

논문 2015-10-15

CoAP 기반 클라우드 환경 IoT 구조 설계 및 구현

(Design and Implementation of CoAP based Cloud-IoT Architecture)

박 영 기, 양 현 식, 김 영 한*

(Young-Ki Park, Hyun-Sik Yang, Young-Han Kim)

Abstract : In the IoT(Internet of Things) environment, methods that user can access sensor node directly to collect sensing data or manage sensor in a gateway have a limitations. To solve this problem, cloud based sensor network architectures are proposed. In this paper, we proposed CoAP based IoT architecture that a lightweight gateway is used for data gathering instead of using a heavy traditional one and users can request sensing data through IoT applications running in the cloud environment and analyze signaling message cost. By doing so, our system can reduce message cost compared to the traditional gateway based system.

Keywords : IoT, Sensor, WSN, Cloud, COAP, Restful

1. 서 론

무선 센서 네트워크(Wireless Sensor Network)는 최근에 다양한 어플리케이션과 함께 많은 곳에서 이용되고 있다 [1]. 특히, 무선 센서 네트워크는 설치가 용이하다는 장점 때문에 다양한 분야에서 활용도가 높다. 예를 들어 무선 센서 네트워크는 군사 분야나 감시 [2, 3], 자연재해 [4], 환자를 위한 건강 모니터링 [5, 6], 유해 환경 탐사 및 지진 감시 [7]와 같은 경우에 사용될 수 있다. 다양한 분야에서 사용되는 만큼, 그에 따른 다양한 어플리케이션 또한 고려되고 있지만 센서가 가지는 제한적 요소로 인해 생기는 컴퓨팅 능력의 한계나 배터리의 한계등이 문제점으로 거론되고 있다. 게이트웨이를 통해 데이터의 재가공이 가능하나, 게이트웨이의 수용능력이 하나의 문제점으로 거론되고 있다. 이와같

*Corresponding Author(younghak@ssu.ac.kr)

Received: 16 Apr. 2015, Revised: 21 May 2015, Accepted: 30 May 2015.

Y.-K. Park, H.-S. Yang, Y.-H. Kim: Soongsil University

※ 본 논문은 중소기업청에서 지원하는 2015년도 산학연협력 기술개발사업(산학연협력 기술개발사업(연구마을), 과제번호: C0221093)의 연구수행으로 인한 결과물임을 밝힙니다.

은 문제를 해결하기 위해 클라우드 기반의 IoT(Internet of Things) 구조가 제안되고 있다 [8]. 전통적인 센서 네트워크의 형태는 센서로부터 수집된 데이터를 받아 단순히 데이터를 보여주는 형태로 되어 있지만, 클라우드 기반의 센서 네트워크에서는 하드웨어의 제한없이 다양한 데이터를 어플리케이션을 통해 가공하여 다양한 서비스를 제공할 수 있다 [9]. 여러 종류의 게이트웨이가 클라우드 기반의 센서 네트워크에서는 REST 기반의 웹 인터페이스와 같은 표준화된 인터페이스를 통해 데이터를 수집하기 때문에 다양한 센서 데이터를 손쉽게 수집 가능하다 [10]. 현재의 센서 네트워크는 인프라를 구성하는 제공자와 정보를 사용하는 사용자가 동일한 환경만을 고려하였다. 하지만 클라우드 기반의 센서 네트워크에서는 모든 데이터가 클라우드로 수집되고 각 데이터들을 세분화 하여 정보 제공자의 동의에 따라 데이터를 제공하고 사용할 수 있는 구조로 구성이 가능하다. 이는 현재 연구되고 있는 클라우드 기반의 빅데이터 처리방안과 비슷한 형태로, 수집된 방대한 데이터를 바탕으로 다양한 어플리케이션을 통해 가공하여, 여러 서비스를 제공하는 것이 가능하다. 따라서 본 제안에서는 클라우드내 제공자 기반의 인스턴스를 생성하고 그에 따른 서비스를 제공하기 위한 구조를 연구하고자 한다. 본 논문의 구성은 다음과 같다.

2장에서 IoT단말의 프로토콜 및 이중데이터 연

동을 위한 게이트웨이, 그리고 클라우드 구조에 대해 연구한다. 3, 4장은 클라우드기반 IoT 구조를 설계하고 제안된 설계에 따라 구현 한다. 마지막으로 본 논문의 연구결과를 통해 마무리 한다.

II. 관련 연구

본 절에서는 본 논문에서 제안하는 클라우드 기반 IoT 시스템 관련 기술들에 대해 기술한다. 클라우드 기반의 IoT 시스템을 위해서는 다양한 기반기술이 필요하다. 전체 시스템은 크게 3가지로 구분할 수 있다. 데이터를 수집하고 송신하기 센서 네트워크 관련 기술부분과 6LoWPAN, ZIGBEE 등의 다양한 인터페이스를 수용하고 데이터를 전송하기 위한 게이트웨이 관련기술부분, 전송된 데이터를 분류/분석 하고 가공하기 위한 클라우드 관련 기술이다.

1. 센서네트워크

저전력 센서네트워크 기술은 IETF에서 표준화한 6LoWPAN과 Zigbee Alliance에서 정의한 Zigbee 스택이 대표적이다. 이 두 기술은 IEEE 802.15.4 IPv6 기반에 네트워크 스택상의 어댑테이션 스택을 추가하여 기존 IPv6 프로토콜에서 사용되는 패킷을 압축하여 사용함으로써, 경량화 패킷을 통해 저전력 네트워크 송수신이 가능 하다. 센서 네트워크의 대표적인 응용기반 기술로는 Constrained Application Protocol(CoAP)이 있다. CoAP은 응용계층의 플랫폼으로 단말간 통신에 있어 HTTP와 같은 RESTful기반 기술을 지원하며, 저전력 센서네트워크를 위해 개발되었다 [11]. 그 외 응용 계층 기술로는 Alljoyn, oneM2M등이 있다. Alljoyn은 웹컴에서 개발한 프레임워크를 AllSeen Alliance의 오픈소스로 편입시켜 개발을 주도하고 있으며, 단말은 네트워크에 독립적인 구조의 프레임워크를 이용하기에 모든 단말과 통신이 가능하다. oneM2M은 사물과 사물에 대한 수평적 플랫폼 제공으로 모든산업분야간 표준화 기술을 정의하고 있다.

2. 이종망 통신을 위한 게이트웨이 기술

클라우드 기반의 IoT 시스템 구축을 위해서는 다양한 인터페이스의 단말들을 수용 할수 있는 구조로 고려해야 하며, 클라우드에서 데이터의 분류/분석을 위한 데이터 포맷 또한 고려해야 한다. 현재 센서 네트워크에서 센서가 외부 네트워크와 통신하기 위한 방법은 직접적으로 데이터를 전송하는 방

법과 게이트웨이를 통해 데이터를 전송하는 방법으로 나눌 수 있다. 직접 통신하는 경우 센서 단말 자체가 외부 네트워크와 연결이 가능한 인터페이스를 가지고 있기 때문에 게이트웨이가 필요 없다. 반면 다양한 인터페이스를 수용하고 각각의 고유한 프로토콜을 사용하는 센서 네트워크로부터 데이터를 수신하기 위해서는 게이트웨이가 필요하다. 한 예로 IEEE 802.15.4 기반의 센서 네트워크의 단말로부터 센싱된 데이터를 송신하는 경우, 외부 네트워크로 데이터를 전송하기 위해 게이트웨이 내 프로토콜 변환기술을 통해 데이터를 변환하여 전송해야 외부 망과의 연동이 가능하다. IEEE 802.15.4의 IoT 단말의 경우 외부 네트워크와 통신하기 위해서는 게이트웨이에서 802.15.4의 스택이 구현되어 있어야 한다. 게이트웨이의 802.15.4 스택은 IoT 단말로부터 전송된 데이터를 외부의 네트워크(IPv4 또는 HTTP) 프로토콜에 적합하게 변환 하여 목적지 노드로 데이터를 전송하게 된다 [12]. 이와 같이 게이트웨이를 이용하면 다양한 단말로부터의 데이터 수집이 가능하며 게이트웨이에 정의된 데이터 형태에 따라 변경도 가능하다.

3. 클라우드 시스템

클라우드 컴퓨팅은 다양한 IT자원들을 제공함으로써 사용자가 가지는 문제점을 극복하기 위한 시스템으로 컴퓨팅 환경 뿐만 아니라 다양한 분야에서 활용되는 기술이다. 클라우드 컴퓨팅은 서비스 제공 모델에 따라 크게 SaaS(Software as a service), PaaS(Platform as a service), IaaS(Infrastructure as a Service)로 구분되어진다. SaaS는 논리적으로 가상화된 컴퓨팅 자원과 스토리지 등의 인프라를 제공하는 것을 의미하며 PaaS는 사용자가 소프트웨어를 개발할 수 있도록 모듈을 제공하는 것을 말하고 IaaS는 인프라 자체를 제공하는 것을 의미한다 [13]. 현재 클라우드 서비스는 클라우드 구조가 가지는 확장성, 서비스, 데이터 처리 능력 등의 다양한 이점들 때문에 여러 분야에서 널리 활용되고 있다. 특히 빅데이터 시대가 도래하면서 수많은 데이터를 클라우드를 통해 수집하고 분석하여 제공하기 위한 플랫폼 및 여러 구조적 모델들이 제안되고 있다. 또한 인터넷만 연결되면 다양한 서비스를 제공할 수 있고 사용자 중심의 서비스 제공 방식을 이용하기 때문에 모바일 시장에서도 클라우드를 연동하기 위한 많은 방법들이 제안되고 있다.

III. 클라우드 기반 IoT 구조 설계

본 절에서는 IoT 단말 및 게이트웨이를 위한 클라우드 구조를 설계하고 6LoWPAN 단말과 라이트 웨이트한 게이트웨이, 그리고 클라우드 인스턴스를 구현하여 테스트베드 구축을 통한 상호연동 테스트를 진행 한다.

1. 디바이스 단말 정보 제공 설계

IoT 단말 정보를 클라우드 인스턴스로 보내기 위해서는 단말의 타입 및 데이터 구조를 정의해야 한다. 단말의 타입에는 주소정보, 데이터 특성 별 구분 정보(온도/습도/조도 등), 제공자 정보 및 위치 정보 등이 있다. 추가적으로 단말 간의 토폴로지를 위해 단말 간 연결성 정보를 포함 할 수 있다. 데이터 정보 구조는 센서 데이터로 구성되며, 복합 센서의 경우 타입 별 데이터를 구분하여 반복적으로 구성하며, 정보전송은 표준화 프로토콜인 CoAP을 사용하여 송수신 한다. 표 1은 IoT 단말의 전송 포맷 구조이며 그림 1은 기능 간 송수신 흐름도이다.

표 1. CoAP 전송에 대한 메시지 포맷
Table 1. CoAP Request Message Format

```
# Type Information
REQUEST
Header:GET(T=NCON,Code=1,MID=0x01a0)
Token: 0x71
Uri-Host:"coap://[aaaa::212:740:13b:277c]"
Uri-Path: "type-info"
Payload: ADDR|TYPES|OWNER*|LBS*
RESPONSE
Header: (T=ACK, Code=0, MID=0x01a0)

# Data Information
REQUEST
Header:GET(T=NCON,Code=1,MID=0x01a0)
Token: 0x71
Uri-Host:"coap://[aaaa::212:740:13b:277c]"
Uri-Path: "data-info"
Payload: TYPE|DATA|TYPE|DATA

# Neighbor Information
REQUEST
Header:GET(T=NCON,Code=1,MID=0x01a0)
Token: 0x73
Uri-Host:"coap://[aaaa::212:740:13b:277c]"Uri-Path: "neighbor"
Payload: NEIGHBOR|NEIGHBOR|...
RESPONSE
Header: (T=ACK, Code=0, MID=0x02b0)
```

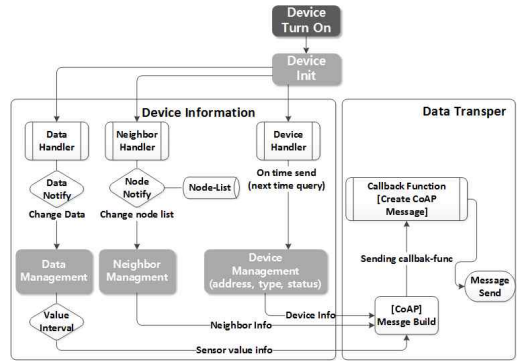


그림 1. IoT 단말의 동작구조 및 송수신 흐름도
Fig. 1 IoT Device Process Flow

IoT 단말은 초기화 이후 단말 상태 관리 및 이웃 노드 관리, 센서 데이터 관리 핸들러에 기능을 위임하며, 네트워크 스택 라우팅 관리부분은 생략되어 있다. 초기화 이후 최초 Device Handler에 의해 단말의 상태(주소, 타입, 소유자 정보 등)를 최상의 노드 및 게이트웨이로 전송하게 된다. Neighbor Handler는 Node List 구조체에 변경이 생기면 단말과 연결된 이웃 노드의 주소를 전송하게 된다. Data Handler의 경우에는 센서로 부터 읽은 데이터가 기존의 값과 다른 경우, 타입 정보와 센서 데이터를 보내게 되며, 데이터 전송에 대해 Interval 값을 세팅함으로써, 불 필요 데이터 전송을 방지할 수 있다.

2. 릴레이기반 게이트웨이 구조 설계

단말에서 전송된 데이터를 클라우드 인스턴스에 전송하기 위해서는 데이터를 릴레이할 수 있는 게이트웨이가 필요하다. 외부의 클라우드 인스턴스와 연동하기 위해서는 최소한의 인증 절차가 이루어져야 하며, 이를 위해서 클라우드 서버에 게이트웨이정보를 등록해야 한다. 등록정보는 게이트웨이의 MAC과 IP Address 주소이며, 등록시 서비스고유 정보인 Universally Unique Identifier (UUID)와 단말의 데이터 저장 처리를 담당하는 인스턴스를 할당 받는다. 그림 2는 게이트웨이에서의 데이터 흐름을 보여준다. 게이트웨이가 클라우드 서버에 등록되기 전에 IoT 단말로부터 메시지를 받은 경우 게이트웨이는 해당데이터를 버리고 단말에 게이트웨이 미등록 ACK를 보낸다.

IoT 단말이 등록 메시지에 대해 미등록 ACK를

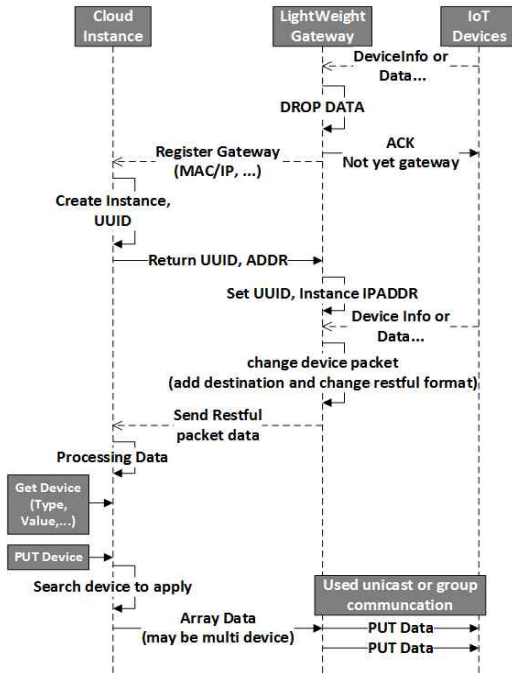


그림 2. 게이트웨이 기반 릴레이 흐름도
Fig. 2 Gateway based relay flow chart

받으면, 단말에서 정해진 시간 이후 재요청하게 된다. 게이트웨이에서 IoT 단말의 데이터를 받으면, 단말의 Source주소와 Payload정보를 분류하여 발급된 클라우드 인스턴스 주소로 데이터를 보낸다. 이때 발송되는 데이터 포맷은 JSON 구조의 RESTful 방식으로 데이터 전송한다. 게이트웨이의 주요기능은 클라우드 서버 등록, 단말로부터 수신된 데이터의 Restful 포맷 변환 통신, 그리고 사용자와 단말 간 통신을 위한 프로토콜 변환 기술이 포함되어야 한다.

3. IoT 단말/게이트웨이를 위한 클라우드 설계

IoT 단말 및 게이트웨이 정보를 저장하기 위해 단말의 유형에 따른 인스턴스를 제공해야 한다. 인스턴스는 게이트웨이 정보 및 IoT 단말의 데이터를 관리하며, 물리적 단말에 대한 논리적 단말 생성 및 유형에 따른 데이터 집합 기능을 제공 한다. 물리적 단말에 대한 논리적 단말 생성은 동일 유형의 단말을 하나의 데이터로 집합하여 하나의 논리적 단말로 사용자에게 제공할 수 있다. 위와 같은 기능을 제공하기 위해서는 클라우드 서버에서의 인스턴스 할당과 게이트웨이에서 전송되는 데이터를 분석하

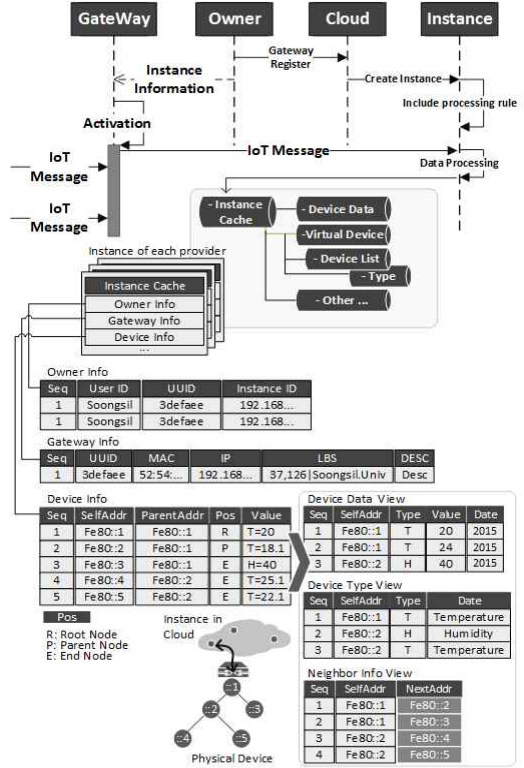


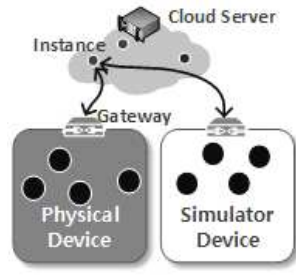
그림 3. 클라우드 및 게이트웨이 구조도
Fig. 3 Cloud and Gateway Structure

여 파싱해야 하며, 분류데이터의 캐시화가 필요하다. 그림 3은 게이트웨이로 부터 전송된 데이터 흐름과 정보 분류에 따른 클라우드 캐시 구조체이다.

캐시구조체는 크게 Gateway Info, Device Info로 분류되며, 세부적인 서비스를 위해 Device Info를 이용하여 단말과 관련된 다양한 View 구조체를 생성한다. 사용자 서비스를 위한 기본적인 View 구조체는 단말의 센서데이터 구조(Device Data), 단말이 사용할 수 있는 유형(Device Type), 단말의 토폴로지 구성을 위한 이웃 단말과의 연결성(Neighbor Info)등이 있다. 그리고 가상단말(Virtual Device) 서비스를 위한 구조체가 있다. 클라우드에서 데이터를 처리하는 경우, 클라우드의 처리 비용을 줄이기 위해 각 센서가 가지는 특징별로 데이터의 요청을 처리한다. 각 센서는 데이터 전송시 자신의 데이터가 유효한 시간을 나타내는 value값을 함께 전송한다. 클라우드는 각 사용자가 동일한 데이터를 요청하는 경우 유효시간 내에서는 기존에 수집된 데이터를 전송하며, 유효시간이 지난 경우 필요

표 2. 구현환경

Table 2. Implementation Environment

Definition	Description
IoTDevice	6LoWPAN, CoAP Device
Gateway	Linux2.6 higher(include tun)
Cloud	Openstack icehouse
Environment	

데이터를 센서에게 요청한다. 유효시간은 각 센서가 수집하는 데이터의 특성에 따라 정의 가능하다.

IV. 시스템 구현

본 절에서는 설계에 따른 IoT단말과 게이트웨이를 구현 하여 클라우드 인스턴스와 상호연동 테스트를 진행한다. 표 2는 각 구조별 구현 환경이다.

1. IoT 단말 메시지 Request / Response 구현

IoT 단말은 6LoWPAN, CoAP 스택이 포함되어 있으며, 단말의 메시지 송수신을 위해 게이트웨이에 설치된 최상의 단말에는 CoAP Server를 그리고 하위의 단말들은 CoAP Client로 동작하게 한다. 부팅 이후 단말의 유형을 게이트웨이로 보내는 Request Device Type과 센서데이터 전송을 위한 Request Sensor Data를 구현 한다. 표 3은 단말에서의 Request Function 구조를 보여준다.

dev_callback은 단말정보를 등록하는 함수이며, 단말의 초기화 및 네트워크 연결이 설립된 후 호출되는 콜백함수 이다. 해당 함수에서는 CoAP헤더를 초기화 하고 단말의 타입구조체를 가져온다. 구조체에는 16바이트 IPv6로 된 자신의 주소와 1바이트의 센서타입, 가변길이 센서설명으로 구성되어 있다. 복합센서를 사용하는 경우 단말의 타입 정의는 THL(temperature/humidity/light)과 같이 연속적으로 나열하여 정의한다. 자신의 주소, 타입, 타입설명을 CoAP Payload에 넣고 UDP로 게이트웨이에 전송한다. 전송 후 ACK를 수신하기 위해 Response에 대한 핸들러를 정의하며, 해당 핸들러에서

표 3. IoT단말의 요청 함수

Table 3. Request Function in IoT Device

```

## Device Information Function
void dev_callback {
  initialization coap header
  IF is coap buff
  create coap packet
  set coap condition = con
  set coap uri = "device info"
  device = get device
  IF device is right
  set coap buffer
  -> self address in device
  -> type in device
  -> description in device
  ENDIF
  assign packet buffer to req payload
  assign packet size to req payload length
  udp packet sent to the gateway
  res handler registered for the req
  ENDIF
}

## Sensor Data Function
void send_sense_data {
  initialization device
  init humidity, temperature sensor
  read new humidity, temperature
  init index set 2
  init buffer array
  IF is different new temperature and old
  set temperature data to buffer, index+ 4
  set separator(:) to buffer, ++index
  ENDIF
  IF is different new temperature and old
  set temperature data to buffer, index+ 4
  set separator(:) to buffer, ++index
  ENDIF
  get rpl-dag
  get destination address from rpl-dag
  udp packet sent to destination address
  res handler registered for the req
}
    
```

NACK를 수신한 경우 정해진 시간이후 재전송을 요청 한다. 클라우드로 등록이 되고 나면, sensor_data는 단말의 타입에 따라 센서보드를 초기화 한다. 센서보드로부터 데이터를 읽어 기존 데이터와 현재 데이터의 차가 정해진 값 이상의 경우 데이터를 전송하게 된다. 전송할 센서데이터 포맷은 그림 4의 센서데이터 포맷과 같다.



그림 4. 센서데이터 포맷구조
 Fig. 4 Request Sensor Data Format

표 4. 게이트웨이 및 클라우드 RestAPI 함수
 Table 4. Function of Gateway and Cloud

```

## Gateway to Cloud function
proto void create service
proto void update sensor data
...
init query data and json object
set uuid to json object from cloud
init restapi global function, headers
set content-type header = json
set restapi url = address and port
->http://url/resource/restapi?q=&json-data
...
## Create Sensor data or update
void update sensor data{
  IF is restapi = true
    set json += device address
    set json += sensor data
    set query command = update_sensedata
    set query data from json
    set postfields from query data
    restapi sent message and cleanup
  ENDF
}
## Response message check
void response check callback{
  IF response is not ok
    restapi response error
  ENDF
}
...
## restapi for cloud side
void restapi request {
  get request param = q
  get request param = json-data
  IF q is create_service
  ...
  ELSEIF q is update_sensedata
    get objects from sensor of json-data
    LOOP objects.size
      get element from objects
      -> element->uuid, sensordata, type, hexdata
      get types from element
      LOOP types.size
        update sensor_info #Latest data
        insert sensor_data #Accumulation data
      ENDF
    ENDF
  END
}
...
}

```

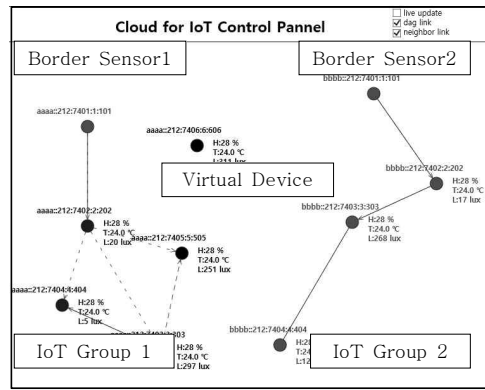
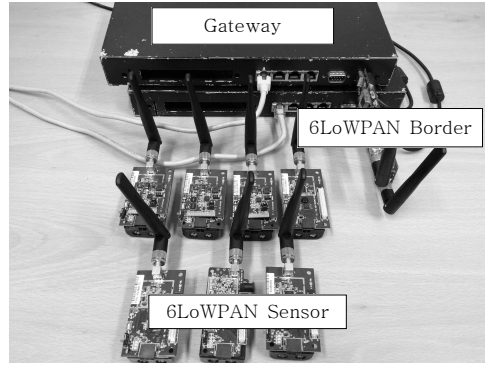


그림 5. 클라우드 기반 IoT 시스템 구현
 Fig. 5 Web interface of Cloud Based IoT

2. 게이트웨이 및 클라우드 시스템 구현

게이트웨이는 IoT단말의 최상의 계층인 Border Router 단말을 포함하고 있다. 리눅스 2.6이상의 커널버전에 구현 되었으며, 게이트웨이 데몬은 tun device를 생성하여 외부의 네트워크와 IoT단말 네트워크와 연동을 한다. 외부연동을 위해서는 JSON 포맷으로 변환 후 RestAPI를 사용하여 클라우드와 통신한다. 표 4는 게이트웨이 프로토 타입과 다양한 함수중 단말의 센서데이터 송수신 소스코드를 보여 준다. update_sensedata 함수는 단말의 센서데이터 저장 및 업데이트 기능을 담당한다. 함수의 기능은 JSON데이터 초기화 후 게이트웨이 고유값 UUID와 update_sensedata_t 구조체에서 단말로부터 수신된 SelfAddr, DataBuffer를 JSON 오브젝트로 변환하여 URI를 통해 클라우드 서버에 전송 한다.

클라우드 기반 IoT 테스트 베드는 그림 5와 같이 두개의 게이트웨이에서 보내는 단말의 정보를 클라우드 인스턴스내 웹으로 구현 하였다.

제공되는 RestAPI는 GW_INFO, TOPOLOGY,

SENSEDATA, FAKE_DEVICE 등이 있다. GW_INFO는 게이트웨이의 UUID 정보(IP_ADDR, MAC_ADDR)를 이용하여 게이트웨이 고유의 정보를 저장 하며, 저장된 게이트웨이 정보의 구분값을 이용하여 센서단말을 관리 하게 된다. TOPOLOGY는 단말들의 연결성 정보를 처리하며, 단말 자신의 주소, 이웃단말의 주소 그리고 부모노드 주소가 필요 하다. SENSEDATA는 센서의 데이터 값을 처리하며, 서비스를 위해 센서데이터에 대한 유효시간을 포함하여 저장한다. FAKE_DEVICE는 물리센서에 대한 가상센서를 처리하며, 다중의 물리센서를 하나의 가상센서로 표현하여 서비스를 제공하게 된다. 가상센서에 대한 데이터 처리는 각각의 물리센서로부터 취합된 데이터를 평균화 처리하며, 데이터의 유효시간이 지난 데이터만 물리센서로 재요청 하여 처리한다. 인스턴스 웹에 접속 하면 두개의 물리적 게이트웨이와 게이트웨이에 접속된 단말의 종류를 보여 준다. 하나의 게이트웨이에는 물리적 단말이 연결된 구조이며, 다른 하나는 가상의 단말이 연결된 상태이다. 물리적 단말의 표현은 주소 및 센서데이터로 표현 하며, 동일 유형의 단말을 결합하여 가상센서를 구성 할 수 있다. 가상 센서의 데이터는 선택된 단말의 센서데이터를 평균화 하여 보여주며, 가상단말에 대한 물리적 단말에 오류가 발생할 경우 데이터 값에 제외 되고 오류 사항을 보여 준다. 가상단말로 데이터를 요청 할 때 가상단말로 구성된 물리단말의 주소로 데이터를 요청하게 된다.

V. 성능 평가

본 장에서는 구현된 시스템을 기반으로 기존 센서 네트워크 시스템과의 성능 분석을 진행하였다. 성능 비교 분석을 위해 기존 센서 네트워크에 사용되는 게이트웨이를 통한 데이터 전달 방식과 클라우드를 통한 데이터 전달 방식에서의 유저수의 변화에 따른 성능을 확인하였다.

다수의 사용자가 동일 데이터를 요청하는 경우에 대하여 성능 분석을 실시하였다. 기본적으로 게이트웨이는 이중망간 데이터 송수신을 위한 맵핑기능을 포함한다. 게이트웨이기반의 환경에서는 데이터 요청 메시지에 대해 맵핑 후 데이터를 전송하는 BYPASS 방식을 이용하여 처리하였고, 클라우드기반에서는 동일 데이터에 대하여 Pending 방식을 사용하여 처리하였다. 그림 6과 같이 기존 게이트웨이에서 데이터를 요청 하는 경우에, 유저수가 증가함

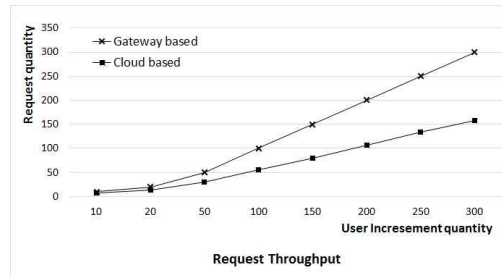


그림 6. 사용자 증가에 따른 처리량

Fig. 6 Throughput by the user increases

에 따라 각 요청 메시지를 개별적으로 처리하기 때문에 요청 메시지도 증가하는 것을 확인할 수 있다. 하지만 클라우드 기반의 IoT 구조에서는 동일 데이터에 대한 요청 메시지를 특정 메시지 수 내에서 단일 메시지로 처리함으로써 유저수 증가에 따른 시스템에 미치는 영향이 더 적어지기 때문에 더 효율적임을 확인 할 수 있었다.

VI. 결 론

다양한 인터페이스를 가진 IoT 단말이 증가로 인해 많은 단말들을 수용하기 위한 네트워크 구조가 요구되고 있다. 또한 IoT 단말이 증가함에 따라 증가하는 데이터를 이용한 다양한 서비스도 요구되고 있다. 하지만 센서 단말이 가지는 자원적 한계로 인해 다양한 서비스를 지원하는데 한계가 있다. 본 논문에서는 다양한 센서 디바이스를 수용하고, 사용자에게 다양한 서비스를 제공하기 위한 클라우드 기반의 IoT 시스템을 구현 하였다. 제안된 구조에서는 클라우드에 생성되는 인스턴스를 통해 데이터를 수집하고, 웹형태로 센서 정보 및 데이터를 확인할 수 있게 하였다. 이에 따라 수많은 센서로부터 데이터를 수집하는 경우에도, 각각의 인스턴스에 따라 분류되어 원하는 데이터를 원하는 사용자에게 제공할 수 있을 것으로 기대된다. 차후 연구에서는 본 논문에서 제안한 구조를 바탕으로 확장성을 고려한 IoT 서비스 구조를 연구하고 개발하고자 한다.

References

[1] L.M. Borges, F.J. Velez, A.S. Lebres, "Survey on the Characterization and Classification of Wireless Sensor Network Applications," IEEE

- Communications Surveys & Tutorials, Vol. 16, No. 4, pp. 1860-1890, 2014.
- [2] G. Simon, M. Maroti, A. Ledeczi, G. Balogh, B. Kusy, A. Nadas, G. Pap, J. Sallai, K. Frampton, "Sensor network-based countersniper system," Proceedings of the Second International Conference on Embedded Networked Sensor Systems, 2004.
- [3] J. Yick, B. Mukherjee, D. Ghosal, "Analysis of a Prediction-based Mobility Adaptive Tracking Algorithm," Proceedings of the IEEE Second International Conference on Broadband Networks, 2005.
- [4] M. Castillo-Effer, D.H. Quintela, W. Moreno, R. Jordan, W. Westhoff, "Wireless sensor networks for flash-flood alerting," Proceedings of IEEE International Caracas Conference on Devices, Circuits and Systems, Vol. 1, pp. 142-146, 2004.
- [5] T. Gao, D. Greenspan, M. Welsh, R.R. Juang, A. Alm, "Vital signs monitoring and patient tracking over a wireless network," Proceedings of the IEEE EMBS Annual International Conference, 2005.
- [6] K. Lorincz, D. Malan, T.R.F. Fulford-Jones, A. Nawoj, A. Clavel, V. Shnayder, G. Mainland, M. Welsh, S. Moulton, "Sensor networks for emergency response: challenges and opportunities," IEEE Pervasive Computing, Vol. 3, No. 4, pp. 16-23, 2004.
- [7] G. Wener-Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees, M. Walsh, "Deploying a wireless sensor network on an active volcano," IEEE Internet Computing, Vol. 10, No. 2, pp. 18-25, 2006.
- [8] A. Alamri, W.S. Ansari, M.M. Hassan, M.S. Hossain, A. Alelaiwi, M.A. Hossain, "A Survey on Sensor-Cloud: Architecture, Applications, and Approaches," International Journal of Distributed Sensor Networks, Article ID 917923, pp. 1-18, 2013.
- [9] S. Madria, V.; Kumar, R. Dalvi, "Sensor Cloud: A Cloud of Virtual Sensors," IEEE Software, Vol. 31, No. 2, pp. 70-77, 2014.
- [10] L. Gao, C. Zhang, L. Sun, "RESTful Web of Things API in Sharing Sensor Data," Proceedings of International Conference on Internet Technology and Applications, pp. 1-4, 2011.
- [11] Z. Shelby, K. Hartke, C. Bormann, "The Constrained Application Protocol (CoAP)," Internet-draft, 2014.
- [12] A. Castellani, S. Loreto, A. Rahman, T. Fossati, E. Dijk, "Guidelines for HTTP-CoAP Mapping Implementations," Internet-draft, 2015.
- [13] M.N.O. Sadiku, S.M. Musa, O.D. Momoh, "Cloud Computing: Opportunities and Challenges," IEEE Potentials, Vol. 33, No. 1, pp. 34-36, 2014.

Young-Ki Park (박영기)



He is currently a Ph.D. candidate in Electrical and Communication Engineering at Soongsil University, Seoul, Korea. He received his M.S. degree in Electronic Engineering from Soongsil University. His current research interests include Internet of Things (IoT), Digital Signage and the Network Function Virtualization (NFV). Currently, He is participating in several IT R&D programs in Korea.

Email: ykpark@dcn.ssu.ac.kr

Hyun-Sik Yang (양 현 식)

He is currently a Ph.D. candidate in Electrical and Communication Engineering at Soongsil University, Seoul, Korea. He received his B.S. degree in Electronic Engineering from Soongsil

University. His current research interests include Wireless Sensor Networks (WSNs), Authentication and Authorization for Constrained Environments (ACE) and the Network Function Virtualization (NFV). Currently, He is participating in several IT R&D programs in Korea.

Email: yangun@dcn.ssu.ac.kr

Young-Han Kim (김 영 한)

He received his B. S. degree in Electronics Engineering from Seoul National University, Seoul, Korea, in 1984, and the M.S. and Ph.D. degrees in Electrical Engineering from Korea

Advanced Institute of Science and Technology, Seoul, in 1986 and 1990 respectively. From 1987 to 1994 he was a senior member of the research staff in the Digicom Institute of Telematics, Seoul, Korea, where he did research on data and computer communication systems. Since September 1994 he has been with Soongsil university, where he is a professor of school of electronic engineering. He is a director of Ubiquitous Network Research Center and a director of Convergence-ITRC(Convergence Information Technology Research Center) project supported by MSIP(Ministry of Science, ICT & Future Planning), Korea. His research interests include wireless networking and future networking, SDN(Software Defined Networking), ICN(Information Centric Networking) and sensor networking. He is a member of IEICE and IEEE.

Email: younghak@dcn.ssu.ac.kr