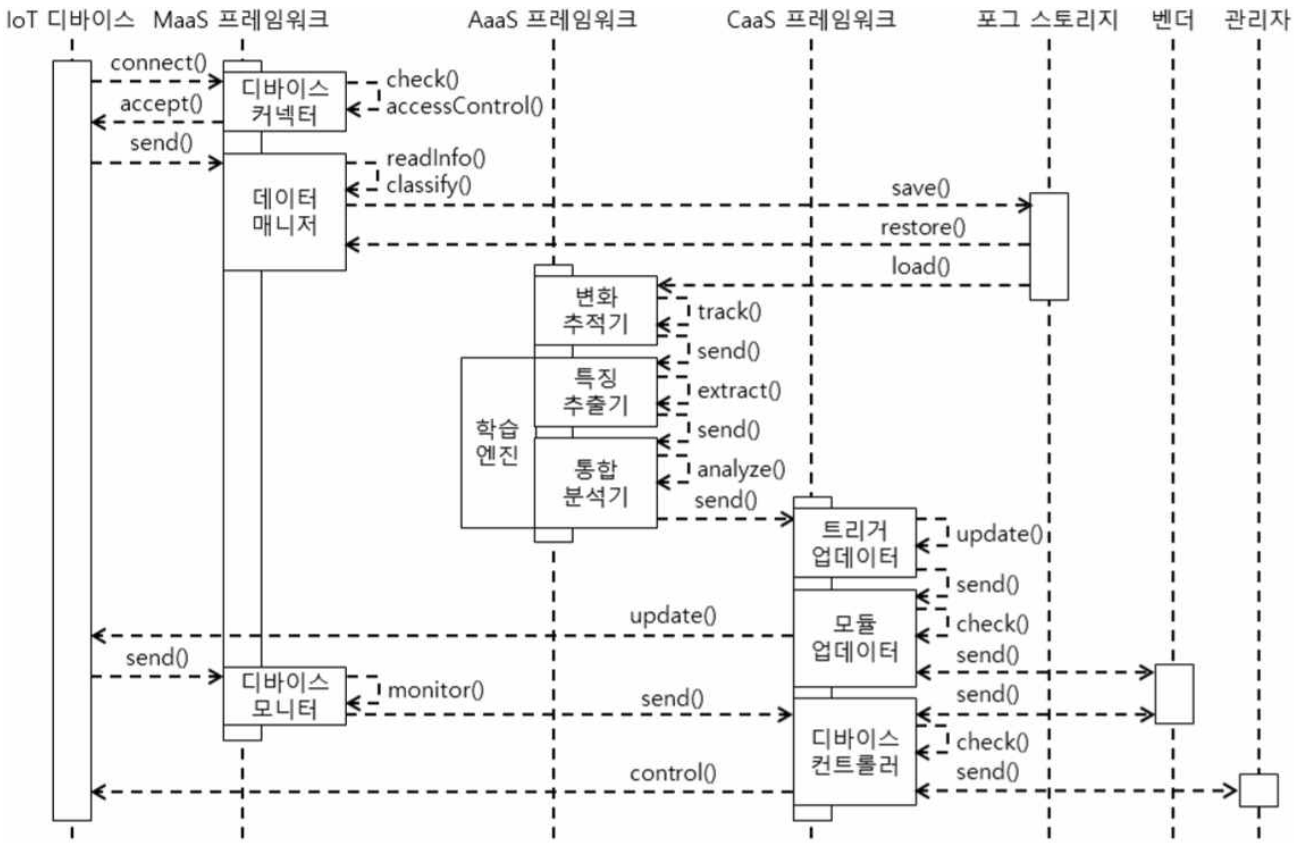


그림 4 제안하는 플랫폼의 함수 플로우

(2) 모듈 업데이터

모듈 업데이터는 트리거 업데이터가 갱신한 트리거를 트리거 업데이터로부터 전달받는다. 모듈 업데이터는 전달받은 트리거를 포함하여 새로이 적용될 모듈을 갱신한다. 모듈 업데이터는 디바이스 컨트롤러가 플랫폼 관리자(Administrator) 또는 벤더(Vendor)에게 보낸 오류 정보를 기반으로 갱신된 모듈을 플랫폼 관리자 또는 벤더로부터 전달받고 이를 IoT 디바이스에 적용하여 모듈을 갱신한다.

(3) 디바이스 컨트롤러

디바이스 컨트롤러는 모니터링 프레임워크 디바이스 모니터가 모니터링한 IoT 디바이스 오류 정보를 디바이스 모니터로부터 전달받는다. 디바이스 컨트롤러는 디바이스 모니터로부터 전달받은 오류 정보의 원인이 플랫폼 내부 설계와 구현에 있는지, 벤더에 의해 제공된 소프트웨어 모듈에 있는지 확인하여 플랫폼 관리자 또는 벤더에게 오류 정보를 보낸다. 또한 디바이스 컨트롤러는 IoT 디바이스를 통해 사용자의 의도를 파악하여 IoT 디바이스를 직접 제어한다.

3. API

앞 절에서 설명한 것과 같이 플랫폼의 서비스 프레임워크와 API가 함께 포그 컴퓨팅 환경을 구성한다. API는 IoT 디바이스 노드와 프레임워크 간 D2F(Device to Framework) API, 프레임워크간 F2F(Framework to Framework) API, IoT 디바이스 노드 간 D2D(Device to Device) API, 그리고 서비스 프레임워크 내부 API 세 유형으로 나누어진다. 이러한 API는 각 서비스 프레임워크의 기능 및 내부 구현에 대한 추상화로 볼 수 있다. 각 영역 내 주요 API의 적용범위, 기능을 나타내는 이름, 출처/목적지(Source/Destination)와 전달/비교 값 등을 포함한 매개변수는 [표 2], [표 3]와 같다.

만약 새로운 IoT 디바이스나 서비스 프레임워크가 MACaaS 플랫폼에 추가되면 이를 위한 별도의 작업 없이 API 추상화가 재사용될 수 있다. 이때 MACaaS 플랫폼 내부 구조에 대한 세부사항을 모르는 채로 IoT 디바이스나 서비스 프레임워크가 추가되기 때문에 API 추상화는 일관성과 확장성을 제공할 수 있다. 본 플랫폼에서 제공하는 API 함수들의 전체 플로우를 도식화하면 [그림 4]와 같다. API 이름, 화살표 방향을 통해 기능과

호출 관계를 확인할 수 있으며 본 논문에서 제안하는 MACaaS 플랫폼의 전체 흐름을 파악할 수 있다.

표 2. 제안하는 플랫폼의 D2F, F2F, D2D API

구분	적용범위	API
D2F	IoT 디바이스 ↔ 디바이스 커넥터	connect()
		accessControl()
		accept()
	IoT 디바이스 ↔ 모듈 업데이터	update()
	IoT 디바이스 ↔ 디바이스 컨트롤러	control()
D2F, F2F, D2D	공통	send()
		receive()

표 3. 제안하는 플랫폼의 서비스 프레임워크 API

구분	적용범위	API
모니터링 프레임워크	디바이스 커넥터	check()
	데이터 매니저	classify()
		readInfo()
		save()
		restore()
	디바이스 모니터	monitor()
분석 프레임워크	변화 추적기	load()
		track()
	특징 추출기	extract()
	통합 분석기	analyze()
컨트롤 프레임워크	트리거 업데이터	update()
	모듈 업데이터	check()
	디바이스 컨트롤러	check()
공통	공통	send()
		receive()

Ⅳ. 평가

본 논문에서 제안하는 MACaaS 플랫폼은 크게 포그 노드와 IoT 디바이스 노드로 나누어지며, 전체 구조를 유지하면서 사용자나 IoT 디바이스의 필요에 따라 프레임워크, IoT 디바이스 등을 쉽게 추가할 수 있어 일관성과 확장성을 지닌다. 본 장에서는 MACaaS 플랫폼에 새로운 IoT 디바이스들을 추가하고 확장하는 것의 용이성을 확인한다. [표 4], [표 5]은 포그 노드와 IoT 디바이스 노드의 구현 환경 명세이다.

표 4. 포그 노드 구현 환경

범주	구분	이름	버전
포그 서버	프레임워크	Node.js	6.12.0
	소프트웨어	Mobius Yellow Turtle	2.0.0
	데이터베이스	MySQL	14.14
	운영체제	Debian (Linux, 64bit)	4.9.2-10
	CPU	Intel i5-4670 3.4GHz	-
포그 스토리지	운영체제	Debian (Linux, 64bit)	4.9.2-10
	NAS	OpenMediaBault (Erasmus)	3.0.86

표 5. IoT 디바이스 노드 구현 환경

범주	구분	이름	버전
고사양 IoT 디바이스	프레임워크	Node.js	6.12.0
	소프트웨어	&Cube Thyme/Lavender	2.0.0
	운영체제	Raspbian (Stretch)	4.9
	하드웨어	Raspberry PI	3
엣지 게이트웨이	프레임워크	Node.js	6.12.0
	소프트웨어	&Cube Thyme/Rosemary	2.0.0
	운영체제	Raspbian (Stretch)	4.9
	하드웨어	Raspberry PI 3	-
저사양 IoT 디바이스	하드웨어	초음파 센서 HC-SR04-P	-
		적외선 센서 HC-SR501	
		카메라 모듈 913-2664	
스마트 폰	운영체제	Android (Nougat)	7.0
	하드웨어	Samsung Galaxy S8	-

본 장에서는 용이성 확인 과정과 결과가 유효함을 보이고자 IoT 표준화 기구인 oneM2M 표준규격을 이용하였으며 oneM2M을 위해 KETI에서 개발한 IoT 플랫폼인 Mobius와 &Cube를 이용하였다. [그림 5]는 저사양 IoT 디바이스로 구성한 스마트 CCTV를 포그 서버에 등록하는 과정을 도식화한 것이다.

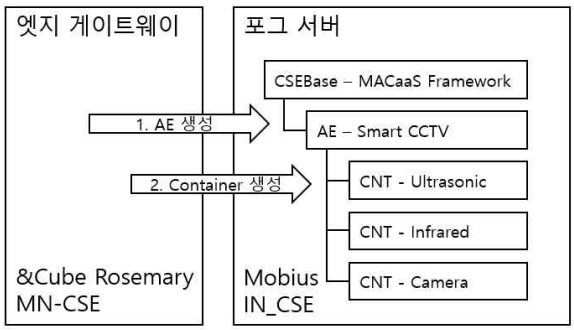
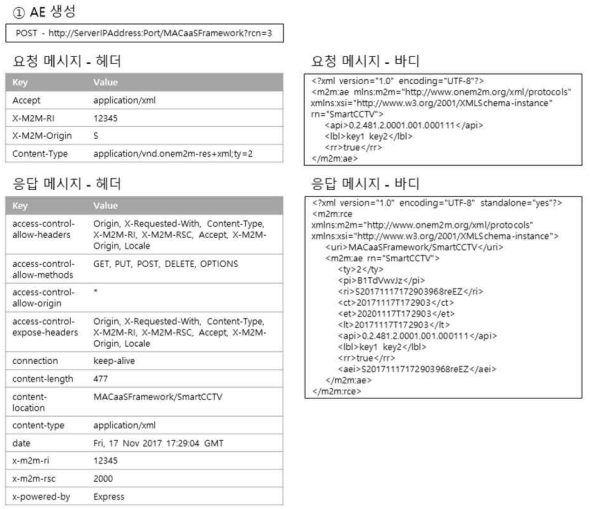


그림 5. 스마트 CCTV 등록 과정

엣지 게이트웨이가 스마트 CCTV에 해당하는 아키텍처 엔티티인 AE를 생성해달라는 요청 메시지를 포그 서버에 전달한다. 포그 서버는 스마트 CCTV AE를 생성하고 엣지 게이트웨이에 응답 메시지를 전달한다. 스마트 CCTV AE가 생성되었으면 엣지 게이트웨이는 초음파 센서, 적외선 센서, 그리고 카메라 모듈에 해당하는 컨테이너(CNT)를 생성해달라는 요청 메시지를 포그 서버에 전달한다. 포그 서버는 앞서 생성한 스마트 CCTV AE 안에 각 센서 모듈에 해당하는 CNT를 생성하고 엣지 게이트웨이에 응답 메시지 를 전달한다. 스마트 CCTV와 센서 모듈들이 모두 등록된 후에 실제 데이터들은 ContentInstance라는 리소스 형태로 CNT 안에 저장된다. [그림 6], [그림 7]은 플랫폼에서 AE와 컨테이너 생성시 주고받는 메시지들이다.


① AE 생성

POST - http://ServerIPAddress:Port/MACaaSFramework?rcn=3

요청 메시지 - 헤더

Key	Value
Accept	application/xml
X-M2M-RI	12345
X-M2M-Origin	S
Content-Type	application/vnd.onem2m-res+xml;ty=2

요청 메시지 - 바디

```
<?xml version="1.0" encoding="UTF-8"?>
<m2m:ae mlns:m2m="http://www.onem2m.org/xml/protocols"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
rn="SmartCCTV">
    <api>0.2.481.2.0001.001.000111</api>
    <lbl>key1 key2</lbl>
    <rr>true</rr>
</m2m:ae>
```

응답 메시지 - 헤더

Key	Value
access-control-allow-headers	Origin, X-Requested-With, Content-Type, X-M2M-RI, X-M2M-RSC, Accept, X-M2M-Origin, Locale
access-control-allow-methods	GET, PUT, POST, DELETE, OPTIONS
access-control-allow-origin	*
access-control-expose-headers	Origin, X-Requested-With, Content-Type, X-M2M-RI, X-M2M-RSC, Accept, X-M2M-Origin, Locale
connection	keep-alive
content-length	477
content-location	MACaaSFramework/SmartCCTV
content-type	application/xml
date	Fri, 17 Nov 2017 17:29:04 GMT
x-m2m-ri	12345
x-m2m-rsc	2000
x-powered-by	Express

응답 메시지 - 바디

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<m2m:rce
xmlns:m2m="http://www.onem2m.org/xml/protocols"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <uri>MACaaSFramework/SmartCCTV</uri>
    <m2m:ae rn="SmartCCTV">
        <ty>2</ty>
        <pi>B1TdVwvJz</pi>
        <ri>S20171117172903968reEZ</ri>
        <ct>20171117T172903</ct>
        <et>20201117T172903</et>
        <lt>20171117T172903</lt>
        <api>0.2.481.2.0001.001.000111</api>
        <lbl>key1 key2</lbl>
        <rr>true</rr>
        <aei>S20171117172903968reEZ</aei>
    </m2m:ae>
</m2m:rce>
```

그림 6. AE 생성 시 요청 메시지와 응답 메시지


② Container 생성

POST - http://ServerIPAddress:Port/MACaaSFramework/SmartCCTV?rcn=3

요청 메시지 - 헤더

Key	Value
Accept	application/xml
X-M2M-RI	12345
X-M2M-Origin	SOrigin
Content-Type	application/vnd.onem2m-res+xml;ty=3

요청 메시지 - 바디

```
<?xml version="1.0" encoding="UTF-8"?>
<m2m:cnt
xmlns:m2m="http://www.onem2m.org/xml/protocols"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
rn="Ultrasonic">
    <lbl>Ultrasonic</lbl>
</m2m:cnt>
```

응답 메시지 - 헤더

Key	Value
connection	keep-alive
content-length	456
content-location	MACaaSFramework/SmartCCTV/Ultrasonic
content-type	application/xml
date	Fri, 17 Nov 2017 23:55:44 GMT
x-m2m-ri	12345
x-m2m-rsc	2001
x-powered-by	Express

응답 메시지 - 바디

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<m2m:cnt
xmlns:m2m="http://www.onem2m.org/xml/protocols"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
rn="Ultrasonic">
    <ty>3</ty>
    <pi>S20171117235429644ppUx</pi>
    <ri>rkzOyh261M</ri>
    <ct>20171117T235543</ct>
    <et>20201117T235543</et>
    <lt>20171117T235543</lt>
    <st>0</st>
    <mni>3153600000</mni>
    <lbl>Ultrasonic</lbl>
    <mbs>3153600000</mbs>
    <mia>31536000</mia>
    <cr>SOrigin</cr>
    <cni>0</cni>
    <cbs>0</cbs>
</m2m:cnt>
```

그림 7. 컨테이너 생성 시 요청 메시지와 응답 메시지

V. 결론

본 논문은 포그 컴퓨팅 환경에서 다양한 IoT 디바이스를 모니터링하고 분석하여 제어하는 서비스를 제공하는 플랫폼을 제안하였다. 기존 IoT를 위한 포그 컴퓨팅 관련 연구들은 포그 컴퓨팅 환경에서 포그 서버의 배치, 특정 서비스를 제공하기 위한 프레임워크에 대한 연구가 주를 이루었다. 이는 사용자나 IoT 디바이스의 필요에 따라 서비스 프레임워크, IoT 디바이스 등을 쉽게 추가할 수 있는 일관성과 확장성을 갖지 못하는 한계를 나타내었다. 따라서 본 논문에서 IoT 디바이스의 상태와 행위를 모니터링하고 IoT 디바이스에 의해 수집되고 생성된 데이터를 저장 및 분석하여 IoT 디바이스를 제어하는 서비스를 제공하는 플랫폼을 제안하였다. 제안한 플랫폼은 API를 사용하여 새로운 응용 프로그램에 쉽게 이식이 가능하고 새로운 서비스를 쉽게 확장할 수 있는 일관되고 통합된 인터페이스를 제공한다. 이를 위해 제안하는 플랫폼이 적용될 영역인 포그 컴퓨팅 도메인과 IoT 디바이스 노드와 프레임워크 간, 프레임워크 간, 그리고 IoT 디바이스 노드 간 API를 정의하였다.

이러한 MACaaS 플랫폼의 구현 환경을 위해 Debian 계열 Linux 기반 운영체제를 사용하는 포그 서버와 포그 스토리지를 구축하였다. 포그 스토리지는 오픈 소스 소프트웨어 NAS인 OpenMediaVault를 이용하였고 포그 서버, 고사양 IoT 디바이스와 엣지 게이트웨이의 소프트웨어는 KETI에서 개발한 oneM2M 표준 기반 IoT 플랫폼을 이용하였다. 또한 사양 IoT 디바이스와 엣지 게이트웨이는 오픈 소스 하드웨어 플랫폼인 Raspberry Pi를 이용하였고 초음파 센서, 적외선 센서, 그리고 카메라 모듈을 엣지 게이트웨이와 결합하여 스마트 CCTV라는 고사양 IoT 디바이스를 구성하였다.

본 논문에서 제안한 MACaaS 플랫폼에 새로운 IoT 디바이스들을 추가하고 확장하는 것의 용이성을 확인하고 그 과정과 결과가 유효함을 보였다. 이를 위해 IoT 표준화 기구인 oneM2M 표준규격을 이용하였으며 oneM2M을 위해 KETI에서 개발한 IoT 플랫폼인 Mobius와 &Cube를 이용하였다. 스마트 CCTV를 포그 서버에 등록하는 과정에서 Mobius 플랫폼이 제공하는 오픈 REST API를 사용하여 쉽게 리소스를 생성할 수 있음을 확인하였다. 또한 리소스를 생성하면 해당 리소스에 접근할 수 있는 API도 자동으로 생성하는 것과 이를 쉽게 수정하는 것이 가능함을 확인하였다. 그러나 MACaaS 플랫폼에 새로운 IoT 디바이스들을 추가하고 확장하는 것의 용이성을 확인하는 제한을 두었다. 따라서 향후 연구에는 함수, 모듈, 서비스 프레임워크 등을 확장하는 추가적인 구현이 필요하며 이를 통한 실효성 입증의 재확인이 요구된다.

REFERENCES

[1] Doukas, Charalampos., and Maglogiannis, Ilias. "Bringing IoT and cloud computing towards pervasive healthcare." *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on. IEEE*. 2012.

[2] Haller, Stephan., Stamatis, Karnouskos., and Christoph, Schroth."The internet of things in an enterprise context." *Future Internet Symposium. Springer*. 2008.

[3] 정희자, "스마트 디바이스 착신정보 중계 기반 손목형 모듈 시스템 설계 및 구현," *스마트미디어저널*, 제5권, 제4호, 131-137쪽, 2016년 12월

[4] 차병래, 최명수, 박선, 김형균, 김용일, 김종원, "IoF-Cloud 기반 분산된 IoT 장비들을 위한 Download Over-the-Air 기능의 개념 설계," *스마트미디어저널*, 제5권, 제4호, 9-17쪽, 2016년 12월

[5] 서희경, "IoT 및 네트워크 관리 지원을 위한 컴포

넌트 아키텍처 개발," *스마트미디어저널*, 제6권, 제2호, 42-49쪽, 2017년 6월

[6] Yi, Shanhe., Hao, Zijiang., Qin, Zhengrui., and Li, Qun. "Fog computing: Platform and applications." *Hot Topics in Web Systems and Technologies (HotWeb), 2015 Third IEEE Workshop on. IEEE*. 2015.

[7] Bonomi, Flavio., Milito, Rodolfo., Zhu, Jiang., and Addepalli, Sateesh. "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] Stojmenovic, Ivan. "Fog computing: A cloud to the ground support for smart things and machine-to-machine networks." *Telecommunication Networks and Applications Conference (ATNAC)*, 2014 Australasian. IEEE. 2014.

[9] Mell, Peter., and Tim, Grance. "The NIST definition of cloud computing." 2011.

[10] Dsouza, Clinton., Ahn, Gail-Joon., and Taguinod, Marthony."Policy-driven security management for fog computing: Preliminary framework and a case study." *Information Reuse and Integration (IRI), 2014 IEEE 15th International Conference on. IEEE*. 2014.

[11] Jain, Akshay., and Singhal, Priyank. "Fog computing: Driving force behind the emergence of edge computing." *System Modeling & Advancement in Research Trends (SMART), International Conference. IEEE*. 2016.

[12] Lee, Gilsoo., Saad, Walid., and Bennis, Mehdi. "An online secretary framework for fog network formation with minimal latency." *Communications (ICC), 2017 IEEE International Conference on. IEEE*. 2017

[13] Nishio, Takayuki., Shinkuma, Ryoichi., Takahashi, Tatsuro., and Mandayam, Narayan B. "Service-oriented heterogeneous resource sharing for optimizing service latency in mobile cloud." *Proceedings of the first international workshop on Mobile cloud computing & networking*. ACM. 2013.

[14] Willis, Dale., Dasgupta, Arkodeb., and Banerjee, Suman. "Paradrop: a multi-tenant platform to dynamically install third party services on wireless gateways." *Proceedings of the 9th ACM workshop on Mobility in the evolving internet architecture. ACM*. 2014.

저 자 소 개

이한솔

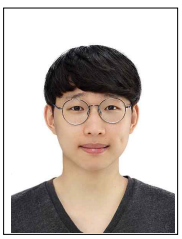
2017년 숭실대학교 컴퓨터학부 학사 졸업.
2019년 숭실대학교 컴퓨터학과 석사 졸업

<주관심분야 : 시스템 소프트웨어, 운영체제>

최정우

2018년 숭실대학교 컴퓨터학부 학사 졸업.
2018년~현재 숭실대학교 컴퓨터학과 석사 재학

<주관심분야 : 시스템 소프트웨어, 운영체제>



변기범

2016년 숭실대학교 컴퓨터학부 학사 졸업.
2018년 숭실대학교 컴퓨터학과 석사 졸업

<주관심분야 : 시스템 소프트웨어, 운영체제>

홍지만(종신회원)

2003년 서울대학교 컴퓨터공학과 박사 졸업
2004년~2007년 광운대학교 컴퓨터공학과 교수.
2007년~현재 숭실대학교 컴퓨터학부 교수

<주관심분야 : 시스템 소프트웨어, 운영체제>