JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 20, No. 11 : 2119~2124 Nov. 2016

IoT 게이트웨이 기반의 이벤트 중심 접근 방식 응용프로그램 설계

라이오넬[1] · 장종욱[2*]

Design of IoT Gateway based Event-Driven Approach for IoT related Applications

Lionel Nkenyereye[1] · Jong-Wook Jang[2*]

[1]Department of Computer Engineering, Dong-Eui University, Busan 47340, Korea
[2*]Department of Computer Engineering, Dong-Eui University, Busan 47340, Korea

요 약

사물 인터넷(IoT)은 효율적인 시간 응답 및 처리를 위해 이벤트 중심으로 접근 할 필요가 있다. IoT에서 모바일 기기의 성장은 IoT 응용 프로그램과 관련이 있는 지능형 건물로 연결이 된다. 예를 들어, 홈 오토메이션 제어 시스템은 홈 서버에 액세스하기 위해 스마트 폰이나 웹 서비스에 클라이언트 시스템과 같은 웹 응용 프로그램을 사용하여 제어 명령을 전송 합니다. 홈 서버는 클라이언트 시스템으로부터 명령을 수신 받은 후 조명 시스템을 제어 한다. 게이트웨이 기반의 클라이언트 처리 담당인 RESTful 기술은 '인터넷상에 숨어있는 다수의 클라이언트들에 대한 증명'을 요청한다. 본 논문에서는 동시성 이벤트를 처리하기 위한 IoT 게이트웨이의 설계 작업을 제안한다. NodeJS의 통신 프로토콜 기반의 메시지 지향 미들웨어인 XMPP는 중앙 허브를 통해 게이트웨이에 접속하여 지능형 빌딩 제어 장치의 통신 부분을 처리한다.

ABSTRACT

The Internet of things (IoT) needs to be an event-driven approach for efficient related time response and processing. The growth of mobile devices in Internet of Things (IoT) leads to a number of intelligent buildings related IoT applications. For instance, home automation controlling system uses client system such web apps on smartphone or web service to access the home server by sending control commands. The gateway based RESTful technology responsible for handling clients'requests attests an internet latency in case a large number of clients' requests submit toward the gateway increases. In this paper, we propose the design tasks of the IoT gateway for handling concurrency events. The gateway based event-driven architecture is designed for building IoT gateway using node.js on one side and communication protocol based message-oriented middleware known as XMPP to handle communications of intelligent building control devices connected to the gateway through a centralized hub.

키워드 : 만약 IoT 게이트웨이, 이벤트 기반 아키텍처, Node.js, 지능형 빌딩 시스템, XMPP, 사물의 인터넷.

Key word : IoT Gateway, Event-driven Architecture, Node.js, Intelligent buildings system, XMPP, Internet of Things.

Received 31 October 2016, **Revised** 02 November 2016, **Accepted** 05 November 2016

* **Corresponding Author** Jong-Wook Jang(E-mail: jwjang@deu.ac.kr, Tel:+82-51-890-1709)
Department of Computer Engineering, Dong-Eui University, Busan 47340, Republic of Korea

Open Access http://dx.doi.org/10.6109/jkiice.2016.20.11.2119 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. INTRODUCTION

The intelligent buildings refers to a range of Internet of Things (IoT) applications such as automatic energy metering, home automation and wireless monitoring[1]. Traditionally, the intelligent buildings system use gateway based client server architecture such as web-based mobile application or web service to handle commands to control devices in IoT environment. As the number of clients' requests increases in simultaneously manner, the Representation State Transfer (REST) architecture to handle those requests encounters limitations for performing them successfully, and therefore, slows down predefined actions that take automatically without the help of humans.

This paper focuses on designing a prototype architecture for a gateway and its components of intelligent buildings for IoT applications. The model that we are looking for should have a gateway component with the ability of processing several requests from multiple smart devices simultaneously without blocking and long execution requests. These features are essential for reducing communication costs. In server-side asynchronous event-driven programming [2], the user requests are treated as application events and inserted into an event queue. This event-driven model has been widely adopted in building scalable web applications, mainly through the Node.js [3] framework. The real-time communication allows the implementation of eXtensible Messaging and Presence Protocol (XMPP) to support the IoT device management framework [3]. The implementation of XMPP provides near-real-time communication between multiple devices over multiple networks.

Ⅱ. PROTOTYPE ARCHITECTURE FOR INTELLIGENT BUILDING RELATED IoT APPLICATIONS WITH ITS GATEWAY

The design prototype shown in Fig. 1 consists of database and virtual machines as resources to carry out several functions from the intelligent building controls units. The intelligent building controls units are responsible of sending the measurement of sensors installed on the microcontroller boards. The components on the cloud store readings and status of these intelligent building control units (IBCU) periodically for further analysis. The components on the cloud are also responsible of sending control and monitoring commands to the sensors connected to these IBCUs. The IBCUs are the IoT devices nodes in charge of manage the data to be sent to the management components services on the cloud. Every IBCUs connected to the Wi-Fi network uses a unique IP address that can be used to reach that IBCU. At the IBCUs level, basic instructions are implemented in order to pre-process information on the main gateway. The client application accessible anywhere through either wearable device or mobile device discovers all the connected IBCUs and enables the user to start interacting with them. Each IBCUs broadcasts itself over the network so that the cloud services can discover these IBCUs for reporting to the user.

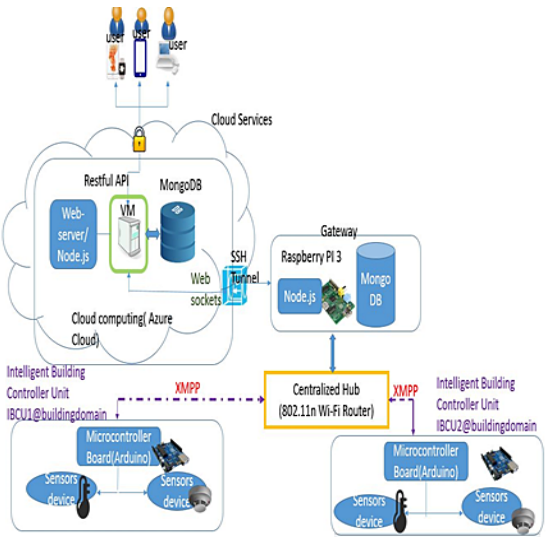

Fig. 1 A prototype architecture for Intelligent Buildings (energy metering, home automation, wireless monitoring) related IoT environment

The Raspberry Pi 3 is the main gateway, whose is to interconnect the IBCUs to the network providers' infrastructure of the system. This gateway runs a node.js in order to process a large number of concurrent connections from the end users using the intelligent building applications. The web socket ensures the communication between the cloud services and the raspberry Pi through an encrypted secure shell (SSH) tunnel. This gateway is also in charge of sending instruction set which is in the format of commands that can be communicated for performing automatic configuration when a new IBCU is connected to the intelligent building related IoT system. The instructions are implemented in JavaScript and are communicated to the microcontroller board (Arduino) devices to request sensors readings or to send commands to control the peripheral sensors.

Finally, the end users are able to interact with the whole intelligent building related IoT applications using a secured REST API, using mobile device and connected wearable devices. A smart phone application through cloud services that give access to the gateway made for the purpose of controlling all the IBCUs currently available on the network. This is done by the functionality of the XMPP protocol. Upon retrieving the IBCUs and make request over the network in the form of messages that are sent to the IBCUs through the centralized hub.

Ⅲ. DESIGN OF THE IoT GATEWAY BASED EVENT-DRIVEN ARCHITECTURE

3.1. Generalized REST Event Model

Adding event-driven processing to REST APIs as an important concept for the emerging Internet of Things.

REST APIs [4] provide for high level system interoperability and software usability, while event driven processing is needed for automatic capability and efficiency An event is a significant change in the state of some element of a system. Events are asynchronous,

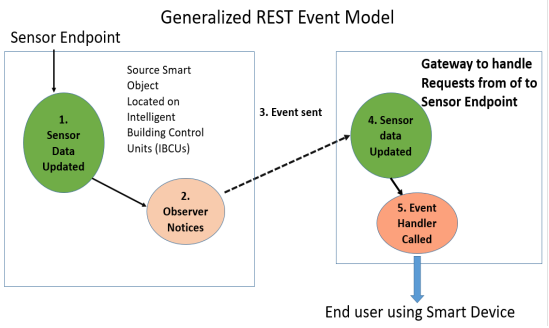

Fig. 2 Generalized REST Event Model to be implemented in the proposed IoT Gateway

that is they occur at arbitrary times relative to other operations in the system. The generalized REST (REpresentation State Transfer) event model is shown on the Fig. 2.

The event of new sensor data updated is processed in the following order :

(1) sensor data are updated when the sensor endpoint makes an observation and sends an update to the IBCUs Observable Property
(2) an Observer Notice resource informs the system to notice updates and forward them according to routing instructions contained in the Observer resource.
(3) The event is routed to the destination resource (Gateway)
(4) The Observable Property (gateway and its associated topic user)is updated.
(5) A handler is invoked based on an observer associated with the local IBCUs. This updates the client via smart phone.

3.2. Publish-Subscriber service based XMPP protocol to handle sensor updated data to the end user

Devices must communicate with each other. Device data then must be collected and sent to the server infrastructure. That server has to share device data, possibly providing it back to devices. The most implemented protocol for allowing communication are XMPP and MQTT [5]. XMPP (eXtensible Messaging

and Presence Protocol) was born as a protocol for messaging apps. Jabber uses it, Google Talk used it. It is a great and reliable messaging protocol, far more reliable than GCM (Google Cloud Messaging)[6] is . MQTT (MQ Telemetry Transport) instead, was born as a communication protocol designed for low-power devices which have limited power capacity and low computational power. It is fast and reliable, and uses far less data and power than XMPP needs to do the same exact work. Both are good and reliable application protocols, even though XMPP relies on HTTP, which makes it automatically more heavyweight but it doesn't mean it's less adequate for this job.

XMPP has a Publish-Subscribe capability, generally referred to as PubSub [7], Publish-subscribe is a well-known technique that is often used in large scale distributed systems. Publish-Subscribe is used when there are lots of events happening, and entities interested in differing subset of the events. A mechanism where each entity independently tried to find out about the events of interest would be complex and inefficient; it requires resource polling which is resource intensive and does not scale to very large deployments.

With the Publish-subscribe, publishers and subscribes are separated. Publishers classify events into topics and each event is sent to the Publish-Subscribe services. Subscribes indicate to the Publish-Subscribe service which topics they are interested in and receive events

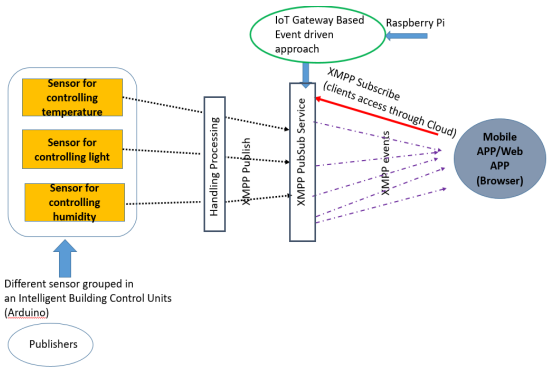

Fig. 3 Design Publish-Subscriber service for collecting and broadcasting new sensor data from IBCUs

associated with each topic as they are sent to the Publish-Subscribe service, giving access to information in real time.

XMPP PubSub was central to the collection of sensor data from IBCUs. It is implemented on the gateway which is implemented on the raspeberry Pi. As shown on the Fig. 3, the brodcasting of new sensed data follows the following flow :

(1) Intelligent Building Controls units as publishers classify events (update of sensor data) into topics (monitoring of interest area by sensor)
(2) Each event is sent to the Publish-Subscribe service on the Gateway
(3) Client users (as subscribers) indicate to the publish-Subscribe service which topics they are interested in
(4) Client users (subscribers) receive message notification associated with interest topic as they are sent request to the Publish-Subscribe service, giving access to information in real time.

XMPP client is run in the browser or Mobile APP on client's smartphone

3.3. Node.js for handling control commands from end users concurrently

Using Node.js allows to write scalable network applications that inherit concurrency in their design. This is accomplished by event-loop. To avoid blocking, all main Application Programming Interface (API)-calls that involve Input/output are asynchronous. Thus, instead of waiting the function to return, the event-loop can execute the next event in the queue, and when the API-call is finished it calls a callback function specified when the call was made. There is an infinite loop running in a single process that continually looks at what event occurred and what callbacks need to be executed. It deals with queuing all events automatically, and it keeps calling the appropriate callbacks as fast as it can. As a developer using Node.js, you do not really need to know this, though you just need to know that

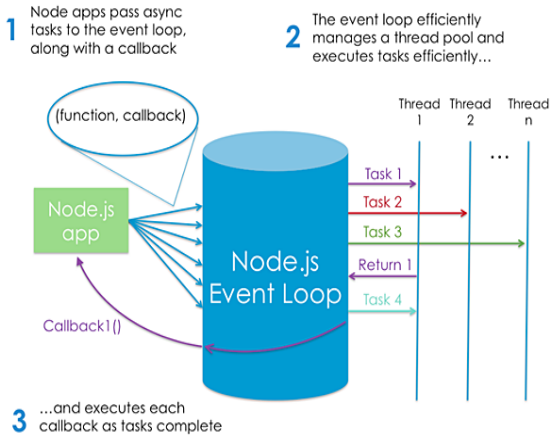

Fig. 4 Configuration diagram for Node.js

your callbacks will be called when events occur as quickly as possible the task completes[8]. See Fig.4 Node.js is based internally around the concept of an event loop. There is an infinite loop running in a single process that continually looks at what event occurred and what callbacks need to be executed. It deals with queuing all events automatically, and it keeps calling the appropriate callbacks as fast as it can [8]. As shown in the Fig. 4. the event loop queues both new requests and blocking I/O requests. The single-thread executes an event loop by setting up a simple mapping of all requests. The event loop gradually dequeuing request from the queue, then processing the request, finally taking the next request or waiting for new requests submitted[8, 9].

Ⅳ. CONCLUSION AND FUTURE WORK

This paper presented a design a prototype architecture for intelligent building that attests the capability of performing a large number of users accessing the intelligent buildings system concurrently. The XMPP protocol allows multiple connected devices to provide real-time communications over multiple networks. With the use of Node.js, an increasing number of requests should not have a negative effect when the information collected from sensors embedded in multiple connected devices is used as rule engines that support the formulation of decision logics. The concurrency models between single-threaded event loop Node.js and multi-thread approach made difference. To test Node.js a higher concurrency level-where it is supposed to surpass multi-threading. The XMPP Publisher subscriber service that creates a subscription to publish updates to the associated REST resource (Observable Property). This creates a graph link back to the data source. This pattern can be used to control HTTP POST, CoAP extended GET, and other asynchronous notification mechanisms from the destination endpoint. Note that Observers and Subscribers are ways of controlling asynchronous updates from either the source or the destination endpoints, or both. If both are used between 2 endpoints, there is a 2 way resource graph connection established. The future work will focus on the implementation of the framework of the proposed prototype with node.js in intelligent buildings related IoT applications such as Home automation.

ACKNOWLEDGMENT

This paper was supported by Dong-Eui University research in 2016, National Research Foundation-BTP-2015DB0160001), Training business area of leading new business(201601 700001) in National Research Foundation(National Research Foundation of Korea) in 2016.

REFERENCES

[1] Continental Automated Buildings Association. Intelligent Building and the Impact of IoT [Internet]. https://www.caba.org/documents/forms/Prospectus-IB-IoT.pdf.

[2] N. Dabek, N. Zeldovich,, F. Kaashoek, D. Mazieres, and R.

Morris. “Event-driven programming for robust software”. in *Proceeding of SIGOPS. European Workshop*, pp.186-189, 2002.

[3] S. Choi, J. Kim, J. Yun and I. Ahan, “A Tutorial for Energy-efficient Communication for XMPP- based Internet of Things,” Smart Computing Review, Vol. 3, no. 6, pp.471-479, 2013.

[4] Wapice Technology Partener. IoT-ticket.com REST API [Internet]. Available : https://www.iot-ticket.com/images/Files/IoT-Ticket.com_IoT_API.pdf.

[5] K. Rose, S. Eldridge and L. Chapin, “The Internet of Things : An Overview. Understanding the issues and Challenges of a More Connected World,” Internet Society, [Internet]. Available : http://www.internetsociety.org/sites/default/files/ISOC-IoT-Overview-20151022.pdf.

[6] Google Cloud Messaging. Engage your users across Android, iOS and chrome [Internet]. Available : https://developers.google.com/cloud-messaging/.

[7] D.Happ, N. Karowski, T.Menzel, V. Handzski and A. Wolisz, “Meeting IoT platform requirements with open pub/sub solutions,” Annals of Telecommunications. [Internet]. Available : http://www.tkn.tu-berlin.de/fileadmin/fg112/Papers/2016/Happ16meeting_iot_platform.pdf.

[8] Y. Z. Daniel, R. Matthew and H. Vijay, “Microarchitectural implications of event-driven server-side web applications,” in *Proceedings of the 48th International Symposium on Microarchitecture*, pp. 762-774, 2015.

[9] F. David, *JavaScript: The Definitive Guide*, 6th ed. United States of America, O’Reilly Media Pub, ch.12, pp. 289-296, 2011.

라이오넬(Lionel Nkenyereye)

동의대학교 컴퓨터공학과 석사과정
※관심분야 : 유무선통신네트워크, 빅데이터, 안드로이드 시스템,사물의 인터넷.

장종욱(Jong-wook Jang)

1995년 2월 부산대학교 컴퓨터공학과 박사
1987년 ~ 1995년 ETRI
2000년 2월 UMKC Post-Doc.
1995년 ~ 현재 동의대학교 컴퓨터공학과 교수
※관심분야 : 유무선통신시스템, 자동차네트워크