Node.js, MQTT 및 MongoDB 를 이용한 효율적이고 유연한 센서 리액션 시스템 개발

민경현*, Rustam Rakhimov Igorevich **
*휘문고등학교 **건국대학교 컴퓨터공학부

Development of Efficient and Flexible Sensor Reaction System based on Node.js, MQTT and MongoDB

Kyung-Hyun Min*, Rustam Rakhimov Igorevich**
*Whimoon High School
** Department of Computer Science and Engineering, Konkuk University, Seoul, 143-701, Korea
E-mail : khmin1104@gmail.com, rusyasoft@gmail.com

요 약

IoT 가 다양한 형태로 IT 기술을 이용하는 사물의 서비스를 제공하고 있다. 보다 스마트한 IoT 환경을 만들기 위하여 센서 데이터를 효율적으로 컨트롤하는 시스템이 필요하다. 본 논문은 계속적으로 생성되는 IoT 센서 데이터를 효율적으로 처리하는 시스템을 Loosely Coupled 하고 확장 가능한 Sensor Reaction System 의 아키텍처로 구성하고 있다. 본 논문에서는 Node.js, MQTT, MongoDB 기술을 사용함으로써 이벤트 기반한 stream 및 batch 처리를 할 수 있도록 센서 데이터를 메시지 큐에서 효율적으로 처리한다. 본 논문에서는 아두이노 보드에 온도 및 빛 센서를 부착한 센서 디바이스로부터 센서 데이터를 받아서 PC 기반의 MQTT Broker / Sensor Reactor / MongoDB 서버 시스템을 구축하고 성능을 분석하였다.

1. 서론

사물인터넷 시대에서 다양한 형태로 IT 기술을 이용한 사물의 서비스가 확산되고 있다. 보다 스마트한 IoT 환경을 만들기 위하여 센서의 계속적으로 생성되는 데이터를 효율적으로 컨트롤하는 시스템이 필요하다.

본 논문은 계속적으로 생성되는 센서 데이터를 효율적으로 처리하고 시스템을 설계 및 구현하였다. 데이터 생성자와 소비자 사이의 Loosely Coupled 된 연결 관계를 구성하며, 실행 시 필요에 따라서 동적으로 설정하며 확장 가능한 Sensor Reaction System 의 아키텍처를 제시한다. 본 연구에서는 데이터의 효율적인 처리를 위하여 Node.js 를 사용하였다.

또한 본 논문에서는 확장된 아키텍처 구조를 제시한다. 확장된 아키텍처 구조에서는 유연한 구조를 제공하는 Publish-Subscribe 형태의 MQTT 기술을 효과적으로 사용하고 있다. 본 시스템의 확장성을 통하여 다양한 IoT 디바이스의 센서 데이터들을 수집, 저장, 분석을 여러 가지 Reaction Server 들을 이용하여 진행할 수 있는 구조를 제공하여, 다양한 IoT 서비스들을 생성할 수 있다.

이와 같이, Node.js, MQTT, MongoDB 기술을 사용하고 아두이노 보드에 온도 및 빛 센서를 부착한 센서 디바이스와 PC 기반의 MQTT Broker / Sensor Reactor / MongoDB 서버를 구축하였다. 그리고 이 서버를 이용하여 Turnaround time 측정과 센서 데이터 분석을 하는 실험을 하였다.

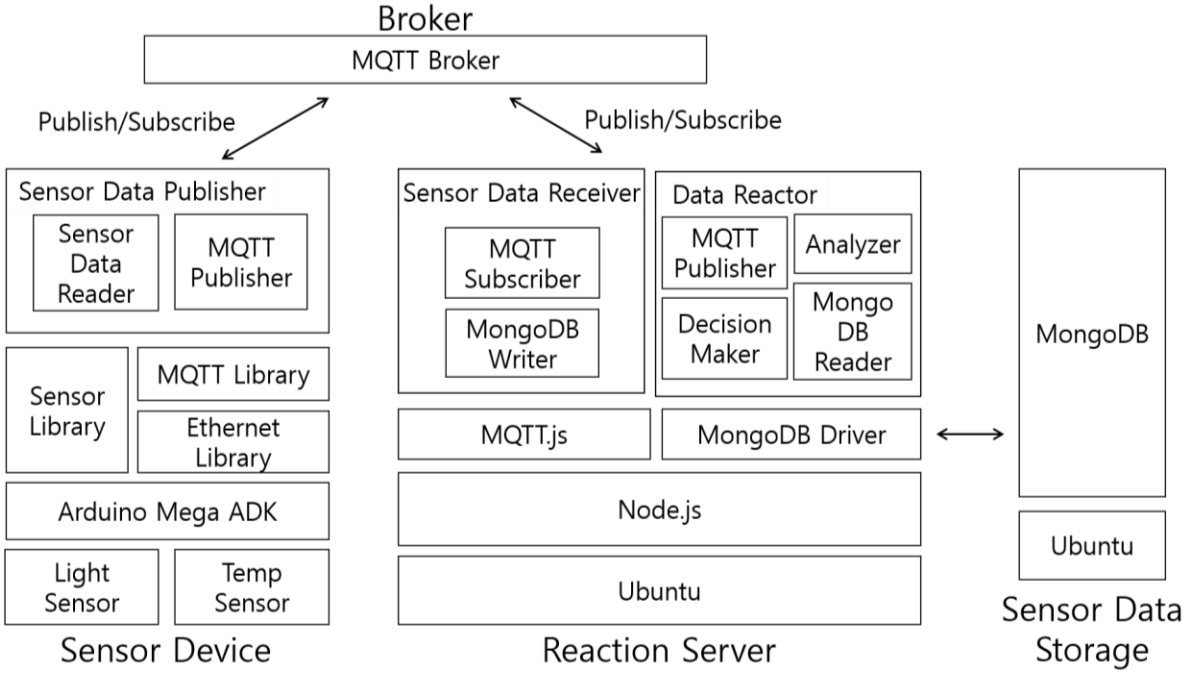
본 논문의 구조는 다음과 같다. 2 절에서는 본 논문에서 제시한 Sensor Reaction System 의 아키텍처 구현에 사용된 요소 기술들을 소개한다. 3 절에서 본 논문의 핵심인 시스템 아키텍처를 제시하고 그 특징과 확장구조를 제공한다. 4 절은 실제 구현하고 성능측정을 해 본 실험환경과 실험결과를 설명한다.

2. 관련 연구

2.1 MQTT

MQTT(Message Queuing Telemetry Transport)는 publish-subscribe 형식을 기반으로 한 가벼운 메시지 프로토콜이다. MQTT 는 M2M(machine-to-machine), IoT device 을 연결하는 프로토콜로 사용된다. MQTT 의 장점은 배터리를 적게 쓰고 데이터 패킷의 크기가 작고 여러 수신자가 있을 때 분배하기 쉽게 만들었다.

Publish-subscribe 구조는 한 데이터를 다른 여러 subscriber 들에게 보내는 일대다 형식이다. Pub-sub 방식에는 non-buffer, buffer 의 두가지 구조가 있다. 이는 buffer 의 유무에 따라 나뉜다



(그림 1) 센서 리액션 시스템 구조

Non-buffered 구조는 subscriber 가 publisher 를 알아서 subscribe 를 신청하면 publisher 가 subscriber 의 신청을 받아주고 subscriber 들을 관리하는 구조이다. publisher 가 publish 할 때 subscriber 에게 데이터를 전달한다. Buffered 구조는 MQTT 의 구조와 같아서 publisher 와 subscriber 사이에 buffer 와 buffer 관리자, 즉 MQTT 에서는 broker 를 둔다. Broker 는 publisher 와 subscriber 를 관리하고 publish 된 데이터 등을 queue 를 통해 관리한다 [2][3]. 이는 Non-buffer 방식과 달리 subscriber 가 publisher 를 몰라도 된다. 또한 publisher 와 subscriber 를 더 확장 시켜도 된다. Subscriber 는 자신이 관심있는 카테고리에 구독하면 된다. 그러면 publisher 가 데이터를 보낼 때 구독한 subscriber 에게 데이터가 전달된다. publisher 와 subscriber 의 관계가 Loosely Coupled 되어 publisher 의 변화에 자유롭다.

2.2 Node.js

Node.js 는 비동기 I/O 를 제공하는 이벤트 기반 아키텍처 기반의 JavaScript 실행환경이다. 따라서 많은 입력 출력 처리를 해야 하는 실시간 응용에서 효율적인 처리가 가능하고 확장성이 좋다. Node.js 는 JavaScript 언어로 되어있고 이벤트 기반, 비동기(Non-Blocking) I/O 모델로 가볍고 효율적이다. 다른 이벤트 기반 모듈과 달리 싱글 스레드를 사용하여 처리를 한다. Npm(Node Packaged Modules)은 JavaScript 언어를 위한 라이브러리를 제공해준다. npm 은 Node.js 를 위한 라이브러리를 제공해준다. 본 논문에서는 npm 이 제공하는 MQTT.js 와 MongoDB Driver 를 사용한다 [4].

2.3 MongoDB

MongoDB 는 문서 지향 데이터베이스, 즉 NoSQL 데이터베이스이다. MongoDB 는 JSON 과 비슷한 형식인 BSON 형식의 데이터를 Document 방식으로 저장하는 No-SQL DB 이어서 다양한 형식의 센서데이터를 저장하고 처리하는데 적합하다. MongoDB 는 데이터의 구조가 다양하든 자주 바뀌든 다 다룰 수 있고 실시간으로 데이터를 받을 수 있을 정도로 빠르다. 그리고 MongoDB 를 분산 서버 형식으로 확장해 나갈 수 있는 구조를 가지고 있다.

3. 시스템 아키텍처

본 논문에서 제시하고자 하는 것은 스트리밍 센서 데이터를 받아 상황에 맞게 리액션해 주는 시스템 구조이다. 시스템의 구조는 크게 Sensor Device, Reaction Server, Broker, Sensor Data Storage 로 되어 있다. Sensor Device 는 센서 데이터를 publish 한다. Reaction Server 는 데이터를 받아 저장한다. Broker 는 publish 와 subscribe 하는 것을 관리한다. Sensor Data Storage 는 센서 데이터를 저장한다. (그림 1)은 이 시스템 아키텍처를 보여주고있다.

3.1 Sensor Device

Sensor Device 는 Sensor Data Publisher 가 빛과 온도 센서의 데이터를 읽어 publish 하는 역할을 한다. Sensor Data Publisher 는 센서 데이터를 읽어 들여서, 그 데이터를 publish 하는 역할을 한다. Sensor Data Publisher 는 빛과 온도의 Sensor Library 를 통하여 읽는다. 그리고 MQTT Publisher 는 MQTT Library 를 통해 그 센서 데이터를 JSON 형식으로 문장을 작성하여 MQTT Broker 에게 publish 한다.

Ethernet Library 는 socket 등의 메소드 콜을 제공해준다. MQTT Library 는 Ethernet Library 위에서 MQTT 프로토콜을 제공한다. Sensor Library 는 각 센서에서 정보를 얻을 수 있게 한다. 이 라이브러리들은 Arduino Mega ADK 가 제공해준다 [1].

3.2 Reaction Server

Reaction Server 는 두가지 프로세스로 나뉘는데 센서 데이터를 받는 프로세스 Sensor Data Receiver 와 데이터를 분석하여 리액션을 해주는 프로세스 Data Reactor 이다.

Sensor Data Receiver 는 subscribe 한 데이터를 Sensor Data Storage 에 저장을 한다. MQTT Subscriber 는 Sensor Data Publisher 가 publish 한 데이터를 MQTT.js 를 통해 subscribe 를 하여 데이터를 받는다. 그리고 MongoDB Writer 가 그 데이터를 MongoDB Driver 를 통하여 MongoDB 에 insert 한다.

Sensor Data Receiver 는 Node.js 에서 작동하고 있다. 이로 인해 비동기처리 방식으로 효율적으로 데이터를 처리한다. MQTT.js 는 Node.js 가 MQTT 프로토콜을 사용할 수 있게 하는 라이브러리이다. MongoDB Driver 는 Node.js 가 MongoDB 의 client 가 되어 MongoDB 를 수정하고 추가할 수 있게 하는 라이브러리이다.

Data Reactor 는 Sensor Data Storage 에 있는 데이터를 읽어 들이고 그 데이터를 가공하여 Broker 에게 publish 한다. MongoDB Reader 는 MongoDB 에 있는 JSON 형식의 센서 데이터를 읽어 들인다. Analyzer 는 그 데이트를 가지고 분석하여 어떤 리액션을 할지 Decision Maker 에게 전달한다. 예를 들어 Analyzer 가 빛 센서 데이터를 통해 빛의 세기가 어두워졌다 라고 분석하면 Decision Maker 가 불을 켜라 라는 리액션을 결정한다. 그럼 MQTT Publisher 가 그 리액션을 Broker 에게 publish 한다. 이렇게 리액션을 Broker 에게 publish 하므로 이 Data Reactor 에게 관심 있는 또 다른 머신이나 장비가 연결되어 사용할 수 있게 하는 장점이 있다.

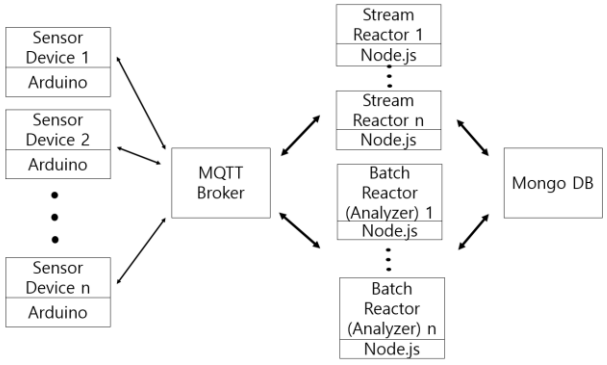
3.3 Broker

Broker 는 MQTT Broker 로 queue(broker)를 사용한 pub-sub 기반의 비동기 커뮤니케이션 구조이다. Broker 는 Sensor Device 와 Reaction Server 에서 publish 하는 데이터들을 queue 에 집어넣어 관리하고, subscriber 에게 전달한다. 그리고 publisher 와 subscriber 를 관리한다. 이러한 Pub-sub 구조의 장점은 하나의 데이터를 다른 여러 머신들에게 나누어 줄 수 있는 것이다. 또한 Broker 를 사용함으로써, subscriber 가 publisher 와 Loosely Coupled 하기 때문에 publisher 가 바뀌더라도 무관하다.

3.4 Sensor Data Storage

Sensor Data Storage 는 Sensor Data Receiver 와 Data Reactor 가 읽고 쓰는 것을 허용하고 그것들을 저장하는 역할을 한다. MongoDB 는 Sensor Data Receiver 가 JSON 형식의 데이터를 MongoDB Driver 를 통해 보내면 그것들을 저장한다. JSON(JavaScript Object Notation)은 가벼운 데이터 교환 방식이다. MongoDB 는 NoSQL 형식으로 JSON 형식과 비슷한 BSON 형식으로 저장하여 데이터의 SQL 에서의 Field 의 변경과 추가가 가능하다. 즉, 데이터의 형식이 변경 가능한 구조이다.

3.5 아키텍처의 확장 구조



(그림 2) 본 시스템의 확장성

본 시스템은 (그림 2)와 같이 확장된 pub-sub 구조의 형태로, 여러 publisher (Sensor Device)들과 여러 subscriber (Reaction Server)들로 확장할 수 있다. Broker 는 각 센서가 보내는 데이터를 카테고리 별로 구분하여 각 queue 에 넣는다.

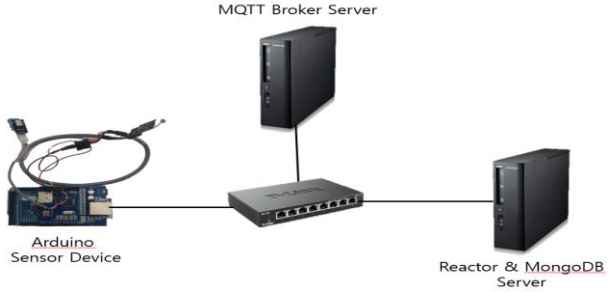
Reaction Server 를 두가지 Stream Reactor 와 Batch Reactor 로 나눌 수 있다. Stream Reactor 들은 관심이 있는 센서 데이터들을 받기 위하여 subscribe 한다. Batch Rector 는 DB 에 저장되어 있는 데이터를 필요에 따라서 분석하여 리액션을 한다. 이렇게 Reaction Server 들은, 해당 분야에 필요한 센서 데이터를 사용하여 분석하여 새로운 정보를 만들어낸다. 이후 MQTT 에 publish 하거나 DB 에 저장할 수 있다.

이러한 Reaction Server 들을 추가함으로써 본 시스템을 더욱 확장할 수 있다. 만일, Broker 서버가 죽으면 (그림 2)의 사례에서는 사용할 수 없으므로 Broker 를 여러 컴퓨터로 나누어 분산적으로 처리하여 서버가 죽지 않게 할 수 있다. 본 시스템의 확장성을 통하여 다양한 IoT 디바이스들의 데이터들을 수집 및 분석하여 IoT 서비스들이 생성되는 구조를 만들 수 있다.

4. 실험

4.1 실험환경

본 실험에서 사용한 시스템의 구조도를 (그림 3)이 나타내고 있다 본 실험에서는 시스템 구조 상 Sensor Data Server 를 Reaction Server 와 함께 두었다. <표 1>은 각 Sensor Device, MQTT Broker Server, Reactor & MongoDB Server 의 세부 실험 환경을 제시한다.



(그림 3) 실험 환경

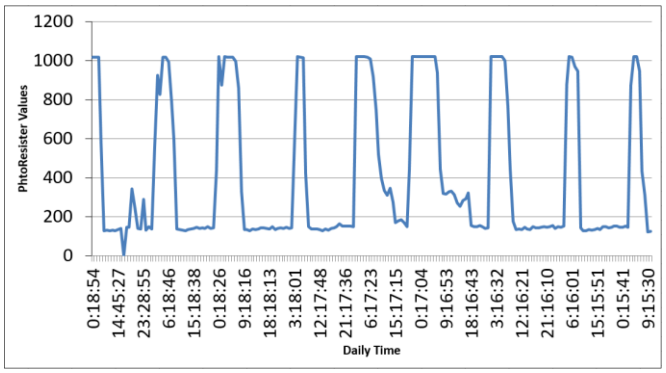
<표 1>세부 실험 환경

기기	기기 사양
Arduino Sensor Device	Arduino Mega ADK, Arduino Ethernet Shield, Photo Resistor, DHT(Digital Humidity Temperature) Sensors
MQTT Broker Server	PC: Intel(R) Core(TM)2 CPU 6300 1.86GHz, RAM 2GB Linux Ubuntu-14.04.1 LTS, Moquette-0.7
Reactor & MongoDB Server	PC: Intel(R) Core(TM)2 CPU 6300 1.86GHz, RAM 2GB Linux Ubuntu-14.04.1 LTS, NodeJS-0.10.25, MongoDB-2.4.9

4.2 실험 결과

첫번째 실험은 하나의 온도 센서 데이터에 관하여 센싱한 장비에서부터 본 논문에서 제시한 리액션 처리를 하고 돌아오는 시간을 측정한 것이다. Sensor Device → Broker Server → Reactor Server → Broker Server → Sensor Device 로 갔다가 오는 시간(Turnaround Time)을 측정했다. 이 실험을 통하여 평균 11ms(millisecend)의 센싱부터 처리까지의 시간을 측정하였다.

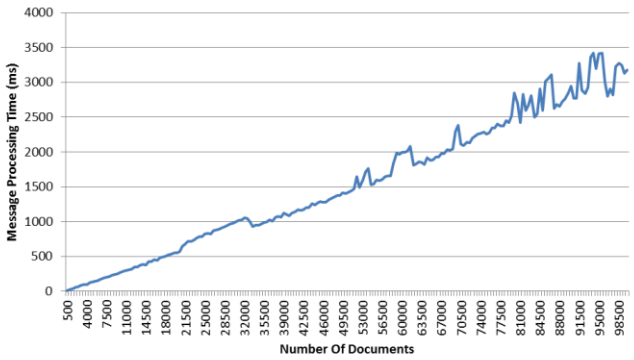
두번째 실험은 몇 일간의 빛 센서 데이터를 모아보고 분석한 것이다. (그림 4)는 빛 센서 데이터를 한시간 마다 묶어 평균을 내어 나타낸 것으로 8 일간의 데이터를 보여준다. 데이터들은 10 분 마다 센서가 측정한 실제 값이다. 센서 데이터들을 0~1024 사이 값으로 나타낸다. 가장 큰 1024 값은 빛을 감지 못한 것으로 가장 어둡다는 것을 의미한다. 0 값은 그 반대로 가장 밝을 것을 의미한다. 주로 0 값은 센서가 햇빛을 직접적으로 받을 때 일어난다. 본 실험에서는 실험 데이터를 통하여 여러 가지 분석을 할 수 있다. 첫째 빛의 주기를 통해 하루를 예측할 수 있다. 둘째 각 하루의 데이터를 구룹핑하여 데이터를 분석하여 구룹 데이터의 양상을 알고 더 나아가 미래를 추측할 수 있다. 또한 빛의 세기가 약하면 '전등을 켜라'라는 결론을 이끌어 낼 수 있다.



(그림 4) 빛 센서 데이터 그래프 (시간당 평균값)

세 번째 실험은, DB 에 저장되어 있는 센서 데이터를 500, 1000, 1500… 개씩 모아서 평균을 계산하는데 걸리는 시간을 측정하는 실험이다. 이러한 실험은 대용량의 데이터를 특성에 맞추어서 그룹 별로 처리해야하는 분석에 필요한 실험이다.

(그림 5)는 데이터를 처리하는데 걸리는 평균 처리 시간을 나타낸다. 이 그래프는 데이터와 데이터 처리 시간이 비례적으로 증가하는 것을 알 수 있다. 데이터가 77000 을 넘은 다음부터 처리시간이 불규칙적으로 증가하는 것을 볼 수 있다. 이는 컴퓨터의 메모리 자원의 부족으로 스와핑 처리로 인한 것으로 추정된다.



(그림 5) Average Document Processing Time

5. 결론

본 논문은 IoT 환경에서 Sensor 들로부터 계속적으로 생성되는 데이터를 즉시 처리해서 Reaction 하거나 저장 후 분석하여 Reaction 하는 Sensor Reactor System 의 아키텍처를 설계하였다. 또한, 이 시스템은 IoT 센서 데이터의 stream 및 batch 데이터 처리를 이벤트기반으로 Node.js, MQTT, MongoDB 기술을 이용하여 구축하여 효율적인 센서 데이터를 처리하도록 구현하였다. Turnaround time 을 통해 성능을 분석하고 센서 데이터 분석을 통하여 데이터의 양상를 알 수 있는 실험을 하였다.

참고문헌

[1] Daniel Palma, Juan Enrique Agudo, Héctor Sánchez and Miguel Macías Macías, "An Internet of Things Example: Classrooms Access Control over Near Field Communication", Sensors Volume-14, 2014.

[2] Andy Stanford-Clark and Hong Linh Truong, "MQTT For Sensor Networks (MQTT-SN) Protocol Specification", Version 1.2, November 14, 2013.

[3] Urs Hunkeler, Hong Linh Truong, Andy Stanford-Clark, "MQTT-S – A Publish/Subscribe Protocol For Wireless Sensor Networks",Communication Systems Software and Middleware and Workshops, COMSWARE 2008. 3rd International Conference, 2008.

[4] Andrew John Poulter, Steven J. Johnston, Simon J. Cox, "Using the MEAN stack to implement a RESTful service for an Internet of Things application", Internet of Things (WF-IoT), IEE World Forum, 2015.