Cloud-based Middleware for Syntactical Interoperability in Internet of Things

Eko Sakti Pramukantoro¹, Fariz Andri Bakhtiar²
¹²-Faculty of Computer Science, Universitas Brawijaya
{¹ekosakti, ²fariz}@ub.ac.id

Received 06 October 2019; accepted 03 January 2020

Abstract. Heterogeneity of protocol communications on the Internet of Things can lead to a syntactical interoperability problem. As far known as sensor nodes have different protocols of communication mechanisms with each other. But, sensor nodes should be able to send data with any kind of protocol communication to IoT applications in the cloud environment. So, the collected data is varied and not limited to specified sensor devices. A middleware that capable of works with much different protocol as a bridge between node sensor and IoT application is one best solution to answer a syntactical interoperability problem. This paper proposed a cloud-based middleware that provides a communication interface to receive data from sensor nodes based on Restful and CoAP. Received data then stored into heterogeneous IoT data storage based on the NoSQL database. Interoperability testing methodology was used for experiments and testing. The result shows proposed middleware can receive data from both protocols. The received data could store based on structure data or unstructured data on IoT data storage.

Keyword: Data, Middleware, HTTP, CoAP

1 Introduction

Generally, a system based on the Internet of Things (IoT) consists of three-component which are Things, Network infrastructure, and Cloud-based applications. However, each component in the IoT could bring through different protocol communications, data formats, and hardware specifications. This led to interoperability problems. Interoperability can be defined as the ability of a system to exchange data with different technologies and protocol communication mechanisms.

Desai has classified interoperability into three categories: Network Interoperability, Syntactical Interoperability, and Semantic Interoperability [1]. The solution for interoperability is a middleware that provides various communication protocols so that the data can be delivered from Things to Cloud-apps [2]. There are three kinds of middleware, i.e. actor-based, cloud-based, and service-based middleware [3]. Actor-based middleware is close to IoT infrastructure; it is suitable for resolving network interoperability problems but has limited computing capacity. Service-based and cloud-based middleware have advantages in computing ability because they are located in a cloud environment with flexible resources, yet service-based middleware is limited to service.
On the other hand, handling the interoperability can also use standard web communication as a messaging protocol. The HyperText Transfer Protocol (HTTP) could solve interoperability issues through homogenizing communications. Unfortunately, not all IoT devices (e.g., sensor nodes) can be homogenized, due to their resource-constrained characteristics such as low computation, memory capacity, and network interface [4]. Various studies have been conducted to formulate solutions for standard web communication for resource-constrained devices. Ruta proposed a Constrained Application Protocol (CoAP) framework for a resource-constrained IoT devices [5]. CoAP uses a smaller transport protocol header and was designed for use in minimum resources, so it is suitable to be implemented in a limited IoT environment.

In this paper, a cloud-based middleware is proposed to resolve interoperability issues by providing HTTP and CoAP as communication interfaces. A detail explanation of this middleware can be found in section 2.

2 Proposed Systems

![Fig 1. The architecture of cloud-based middleware](image1)

![Fig 2. The sequence diagram of cloud-based middleware](image2)
As depicted in figure 1, the proposed cloud-based middleware consists of several subsystems. (1) API gateway, consist of two protocol communication based on HTTP and CoAP; (2) authentication and authorization mechanisms through JSON Web Token (JWT); (3) Semantic filtering, with the ability to determining kind of data which sent by sensor node [6]; (4) NoSQL based data storage for storing heterogeneous data from sensor nodes [7].

Figure 3 shows how our system works. First, the CoAP client must be authenticated through the JSON Web Token (JWT). Authentication is done by sending the token key and the secret key obtained from the device registration process on the IoT applications to the semantic IoT Web service through the CoAP server. The token key and the secret key will be validated on a semantic IoT Web service to obtain a JWT token. If valid then the JWT token will be sent by semantic IoT Web service to the CoAP client via the CoAP server. But if it is invalid, the CoAP client will receive an error message.

JWT tokens that have been obtained will be sent back along with the data and topics to the CoAP server and then forwarded to a semantic IoT Web service. Topics that have been received are validated. The validation process must include the JWT token as an authorization requirement. If valid, then proceed with the data storage process via semantic filtering method. There are two types of storage data: 1) MongoDB for storing structured data; 2) GridFS to store unstructured data. After the storage process, the semantic IoT Web service will send a response to the CoAP client for either a successful response or a failed state.

3 Experiment Setup

To asses our system, we used the interoperability testing methodology as shown in Table 1. This testing aims to determine whether a CoAP and Restful Webservice based on HTTP can work together as a communication protocol in Cloud middleware. The testing method is based on the interoperability assessment methodology with several parameters known as protocol, interpretation and information utilization. For protocol, testing is conducted by sending the data through CoAP and HTTP concurrently to cloud-based middleware. while Interpretation and information utilization conducted as one testing that we named as data integrity testing. The data used in this testing are the sensor's data with several formats like JSON, image, and video files.

| Parameter                  | Scenario                                                                 |
|----------------------------|--------------------------------------------------------------------------|
| Sending data               | - sent via CoAP and HTTP communication protocols concurrently to Cloud-based middleware |
|                           | - independently sending data via CoAP and HTTP to Cloud-based middleware. |
| Monitoring send data       | During the process of sending data, Wireshark is used to monitoring transferred data. |
| Checking the data model    | Validating data by matching data form sensor nodes and data which stored in data storage. |

Table 1. Interoperability testing scenario
4 Experiment Result and discussion

4.1 Interoperability Testing

Table 2 is a sample of testing result which shows data transmission from the sensor nodes to cloud-based middleware. The first sensor node with IP address 182.1.64.199 using HTTP, the second sensor node has with IP address 182.1.77.223 using CoAP, and IP address for cloud-based middleware is 206.189.94.98. Both sensor nodes sent the sensor’s concurrent to cloud-based middleware. There are four packet data obtained from transmission using both protocols. One pair is the authentication process, and the second pair is processed sending data using the POST method. Data that is sending using CON, it means the server should replay to the sender with the ACK packet. This is the feature of reliability in CoAP. The interoperability testing was performed and shows data can be sent concurrently using CoAP and HTTP to cloud-based middleware.

Table 2 Concurrent sending data

| Source       | Destination | Protocol | length | Info                                                                 |
|--------------|-------------|----------|--------|----------------------------------------------------------------------|
| 182.1.65.236 | 206.189.94.98 | HTTP     | 152    | POST /user/login HTTP/1.1 (application/x-www-form-urlencoded)         |
| 206.189.94.98 | 182.1.65.236 | HTTP     | 725    | HTTP/1.1 200 OK (application/json)                                  |
| 182.1.64.199 | 206.189.94.98 | HTTP     | 170    | POST /topic/jsonhttp/topic HTTP/1.1 (application/x-www-form-urlencoded) |
| 206.189.94.98 | 182.1.64.199 | HTTP     | 482    | HTTP/1.1 200 OK (application/json)                                  |
| 206.189.94.98 | 182.1.77.223 | CoAP     | 48     | ACK, MID:59253, 2.01 Created, TKN:4e 71, /login                   |
| 182.1.77.223  | 206.189.94.98 | CoAP     | 56     | CON, MID:59254, GET, /login                                         |
| 206.189.94.98 | 182.1.77.223 | CoAP     | 445    | ACK, MID:59254, 2.05 Content                                        |
| 182.1.77.223  | 206.189.94.98 | CoAP     | 527    | CON, MID:59255, POST, TKN:73 4b, /postdata                         |

4.2 Data Integrity

The next testing parameter is data integrity during transmitting the sensor’s data. Figure 3 is an example of data in JSON format which consists of humidity, nitrogen, and pH. This data sent by the sensor node to Cloud-based middleware. Figure 4 is data from the sensor node that is successfully saved into the data storage. The data which is stored in data storage can be displayed through IoT applications. The result of the data displayed in the IoT applications can be seen in Figure 5.
The second sample of data is image data. To send data using CoAP need to
change the format as pickle, while HTTP as raw format. Figure 6 is an example of image data sent to the Cloud-based middleware through each protocol. According to the semantic filtering process, this data detected to be unstructured data. So, the data will be saved into GridFS. Figure 7 shows the image data that has been stored on the GridFS data storage. From integrity testing results, cloud-based middleware can receive structured and unstructured data. The semantic filtering functioned as expected, could save data based on the kind.

5 Conclusion

Based on the experiment, cloud-based middleware is able to resolve interoperability problems especially syntactical interoperability by providing communication protocol via CoAP and HTTP to sensor nodes. The CoAP and HTTP protocols can work independently or concurrently in the process of transmission data from the sensor node to an IoT data storage. In terms of data integrity, data transmitted by the sensor nodes that are JSON, images, and video have the same structure and shape as the data displayed in the IoT application, as well as the data rewritten on a file (video data). For future work, another protocol communication will be added to cloud-based middleware.

References

1. P. Desai, A. Sheth, and P. Anantharam, “Semantic Gateway as a Service Architecture for IoT Interoperability,” in Proceedings - 2015 IEEE 3rd International Conference on Mobile Services, MS 2015, pp. 313–319 (2015).
2. E. S. Pramukantoro and H. Anwari, “An Event-Based Middleware For Syntactical Interoperability And Enabling Web-Oriented In Internet Of Things,” Int. J. Electr. Comput. Eng., vol. 8, no. 5, (2018).
3. M. A. Razzouque, M. Milojevic-Jevric, A. Palade, and S. Cla, “Middleware for internet of things: A survey,” IEEE Internet Things J., vol. 3, no. 1, pp. 70–95, (2016).
4. E. Dalipi, F. Van Den Abeele, I. Ishaq, I. Moerman, and J. Hoebeke, “EC-IoT: An easy configuration framework for constrained IoT devices,” 2016 IEEE 3rd World Forum Internet Things, WF-IoT 2016, pp. 159–164, (2017).
5. M. Ruta et al., “A CoAP-based framework for collaborative sensing in the Semantic Web of Things,” Procedia Comput. Sci., vol. 109, pp. 1047–1052, (2017).
6. E. S. Pramukantoro, F. A. Bakhtiar, and A. Bhawiyuga, “A Semantic RESTful API for Heterogeneous IoT Data Storage,” (2019).
7. E. S. Pramukantoro, W. Yahya, G. Arganata, A. Bhawiyuga, and A. Basuki, “Topic based IoT data storage framework for heterogeneous sensor data,” in 11th International Conference on Telecommunication Systems Services and Applications (TSSA), (2017).