JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 20, No. 4 : 728~738 Apr. 2016

안정적인 사물인터넷 플랫폼을 위한 MQTT 기반 데이터 수집 솔루션 관한 연구

김상현[1] · 김동휘[1] · 오형석[1] · 전현식[1] · 박현주[2*]

The Data Collection Solution Based on MQTT for Stable IoT Platforms

Sang-hyun Kim[1] · Dong-hwi Kim[1] · Hyeung-seok Oh[1] · Hyun-sig Jeon[1] · Hyun-ju Park[2*]

[1]Department of Radio-wave Engineering, Hanbat National University, Daejeon 34158, Korea
[2*]Information and Communication Engineering, Hanbat National University, Daejeon 34158, Korea

요 약

ICT 기술의 발전은 인간이 생산해내는 정보뿐 아니라 일상 사물을 인터넷에 연결하여 사물의 정보와 사물이 센싱하는 환경 정보까지 공유할 수 있는 사물인터넷 시대의 도래를 촉진하고 있다. 이에 사물과 사물 간 데이터를 전송하는 많은 방법들이 제안되고 있으며 대표적으로 HTTP 프로토콜을 활용한 데이터 전송 방법이 대중적으로 사용되고 있다. 하지만 방대한 데이터들을 효과적으로 처리하기 위하여 사물 인터넷 플랫폼 시장에서는 좀 더 신속하고 안정적인 통신 프로토콜을 지향하고 있다. HTTP 프로토콜을 지원하는 시스템에서는 헤더가 데이터에 비해 상대적으로 크기에 전송 효율의 문제를 보인다. 또한 시스템 과부하의 문제점이 대두되고 있으며 기업 간 자체 사물인터넷 표준은 데이터 교환 호환성을 방해하고 있다. 이에 본 논문에서는 경량의 표준 사물인터넷 프로토콜인 MQTT와 Web Socket을 활용하여 전송 효율을 개선하고 신속하고 안정성 있는 사물인터넷 플랫폼을 위한 데이터 수집 솔루션을 제안한다.

ABSTRACT

We are currently able to share not only the information from humans but also the data from connected things on the Internet. We are getting close to IoT era because of progress of Information Communication Technology. Therefore, the ways of data transfer are offered between things to things. One of typical way is the HTTP protocol. However, The field of Internet of Things platforms requires more fast and more stable communication protocol to handle massive data. The system supporting HTTP protocol, there is a problem of transmission efficiency in a relatively large header compared to data. also HTTP protocol system overload situations and the problem of data compatibility happens due to each standard of many organizations. Thus, To solve these problems, I suggest the data collection solution based on MQTT protocol for the operation of the stable IoT platforms.

키워드 : MQTT, 프로토콜, 사물인터넷 플랫폼, Broker, Publish/Subscribe

Key word : MQTT, Protocol, IoT Flatform, Broker, Publish/Subscribe

Received 27 January 2016, **Revised** 17 February 2016, **Accepted** 09 March 2016

* **Corresponding Author** Hyun-Ju Park (E-mail:phj@hanbat.ac.kr, Tel:+82-042-821-1220)
Information and Communication Engineering, Hanbat National University, Daejeon 34158, Korea

Open Access http://dx.doi.org/10.6109/jkiice.2016.20.4.728 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

최근 다양한 사물이 서로 연결되는 사물인터넷(Internet of Thing)에 대한 기대와 관심이 커지고 있다. 세계적인 시장조사기관 가트너는 그림 1과 같이 2015년 '인터넷 연결 기기(connected things)'의 대수가 올해보다 30% 증가한 49억대, 2020년에는 250억 대의 장치들이 무선 인터넷을 기반으로 연결될 것이라고 예상하였다[1]. 장소에 구애받지 않고 언제 어디서나 컴퓨팅 환경에 접속할 수 있는 유비쿼터스 패러다임이 확대되면서 다양한 기기가 인터넷에 연결되어 연동되고 있다.

사물인터넷 기술에 대한 기대와 관심이 고조되고 있는 가운데 국내・외 단체에서 사물인터넷 플랫폼에 관한 연구를 진행하고 있다. 사물인터넷 플랫폼이란 인터넷에 연결된 모든 기기를 하나의 시스템 안에서 통합・관리 할 수 있는 운용체계이다. 위에 언급한 바와 같이 사물인터넷 기기들이 폭발적으로 늘어나고 있기 때문에 이를 수용하기 위한 사물인터넷 플랫폼 개발에 대한 연구는 필수적이라 판단된다. 또한 현재 등장하는 많은 기업들의 자체 프로토콜 표준은 사물인터넷 플랫폼 시장을 혼란스럽게 만들고 있고, HTTP 프로토콜을 지정하는 기존 시스템에서는 전송 효율 문제, 시스템 과부하의 문제점이 대두되고 있다.

본 논문에서는 안정적인 사물인터넷 플랫폼에 대한 연구와 기존 시스템의 전송 효율 문제를 보완하기 위하여 보편성, 유연성, 경량성을 가진 사물인터넷 프로토콜인 MQTT를 사용한 데이터 수집 솔루션을 제안한다. 2장에서는 MQTT 프로토콜과 Web Socket 기술에 대하여 언급하고, 3장에서는 제안하는 시스템의 설계와 구현, 4장에서는 제안하는 시스템의 성능평가에 대하여 논의하고 결론을 도출한다.

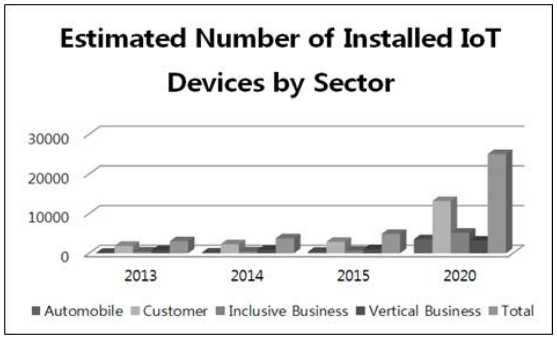

Fig. 1 Estimated Number of Installed IoT Devices by Sector

Ⅱ. 관련 연구

이 장에서는 안정적인 사물인터넷 플랫폼 운영을 위한 MQTT 프로토콜 기반 데이터 수집 솔루션을 구현하기 위한 기술에 관하여 설명한다.

2.1. MQTT 프로토콜

MQTT는 1999년 IBM에서 개발된 프로토콜로서 제한된 컴퓨팅 성능과 네트워크 연결 환경에서의 동작을 고려하여 설계된 대용량 메시지 전달 프로토콜이다. 2013년엔 OASIS (Organization for the Advancement of Structured Information Standards)에 의해 사물인터넷을 위한 메시지 프로토콜로 선정되었을 뿐만 아니라 배터리 에너지 소모 측면에서 효율적인 프로토콜임이 검증되고 있다[2,3]. 또한 모바일 기기와의 효율적 통신을 위한 양방향 Publish/Subscribe 특성을 가지고 있기 때문에 메시징 크기에 제약이 없으며 이기종 플랫폼 간 개발이 용이하다. 이와 같은 특징 때문에 사물인터넷 환경에서 널리 사용되고 있다.

2.1.1. Publish/Subscribe 구조

MQTT의 가장 큰 특징은 Publish/ Subscribe 구조를 가진다. Publisher와 Subscriber는 특정 Topic을 사용하여 MQTT Broker를 거쳐 통신이 가능하다. 그림 2는 Publisher와 Subscriber가 통신하는 기본 구조를 나타낸 것이다. 하나 이상의 Subscriber가 특정 Topic을 Subscribe한 상태라면 Publisher에 의해 특정 Topic으로 Publish된 데이터는 MQTT Broker를 거쳐 Subscriber에게 전달된다.

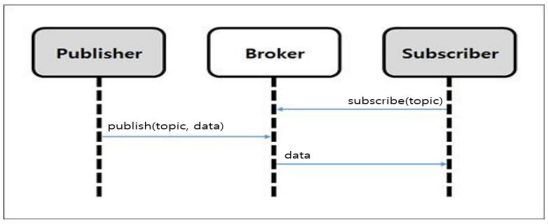

Fig. 2 Publish/Subscribe structure

2.1.2. Topic Format

그림 2에서 Topic은 문자열로 표현하고 하위 항목의 각 계층은 슬래쉬(/)문자로 구분하여 계층적으로 구성한다[4]. 계층적 구성을 통하여 다양한 사물인터넷 디바이스를 효율적으로 관리할 수 있다. 예를 들어 학교의 각층과 강의실을 위한 온습도 센서가 존재한다고 가정하였을 때, 그림 3과 같은 다이어그램으로 데이터를 표현할 수 있다.

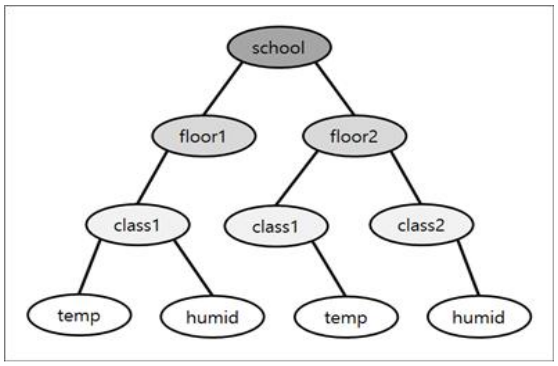

Fig. 3 Hieraarchical structrue according to the MQTT Topic

그림 3에서 학교 1층에 있는 1번 강의실의 온도 센서를 MQTT Topic으로 표현하면 다음과 같이 표현하는 것이 가능하다.

- school/floor1/class1/temp

또한 다중범위의 데이터를 표현하는 와일드카드 기능이 있다. “+”문자는 같은 레벨에서의 모든 Topic을 의미하고 “#”문자는 아래 계층까지의 모든 Topic을 의미한다. 예를 들어 학교의 모든 데이터 정보를 수집하고 싶다면 다음과 같이 표현할 수 있다[5].

- +/+/+/+
- #

이 표현식은 Mosquitto MQTT의 Broker의 환경내용을 기재하였으며 Broker의 종류에 따라 약간 상이하다.

2.1.3. Message Format

MQTT Message Format은 표 1과 같이 기본적으로 2Byte의 최소 헤더를 가지며, 데이터를 Publish할 때의 2Byte의 필수 정보를 담고 있다. 이 때 메시지의 타입과 데이터의 크기에 따라 헤더의 크기는 변한다. 최소 2Byte로 동작할 수 있기 때문에 저전력으로 운용하는 것이 가능하고 낮은 대역폭에서의 성능이 우수하다.

Table. 1 MQTT Header Message Format

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT control Packet type				DUP flag	QoS level		RETAIN
	0	0	1	1	X	X	X	X
byte 2	Remaining Length							

2.1.4. QoS (Quality of Service)

MQTT는 메시지 처리에 대한 신뢰성 보장을 위하여 3단계의 QoS(Quality of Service)를 지원한다[6].

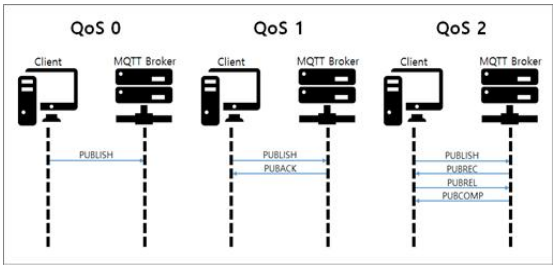

Fig. 4 Packet transfer mode according to the QoS value

그림 4와 같이 QoS 0은 메시지를 한 번만 전달하고 전달 여부는 확인하지 않는다. 따라서 큰 페이로드를 가지는 메시지일 경우, 패킷 손실이 발생하면 메시지가 전달되지 않고 유실될 가능성이 높다. QoS 1은 메시지를 최소 1번 이상 전달하고 전달 여부를 확인한다. 하지만 PUB ACK 패킷이 유실된다면 메시지가 불필요하게 중복으로 전달될 가능성이 있다. QoS 2 는 4-way handshake을 통해 정확하게 한 번만 전달한다. QoS 2를 사용한다면 메시지가 유실될 가능성은 없지만 4-way handshake 과정에 의하여 종단 간 지연이 늘어난다. QoS 레벨이 높을수록, 패킷을 교환하는 횟수가 늘어나기 때문에 상대적으로 종단 간 지연은 늘어난다[7].

Table. 2 QoS Message Format

QoS value	Bit 2	Bit 1	Description
0	0	0	At most once delivery
1	0	1	At least once delivery
2	1	0	Exactly once delivery
-	1	1	Reserved – must not be used

QoS value에 따라 표 2와 같이 정의 된다. 2개의 비트로 구분되며 Bit 2와 Bit 1의 조합은 정의하지 않는다.

2.1.5. MQTT Broker의 종류와 장단점

MQTT는 다양한 종류의 Broker를 가지고 있다. 그 중에서도 가장 대중적으로 활용되는 Mosquitto와 RabbitMQ가 있다. Mosquitto 브로커는 MQTT 전용 브로커로 경량이며 브로커가 가져야 할 대부분의 기능을 충실히 지원한다는 장점이 있다. 그렇기 때문에 소형 디바이스를 사용하는 사물인터넷 환경에서 사용하기에 적합하다. 본 논문에서는 다른 상용제품에 비해 성능이 우수하고 오픈소스로 활용할 수 있는 Mosquitto MQTT broker를 사용한다. 표 3에 Mosquitto와 RabitMQ의 장단점을 정리한다.

Table. 3 Comparison of Mosquitto and RabbitMQ

Feature	Mosquitto	RabbitMQ
Lightweight program	O	X
To support a variety of protocols	X	O
Three levels of QoS support	O	X
Wildcard support	O	O
Support for clustering	X	O

2.2. Web Socket

Web Socket 프로토콜은 전이중(full-duplex)을 제공하는 양방향 통신채널이다. 하나의 소켓을 통해 확장성과 실시간을 보장하여 웹 애플리케이션을 구축하는 것이 가능하다[8]. Web Socket은 웹 서버와 웹 브라우저 상에 구현되어, 두 지점 간에 실시간 상호 작용하도록 지원한다. 실시간이 요구되는 응용 프로그램을 한층 효과적으로 구현할 수 있다. 이러한 특징 때문에 본 논문에서는 Push Server를 Web Socket을 사용하여 구현한다. Web Socket 프로토콜은 2011년 인터넷 표준화 기구인 IETF에서(IETF RFC 6455) 표준화되었고, 웹 소켓 응용 프로그래밍 인터페이스(API)는 월드 와이드 웹 컨소시엄인 W3C에서 표준화를 완료했다. 그림 5는 현재 Web Socket을 지원하는 웹 브라우저를 보여주는 그림이다. 이 그림에 의하면 Opera Mini 브라우저를 제외한 모든 브라우저의 최신 버전은 Web Socket을 지원한다. Android Browser와 iOS Safari 또한 지원하기 때문에 모바일 환경에서의 서비스도 가능하다[9].

Fig. 5 The type of browser that supports Web Socket

따라서 Push Server는 Web Socket의 일종인 Socket.io를 사용하여 구현한다. Socket.io란 JavaScript를 이용하여 브라우저 종류에 관계없이 다양한 방식의 실시간 웹 기술을 구현할 수 있는 기술이다. Web Socket과 달리 Socket.io는 표준 기술이 아니고 Node.js 모듈로서 LearnBoost 회사의 저작물이며 MIT 라이센스를 가진 오픈소스이다.

Ⅲ. 제안하는 데이터 수집 솔루션

이 장에서는 안정적인 사물인터넷 플랫폼을 위한 MQTT 기반 데이터 수집 솔루션에 대해 설명한다. 사물인터넷 플랫폼 구축을 위한 데이터 수집 솔루션을 위해서는 표준화된 프로토콜 사용이 필요하다. 세계의 표준화 단체에서는 사물인터넷 관련 기술 및 서비스에 대한 표준화를 진행하고 있으며, 작은 메모리와 낮은 처리 성능을 가진 노드들도 사용할 수 있는 가볍고 빠른 통신 프로토콜에 대한 연구가 활발히 이루어지고 있다. 그 중에서도 모바일 기기와의 효율적 통신을 위한 양방향 Publish/Subscribe 메시징 서비스를 구현하는 MQTT (Message Queue Telemetry Transport)도 개방형 표준 프로토콜로 배포되고 있다. 사물인터넷 단말에서 발생되는 방대한 데이터들에 의해서 전체 플랫폼이 동작하기 때문에 시스템은 데이터의 폭발적 증가를 감당하는 것이 요구되고 있다.

본 논문에서는 사물인터넷을 서비스하기 위하여 불안정한 네트워크 상황에서도 높은 효율을 보이는 사물인터넷 표준 프로토콜인 MQTT를 활용하고, 사용자들에게 빠른 알림 서비스를 제공하기 위하여 Web Socket

기술을 활용한다. 기본적으로 제안하는 솔루션에서는 데이터를 수집하기 위한 IoT Device(IoT Sensor Node, IoT Actuator Node), 수집된 데이터를 처리하는 MQTT IoT Broker Server, 데이터 저장소 Data Server, 클라이언트를 위한 Push Server가 구성된다. 그림 6은 본 논문에서 제안하는 사물인터넷 플랫폼을 위한 MQTT 데이터 수집 솔루션의 전체 구성도이다.

3.1. 시스템 아키텍처

안정적인 사물인터넷 플랫폼을 위한 데이터 수집 솔루션을 위하여 그림 6과 같은 아키텍처를 설계한다. 크게 역할별로 5가지 개체로 IoT Device(IoT Sensor Node : ISN, IoT Actuator Node : IAN, IoT Complex Node : ICN), IoT Broker Server(IBS), Data Server, Push Server, Client로 분류한다.

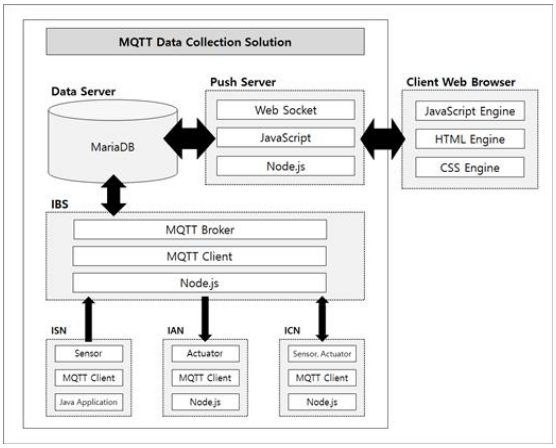

Fig. 6 System architecture to be proposed

3.1.1. 각 개체별 담당 역할

각 개체를 분류하고 담당 역할을 간단히 소개하고자 한다.

3.1.1.1. IoT Device (ISN, IAN, ICN)

IoT Device는 ISN와 IAN, ICN으로 구분한다. 일반적으로 ISN과 IAN은 구체적인 센서나 액추에이터를 가지고 구현을 하지만 본 논문에서의 주제는 안정적인 플랫폼을 위한 데이터 수집 솔루션이기 때문에 가상의 ISN과 IAN으로 대체한다. ISN은 일정 범위의 온도 값을 MQTT Topic에 맞게 주기적으로 송신한다. IAN은 IBS의 Topic을 Subscribe하여 특정 값이 수신될 때 동작한다.

3.1.1.2. IoT Broker Server (IBS)

IBS는 IoT Broker Server의 약자로 크게 MQTT Broker와 MQTT Client로 구성되어 있다. 본 논문에서 제안하는 플랫폼에서 데이터 Topic을 관리하는 중추적인 역할을 하고 있다. ISN으로부터 센싱 데이터가 Publish되면 IBS에 구성된 MQTT Client가 센싱 데이터를 Subscribe하여 Data Server로 저장한다. 또한 사용자가 Push Server로 송신한 명령이 존재하면 해당 명령 주제를 Subscribe하여 IAN을 제어한다.

3.1.1.3. Data Server

Data Server는 IBS로부터 수집된 데이터를 저장하는 역할과 Push Server에게 데이터를 넘겨주는 두 가지의 역할을 한다. 첫 번째 역할로 ISN과 Client를 거쳐 IBS에 수집된 모든 데이터를 수집한다. ISN으로부터 수신된 데이터는 IBS의 MQTT Client가 데이터베이스 커넥션을 이용하여 Data Server로 데이터를 주기적으로 저장한다. 또한 Push Server의 요청에 따라 데이터를 가공하여 특정 데이터를 전달한다.

3.1.1.4. Push Server

Push Server는 사용자에게 신속한 알림 서비스를 제공하는 역할을 한다. 신속한 사용자 알림 서비스를 위하여 Web Socket의 일종인 Socket.io로 구현하였으며 특정 데이터가 수신되면 Client의 웹 브라우저로 알림 메시지를 보내는 역할을 한다.

3.1.1.5. Client

JavaScript Engine, HTML Engine, CSS Engine 등이 포함된 웹 브라우저는 모두 Client 역할을 수행한다. 일반적인 PC나 스마트폰, 태블릿 PC를 Clilent로 사용하는 것이 가능하다. Push Server와 연동되어 고객들에게 서비스를 하는 역할을 한다.

3.1.2. Message Sequence

본 논문에서 제안하는 데이터 수집 솔루션의 이해를 돕고자 Message Sequence Diagram을 작성한다. 기본적으로 IBS를 중심으로 연결되어 ISN에서 Client까지 메

시지 전달이 가능하도록 Publish/Subscribe 및 Push 절차를 설계한다.

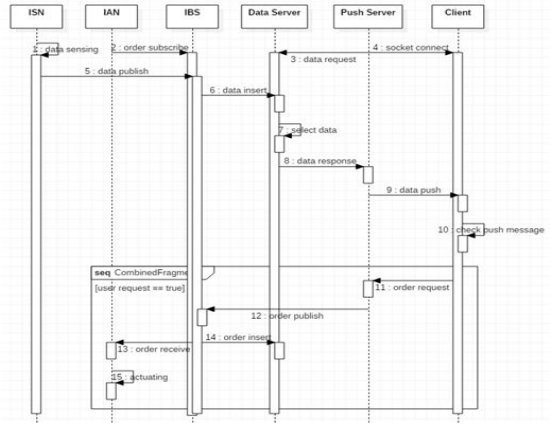

Fig. 7 Message Sequence Diagram in accordance with the user's command

그림 7은 현재 설계된 데이터 수집 솔루션에서 센싱 데이터를 수집하여 사용자의 명령으로 IAN을 제어하기까지의 Sequence Diagram 이다.

3.2. MQTT Topic 설계

안정적인 사물인터넷 데이터 수집 솔루션을 위하여 주제를 분류하여 MQTT Topic을 설계하였다. Subscriber는 각 항목별로 주제를 분석하여 원하는 동작을 수행한다. 아래의 표처럼 데이터는 크게 Location, Device ID, Sensor Type, Subject 네 가지의 주제로 분류되며 나머지는 옵션 Topic이다.

Table. 4 MQTT Topic Design

Topic	Option	Meaning
Location	X	Location information of the ISN and IAN
Device ID	X	Unique identifier of the ISN and IAN
Sensor Type	X	ISN or IAN
Subject	X	Subject of data
Action	O	The operation of IAN
Condition	O	IAN operating conditions
Value	O	Value of detailed data

표 4에서 표현하는 바와 같이 Publisher와 Subscriber는 그림 8과 같이 기본적으로 “location/deviceID/sensor Type/subject” 형태의 Topic 구조를 통하여 데이터를 전달한다.

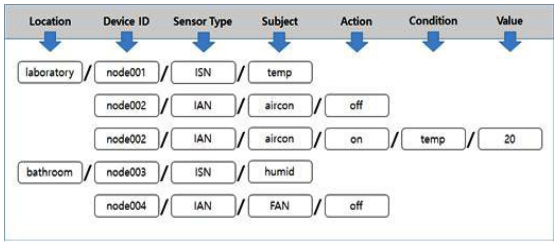

Fig. 8 Hierarchical data structure using the designed Topic

Action, Condition, Value의 3가지의 값은 선택적으로 사용한다.

3.3. 데이터 수집 시스템 구현

데이터 수집을 위한 시스템 구현을 소개하고 있으며, 브로커 테스트와 가상의 ISN, IAN, IBS, Data Server, Push Server 구현에 대한 상세 설명을 기술하였다.

3.3.1. 브로커 테스트

broker는 다른 상용제품에 비해 성능이 우수하고 오픈소스로 활용할 수 있는 Mosquitto MQTT broker를 사용한다. 경량의 프로그램으로 소형 디바이스에 사용하기 적당하다. Mosquitto 설치 후 커맨드 창에 mosquitto -v 명령을 실행한다. 그림 9와 같은 1883 기본포트를 수신하고 있다.

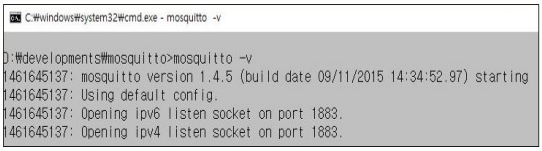

Fig. 9 Mosquitto Broker Execution screen

이후 다른 커맨드 창을 연 후, mosquitto_sub –h IP –t /topic 을 입력하면 IP 주소로 수신되는 topic이라는 주제의 데이터를 Subscribe 하게 된다. 또 다른 커맨드 창에서 mosquitto_pub -h IP -t /topic –m “test”를 입력하면 입력한 IP 주소의 topic으로 “test”란 메시지를 Publish한다.

현재의 테스트는 같은 로컬 망 안에서 이루어 졌지만 네트워크에 연결되어 있는 경우 외부 망에서도 IP와 포트를 통하여 데이터를 Publish하거나 Subscribe하는 것이 가능하다.

3.3.2. 가상의 ISN, IAN 구현

ISN과 IAN는 실제가 아닌 가상의 ISN과 IAN을 구현하여 데이터 수집 솔루션을 설계한다. ISN은 앞에서 설명한 데이터 Publish 방법을 응용하여 구현한다. ISN은 Java 시스템 커맨드 프로그램을 사용하여 1초에 한 번씩 동일한 Topic에 20~24도의 온도 값을 IBS로 Publish한다. 이때 Topic 값은 “laboratory/node001/ISN/temp” 하여 온도 값을 전송한다. 그림 10은 메시지 수신 확인을 위한 기본 테스트이다.

IAN은 MQTT Client의 역할을 해야 하기 때문에 별도의 모듈을 설치해야 한다. 본 논문에서는 Node.js 환경에서 npm의 MQTT 모듈을 활용하였다. IBS에서는 ISN에서 특정데이터를 수신하거나 Client의 요청으로 데이터가 감지되었을 때 IAN을 제어하기 위한 topic에 특정 데이터를 Publilsh한다. 이 Topic을 Subscribe하고 있는 IAN은 데이터를 수신하고 메시지 수신 이벤트 때의 콜백 함수로 원하는 액추에이터를 동작시킨다.

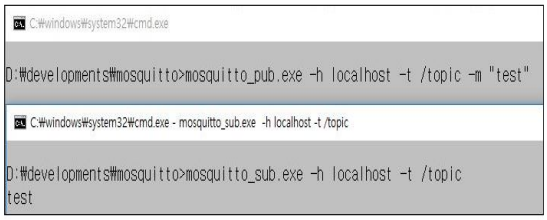

Fig. 10 Publish/Subscribe test using Mosquitto Broker

3.3.3. IBS의 구현

IBS는 크게 두 가지의 역할을 한다. 데이터 Topic을 관리하는 MQTT Broker의 역할과 Data Server로 데이터를 저장하거나 조회하는 MQTT Client 역할을 한다. MQTT Broker는 위의 3.2 장에서 설계한 topic들의 주제를 모두 관리한다. 따라서 센싱 데이터의 관리나 액추에이터 명령 데이터 등 모든 데이터의 구조관리를 하며 Publisher와 Subscriber 사이에서 중개자 역할을 한다. 또한 ‐q 옵션으로 QoS를 설정하여 3단계로 데이터의 신뢰성을 보장할 수 있다. 본 논문에서는 반복적인 데이터를 처리하기 때문에 디폴트 값인 QoS 0 값으로 테스트를 진행한다. MQTT Client는 다른 ISN과 IAN처럼 일반적인 Publisher와 Subscriber의 역할을 수행한다. IBS에서의 MQTT Client는 ISN으로부터 Topic으로 전달된 모든 데이터를 Data Server에 주기적으로 저장한다. 실제로 “laboratory/node001/ISN/+” 와 같이 와일드카드 Topic을 사용하여 데이터를 Subscribe한 결과 ISN으로부터 “laboratory/node001/ ISN/temp” Topic으로 Publish 된 데이터를 주기적으로 수신하는 것을 확인할 수 있었다. 또한 데이터를 받는 message 이벤트 때 콜백 함수로 데이터베이스에 저장하는 쿼리를 수행함으로서 데이터를 받음과 동시에 Data Server에 저장하였다. 반대로 Client로부터 온 명령 Topic을 Subscribe하고 있는 IAN에게 전달하여 액추에이터를 동작시키는 역할도 수행한다.

3.3.4. Data Server 구현

Data Server는 IBS로부터 수집된 데이터를 저장하는 역할을 한다. Data Server에 저장되는 데이터는 MQTT Topic과 일치하는 속성 값으로 구성한다. 그림 11은 IoT Device를 테이블 형태로 표현한 데이터 구조이다.

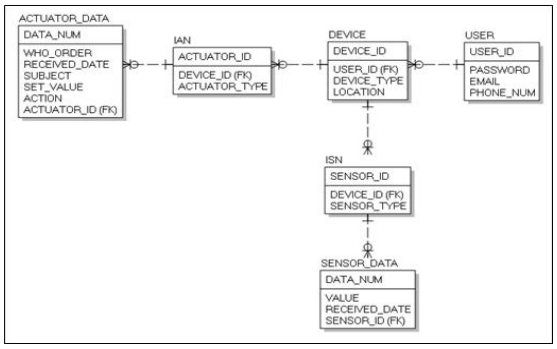

Fig. 11 Data table structure of IoT Device

3.3.5. Push Server 구현

Push Server는 Socket.io를 사용하여 구현한다. ISN, IAN과 IBS의 프로토콜과는 다르게 Push Server는 HTTP 프로토콜을 사용하여 Client와 통신한다. 일반적으로 ISN과 IAN은 저사양의 디바이스를 사용하기 때문에 네트워크 환경과 자원을 고려하여 가벼운 MQTT 프로토콜을 사용할 필요가 있지만, Push Server는 일반적으로 고사양의 PC를 사용하기 때문에 HTTP 프로토

콜을 사용한다. 또한 대중적인 프로토콜이기 때문에 Client에서 별도의 애플리케이션 설치 없이 웹 브라우저 하나로 손쉽게 사용이 가능하다는 장점이 있다. 구현을 위하여 Node.js 서버와 npm의 socket.io 모듈을 사용한다. Push Server는 JavaScript, HTML 파일로 구성되는데 JavaScript 파일에서 특정 포트로 connection 이벤트를 설정하여 접속할 수 있도록 구현한다. connection이 완료되면 웹 서버는 최근 온도 값 10개의 평균을 구하여 설정 값 보다 높을 때 사용자에게 알림을 전달한다.

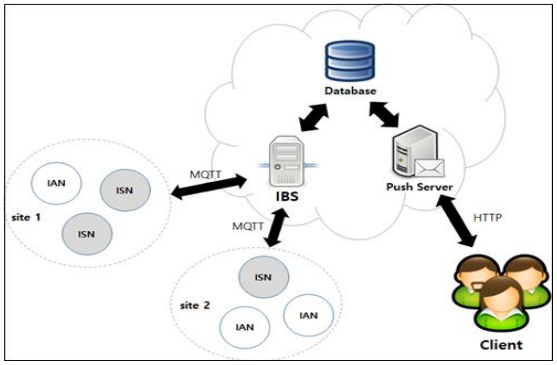

Fig. 12 IBS, Database, the role of the representative of the Push Server

Fig. 13 Notification message that Client has received from the Server

그림 12와 같이 IBS, Database, Push Server는 IBS를 중심으로 연결되어 Client와 IoT Device 사이를 연결시켜 준다. ISN을 통하여 IBS로 수집된 센싱 데이터는 곧바로 Database에 저장되며 Push Server를 통해 Client에게 실시간으로 서비스된다. 그림 13은 Client가 받은 알림 메시지와 IAN을 제어했을 경우에 받는 메시지를 캡처한 것이다. Client는 알림 메시지를 받고 명령 패킷을 IBS에 전송하여 특정 IAN을 제어하는 것이 가능하다.

Ⅳ. 성능 테스트

제안하는 시스템을 검증하기 위하여 성능 테스트를 통해 MQTT와 HTTP 프로토콜 특성을 비교 분석한다.

4.1. 시스템 환경 구축

실제와 유사한 환경에서 테스트를 진행하기 위하여 데이터 수집 서버인 IBS를 로컬 환경에 구축하였다. IBS에는 시스템에서 사용하는 Mosquitto MQTT Broker와 비교 대상인 Node 기반 HTTP 웹서버를 설치하였다. 그림 14는 테스트 환경을 도식화하였다.

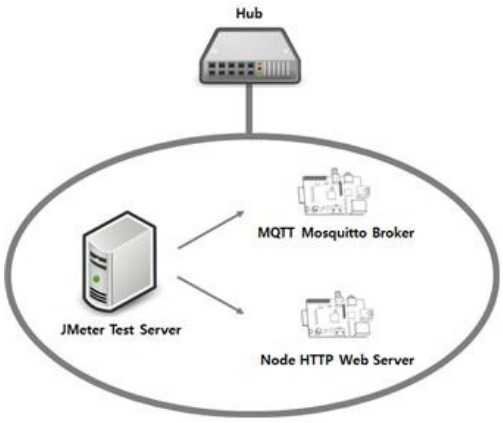

Fig. 14 Configuration of the test environment

4.2. 시나리오 적용

본 실험에서는 수집되는 센싱 데이터의 처리량과 처리속도에 초점을 맞추어 실험을 진행한다. 주기적으로 센싱 데이터를 추가하는 Stepping Test로 실험을 진행하였으며, 표 5와 같이 실험 기준을 정하고 실험 조건을 기술하였다.

- 센싱 데이터를 서버로 10초에 100개씩 전송하여 500개까지 증가시킨다.
- 각 센싱 데이터들을 100단위로 5초 동안 유지시킨다.
- 최대치 500이 되면 10초 동안 상태를 유지한다.

Table. 5 Scenario

Test time (second)	Sensing data (count)	Interval of increase (second)	Hold Time (second)
40	0 ~ 400	1	5
	500	X	10

본 실험은 센싱 데이터가 동시에 100개부터 500개까지 동시에 서버로 전송되는 시나리오를 나타낸다. 센싱 데이터가 많지 않은 소규모 사물인터넷 시스템과 센싱 데이터가 많은 대규모의 사물인터넷 시스템에서 어떤 프로토콜이 적당한지 해당 실험을 통해 파악하는 것이 가능하다.

4.3. 성능 분석

HTTP와 MQTT 처리량 분석과 분석에 대한 성능 테스트 비교를 기술하였다.

4.3.1. HTTP Node Web Server 처리량 분석

그림 15는 Node Web Server에서 HTTP 프로토콜의 처리량을 보여주는 그래프이다. 센싱 데이터가 전송되는 5초 구간에 처리량이 증가하여 종료되는 시점까지 평균적으로 초당 약 180의 처리량을 유지한다.

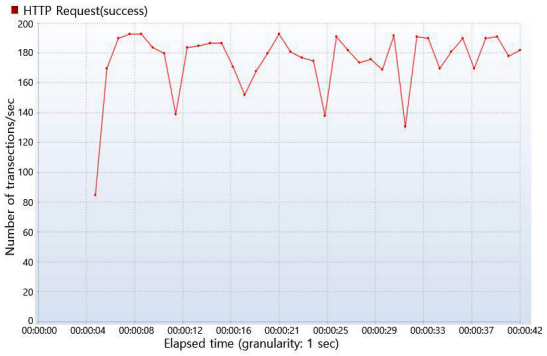

Fig. 15 HTTP Node Web Server throughput graph

4.3.2. MQTT Mosquitto Broker 처리량 분석

그림 16은 Mosquitto Broker에서 MQTT 프로토콜의 처리량을 보여주는 그래프이다. 센싱 데이터가 전송되는 5초 구간에 처리량이 증가하여 초당 약 12,000의 처리량을 유지한다.

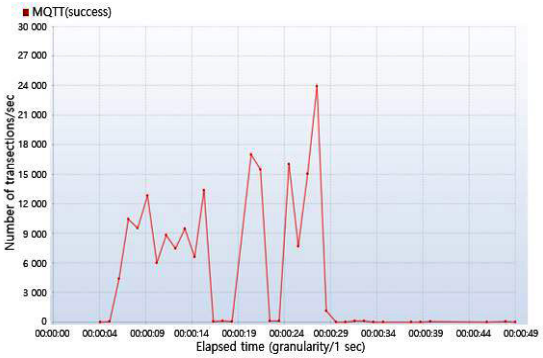

Fig. 16 MQTT Mosquitto Broker throughput graph

하지만 30초 구간에 500개의 센싱 데이터가 추가되는 시점부터 데이터가 유실되어 처리하지 못하는 것으로 판단된다. 이는 Mosquitto Broker의 최대 커넥션 허용 개수가 1,024개인 시점에서 Publisher와 Subscriber가 각각 500개씩 커넥션을 사용한 결과로 판단된다. 또한 QoS 0으로 센싱 데이터를 전송했기 때문에 유실된 데이터를 끝까지 처리하지 않은 것으로 간주된다. 그럼에도 불구하고 186,916개의 센싱 데이터를 처리했으며 초당 처리량 4,301.7을 기록했다. 이는 HTTP 처리량의 24배에 육박하는 수치이다.

4.4. 성능 테스트 요약

앞에서 진행했던 성능 테스트와 추가적으로 진행한 테스트에 대하여 요약한다. 다음 표는 앞에서 진행했던 결과와 추가적으로 진행한 테스트에 대한 성능 테스트를 요약한 것이다.

표 6에서 성능 테스트를 위해 각 서버의 응답시간과 처리량에 대하여 분석하였다. MQTT Mosquitto Broker는 Node Web Server보다 초당 처리량이 약 24배 우수하고 응답시간이 약 448배 빨라 센싱 데이터를 처리하는데 높은 안정성을 보였다. MQTT의 경우에는 1,024 이상의 커넥션을 지원해 주는 Broker를 사용하였을 때 안정성이 더 향상될 것으로 판단되고, HTTP의 경우에는 시스템 사양이 높을 때 안정성이 더 향상될 것이라 판단된다. 해당 실험을 통하여 저사양의 소형 PC에서 기존에 주로 사용했던 HTTP보다 MQTT 프로토콜이 높은 안정성을 보이는 것을 증명하였다.

Table. 6 Performance Test Summary Table

Division	MQTT Mosquitto Broker (QoS 0)	MQTT Mosquitto Broker (QoS 1)	HTTP Node Web Server
The number of the processed sensor data(count)	186,916	13,210	6,663
Minimum response time(ms)	0	16	13
Maximum response time(ms)	1,621	4,718	5,674
Average response time(ms)	4	664	1,790
Error rate(%)	0.00	0.00	0.00
throughput (The number of jobs/sec)	4,301.7	351.9	176.8
Processing data size(KB/sec)	0.00	0.00	25.04
Data limits(count)	500	500	700

V. 결론 및 향후연구

현재 방대한 데이터들을 처리하기 위하여 사물인터넷 플랫폼 시장에서는 좀 더 신속하고 안정적인 통신 프로토콜을 지향한다. 하지만 기존의 HTTP 시스템의 경우에는 데이터 전송 효율과 안정성이 부족한 상황이고, 기업들이 제시하고 있는 자체 사물인터넷 표준은 상호 데이터 교환 호환성의 문제가 있다.

이에 본 논문에서는 안정적인 사물인터넷 플랫폼을 위한 MQTT 기반의 데이터 수집 솔루션을 제안하였다. 본 논문에서 제안한 데이터 수집 솔루션을 이용하여 안정적이고 신속한 사물인터넷 플랫폼을 구축하는 것이 가능하였다. 또한 표준 사물인터넷 프로토콜을 사용하였기 때문에 사물인터넷 개발자들의 생산성을 높일 수 있고 클라이언트에게 빠른 응답을 기대하는 것이 가능하다. 향후 연구에서는 이식성 좋은 사물인터넷 플랫폼을 위하여 IEEE 802.15.4에서 사용할 수 있는 MQTT-SN 프로토콜을 활용한 데이터 수집 솔루션에 대하여 연구할 계획이다.

ACKNOWLEDGMENTS

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(2015R1D1A1A01059862) and R&BD Program through INNOPOLIS funded by the Ministry of Future Creation and Science (A2015DD122)

REFERENCES

[1] Gartner, Inc. Gartner Says 6.4 Billion Connected “Things” Will Be in Use in 2016, Up 30 Percent From 2015[Internet] Available: http://www.gartner.com/newsroom/id/3165317

[2] Raphael Cohn and Richard Coppen. OASIS Message Queuing Telemetry Transport (MQTT) TC[Internet]. Available: https://www.oasis-open.org/committees/tc_home.php?wgabbrev =mqtt

[3] M. Prihodko, “Energy Consumption in Location Sharing Protocols for Android Applications,” Master's thesis, Linkoopings University, Sweden, 2012.

[4] Seung-Hyun Shim, and Hak-Beom Kim, “Internet of Things and MQTT technology,” *REVIEW OF KIISC*, vol. 24, no. 6, pp. 40-41, Dec. 2014.

[5] The HiveMQ Team. MQTT Essentials Part 5: MQTT Topics & Best Practices[Internet]. Available: http://www.hivemq.com/blog/mqtt-essentials-part-5-mqtt- topics-best-practices

[6] S. Behnel, L. Fiege, G. Muehl, “On Quality-of-Service and Publish-Subscribe,” in *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems Workshops*, Lisboa, Portugal, pp. 20, 2006.

[7] Lee, Shinho; Kim, Hyeonwoo; Hong, Dong-kweon; Ju, Hongtaek “Correlation analysis of MvQTT loss and delay according to QoS level,” *The International Conference on Information Networking 2013 (ICOIN)*, pp.714-717, Jan. 2013.

[8] V.Pimentel, B.G.Nickerson, “Communicating and Displaying Real-Time Data with WebSocket,” *IEEE Internet Computing*, vol. 16, no. 4, pp.45-53, May 2012.

[9] Web Sockets: Bidirectional communication technology for web apps[Internet]. Available: http://caniuse.com/#search =WebSocket

김상현(Sang-Hyun Kim)

2014년 2월: 한밭대학교 전파공학과 (공학사)
2016년 2월: 한밭대학교 전파공학과 (공학석사)
※관심분야 : 사물인터넷, 웹 프로그래밍, 데이터베이스

김동휘(Dong-Hwi Kim)

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

오형석(Hyeong-Seok Oh)

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

전현식(Hyun-Sig Jeon)

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

박현주(Hyun-Ju Park)

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