IoT 디바이스 간 상호 통신을 위한 사물 인터넷 브로커 서버 설계

이동욱*, 김종현*, 신연순*, 이강우*
*동국대학교 컴퓨터공학과
e-mail : kimhwamin@dongguk.edu
qahira@naver.com
ysshin@dongguk.edu
klee@dongguk.edu

The Design of IoT broker server for two-way communication of IoT devices

Dong-Wook Lee*, Jong-Hyun Kim*, Youn-Soon Shin*, Kang-Woo Lee*
*Dept. of Computer Science Engineering, Dong-guk University

요 약

최근 IoT 보급이 가속화되면서, 글로벌기업들의 IoT 시장의 주도권 선점을 위한 혁신적 기술력 확보 경쟁이 치열해지고 있다. 사물 인터넷 기술을 지원하는 플랫폼 표준화 작업 외에도, 글로벌 대기업 중심으로 다수의 플랫폼 기술이 난립하고 있으며, 스마트 디바이스 통신에는 MQTT, CoAP, XMPP, HTTP 등 다양한 프로토콜이 사용되고 있다. 이렇게 다양한 플랫폼과 프로토콜을 사용하는 환경에서 디바이스간의 상호 연동 서비스가 중요한 문제로 대두되고 있다. 본 논문에서는 MQTT, CoAP 등 서로 다른 프로토콜을 사용하는 디바이스 간의 상호 통신을 용이하게 하고 향후 플랫폼 설계에 적용될 브로커 서버의 설계를 제안하고 구현한다.

1. 서론

최근 IoT(Internet of Things) 기술의 발전이 가속화되면서, 글로벌기업들의 IoT 시장의 주도권 선점을 위한 다양한 시도와 경쟁이 이루어지고 있다. Cisco 의 향후 IoT 전망에 따르면, 사물 인터넷을 구성하는 디바이스의 개수는 2020 년 500 억개로 눈에 띄게 증가할 것으로 예측한다. SK 의 ThingPlug, 삼성의 ARTIK, LG 의 IoT@home 과 같은 다양한 IoT 디바이스를 위한 플랫폼 또한 보급되고 있다. 본 논문에서 다루는 사물인터넷(IoT)기술이란 사물지능통신(Machine to Machine: M2M)으로부터 뻗어 나온 새로운 기술로서 디바이스들을 인터넷에 연결하여 발생하는 정보들을 실체화하고 사용자들의 접근을 용이하게 만들어 다양한 서비스가 가능하도록 구현한 기술이다[1]. 앞서 언급한 ThingPlug, ARTIK, IoT@home 등의 플랫폼이 각 사의 사물 인터넷 기술을 지원하기 위해 이미 사용되고 있으며, 스마트 디바이스 간 통신을 위해 MQTT, CoAP, XMPP, REST, HTTP 등 다양한 프로토콜이 사용되고 있다. 하지만, 통신을 위한 네트워크 프로토콜의 다양성으로 인한 서로 다른 프로토콜로 통신을 하는 디바이스 간 또는 플랫폼 간의 상호 통신 문제 해결이 사물 인터넷 분야에서 문제점으로 대두되고 있다. 본 논문에서는 사물 인터넷의 프로토콜인 MQTT, CoAP 을 각각 통신 프로토콜로서 사용하는 두 스마트 디바이스를 사용하여 디바이스간의 데이터 송수신을 위한 아키텍처를 설계하고, 결과 분석을 통해 추후에 상호 통신을 위한 플랫폼의 브로커 서버 설계에 대해 제안한다.

2. 관련 연구

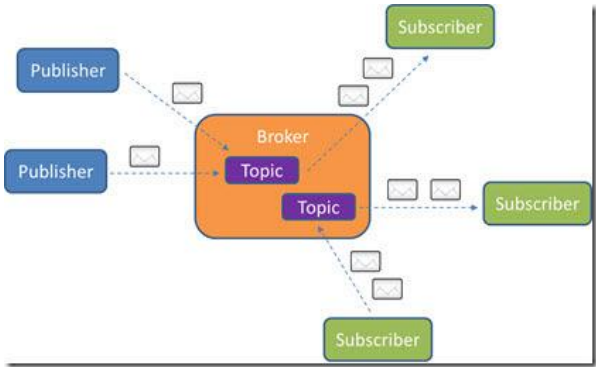
2.1 통신원리

사물인터넷 기술은 앞서 언급한 것과 같이 사물지능통신(Machine to Machine:M2M)으로부터 뻗어 나온 기술로서, 삼성 갤럭시 기어와 같은 스마트 워치, 스마트 홈, 스마트 카 등, 모든 기술이 IoT 기술에 해당된다. 사물인터넷은 사람과 사물, 사물과 사물이 정보를 송수신 할 수 있는 통신 방식을 가지고 있다.[3] 데이터 통신을 위한 단말기로는 스마트 디바이스가 사용될 수 있고, 사람이 송신자 혹은 수신자가 될 수 있다. 이런 IoT 네트워크 통신을 가능하게 하는 네트워크 프로토콜의 예로 MQTT 와 CoAP 을 2.2, 2.3 절에서 소개한다.

2.2 MQTT Protocol

2.2.1 MQTT Protocol 정의

MQTT(Message Queue Telemetry Transport) Protocol 은 작은 규모의 Publish/Subscribe 형태의 비동기 메시지 시스템을 위한 프로토콜이며 메시지 송수신의 QoS(Quality of service)를 보장한다. MQTT 서버를 통한 클라이언트 간의 통신은 연결, 토픽의 등록, 송수신, 연결 종료의 총 4 단계로 구성된다. MQTT 프로토콜은 기본적으로 Publisher 와 Subscriber 간 토픽을 기반으로 한 연결을 통해 송수신을 한다.



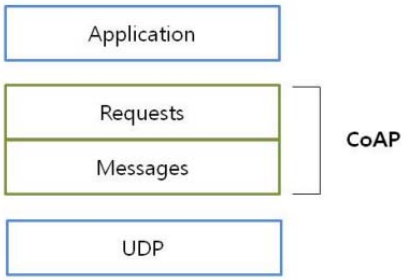
(그림 1) MQTT 프로토콜 통신 구조 [2]

2.2.2 Paho 클래스

Paho 는 이클립스 재단에서 시작된 사물인터넷을 위한 프로젝트 그룹 중 하나로 MQTT 프로토콜의 개발 언어에 대한 클라이언트 라이브러리를 제공한다. Paho 클래스의 내부 클래스들을 활용하여 서버와의 연결, 토픽 등록, 메시지 송수신, 연결 해지 등에서 끝날 때까지 블로킹되는 동기 방식, 리스너를 통해 결과를 알려주는 비동기 방식을 제공한다. 또한 서버와의 연결 옵션을 설정, 통신 결과를 Callback 할 수 있는 기능을 제공하는 클래스 또한 갖추고 있다.

2.3 CoAP Protocol

2.3.1 CoAP Protocol 정의



(그림 2) CoAP 프로토콜의 구조[3]

CoAP(the Constrained Application Protocol)은 전송지연과 패킷의 손실률이 높은 네트워크 환경에서 저 사양 센서 디바이스의 RESTful 웹서비스 제공을 위한 Client/Server 구조의 비동기적 메시징 모델이다.[4] CoAP 의 메시지는 CON(Confirmable), NON(Non-confirmable), ACK(Acknowledgement), RST(Reset)의 4 종류가 있으며, 메시지 중복을 방지하기 위해 0x7d34 같은 2 바이트의 메시지 ID 를 가진다.

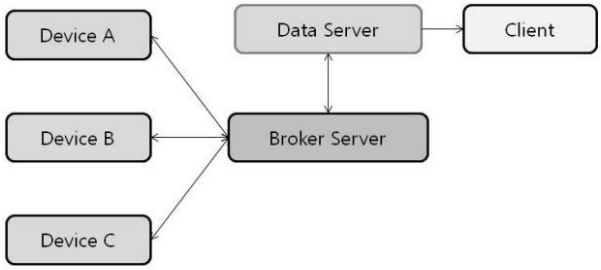
2.3.2 Californium 클래스

Californium 은 CoAP 프로토콜을 자바 언어로 구현한 오픈 소스 프로젝트로서 이클립스 재단에서 관리하고 있다. Californium 주요 클래스는 CoapClient, CoapServer, CoapHandler, CoapResource, CoapResponse 클래스가 있다. CoapClient 클래스를 통해 서버의 리소스를 호출하는 URI 를 설정할 수 있고, CoapHandler 클래스를 사용하여 비동기 방식으로 등록한다. 공통적인 응답에 대한 결과는 CoapResponse 객체로 수신된다. Californium 패키지 내부 클래스들을 사용하여, 비동기 호출, 리소스 감시, 리소스 탐색, 자동 제어 CoAP 서버의 구현이 가능하다.

3. 본론

3.1 시스템 개요

현재 IoT 동향은 MQTT, CoAP, XMPP, REST, HTTP 등 상황에 적합한 다양한 프로토콜을 디바이스에 적용, 통신을 하는데 사용한다. 본 논문에서는 서로 다른 프로토콜을 사용하는 디바이스간의 통신을 가능하게 하는 통신 프로토콜 스택간의 호환 기능을 제공하는 브로커 서버 시스템에 대해서 설명한다. 브로커 서버 시스템은 기본적으로 디바이스의 역할을 하는 오픈 소스 하드웨어(OSHW), 디바이스간의 통신과 수집된 데이터 처리를 담당하는 브로커 서버(Broker server), 클라이언트용 Application, 그리고 데이터를 저장하는 Data Server 로 간단하게 구성된다. 그러나 본 논문에서는 Data Server, Client Application 부분에 대한 자세한 설명은 생략하기로 한다. 개발을 위한 툴로 Java 언어 기반의 Java Embedded 를 활용하여 고기능성과 확장성을 확보하고, 디바이스간의 이식성을 지원하였다. 여기서 대용량의 스트림 데이터를 처리할 필요는 없기 때문에 Oracle 에서 제안하는 Oracle Fusion Middleware 는 사용하지 않는다.[5]



(그림 3) 전체 시스템 개요

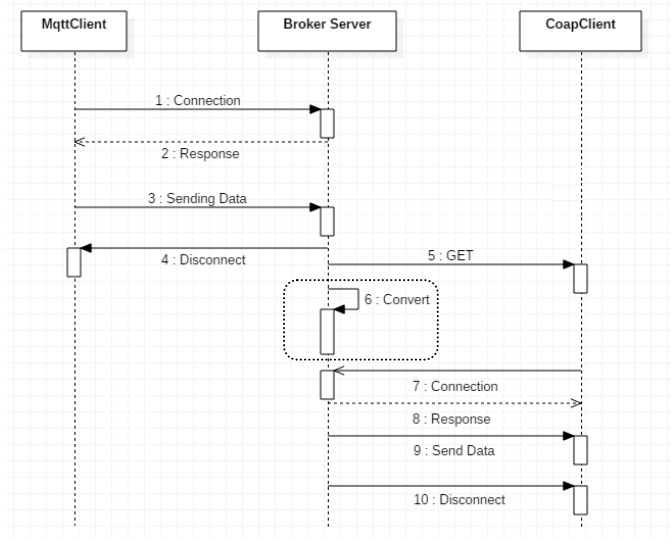
3.2 시스템 흐름

(그림 3)에서와 같이 데이터 혹은 메시지의 흐름은

Device A, B, C 중 한 디바이스에서 시작한다. 어떤 디바이스가 브로커 서버에게 메시지 혹은 데이터를 송신하면, 브로커 서버는 데이터를 데이터 서버에 저장한다. 여기서는 Device A 로부터 Device B 로의 통신을 예로 든다. Device A 와 브로커 서버의 통신 프로토콜이 MQTT, Device B 와 브로커 서버의 통신 프로토콜이 CoAP 이고 각각의 디바이스는 나머지 통신 프로토콜에 대한 송수신 기능이 구현되어 있지 않다고 가정한다. 브로커 서버는 MQTT 통신을 통해 Device A 로부터 전달받은 메시지 혹은 데이터를 Device B 로 보내기 위해 메시지를 CoAP 프로토콜에 맞게 매핑하여 송신한다. 그에 대한 Device B 로부터의 응답도 이와 같은 방식으로 전달된다. Device 로부터 클라이언트에게 메시지를 전달하는 경우는 실제로 널리 사용되는 HTTP 프로토콜에 기반하여 동작한다.

3.3 브로커 서버 동작원리

브로커 서버의 동작원리의 사례 중 하나인 MQTT 프로토콜과 CoAP 프로토콜 간의 통신 구현을 위해 앞서 소개하였던 Paho 클래스와 Californium 클래스를 사용한다. 브로커 서버는 MQTT 클라이언트로부터 송신 된 메시지의 XML 혹은 JSON 형식으로 직렬화 된 페이로드의 메시지 프리미티브에서 데이터를 분리한다. 이 데이터를 CoAP 클라이언트로 다시 전달하기 위해 브로커 서버에서는 CoAP 클라이언트 프로토콜 메시지 형식으로 매핑하여 보내게 된다. 이 과정에서 MQTT 통신에서의 Subscriber 는 브로커 서버가 되며, 브로커 서버는 수신한 데이터를 데이터 서버에 저장 및 클라이언트의 어플리케이션에 송신하는 역할을 한다. 결론적으로, 두 프로토콜 스택의 중간 계층에서 송수신한 데이터를 수집 및 저장하는 것이 브로커 서버의 역할이 된다.

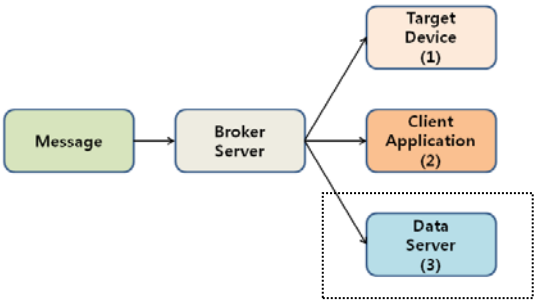


(그림 4) 브로커 서버 동작 시퀀스 다이어그램

3.4 메시지 Convert 과정

(그림 4)는 3.3 절에서 소개한 브로커 서버의 통신 프로토콜 스택의 호출 과정을 시퀀스 다이어그램으로 보여준다. 앞서 언급하였듯이, 브로커 서버는 두 통신 프로토콜간 데이터의 송수신 및 매핑 기능, 클라이언트 어플리케이션으로의 데이터 송신, 데이터 서버로 데이터 저장 기능을 담당하고 있다. Convert 과정에서 브로커 서버는 MQTT Client 로부터 수신한 메시지로부터 데이터를 가져온다. 이 데이터는 상황에 맞게 다음과 같은 과정을 거칠 수 있다.

1. 타겟 디바이스 프로토콜 메시지에 적합하게 데이터 매핑
2. 클라이언트 애플리케이션에 데이터 송신
3. 데이터 서버의 스키마에 적합한 데이터 가공 및 저장



(그림 5) 브로커 서버 제공 기능

데이터가 데이터 서버에 저장될 필요가 없거나, 클라이언트 애플리케이션에 송신할 필요가 없는 단순한 두 IoT 디바이스 간의 통신은 (1)번 과정만 진행한다.

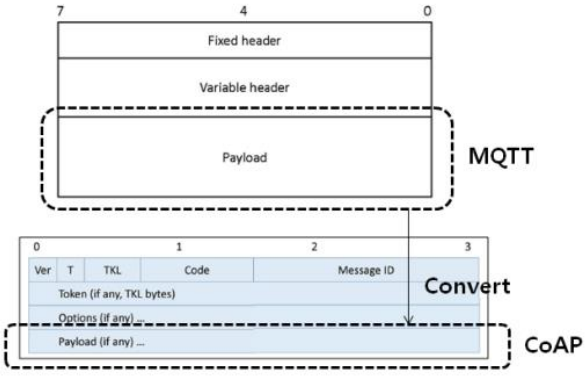
```
from <- sender
to <- receiver

begin

IF from null THEN
        messageFailure() // callback
ENDIF
IF from not null THEN
        data <- sender.get()
        to.payload <- data
ENDIF
If to not null THEN
        to.Send()
        messageSuccess() // callback
ENDIF
end
```

(그림 6) Convert Pseudo Code

(그림 7)은 브로커 서버에서 MQTT 메시지를 CoAP 메시지로 매핑되는 과정을 보여주는 그림이다.



(그림 7) 매핑 과정[3]

3.5 구현 및 분석

본 절에서는 제시한 브로커 서버의 동작 환경과 기능 구현 및 간단히 구현한 UI 를 보여준다.

서로 다른 프로토콜 네트워크 환경에서 브로커 서버를 통해 하나의 디바이스로부터 다른 디바이스로 송신한 데이터가 손상 없이 통신이 되는지 확인할 수 있는 가상의 시스템 환경을 구축하였다. 전송할 임의의 데이터는 간단한 String 형태의 메시지로 설정하였다. 데이터를 전송하는 디바이스는 Raspberry PI 3 모델을 사용하였고, 브로커 서버로는 Linux OS 기반 Desktop 을 사용하여 종단간 통신을 위해 데이터를 변환하는 코드를 직접 구현하였다. 인터넷 환경은 모두 유선으로 연결하여 서로 같은 내부망에 위치하도록 하였다.

개발 환경		버전
디바이스	Raspberry PI	V3
개발환경	Windows 7	Professional 7k
서버	Fedora (Linux)	V10
개발도구	Eclipse	IDE 4.5.0
개발언어	Java	Java SE 1.8

<표 1> 브로커 서버 구현 환경

위와 같은 환경에서 브로커 서버를 구축해 MQTT 디바이스로부터 CoAP 디바이스로의 메시지 송수신을 테스트하고 그 결과를 UI 를 구현하여 확인한다.

```
MqttClient [Java Application] C:₩Program Files₩Java₩jre1.8.0_91₩bin₩javaw.exe (2016. 9.
Connection: 192.168.0.6
cmd: cd'/home/pi/dist'
text: Hello World!! everybody
payload: 23bytes
Sep 26, 2016 03:32:32 PM BrokerServer.Broker.Server$1 deliveryComplete
Delivered: [Mon Sep 26 15:32:32 UTC 2016]
```

(그림 8) MQTT 디바이스 메시지 송신

(그림 8)은 MQTT 클라이언트가 브로커 서버에게 Hello World!! everybody 라는 스트링을 페이로드에 담아 메시지를 송신한 상태를 보여주고 있다. 브로커 서버는 이를 CoAP 디바이스에 보내기 위해 데이터를 CoAP 프로토콜 메시지 형식으로 매핑하여 송신한다.

```
CoapClient [Java Application] C:₩Program Files₩Java₩jre1.8.0_91₩bin₩javaw.exe
code: 2.05
options: {"Content-format": "text/plain"}
payload:48656c6c6f20576f726c6421212065766572279626f6479
text: Hello World!! everybody
ip: 192.168.10
payload: 23bytes
Type : ACK
Delivered: [Mon Sep 26 15:32:34 UTC 2016]
```

(그림 9) CoAP 디바이스 메시지 수신 내용

(그림 9)는 서버로부터 수신한 CoAP 메시지의 내용을 보여준다. payload 부분은 수신한 데이터를 byte 배열의 형식으로 출력한 것 이며, text 부분은 페이로드 부분을 스트링으로 출력한 것이다.

두 그림에서 보듯이 본 논문에서 제시한 브로커 서버를 통해 이종의 프로토콜을 사용하는 디바이스 간의 송수신이 오류나 손실 없이 정확히 전달되고 있음을 알 수 있다.

4. 결론

본 논문에서는 IoT 네트워크 환경에서 사용되는 MQTT, CoAP, XMPP, REST, HTTP 와 같은 다양한 프로토콜과 그 프로토콜간 상호 연동을 통한 통신이 가능하도록 브로커 서버를 제안하였다. 구현된 시스템은 한가지의 프로토콜 통신밖에 지원하지 않는 디바이스 간의 통신을 브로커 서버를 통해 가능하게 하여, 별도로 새로운 통신환경을 디바이스와 서버간에 구축하지 않고도 다양한 프로토콜 디바이스 간의 통신이 이루어지게 하였다.

향후 연구로는 보다 다양한 IoT 네트워크 프로토콜에 대한 설계를 포함하여 성능 평가를 진행하고 전체적인 서비스 플랫폼을 구현하기 위한 방향을 제시한다.

사사

본 연구는 미래창조과학부 및 정보통신기술진 센터의 SW 중심대학지원 사업의 연구결과로 수행되었음(R7116-16-1014).

참고문헌

[1] 김상현, 이정혁, 오창세, 김동휘, 박현주 MQTT 프로토콜과 Web Socket 을 활용한 사물인터넷 플랫폼 설계, 한국정보과학회 학술발표논문집, 2015.12, 425-427 (3pages)

[2] 심승현, 김학범, 사물인터넷과 MQTT 기술, 정보보호학회지 24(6), 2014.12, 37-47(11pages)

[3] Z.Shelby, C.Bormann, The Constrained Application Protocol(CoAP), IETF7252, 2014.6,

[4] 한국정보통신기술협회, oneM2M 서비스 플랫폼 표준 해설서, TTA-14092-SD, 2014.11

[5] 장양자, IoT 플랫폼 기술동향, 정보과학회지 32(6), 2014.6, 19-24(6pages)