https://doi.org/10.7840/kics.2017.42.4.871

선택적 프로토콜 기반의 사물인터넷 데이터 수집 플랫폼

오 형 석◆, 김 동 휘*, 전 현 식*, 박 현 주°

IoT Data Collection Platform Based on Selective Protocols

Hyeong-Seok Oh◆, Dong-Hwi Kim*, Hyun-Sig Jeon*, Hyun-Ju Park°

요 약

사물 인터넷에 대한 관심이 증가되고 사물인터넷 시장 규모가 증가함에 따라 사물인터넷 관련 기기와 프로토콜이 발전하고 있다. 상호 운용성에 대한 문제가 발생됨에 따라 사물인터넷 디바이스뿐만 아니라 시스템 사업자와 서비스 및 애플리케이션 시장과 사물인터넷 표준 정립은 혼란을 겪게 됐고, 사물과 사물 간의 통신은 서로 다른 플랫폼에 의해 방해되고 있다. 현재 다양한 사물인터넷 플랫폼 개발과 이기종 플랫폼과의 상호 운용성을 위한 연구가 진행되고 있지만, 여전히 각각의 플랫폼에서 사용되는 프로토콜은 제한적이며 일반화된 구조로 설계되어 있지 않다. 본 논문에서는 사물인터넷 시장에서 대표적으로 사용되는 HTTP, CoAP, MQTT 프로토콜을 분석하여 프로토콜을 선택적으로 사용하는 서비스와 플랫폼 사용의 일반화를 위해 RESTful API를 적용한 자원을 설계하고, 수집된 데이터의 빠른 처리와 안전성을 위한 Database 모델링 과정을 진행하여 사물인터넷 데이터 수집 플랫폼을 구현했다. 이러한 과정을 통해 서로 다른 프로토콜로 통신하는 디바이스들이 한 플랫폼에서 연동되며, 다양한 프로토콜을 적용할 수 있는 일반화된 선택적 프로토콜 기반의 사물인터넷 데이터 수집 플랫폼을 제안한다.

Key Words : IoT, HTTP, CoAP, MQTT, RESTful API Platform

ABSTRACT

As the interest of things to the Internet increases, the market of IoT grows larger and devices and protocols related with IoT are evolving.As these IoT devices and protocols evolve, There was a problem with interoperability. As a result, market and standard of IoT are confused, and communication between objects and objects is being hindered by different platforms. Currently, various IoT platforms are being developed and interoperability with heterogeneous platforms is under study, but the protocols used on each platform are limited and designed to have no generalized structure. Therefore, in this paper, we analyze services of HTTP, CoAP, and MQTT protocols, which are typical in the Internet market, and design services that enable selective protocol communication. We also design resources that apply the RESTful API to generalize platform usage. and We implemented the platform through database modeling for quick processing and safety of the collected data. Through this process, devices communicating with different protocols can be interworked on one platform and We propose a "generalized selective protocol based Internet object data collecting platform" that can be applied to various protocols.

◆ First Author : Hanbat national University Department of radio wave engineering, wellstone@hanbat.ac.kr, 학생회원
° Corresponding Author : Hanbat national University Department of radio wave engineering, phj@hanbat.ac.kr, 정회원
* Hanbat national University
논문번호 : KICS2016-12-369, Received December 1, 2016; Revised March 16, 2017; Accepted March 24, 2017

Ⅰ. 서 론

사물인터넷에 대한 관심이 증가하면서 사물인터넷 기기 대수가 증가하고 있고, 사물인터넷 시장 가치 또한 높게 평가받고 있다. 세계적인 시장조사기관 가트너에 의하면 2020년에는 208억 대의 '인터넷 연결 기기'가 무선 인터넷을 기반으로 연결 될 것으로 예상하며[1], 다른 조사 기관 및 전문가들은 50억 대에서 500억 대까지 연결될 것으로 예상한다[2].

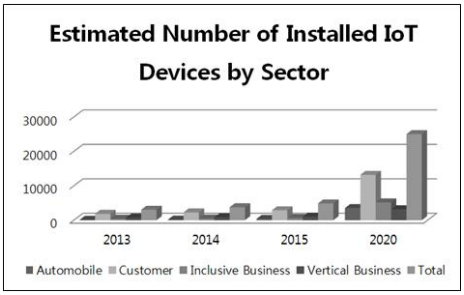

그림 1. 분야별로 설치된 IoT 장치의 추정 된 수
Fig. 1. Estimated Number of Installed IoT Devices by Sector

다양한 사물 인터넷 기기들이 급증함에 따라 국내·외 단체와 협회에서는 사물인터넷 플랫폼에 관한 연구가 활발히 진행되고 있다. 또한 국가간의 새로운 플랫폼의 등장으로 다양한 통신 프로토콜이 생겼다. 이는 서로 다른 통신 규약으로 인해 사물인터넷 플랫폼 시장을 혼란스럽게 만드는 문제가 발생하게 된다. 따라서 이러한 규약과 급증하는 사물인터넷 플랫폼은 진보되어야 한다. 이 점을 착안하여 본 논문에서는 안정적인 사물인터넷 플랫폼과 다양한 프로토콜을 수용할 수 있는 일반화된 플랫폼 연구와 확장성에 용이한 선택적 프로토콜 기반의 사물인터넷 데이터 수집 플랫폼을 제안한다.

Ⅱ. 관련 연구

2.1 CoAP

CoAP(Constrained Application Protocol)은 M2M (Machine-to-Machine) 및 IoT(Internet of Things) 환경에서 센서 및 각종 디바이스 간에 상호 통신할 수 있는 경량 프로토콜로서, IETF(Internet Engineering Task Force) 워킹 그룹 드래프트 18에서 RFC 7252 표준으로 채택되었다[3]. 부가 확장 및 보안 부분은 표준화가 이루어지고 있으며, 현재 oneM2M, ETSI(European Telecommunications Standards Institute), OMA (Open Mobile Alliance) 등의 표준화 단체에서는 CoAP을 채택한 표준을 개발 중이다.

CoAP은 저전력, 작은 메모리와 낮은 처리 성능을 가진 디바이스들을 표준적인 인터넷 환경에서 사용하기 위한 목적으로 개발된 웹 전송 프로토콜로서 차세대 센서 접속 프로토콜로 많이 사용될 것으로 예상된다[4].

CoAP은 HTTP(Hyper Text Transfer Protocol)보다 비교적 경량이면서 Publish/Subscribe 통신 모델까지 커버할 수 있으며, RESTful(Representational State Transfer ful) 구조를 지원하여 CoAP을 사용하는 디바이스들이 기존 웹 서비스 구조와 쉽게 결합하여 웹 기반 사물인터넷 서비스를 구성할 수 있다.

CoAP은 데이터그램 방식으로 트랜스포트 계층 위에서 비동기적으로 데이터를 전송한다. 따라서 데이터 전송에 있어서 신뢰성을 보장하기 위한 재전송 및 타이머 관리 옵션을 포함하고 있으며, 신뢰성과 비신뢰성 메시지 전송을 제공하기 위해 확인형(confirmable), 비확인형(non- confirmable), 승인(acknowledgement), 리셋(reset)의 4가지 메시지 타입을 정의한다. 신뢰성 메시지 전달은 메시지 요청(request)에 메시지 ID를 포함한 확인형 메시지를 전송하여 같은 메시지 ID를 포함한 응답(response)으로 승인 메시지를 받는 절차를 진행하고, 비신뢰성 메시지 전달은 비확인형 메시지로 요청하고 응답은 받지 않는다[4].

GET 요청 시 응답은 piggy-back 형태로 응답에 GET 요청에 대한 데이터 담아 보내는 방법과 일단 승인 메시지만 보낸 후 데이터가 준비되면 데이터를 담아 응답하는 두 가지 형태가 있다. 전자는 메시지 ID를 통해, 후자는 토큰을 통해 신뢰성을 보장한다.

보안을 위해 UDP 계층과 CoAP 계층 사이에 DTLS(Datagram Transport Layer Security)를 사용할 수 있다. DTLS는 UDP 전송 계층 프로토콜을 이용한 실시간 데이터 처리가 증가함에 따라 이를 보호하기 위한 DTLS가 2006년 IETF에 의해 추가적으로 표준으로 제정되었다[5].

2.2 MQTT

MQTT는 IBM에서 개발된 프로토콜로서 저대역폭, 저전력 프로토콜로 불안정한 네트워크 환경에서 IoT 기기간의 경량 메시지 통신을 목적으로 설계된 메시지 전달 프로토콜이다[6]. 2013년에는 OASIS (Advancing open standards for the Information society)에서 사물인터넷을 위한 메시지 프로토콜로

선정되었고 전력 소모 측면에서 효율적인 프로토콜임이 검증되었다. 또한, 이 프로토콜은 다양한 사물인터넷 기기간의 통신과 이기종 플랫폼간의 개발이 가능하다. MQTT는 이미 다양한 임베디드 시스템에서 폭넓게 사용되고 있다. 가전기기, 도시, 산업, 의료 등 다양한 영역에서의 정보 수집이 가능하며 역으로 수집된 정보를 통해 소형 디바이스 제어가 가능하다. 이 프로토콜은 모바일 환경에서도 사용되며 대표적으로 Facebook 메신저에서 이 프로토콜을 사용하고 있다. 이처럼 MQTT는 제한적인 통신 대역폭, 배터리 용량, 성능 등에서 모바일 환경과 원격 검침 등의 모니터링을 목적으로 많이 사용되는 프로토콜이다.

MQTT는 Publish/Subscribe 특징을 가지고 있는 프로토콜로서 Publisher, Subscriber, Broker라는 세 개체간의 통신을 통해 데이터를 교환한다. Publisher와 Subscriber 간의 통신을 중계하는 Broker는 단일 프로토콜을 지원하는 경량의 Broker에서부터 다양한 표준 통신 프로토콜을 지원하는 상호 운용적인 Broker까지 활발한 개발이 이루어지고 있다. 이 개체들 간의 통신은 Topic이라는 문자열을 통해 송수신하며, 슬래쉬(/)를 통해 계층적 구조를 표현하여 다양한 IoT 기기를 효율적으로 관리할 수 있다.

메시지 전송에 대한 신뢰성을 보장하기 위해 MQTT에서는 3 단계의 QoS Level을 지원한다. QoS 0은 메시지를 한 번만 전달하고 전달한 메시지에 대한 응답을 확인하지 않는다. 중요도가 높은 메시지에 대한 유실은 책임지지 않기 때문에, 센싱 값과 같이 중요도가 낮은 메시지에 대해서 효율적으로 사용할 수 있다. QoS 1는 최소 한 번 이상에 대한 전송과 그에 대한 응답을 확인한다. QoS 0에 비해 메시지 유실 가능성은 낮지만, PUBLISH에 대한 응답인 PUBACK가 유실될 경우 메시지 중복에 대한 단점을 가진다. QoS 2는 4-way handshake를 통해 정확한 한 번의 메시지 송수신을 한다. 메시지 유실 가능성은 없지만 종단 간의 지연이 발생하기 때문에, 중요한 데이터를 교환할 때 사용하기 적합하다[7].

2.3 REST API

REST는 웹 창시자중 한 사람인 Roy Fielding가 소개한 아키텍처로 리소스, 메소드, 메시지 3가지 요소로 구성된다. REST는 HTTP 표준만 따르면 특정 언어나 플랫폼에 종속 받지 않고 사용 가능한 형태의 구조로 모든 자원은 고유의 식별자인 URI(Uniform Resource Identifier)가 있으며, 이런 REST 아키텍처 스타일에 따라 정의되고 이용되는 서비스나 응용 프로그램을 RESTful 웹 서비스라고 한다.

RESTful 웹 서비스는 리소스를 등록하고 저장해두는 중간 매개체 없이 리소스 제공자가 직접 리소스 요청자에게 제공하는 ROA(Resource Oriented Architecture)방식으로 사용자나 클라이언트의 컨텍스트를 저장하지 않는 형태로 서버에 들어오는 메시지로만 처리하기 때문에 구현이 단순해진다.

REST API 설계의 기본원칙은 리소스는 명사를 사용하여 직관적이고 단순하게 설계하며 리소스당 2개의 기본 URI를 사용하여 자원의 집합이나, 한 개의 자원을 나타내야 한다. 또한 메소드는 HTTP의 기본 메소드인 POST(CREATE), GET(READ), PUT(UPDATE), DELETE(DELETE) 사용하여 리소스의 추가, 조회, 갱신, 삭제 기능을 수행한다[8].

Ⅲ. 제안하는 데이터 수집 플랫폼

본 절에서는 안정적인 사물인터넷 플랫폼과 다양한 프로토콜을 수용할 수 있는 일반화된 구조로 선택적 프로토콜 기반의 사물인터넷 데이터 수집 플랫폼을 설명한다. 제안하는 선택적 프로토콜 기반의 플랫폼에서 제공하는 프로토콜은 현재 많이 사용되는 표준 프로토콜인 HTTP, CoAP, MQTT를 제공한다. CoAP과 MQTT는 사물인터넷에서 사용하기 적합한 프로토콜로서 현재 다양한 곳에서 사용되고 있는 프로토콜이다. 이러한 프로토콜은 Data Collection & Control Server(CnC)를 통해 통신이 가능하도록 설계했고, 다양한 프로토콜과 사물인터넷 기기를 수용 및 제어하기 위한 일반화된 구조를 설계했다.

제안하는 플랫폼에서는 다양한 프로토콜을 사용하

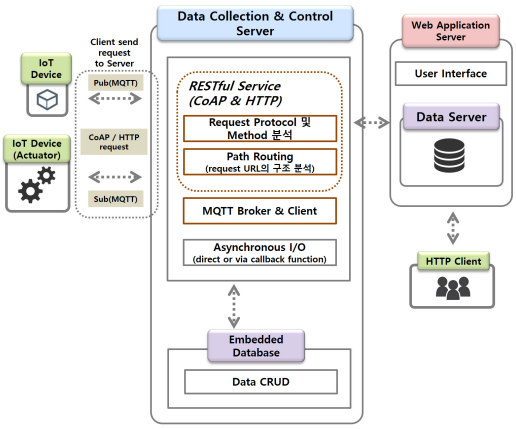

그림 2. 솔루션 구성도
Fig. 2. Structure of Proposed Solution

기 위한 IoT Device, 데이터 수집 및 IoT Device를 제어하기 위한 CnC, 데이터 전달의 신뢰성을 위한 캐시 용도의 Embedded Database , 데이터가 최종적으로 저장되는 메인 Database, 사용자에게 다양한 인터페이스 제공과 모니터링을 위한 Web Application Server(WAS)로 구성된다. 제안하는 데이터 수집 솔루션의 전체 구성도는 그림 2와 같다.

3.1 Database 모델링

Database는 플랫폼의 핵심인 CnC의 Embedded Database에는 사용하는 CnC, Device, Sensor, Actuator, Model에 관한 정보와 사용자의 명령 값, Sensor의 센싱 값 등이 저장되며 그림 3과 같이 모델링했다.

각 테이블은 식별을 위한 id를 기본키로 설정하고 id, 이름, 상태, 수정시간, 설명 등의 기본정보를 가진다. device_sensor, device_actuator 테이블은 Device에 연결된 Sensor나 Actuator의 동작 값을 히스토리화하기 위한 테이블이다. Model과 Model_ Type의 경우 Model_ Type이 온도센서라면 Model은 RT- 02K, RT-03K, PT-03 등이 있다.

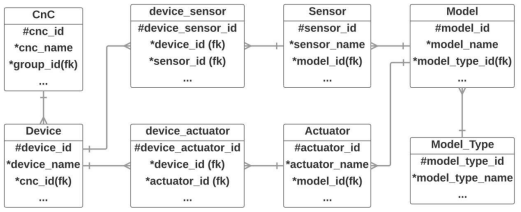

그림 3. 데이터베이스 모델링
Fig. 3. Database Modeling

표 1. Database table description
Table 1. Database table description

table name	description
CnC	CnC Store basic information, location, and ip address information
Device	device Store basic information, CnC id, battery, ip address, protocol
Sensor	Sensor Store basic information Device id
Actuator	Actuator Store basic information, Device id
device_sensor	device_sensor Store basic information, device id, sensor id, sensing value
device_actuator	device_actuator Store basic information, device id, actuator id, actuating value
Model	Model Store basic information
Model_Type	Model_Type Store basic information

WAS의 경우 사용자 정보나 그룹생성을 위한 정보를 위해 추가로 Group, Account, Account_Group을 모델링했다. Account 테이블에는 사용자의 정보가 저장되며, 등록된 사용자들은 Group에 포함되어야 한다. 사용자는 한 개 이상의 Group에 속할 수 있고, Group은 한 명 이상의 사용자를 가져야 한다. 이때 Account와 Group은 N:M 관계를 가지게 되며, 이를 정규화하여 Account_Group 테이블을 정의했다.

3.2 REST API 설계

본 절에서는 WAS와 CnC에서 공통으로 사용하는 주요 API를 설명한다. CoAP과 HTTP는 REST 기반으로 각 요청에 대한 RESTful Service를 표 2와 같이 설계했다. MQTT의 경우 REST API를 지원하지 않지만, 일반화를 위한 공통된 API를 사용하기 위해서 표 2와 동일하게 Topic을 설계했고, CRUD 기능을 지원하기 위해 MQTT의 Payload에 method 속성을 추가하여 전송하도록 설계했다. 리소스는 디바이스에 대한 정보를 저장하는 devices, 센서에 대한 정보를 저장하는 sensors, 엑츄에이터에 대한 정보를 저장하는 actuators, 각 디바이스에서 들어오는 엑츄에이터의 상태 정보 및 명령을 저장하는 deviceActuators, 각 디바이스에서 들어오는 센싱 데이터를 저장하기 위한 deviceSensors로 구성했다. 메소드는 CnC의 특성상 DELETE 요청을 제외한 나머지를 구현했다. API /api/v1/devices/{id}/ actuators 의 경우는 device의 특성상 통신이 바로 불가능할 경우를 대비한 API로 device가 통신이 가능 할 때 자신에게 온 명령을 다음 API를 통해 요청한다.

표 2. platform REST API
Table 2. platform REST API

main	Method	REST API
device Actuator	POST	/api/v1/deviceActuators
	GET	/api/v1/deviceActuators
	GET	/api/v1/deviceActuators/{id}
	PUT	/api/v1/deviceActuators/{id}
device Sensor	POST	/api/v1/deviceSensors
	GET	/api/v1/deviceSensors
	GET	/api/v1/deviceSensors/{id}
devices	GET	/api/v1/devices/{id}/actuators

3.3 데이터 흐름 처리

3.3.1 Sensing 관련 데이터 처리 흐름

그림 4는 특정 프로토콜을 사용하는 디바이스가 CnC에게 측정된 센서의 센싱 값을 전달하고, 수신한

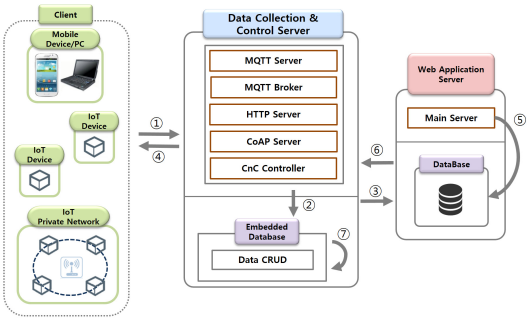

그림 4. 센싱 데이터 처리 흐름
Fig. 4. Data Processing Flow about Sensing

센싱 값을 WAS에게 전달하여 Database에 저장하는 과정에 대한 흐름도이다.

그림 4의 흐름을 요약하면 다음과 같다.

① 특정 프로토콜을 사용하는 디바이스는 CnC에게 센싱 값을 전달한다.

② CnC는 디바이스로부터 전달 받은 센싱 값을 CnC의 Embedded Database에 저장한다. CnC의 Embedded Database는 캐시로 사용되는 임시 저장소이다.

③ CnC에서는 전달받은 센싱 값을 WAS에게 전달한다.

④ 데이터 전송 완료를 응답한다.

⑤ REST API를 통해 전달받은 센싱 값을 WAS의 Database에 저장한다.

⑥ WAS는 데이터 저장에 대한 성공 여부를 CnC에게 응답한다.

⑦ WAS로부터의 응답이 성공적일 경우, CnC의 Embedded Database에 저장되어 있던 데이터를 삭제한다. 실패 시 Embedded Database에 데이터를 재전송 한다.

3.3.2 Actuating 관련 명령 처리 흐름

그림 5는 사용자가 특정 프로토콜을 사용하는 디바이스에게 명령을 전송하기 위한 처리 흐름으로, 사용자로부터 수신된 명령이 WAS와 CnC를 거쳐 디바이스에게 전달되는 과정에 대한 흐름도이다.

그림 5는 사용자가 디바이스에게 어떠한 명령을 전송했을 때의 흐름을 표현한다.

① Client는 WAS에게 특정 명령을 전송한다.

② Client로부터 받은 명령을 WAS의 Database에 저장한다.

③ 저장한 명령을 CnC에게 전달한다.

④ CnC는 전달받은 명령을 CnC의 Embedded Database에 저장한다.

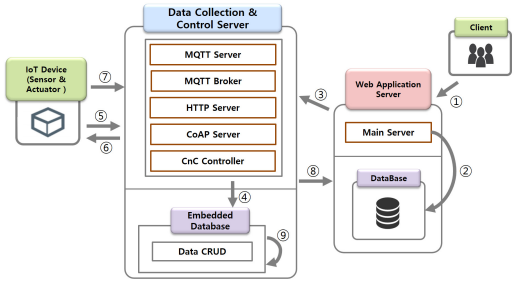

그림 5. 엑츄에이팅 명령 처리 흐름
Fig. 5. Command Processing Flow about Actuating

⑤ IoT Device는 CnC에게 요청을 통해 자신에게 온 명령이 있는지 주기적으로 확인한다.

⑥ CnC는 해당 IoT Device에 대한 명령이 있을 경우, 명령 데이터를 전송한다.

⑦ IoT Device는 명령에 대한 실행이 완료되면 그에 대한 결과 값을 CnC에게 전송한다.

⑧ CnC는 ⑦에 대한 명령이 들어오면 WAS에게 전달한다.

⑨ CnC는 ⑧과 같이 명령을 전달한 후 CnC의 Embedded Database에 저장되어 있던 명령을 삭제한다. 전송 실패 시 Embedded Database는 삭제하지 않는다.

3.4 CnC 구조

플랫폼 구조를 살펴보면 CnC는 Node.js 기반으로 모듈(cncController, service, serviceController, dto, Repositrory)을 구성했다.

cncController 모듈은 Database 연동, Domain 생성, 요청 처리 등의 CnC의 전체적인 동작을 제어하는

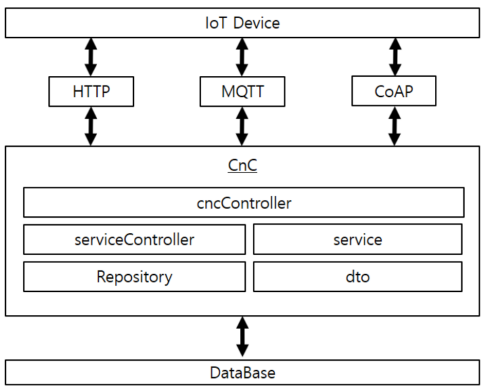

그림 6. CnC 구조
Fig. 6. CnC Structure

모듈이다. service 모듈은 각기 다른 프로토콜에 대한 처리와 API를 등록하는 모듈이고, serviceController 모듈은 각기 다른 프로토콜의 데이터를 분석하여 dto 객체를 생성하거나 cncController 모듈로 요청을 전달하는 모듈이다. dto 모듈은 Database 모델링에 따른 데이터 속성을 정의하는 객체이고 dtoRouter 모듈은 어떤 dto 모듈을 사용할지 판단하여 dto 객체를 만드는 모듈이다. 마지막으로 Repository 모듈은 Database와 연동하여 쿼리를 수행하는 모듈이다.

플랫폼은 HTTP, CoAP, MQTT를 선택적으로 사용할 수 있도록 service 모듈에서 각각 다른 포트를 통해 요청을 처리한다. 따라서 프로토콜이 사용하는 포트를 통해서 데이터를 송수신한다. 또한 모든 데이터는 프로토콜과 관계없이 JSON 형식으로 전달되고 Database에 데이터를 저장하여 프로토콜에 상관없이 IoT Device가 요청을 처리할 수 있도록 구현했다.

3.5 요청에 대한 CnC 내부 처리 흐름

CnC의 요청은 REST API 기반으로 POST / GET / PUT 요청을 처리할 수 있도록 구현했다. GET과 PUT 요청에 대한 처리 흐름은 그림 7과 같으며, POST 요청에 대한 처리 흐름은 그림 8과 같다.

IoT Device의 Actuator가 스마트전구라고 가정하면 그림 5의 ④번까지의 과정은 POST 요청을 통해 전구 점등이라는 명령을 Embedded DataBase에 저장한다. 이후 IoT Device는 자신에게 온 명령을 service 모듈에게 제공하는 API와 GET으로 요청한다. 만약 IoT Device의 device_id가 d1이라면 /api/v1/devices/d1/actuators이라는 API로 요청한다. service 모듈은 device_id인 d1을 포함하여 serviceController 모듈에 GET 요청을 전달하고 serviceController 모듈은 cncController 모듈에게

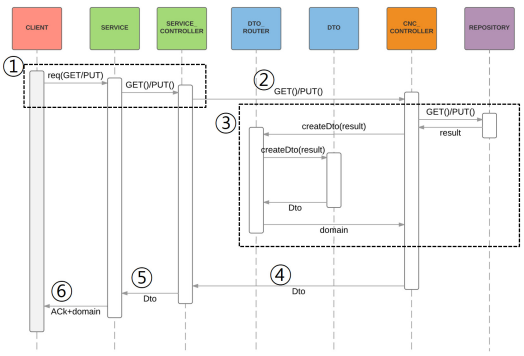

그림. 7. GET/PUT 요청
Fig. 7. GET/PUT Request

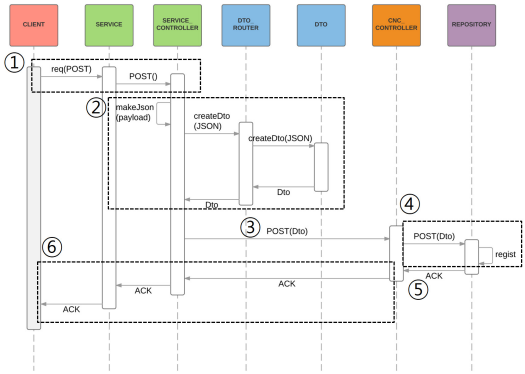

그림 8. POST 요청
Fig. 8. POST Request

device_id가 d1인 IoT Device에 대한 명령을 보내달라는 요청이 왔음을 알려준다. 요청을 전달 받은 cncController 모듈은 Repository 모듈을 통해 device_id가 d1인 IoT Device에 관한 모든 명령을 device_actuator 테이블에서 찾아 dto 객체로 만든다. 이 과정에서 앞서 말한 전구 점등이라는 명령에 대한 정보가 포함된 dto가 생성된다. 이 dto는 cncController 모듈을 통해 serviceController 모듈에게 전달되고 serviceController 모듈은 해당 dto를 JSON 형식으로 만들어 service 모듈에 전달한다. 만약 전구의 actuatorId가 a1이고 전구 점등이라는 명령이 ON이라면 {deviceId: d1, actuatorId: a1, actuatingValue: ON}이라는 JSON 형식의 데이터를 만든다. 이 과정이 완료되면 service 모듈은 IoT Device로 다음의 데이터를 전달한다.

예를 들어 제안하는 플랫폼에서 Clinet를 온도 센서가 장착된 IoT Device라 가정한다. Client는 온도 센서의 센싱값을 특정 프로토콜로 Service 모듈에게 전송한다. 이 때 Client는 /api/v1/devcieSenosrs라는 API로 POST 요청를 통해 Service 모듈에게 전달한다. Service 모듈에서는 Client가 사용하는 해당 프로토콜의 Service 모듈로 POST 요청을 수신하고, 이를 serviceController 모듈에게 전달한다. serviceController 모듈은 특정 프로토콜의 요청에 대한 특정 동작을 Routing하기 위한 모듈이다. 수집된 센싱 값은 dto 모듈을 통해 JSON 형식의 dto 객체를 만든다. 이 dto 객체는 제안하는 플랫폼에서 지정한 형식으로, 수신한 센싱 값과 device_id, sensor_id 등이 담긴다. 이렇게 만들어진 dto 객체는 cncController 모듈에게 전달된다. cncController 모듈은 Repository 모듈에게 dto 객체를 전달하고, Repository 모듈은 전

달받은 dto 객체를 Embedded Database에 저장한다. 저장이 성공적이라면 그에 대한 응답은 cncController 모듈을 거쳐 Client에게 전달되고, 잘못된 형식의 dto 객체일 경우 Error를 반환한다.

Ⅳ. 결 론

현재 사물인터넷 기기와 플랫폼의 개발이 급증함에 따라 기업들은 자체 사물인터넷 프로토콜을 제시하고 있다. 또한 이러한 기업들이 국내·외 단체 및 협회에서 개발한 국제 표준 프로토콜을 사용하더라도 각각의 표준 프로토콜간 통신 호환성 문제는 여전히 남아있다. 이로 인해 사물인터넷 시장은 혼란이 일어나고 사물인터넷 관련 연구들은 지체될 것을 예상한다. 본 연구에서는 현재 많이 사용되는 프로토콜인 HTTP, CoAP, MQTT 프로토콜을 선택적으로 사용하며 설계된 API를 통해 데이터를 수집하거나 디바이스에게 명령을 전달할 수 있고, Embedded Database를 통해 데이터 캐싱 기능을 추가하여 데이터 전송시 분실되는 데이터를 플랫폼의 Embedded Database에 저장하여 WAS로 안전하게 전달될 때까지 데이터를 보존하여 데이터 전송의 안정성 및 신뢰성을 보장하는 플랫폼을 구현했다. 결과적으로 다양한 프로토콜을 수용하면서 안정적인 데이터 수집이 가능한 플랫폼을 통해 사물인터넷 발전에 이바지할 것을 예상한다.

향후 연구로는 현재 CnC에서 Embedded Database로 사용 중인 PostgreSQL을 NoSQL 데이터베이스인 Redis 변경하여 Scale Out과 Scale Up에 제한적인 RDMBS의 단점을 보완하고 사물인터넷의 방대한 데이터를 분산시키는 data sharding을 지원하고자 한다. 또한 Redis는 In memory Database로써 다양한 데이터 구조체를 지원함과 동시에 더 빠른 응답을 제공함으로써[9] 효율적인 플랫폼을 구성할 계획이다.

References

[1] Gartner Inc., *Gartner Says 6.4 Billion Connected "Things" Will Be in Use in 2016, Up 30 Percent From 2015*(2015), Retrieved March, 3, 2016, from http://www.gartner.com/newsroom/id/3165317

[2] D. Evans, *The Internet of Things(2011),* Retrieved March, 3, 2016, from http://www.cisco.com

[3] Z. Shelby, B. Frank, and D. Sturek, *Constrained Application Protocol (CoAP),* RFC 7252, Jun. 2014.

[4] S. K. Ko, I. K. Park, S. C. Soon, and B. T. Lee "Trends of IETF CoAP based sensor connection protocol technology," *Electron. and Telecommun.*, vol. 28, no. 6, Dec. 2013.

[5] H. Kwon and N. Kang, "Analysis on energy consumption required for building DTLS session between lightweight devices in internet of things," *J. KICS*, vol. 40, no. 08, pp. 1589-1590, 2015.

[6] IBM, *The MQTT protocol,* Retrieved Nov., 7, 2014, from http://www.mqtt.org.

[7] S. Lee, H. Kim, and H. Ju, "Design of the high-level architecture of mobile integration SNS gateway and the MQTT based push notification protocol," *J. KICS*, vol. 38B, no. 05, pp. 344-354, 2013.

[8] S. Kim and H. Kim, "API selection and automatic open API composition method based on REST protocol," *J. KICS*, vol. 38C, no. 07, pp. 587-594, 2013.

[9] D. H. Cho, *In memory dictionary Redis Introduce*, Retrieved Apr, 22 2016, from http://bcho.tistory.com/654

오 형 석 (Hyeong-Seok Oh)

2015년 2월 : 한밭대학교 전파공학과(공학사)

2015년 9월~현재 : 한밭대학교 전파공학과 석사과정

<관심분야> 사물인터넷, 시스템 프로그래밍, IoT 서버, 데이터베이스

김 동 휘 (Dong-Hwi Kim)

2015년 2월 : 한밭대학교 전파공학과(공학사)
2015년 9월~현재 : 한밭대학교 전파공학과 석사과정
<관심분야> 사물인터넷, 시스템 프로그래밍, 데이터베이스

전 현 식 (Hyun-Sig Jeon)

2005년 2월 : 한밭대학교 전파공학과(석사과정)
2011년 2월 : 한밭대학교 전파공학과 (박사과정)
<관심분야> 데이터베이스, 공간 데이터베이스, 위치 추정 알고리즘, ESL etc.

박 현 주 (Hyun-Ju Park)

1990년 2월 : 서울시립대학교 전산통계학과(공학사)
1992년 2월 : 서울대학교 전산과학과(공학석사)
1997년 2월 : 서울대학교 전산과학과(공학박사)
1998년 4월~2000년 3월 : 대전산업대학교 정보통신공학과 전임강사
2000년 4월~현재 : 국립 한밭대학교 정보통신공학과 교수
<관심분야> 프로그래밍 언어, 운영체제, 데이터베이스