An MQTT-Based Context-Aware Autonomous System in oneM2M Architecture

Geonwoo Kim, Seongju Kang, Jiwoo Park, and Kwangsue Chung, *Senior Member, IEEE*

***Abstract*—Having a seamless connection between independently developed platforms has recently become an important issue. In addition, research is underway on autoconfiguration to minimize human interference while configuring devices for users to utilize Internet of Things (IoT) applications. OneM2M is a standardization group established to develop a standard Internet service platform. In the oneM2M environment, data exchange between each component is performed through HTTP, constrained application protocol (CoAP), and message queuing telemetry transport (MQTT). MQTT enables efficient data transmission in low-power and unreliable networks. However, MQTT is not suitable for service discovery for autoconfiguration given that it does not support multicasting and resource directories for managing IoT resources. In this paper, we present an architecture to provide an autoconfiguration mechanism by using the MQTT publish/subscribe messaging pattern. To realize fully autonomous systems, the proposed system autonomously interacts with devices by recognizing the context using semantic Web technology. We verify the feasibility of the proposed system through its implementation in the real environment. It is confirmed that the proposed approach is able to automatically configure MQTT-based devices and provide context-aware services using a standard interface.**

***Index Terms*—Autonomous system, Internet of Things (IoT), message queuing telemetry transport (MQTT), oneM2M standard, semantic Web, service discovery.**

I. Introduction

THE INTERNET of Things (IoT) is a huge and heterogeneous collection of protocols, services, and solutions. Due to the advent of new technologies for low-power wireless communications and low-cost processors, the IoT paradigm has taken the next step in networking devices, resulting in building automation, smart grids, intelligent supervisory systems, and smart homes. With the heterogeneity of independent IoT platforms, numerous protocols and standards have been developed [1]. Nevertheless, even within standardized protocols, there is a large variety to choose from. These platforms are the result of evolving requirements and technology, leading to a highly dynamic ecosystem of coexisting protocols that cannot work with each other [2]. One of the key points of IoT deployment is the interoperability between devices and applications across multiple architectures, platforms, and networking technologies. However, due to the rapid growth of the IoT market, the number of IoT devices has increased dramatically, thereby making it difficult to connect independently developed platforms and to configure devices and services manually [3]. Therefore, to easily provide interoperability between platforms and to minimize user interference during device configuration, research on autoconfiguration is underway [4].

An international partnership project, the oneM2M global initiative was created to produce a worldwide M2M service layer specification for machine-to-machine (M2M) solutions [5]. The data exchange between each component in the oneM2M environment is accomplished through HTTP, constrained application protocol (CoAP), and message queuing telemetry transport (MQTT).

MQTT is a messaging protocol for the IoT selected by the organization for the advancement of structured information standards (OASIS). It enables efficient data transmission in low power and unreliable network [6]. MQTT is suitable for exchanging messages between backend applications and edge network equipment because the broker sends messages to subscribers who subscribe to a specific topic, and it uses a small size message header to minimize the overhead. In addition, it provides three levels of quality of service (QoS) to ensure reliability in the transmission of publish/subscribe-based messages, thereby ensuring the reliable transmission of important messages that must be transmitted. If the message is transmitted using the publish/subscribe method, 1:N, N:N, and 1:1 communications can be efficiently implemented. A typical commercial service using MQTT is Facebook Messenger, and many open source projects are based on MQTT [7], [8]. However, because MQTT operates on the basis of TCP, it does not support multicasting and resource directories for managing IoT resources. Therefore, MQTT is not suitable to discover services for automatically configuring the IoT devices.

To address this issue, we present an autoconfiguration system that enables service discovery and device operation using the MQTT publish/subscribe messaging pattern. In the proposed system, devices register the service based on discovering the gateway using the service discovery protocol, and it is able to discover the service in the local network using the MQTT topic configuration. In addition, to realize fully autonomous systems, semantic Web technology is applied to the gateway for autonomously performing the interaction between devices by recognizing the context. Abowd *et al.* [9]

Manuscript received March 25, 2019; revised April 29, 2019; accepted May 23, 2019. Date of publication May 30, 2019; date of current version October 8, 2019. This work was supported by the Institute for Information and Communications Technology Promotion grant funded by the Korea Government (MSIT, Development of Human Implicit/Explicit Intention Recognition Technologies for Autonomous Human Interaction) under Grant 2017-0-00167. *(Corresponding author: Kwangsue Chung.)*

The authors are with the Department of Electronics and Communications Engineering, Kwangwoon University, Seoul 01897, South Korea (e-mail: gwkim@cclab.kw.ac.kr; jwpark@cclab.kw.ac.kr; kchung@kw.ac.kr).

Digital Object Identifier 10.1109/JIOT.2019.2919971

defined the context as follows: "any information that can be used to characterize the situation of an entity, where an entity can be a person, or a physical or computational object." We show the mechanism of using semantic Web technologies to describe the IoT data and methods of associating IoT data with semantic metadata for context-aware service.

The rest of this paper is organized as follows. In Section II, the oneM2M standard, MQTT, the service discovery protocol, and semantic Web technology are discussed. In Section III, we present a system that autonomously performs the interaction through the request/response scheme using MQTT and semantic Web technology is discussed. In Section IV, the feasibility of the proposed system is verified through a real-world implementation and the performance is analyzed. Finally, Section V offers concluding remarks.

II. Background and Related Works

Without standards, IoT systems and services would be developed independently for different vertical domains, thereby causing high fragmentation problems and increasing development and maintenance costs. To mitigate the fragmented IoT ecosystem, several industry and international standards organizations corroborated to publish standard specifications for the IoT system. These specifications include lightweight communication protocols, service discovery mechanisms, and semantic Web technologies. This section presents the research works relevant to this focus.

A. OneM2M Standard

The oneM2M global initiative has been committed to standardizing a common service layer platform for M2M/IoT services that are globally applicable and access-independent. Swetina *et al.* [10] summarized the oneM2M standardization activities well. The oneM2M initially collected a variety of use cases from a wide range of vertical business domains. After that, it formulated requirements for the oneM2M common service layer and then designed the system architecture [11]. The primary goal of oneM2M is to define a globally agreed-upon M2M service platform that incorporates the currently isolated M2M service layer standards activities. The oneM2M standard adopted a RESTful architecture, and thus all services are represented as resources to provide the defined functions [5].

The oneM2M reference architecture model is shown in Fig. 1. The oneM2M architecture divides the M2M/IoT environments into two domains (infrastructure and field domain) and defines four types of nodes that reside in each domain considering the configuration scenario where oneM2M systems are deployed: infrastructure node (IN), middle node (MN), application dedicated node (ADN), and application service node (ASN). The IN is located in the network infrastructure and provides M2M service. The MN acts as a gateway between the devices and the network infrastructure. The ADN is a constrained device with limited functionality that includes only M2M service logic. The ASN is a generic node that includes common service functions (CSFs) as well as M2M application.

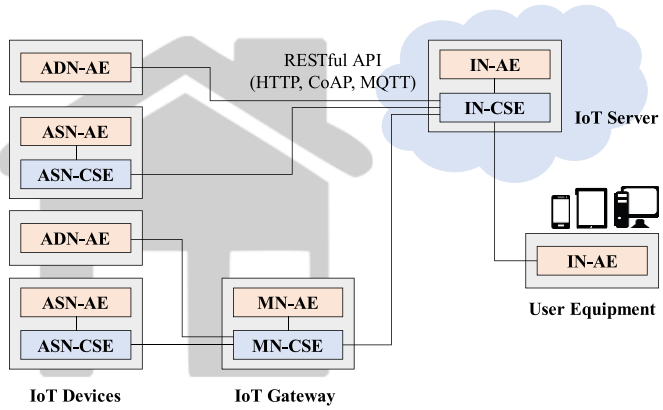

Fig. 1. OneM2M reference architecture model.

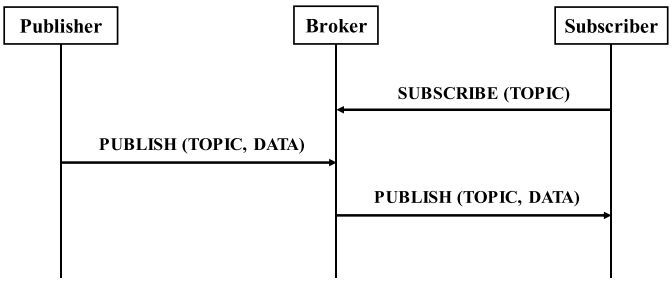

Fig. 2. Message delivery method of MQTT.

The oneM2M architecture is based on a hierarchical entity model, which comprises the application entity (AE), the common service entity (CSE), and the underlying network service entity (NSE). The AE provides the application logic for an end-to-end IoT solution and represents the application services located in a device, gateway, or server. The CSE is a platform that provides CSFs, an M2M service function that can be used by various application entities in common. The CSFs include registration, security, application, service, data and device management, etc. The NSE provides network services to the CSE.

B. MQTT Protocol

MQTT is a communication protocol designed to enable the interaction of multiple peers within a publish/subscribe architecture. The basic concept of message delivery through the MQTT protocol is illustrated in Fig. 2. A client is a peer that represents a program or device that exchanges application messages over a given topic with another client through a server, which is also called a message broker. Client interactions are always moderated by the server, and so clients do not need to know each other. Clients communicate by requesting the server to subscribe or publish messages about a given topic. Clients who want to publish messages will jointly specify the payload and topic of the message, whereas clients who want to receive the message subscribe to the specified set of topics through the topic filter expression [6].

Topics are organized hierarchically in a tree structure, and the topic filter expressions allow one to refer all the elements at a given hierarchal level in the tree. This allows the delivery

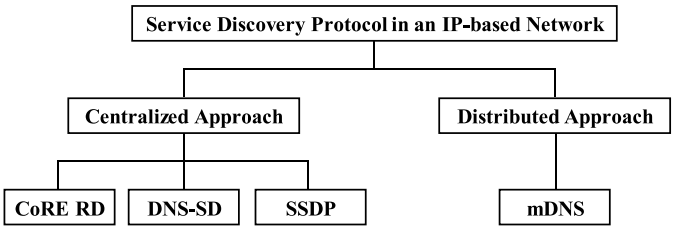

Fig. 3. Service discovery protocol in an IP-based network.

of messages that reference multiple topics based on the same routing criteria. A topic is a string of alphanumeric tokens, which are called topic levels, that are separated by a topic level separator. A topic filter is a string structured as a sequence of topic levels, each specifying an alphanumeric token or the wildcard character "+" or "#." A wildcard "+" can be replaced by a single token, whereas a wildcard "#" is replaced with a token sequence of an arbitrary length. For example, the topic filter expression "house/firstfloor/bedroom/humidity" is the topic sequence of the messages that indicates the humidity of the bedroom on the first floor of the house, while the expression "house/firstfloor/+/humidity" indicates the humidity of all the rooms on the first floor and the expression "house/#" indicates all physical quantities that have been measured in the house.

C. Service Discovery Protocol

Access and discovery are two basic functionalities that are essential to IoT applications [12]. The role of the service discovery protocol is to provide a method of accessing a device by discovering a service in the local network. The service discovery protocol is divided into the centralized approach and the distributed approach. The centralized approach manages the service information in the network through one or more resource directories. The distributed approach discovers the services through interdevice interaction without a resource directory. The service discovery protocols used in an IoT environment with limited resources are the constrained restful environments resource directory (CoRE RD) [13] and domain name system based service discovery (DNS-SD) [14] that use a centralized approach, and the multicast DNS (mDNS) [15] that uses a distributed approach.

The service discovery protocol developed so far in the IP-based network is classified as shown in Fig. 3. The main functions and procedures of the service discovery protocol are publication, registration, discovery, and resolution [16]. The publication is performed by a service discovery protocol in a distributed approach, and it informs another device of the service information, such as the service type, access method, and service characteristics. The registration is a function of storing the service information of a device in a specific element in the network, such as a resource directory so that service discovery can be performed. To register the service, the device must know the address of the resource directory or know the information of the resource directory through the multicast search method. The discovery and resolution are functions of verifying the address or name of a device having an associated service. It is very important to the discovery of resources without previous knowledge of the structure and format of the data hosted by connected things. In previous studies, CoRE related standards were mainly used as a discovery mechanism [17], [18]. Bonjour [19] developed by Apple is a representative software using a service discovery protocol. Bonjour is able to discover and register Apple products on the local network through mDNS with the DNS-SD capability.

D. Semantic Web Technology

The existing Web is a document-oriented Web. It is not suitable for machines to understand and handle the semantically described IoT data given that the document-oriented Web is based on what people read and understand. In order to communicate a large number of IoT devices with heterogeneous capabilities, the semantic Web technologies are promising tools for this purpose to share and exchange data efficiently [20]. Semantic technology allows to understand the meaning of data and to infer new knowledge through relationships between data. The semantic Web is an extension of the existing Web with a machine interpretable meaning, thus establishing data integration, data sharing, and interoperability among interconnected machines [21].

In the semantic Web environment, each resource is identified through an International Resource Identifier (IRI) and is defined through a data model called the resource description framework (RDF) and the RDF Schema. The RDF defines the relationship between resources through a triple structure consisting of a subject, a predicate, and an object. The ontology defines the concept of each resource and the relation between the concepts as a property to provide a core base for the machine to autonomously process the data. Semantic protocol and RDF query language (SPARQL) is a semantic query language for interacting with the repository of the data defined by the RDF triple structure. Semantic technologies are able to play critical roles in data and knowledge management for context-aware service in IoT platforms. By representing context-aware data based on common semantic references, we can provide a wide range of services, such as abstraction of heterogeneous IoT platforms, context-awareness services, and new services through semantic mashup [22]. It is possible to realize high-level interaction services such as dynamic user define-rule generator through autonomous inference based on semantic data modeling in the IoT domain.

To realize fully autonomous systems for providing context-aware services in the IoT environment, it is necessary to support interoperability at the semantic level of resources beyond communication level interoperability [23]. Therefore, the necessity and usefulness of the semantic Web that enables the machine to autonomously process the meaning of data have been studied [24]–[28].

To the best of our knowledge, no work has yet been done on service discovery for autoconfiguration using MQTT. Park *et al.* [29] proposed an SDN controller using the extended MQTT block for multicasting. In the autoconfiguration system, all the procedures must be processed dynamically

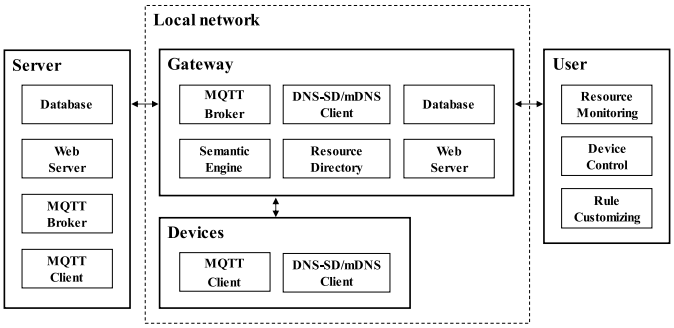

Fig. 4. MQTT-based context-aware autonomous system.

without human intervention when a device is connected to the network. Park *et al.* [29] described how to support multicast in the state that it is already connected to an MQTT broker. In this paper, we propose an autoconfiguration system that employs the DNS-SD/mDNS service discovery protocol for connecting the MQTT client to the broker autonomously. In addition, the proposed system shows an autoconfiguration mechanism using the MQTT publish/subscribe messaging pattern in the international standard oneM2M environment and further realizes a fully autonomous system.

III. Proposed Scheme

In this section, we describe in detail the MQTT-based context-aware autonomous system. The proposed system has an autoconfiguration and a context-aware service component. Based on the oneM2M standard, the proposed system is able to provide automatic services that ensure interoperability with heterogeneous platforms and minimize human interference.

A. System Overview

The overall structure of the proposed MQTT-based autonomous system is shown in Fig. 4. The gateway and the device exist in the same network and operate as an mDNS node implementing the DNS-SD function for service discovery. The gateway and devices have an MQTT communication module for service registration, discovery, and data communication. The broker and resource directory exist in the gateway. The resource directory manages the device and service information in the local network and returns the result of the lookup operation in the JSON format according to the discovery request. The gateway acts as an MN-CSE in oneM2M architecture, which manages the Web server and the resource and also provides CSFs.

The Semantic Engine receives the device information through the resource directory and creates RDF graphs. The server platform communicates with each local gateway and manages the entire gateway. Users are able to monitor and operate devices in the local network using a Web-based interface and receive automated services according to user-defined rules.

B. Service Registration

The procedure of registering the service to the resource directory is shown in Fig. 5. MQTT does not communicate directly with each client, but the broker delivers the message to the subscriber. Therefore, in order to configure the device automatically, it is necessary to discover a gateway that acts as an MQTT broker. The device discovers the gateway service in the local network through a multicast query using DNS-SD/mDNS.

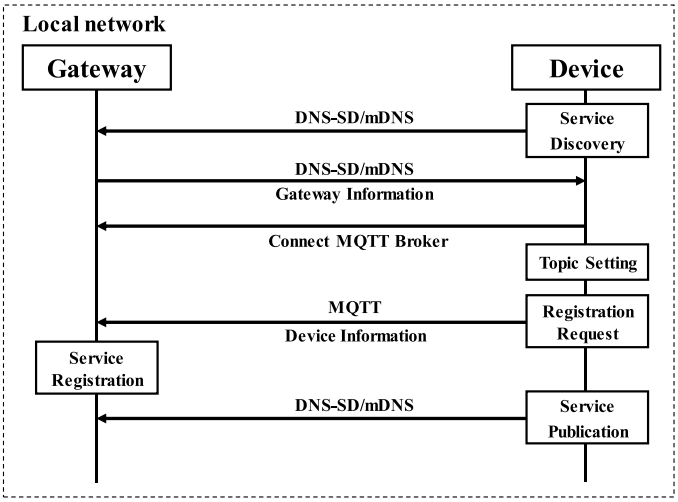

Fig. 5. Service registration procedure of an MQTT-based device.

In the DNS-based service discovery protocol, the service is expressed in the form of *<Instance>.<ServiceType>.<Domain>* [16]. The *<Instance>* part identifies a specific instance of the service and *<ServiceType>* can include a subtype such as *_onem2m._sub*. In the proposed system, the service of the gateway is defined as *gateway._onem2m._sub._tcp.local*. In the oneM2M standard, MQTT topics are expressed in the form */oneM2M/req/<originator>/<receiver>*, and there are differences depending on resource registration, request, and response [5]. Because of security issues, the initial configuration is required to configure the topic with the credential identifier before requesting a resource, and *<originator>* and *<receiver>* contain the identifier of the oneM2M entity. Accordingly, the gateway transmits the gateway identifier together with the service information including the IP address of the gateway in response to the multicast query.

The device accesses the MQTT broker using the IP address of the gateway and sets the communication topic with the gateway by using the identifier of the gateway and the device's IP address allocated from the dynamic host configuration protocol (DHCP) server. Since the IP address assigned to the device in the local network is unique, it is able to identify each device at the gateway using the IP address of the device. After the communication topic setting is completed, the device becomes an oneM2M resource by registering its service to the resource directory using the MQTT. The MQTT message is configured in the JSON format and includes the device name, service, and IP address. The devices publish the service using the DNS-SD/mDNS together with the MQTT-based service registration. Therefore, the service discovery using the multicast query among the devices is possible in addition to look up the resource directory.

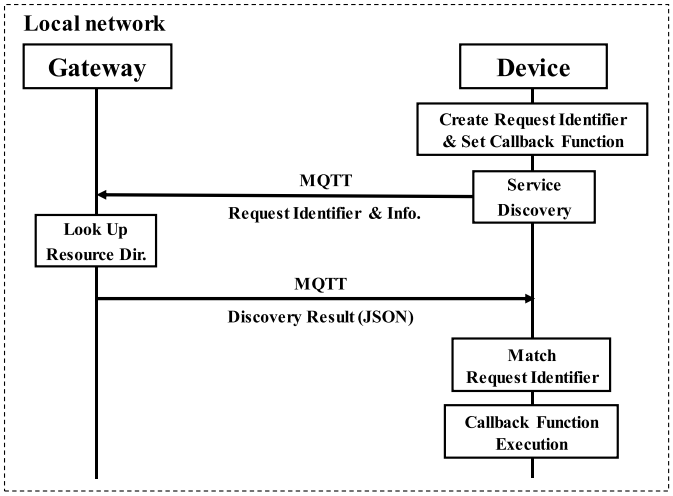

Fig. 6. Service discovery procedure using MQTT.

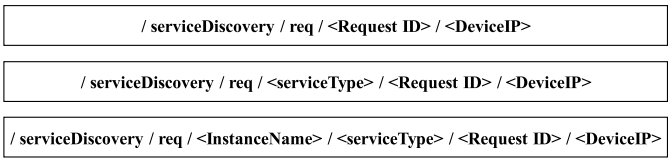

Fig. 7. MQTT topic structure for service discovery.

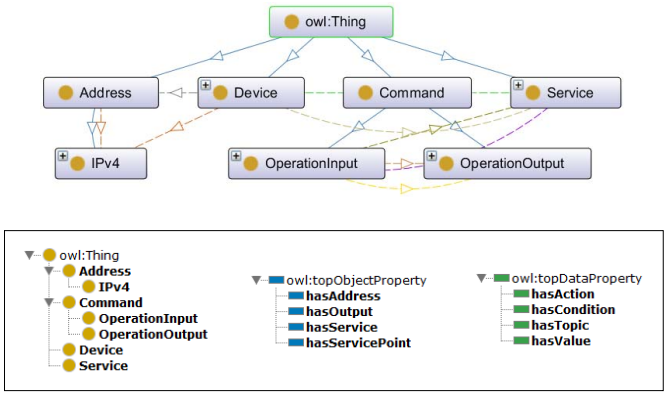

Fig. 8. Designed ontology model.

C. Service Discovery

Fig. 6 shows the procedure of discovery for a service after the service is registered in the resource directory. MQTT supports asynchronous message processing according to real-time push transmissions for minimizing power usage and does not act as the request/response method for each message as HTTP and CoAP do. However, in the asynchronous message processing method of MQTT, it is difficult to confirm whether the message is a discovery result for a specific service discovery request. In this paper, we propose an asynchronous request/response scheme by constructing a message identifier for each message to enable service discovery in the local network.

The user or device stores a behavior to be performed when receiving a discovery response in a hash map together with a unique message identifier. The hash map consists of *<key, value>* pairs. The *key* is a unique message identifier and the *value* is a function that is executed when a service discovery response is received. When the gateway receives the service discovery request, it analyzes the topic structure to check the IP of the device that requested the discovery and what type of discovery request it is and checks the resource directory accordingly. The service discovery result is configured using the IP and message identifier of the device that was checked when the request was received and then delivered in the JSON format using MQTT. If a subscriber receives a discovery result from the gateway, it executes the callback function mapped to the hash map using the message identifier included in the MQTT topic.

Gateways and devices use MQTT topic wildcards when configuring topic subscriptions since they must be able to process messages for dynamically changing topics, such as service discovery types and message identifiers. The user configures a topic as shown in Fig. 7 to discover the entire service in the local network, the devices having a specific service type, and the communication channel for a device having a specific service. For example, to discover a device that has the *oneM2M_light._onem2m._sub._tcp.local.* service type in the local network, the publisher configures an MQTT topic such as */serviceDiscovery/req/onem2m_light/LbgNp-3/223.194.33.64:192.168.86.65*. In this case, *LbgNp-3* is a unique random string composed of alphanumeric characters that are generated when requesting a service discovery.

D. Device Operation

Users are able to know the service and access path of each device using the service discovery. When the device is connected to the local network, the gateway automatically collects device information and stores it, so that the device is able to be controlled through the gateway access. In addition, it is possible for the users to control the device by receiving the MQTT topic information that is able to access the corresponding device through the communication channel discovery.

In Fig. 7, *DeviceIP* represents the IP address of the device requesting service discovery. Considering global discovery, it is used together with the public IP address to prevent the duplication of MQTT topics. The *DeviceIP* part format is *public IP:private IP* such as *223.194.33.64:192.168.86.65*.

E. Ontology Modeling

In the proposed system, the ontology is designed as shown in Fig. 8 so that the machine understands the meaning of the data and provides context-aware service. Ontologies such as the W3C semantic sensor network (SSN) ontology [30], which is mainly used in the IoT environment, and the oneM2M Base ontology [31], which is used in the oneM2M standard, have a complicated resource relationship. Therefore, we design the optimized ontology according to the proposed system. The *Device*, *Service*, and *Address* are the classes for representing devices in the local network. The *Command* is a class for rule matching based on user-defined rules. When a device is connected to the local network, it is mapped to the *Device*, *Service*, and *Address* classes based on the device information

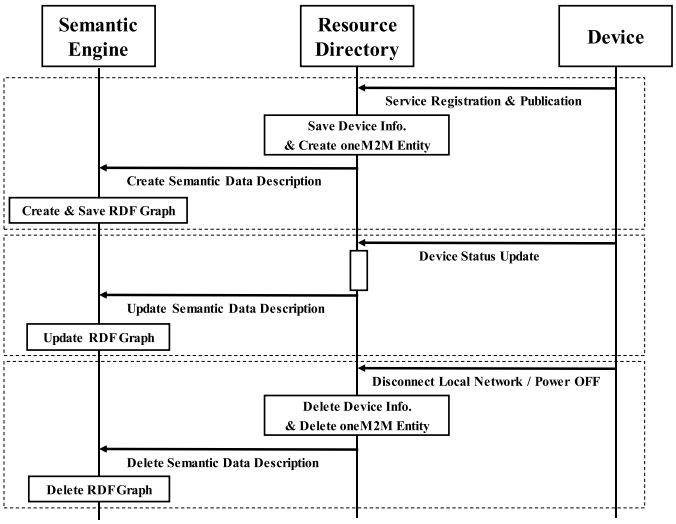

Fig. 9. Semantic data description procedure.

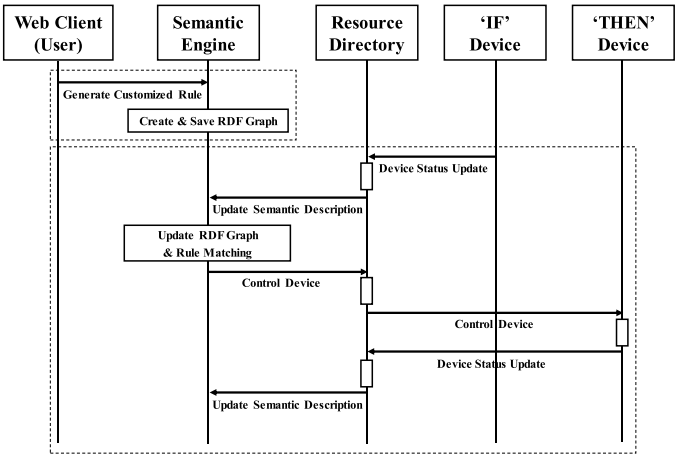

Fig. 10. Rule creation and matching procedure.

and is expressed as semantic data. The *OperationInput* and *OperationOutput* classes are mapped to the "IF" and "THEN" conditions of the user-defined rule, respectively.

F. Context-Aware Service

Fig. 9 shows the procedure in which the state information of devices in the local network is represented by semantic data. When the device accesses the local network, it registers the device and its service to the resource directory as described in Section III-B. In the resource directory, the device information is stored, and an oneM2M resource is created for the connected device. Each device is resource-wise incorporated into an AE/CNT/CNT structure and mapped to a *gateway/device/service*. The Semantic Engine generates the RDF graph according to the ontology using the information received from the resource directory and stores it in the database for the semantic query. When the device state is changed, the RDF graph stored in the semantic database is updated. SPARQL does not provide a query function for updates [32]. To prevent the generation of duplicate semantic data, the Semantic Engine deletes the previous data and inserts new data into the semantic database. When the device terminates the connection in the local network or the power is turned off, the device information stored in the resource directory and the oneM2M resource are deleted. Like the semantic data creation procedure, the Semantic Engine removes the RDF graph stored in the semantic database using the information received from the resource directory.

A rule matching procedure is illustrated in Fig. 10 for generating user-defined rules through the Web client and controlling the devices according to the current status. The user confirms the device information collected through the service discovery and sets the IF condition and the THEN condition. The condition comparison of the user-defined rule is performed according to three situations of higher, lower, and equal. For example, if a user-defined rule is created to "set the desired temperature of the air conditioner to 21 degrees if the temperature in the room is higher than 28 degrees," the thermostat's current temperature of higher than 28° corresponds to the IF condition and the desired temperature of 21° corresponds to the THEN condition. The Semantic Engine generates RDF graphs using information about the IF and THEN conditions. If the status of the IF device changes, the status of that device is updated in the Semantic Engine and the rule matching process is performed. When it is confirmed that the status of the updated device corresponds to the user-defined rule by using the *hasCondition* and *hasServicePoint* attributes of the *OperationInput*, the control request is made to THEN device through the *hasAction* attribute defined in the *OperationOutput*. After the control for the THEN device is performed, it is updated to the Semantic Engine via the resource directory again.

G. Web Client

Through the Web-based interface, users can receive services without needing specialized knowledge such as the ontology and syntax used in the system. To realize a fully autonomous system, the proposed system is autonomous in all the procedures. The user can monitor and control the status of the devices in real time through the service discovery from the time when the device connects to the local network. It is possible to create and delete user-defined rules dynamically using the graphic user interface (GUI) through the Web-based interface so that it is able to provide autonomous services efficiently.

IV. Implementation and Evaluation

In this section, we verify the feasibility of the MQTT-based context-aware autonomous system and analyze its performance through real-world implementation and experimentation. It is proved that our proposed system provides an autoconfiguration mechanism, device operations according to service discovery, and context-aware service based on semantic descriptions of the user-defined rules and device properties. To analyze the performance, we measure the service matching time according to the number of services and the rule matching time according to the number of rules.

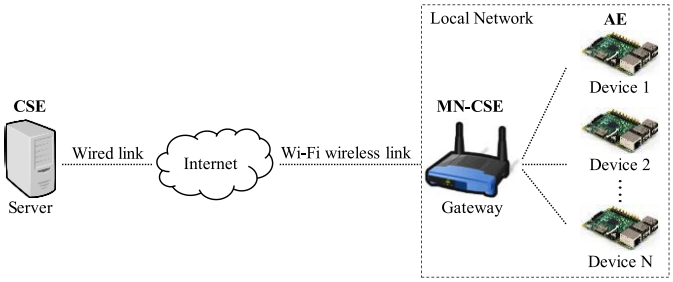

Fig. 11. Network environment.

Device Log

```
Service Registration:
{ interfaceIndex: 3,
  type:
   ServiceType {
     name: 'gateway',
     protocol: 'tcp',
     subtypes: [],
     fullyQualified: true },
  replyDomain: 'local.',
  flags: 2,
  name: 'Gateway',
  networkInterface: 'wlan0',
  fullname: 'Gateway._gateway._tcp.local.',
  host: 'linaro-alip.local.',
  port: 1883,
  rawTxtRecord: <Buffer 20 47 61 74 65 77 61 79 49 44 3d
  txtRecord: { GatewayID: 'S20180511085557067IbqN' },
  addresses: [ '192.168.86.140' ] }
[MQTT Broker Connection Success]
Service Publication:
{ name: 'oneM2M-Lamp',
  type:
   ServiceType {
     name: 'onem2m_light',
     protocol: 'tcp',
     subtypes: [],
     fullyQualified: true },
  domain: 'local.',
  flags: 0 }
```

Fig. 12. Service registration request log for device.

A. Implementation Environment

To verify the proposed MQTT-based context-aware autonomous system, we constructed the experimental environment shown in Fig. 11. The gateways and devices are implemented using a Tinker board and Raspberry Pi 3, which are off-the-shelf embedded boards, and OCEAN [33], which is an open source platform that complies with the oneM2M standard. The MQTT broker of the gateway uses Mosquitto [34], and the protocol DNS-SD/mDNS for service discovery and publication is implemented using Node-mDNS [35]. The Semantic Engine is implemented using the Jena framework of, the Java open source library [36], which supports the development of semantic functions, such as the RDF, semantic database, and SPARQL.

B. Autoconfiguration Experiment

As shown in Fig. 12, the device registers its service after discovering the gateway service in the local network using the DNS-SD/mDNS protocol. The device uses the service information of the gateway to connect the MQTT broker and set the communication topic using the IP address of each device. After setting the communication topic, the service is registered in the resource directory using MQTT and published using the DNS-SD/mDNS for an interdevice multicast query. Fig. 13 shows the device information stored in the resource directory after the service is registered in the gateway.

Gateway Log

```
----> /serviceRegister
 === [Resource Directory] ===
{ device: 'oneM2M-Temperature',
  service: 'onem2m_temp',
  ip: '192.168.86.65' }
{ device: 'oneM2M-Lamp',
  service: 'onem2m_light',
  ip: '192.168.86.66' }
{ device: 'oneM2M-Lamp',
  service: 'onem2m_dim',
  ip: '192.168.86.66' }
{ device: 'oneM2M-Ultrasonic',
  service: 'onem2m_distance',
  ip: '192.168.86.64' }
 === [Resource Directory] ===
----> /oneM2M/req/S20180511085557067IbqN/rosemary/xml
```

Fig. 13. Service registration log for gateway.

All Service Discovery

```
[Service Discovery Result]
[ { device: 'oneM2M-Temperature',
    service: 'onem2m_temp',
    ip: '192.168.86.65' },
  { device: 'oneM2M-Ultrasonic',
    service: 'onem2m_distance',
    ip: '192.168.86.64' },
  { device: 'oneM2M-Lamp',
    service: 'onem2m_light',
    ip: '192.168.86.66' },
  { device: 'oneM2M-Lamp',
    service: 'onem2m_dim',
    ip: '192.168.86.66' } ]
```

Specific Service Discovery

```
[Service Discovery Result]
[ { device: 'oneM2M-Lamp',
    service: 'onem2m_light',
    ip: '192.168.86.66' } ]
```

Communication Topic Discovery

```
[Service Discovery Result]
{ path: '/rosemary/Gateway/oneM2M-Lamp/onem2m_light',
  topic: '/oneM2M/req/192.168.86.66/Gateway/xml' }
```

Fig. 14. Service discovery result using MQTT.

The service discovery result using MQTT after the service is registered in the resource directory is shown in Fig. 14. The result illustrates the discovery of the whole service in the local network, the device having a specific service type, and the communication channel for the device having a specific service, respectively. The service discovery result indicates the device name, service type, and IP address information in the JSON format in consideration of one or more devices having the same service.

Fig. 15 shows how devices are controlled using the MQTT topic information through the communication channel discovery. The user stores a callback function to be executed

Callback Function

```
var requestID = makeRequestID();
discoveryMap.set(requestID, function(message){
        var jsonObj = JSON.parse(message.toString());
        console.log(jsonObj);
        var commTopic = jsonObj['topic'];
        onem2mCreateCin(commTopic,'oneM2M-Lamp','onem2m_light','OFF
    }
)
mqttClient.publish('/serviceDiscovery/req/'+requestID+'/oneM2M-Lamp
```

User

```
[Service Discovery Result]
{ path: '/rosemary/Gateway/oneM2M-Lamp/onem2m_light',
  topic: '/oneM2M/req/192.168.86.66/Gateway/xml' }
```

Device

```
[Device Control]: Set [onem2m_light] to OFF
```

Fig. 15. Device control using MQTT channel discovery.

Semantic Description

```
sm_gateway:oneM2M-Lamp
        a                   owl:NamedIndividual , sm_gateway:Device ;
        sm_gateway:hasAddress  sm_gateway:192.168.86.66 ;
        sm_gateway:hasService  sm_gateway:oneM2M-Lamp.onem2m_dim , sm_gateway:oneM2M-Lamp.onem2m_l
        sm_gateway:hasTopic    "/oneM2M/req/192.168.86.66/Gateway/xml" .

sm_gateway:oneM2M-Lamp.onem2m_dim
        a                   owl:NamedIndividual , sm_gateway:Service ;
        sm_gateway:hasValue  "50" .

sm_gateway:oneM2M-Lamp.onem2m_light
        a                   owl:NamedIndividual , sm_gateway:Service ;
        sm_gateway:hasValue  "ON" .

sm_gateway:192.168.86.66
        a       owl:NamedIndividual , sm_gateway:IPv4 .
```

Fig. 16. RDF triples of device information stored in the semantic database.

Semantic Engine Log

```
[UPDATE]: oneM2M-Temp/onem2m_temp/28
[Rule Matching...]
[MATCHED]: oneM2M-Temp.onem2m_temp > 25
[ACTION]: oneM2M-Lamp.onem2m_dim 20
[Rule Matching End]
[UPDATE]: oneM2M-Lamp/onem2m_dim/20
```

Fig. 17. Rule matching log of the semantic engine.

when receiving a discovery result in a hash map together with a message identifier. The user is able to operate the device by sending a control request to the gateway and the corresponding device using the oneM2M resource path or the MQTT topic information of the device. Through the implementation result, we confirmed that we are able to discover and operate devices and services in the local network.

C. Context-Aware Service Experiment

The device information stored in the semantic database is shown in Fig. 16. When a lamp device is connected to the local network, the properties of the lamp are defined according to the ontology in the Semantic Engine, and the service of the corresponding device, such as its brightness and ON/OFF control are expressed in the RDF triple structure. User-defined rules are stored in the semantic database as the *Command* type. When the device status is changed, the rule matching is performed via the SPARQL query on the semantic database.

Fig. 17 shows the log of the Semantic Engine where the room temperature has been changed to 28° when the rule "Set the lamp's brightness to 20 if the temperature is above 25 degrees" is defined. When a temperature of 28° is measured, the value is updated in the Semantic Engine via the resource directory. Because the status of the device has changed, the rule matching will be performed and the IF condition will be found that matches *If the temperature is above 25 degrees*. In the user-defined rule, the *OperationInput* class corresponding to the IF condition is associated with the *OperationOutput* class corresponding to the THEN condition through the *hasOutput* attribute. Accordingly, the final state of the device is updated by performing the THEN condition defined in the *hasAction* property of the *OperationOutput* class "Setting the brightness of the lamp to 20."

Fig. 18. Web-based interface.

The Web-based interface to utilize the service without requiring specialized knowledge, such as ontology and syntax used in the system, is shown in Fig. 18. The user is able to monitor the devices in real time and control the devices using the GUI. The Web-based data visualization library d3.js [37] is used to represent the devices and services in a tree form and to check the updated device status. Each node from the root node to the leaf node represents MN-CSE (gateway)/MN-AE (gateway)/CNT (device)/CNT (service)/CIN (value). Moreover, in the user-defined rule setting page, a list of rules stored at the semantic database can be checked, and new rules can be dynamically created and deleted as needed.

D. Performance Evaluation

To analyze the performance of the proposed MQTT-based context-aware autonomous system, we measured the service discovery time depending on the number of services in the local network and the rule matching time depending on the

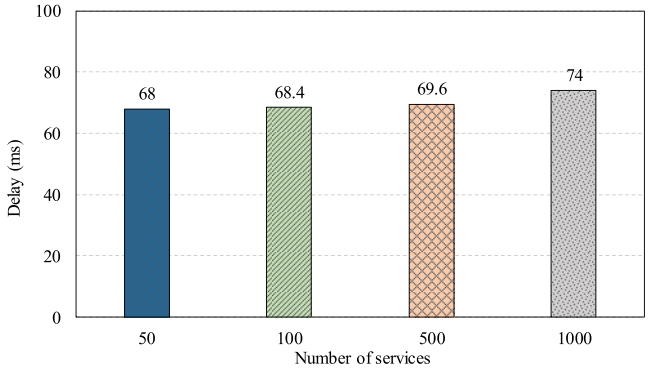

Fig. 19. Average service discovery time depending on the number of services.

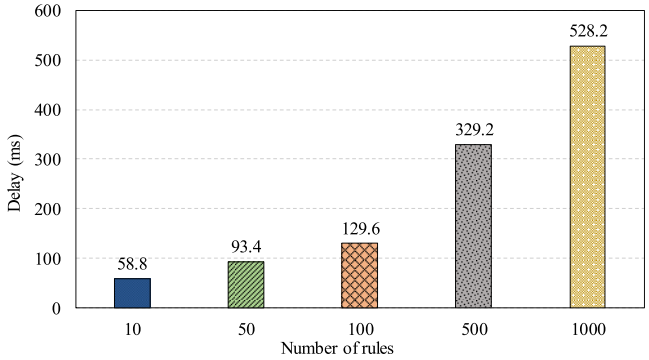

Fig. 20. Average rule matching time depending on the number of rules.

number of user-defined rules. The average delay is calculated based on ten measurements in every case.

The service discovery time measures the delay from when the user and the device publish the service discovery request using MQTT to when the result is received. Fig. 19 shows the results of measuring the average service discovery time by increasing the number of services from 50 to 1000. Because service discovery is performed in the resource directory without going through the Semantic Engine, real-time discovery is possible except for the MQTT communication time. The experimental results show that an average of 74 ms is consumed in an environment where there are 1000 services in the local network and that the delay has some differences depending on the MQTT communication time. As a result, even if the number of services increases, the discovery results are returned directly from the resource directory, so that it is able to discover in real time without significant impact on delay.

The results of measuring the average rule matching time depending on the number of user-defined rules while increasing the number of rules from 10 to 1000 are shown in Fig. 20. Since the rule matching procedure is performed through the SPARQL query on the semantic database, it takes more time compared to the service discovery time. Also, as the number of user-defined rules increases, the delay increases because of the number of RDF graphs defined for each of the IF condition and the THEN condition increases. However, despite using a constrained off-the-shelf embedded board with limited resources, it is confirmed that a maximum delay time of approximately 500 ms occurs in an environment where there are 1000 user-defined rules in the local network. If the number of user-defined rules stored in the semantic database is more than 1000, it may be inconvenient to provide real-time service due to a long delay. However, it is not a critical problem because it is able to reduce the delay by using the high-performance embedded board and optimizing the query. Finally, it is confirmed that the service discovery and rule matching are possible with limited overhead in real time.

V. Conclusion

In this paper, an MQTT-based context-aware autonomous system to provide autonomous services in an environment configured with MQTT-based devices has been proposed. The proposed system performs service discovery by implementing an asynchronous request/response scheme using the MQTT publish/subscribe messaging pattern. Moreover, to realize a fully autonomous system, the gateway autonomously interacts with devices by recognizing the context using semantic Web technology.

Considering the characteristics of MQTT, the process of discovering a gateway to register services uses DNS-SD /mDNS. After the services are registered in the resource directory, the devices become oneM2M resources and are able to access devices and discover services using MQTT.

In the real environment, we implemented the gateway and the device using a small embedded board and analyzed the feasibility and the performance of the proposed system. Through the results, it is confirmed that the devices and services are able to configure automatically in an environment composed of MQTT-based devices and that the proposed system is able to provide context-aware services by understanding the device status. Additionally, through the performance analysis, it is confirmed that the service discovery and rule matching are possible with limited overhead in real time.

References

[1] J. A. Stankovic, "Research direction for the Internet of Things," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 3–9, Feb. 2014.

[2] H. Derhamy, J. Eliasson, and J. Delsing, "IoT interoperability—On-demand and low latency transparent multiprotocol translator," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1754–1763, Oct. 2017.

[3] S. Kim, K. Lee, and J. Jeong, "DNS naming services for service discovery and remote control for Internet of Things devices," in *Proc. Int. Conf. ICT Converg.*, Oct. 2017, pp. 1157–1162.

[4] A. Bröring, S. K. Datta, and C. Bonnet, "A categorization of discovery technologies for the Internet of Things," in *Proc. Int. Conf. Internet Things*, Nov. 2016, pp. 131–139.

[5] *oneM2M—Standards for M2M the Internet of Things*. Accessed: Oct. 19, 2017. [Online]. Available: http://www.onem2m.org

[6] A. Banks and R. Gupta, *MQTT Version 3.1.1*, OASIS Stand., Burlington, MA, USA, Apr. 2014.

[7] H. C. Hwang, J. Park, and J. G. Shon, "Design and implementation of a reliable message transmission system based on MQTT protocol in IoT," *Wireless Pers. Commun.*, vol. 91, no. 4, pp. 1765–1777, Dec. 2016.

[8] K. Liao and C. Lin, "Implementation of IoT applications based on MQTT and MQTT-SN in IPv6 over BLE," *Int. J. Design Anal. Tools Integr. Circuits Syst.*, vol. 6, no. 1, pp. 48–49, Oct. 2017.

[9] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and contextawareness," in *Proc. 1st Int. Symp. Handheld Ubiquitous Comput.*, 1999, pp. 304–307.

[10] J. Swetina, G. Lu, P. Jacobs, F. Ennesser, and J. Song, "Toward a standardized common M2M service layer platform: Introduction to oneM2M," *IEEE Wireless Commun.*, vol. 21, no. 3, pp. 20–26, Jun. 2014.
[11] J. Kim, S.-C. Choi, I.-Y. Ahn, N.-M. Sung, and J. Yun, "From WSN towards WoT: Open API scheme based on oneM2M platforms," *Sensors (Basel)*, vol. 16, no. 10, pp. 1645–1667, Oct. 2016.
[12] G. Tanganelli, C. Vallati, and E. Mingozzi, "Edge-centric distributed discovery and access in the Internet of Things," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 425–438, Feb. 2018.
[13] Z. Shelby, C. Bormann, and S. Krco, "CoRE resource directory," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ietf-core-resource-directory-04, Jul. 2018. [Online]. Available: http://tools.ietf.org/id/draft-ietf-core-resource-directory-04.txt
[14] S. Cheshire and M. Krochmal, "DNS-based service discovery," Internet Eng. Task Force, Fremont, CA, USA, RFC 6763, 2013.
[15] S. Cheshire and M. Krochmal, "Multicast DNS," Internet Eng. Task Force, Fremont, CA, USA, RFC 6762, 2013.
[16] B. C. Villaverde, R. De Paz Alberola, A. J. Jara, S. Fedor, S. K. Das, and D. Pesch, "Service discovery protocols for constrained machine-to-machine communication," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 41–60, 1st Quart., 2014.
[17] F. Khodadadi and R. O. Sinnott, "A semantic-aware framework for service definition and discovery in the Internet of Things using CoAP," in *Proc. Int. Conf. Emerg. Ubiquitous Syst. Pervasive Netw.*, vol. 113, Nov. 2017, pp. 146–153.
[18] S. Cirani *et al.*, "A scalable and self-configuring architecture for service discovery in the Internet of Things," *IEEE Internet Things J.*, vol. 1, no. 5, pp. 508–521, Oct. 2014.
[19] *Bonjour*. Accessed: Oct. 18, 2017. [Online]. Available: http://developer.apple.com
[20] S. Jabbar, F. Ullah, S. Khalid, M. Khan, and K. Han, "Semantic interoperability in heterogeneous IoT infrastructure for healthcare," *Wireless Commun. Mobile Comput.*, vol. 2017, Mar. 2017, Art. no. 9731806.
[21] A. I. Maarala, X. Su, and J. Riekki, "Semantic reasoning for context-aware Internet of Things applications," *IEEE Internet Things J.*, vol. 4, no. 2, pp. 461–473, Apr. 2017.
[22] A. J. Jara, Y. Bocchi, D. Fernandez, G. Molina, and A. Gomez, "An analysis of context-aware data models for smart cities: Towards fiware and ETSI SIM emerging data model," *Int. Archives Photogrammetry Remote Sens. Spatial Inf. Sci.*, vol. XLII-4/W3, pp. 43–50, Oct. 2017.
[23] S. Mayer, R. Verborgh, M. Kovatsch, and F. Mattern, "Smart configuration of smart environments," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 3, pp. 1247–1255, Jul. 2016.
[24] C. E. Kaed, I. Khan, A. V. D. Berg, H. Hossayni, and C. Saint-Marcel, "SRE: Semantic rules engine for the industrial Internet of Things gateway," *IEEE Trans. Ind. Informat.*, vol. 14. no. 2, pp. 715–724, Feb. 2018.
[25] E. Kovacs, M. Bauer, J. Kim, J. Yun, F. Le Gall, and M. Zhao, "Standards-based worldwide semantic interoperability for IoT," *IEEE Commun. Mag.*, vol. 54, no. 12, pp. 40–46, Dec. 2016.
[26] A. Mazayev, J. A. Martins, and N. Correia, "Interoperability in IoT through the semantic profiling of object," *IEEE Access*, vol. 6, pp. 19379–19385, 2017.
[27] Z. Meng and J. Lu, "A rule-based service customization strategy for smart home context-aware automation," *IEEE Trans. Mobile Comput.*, vol. 15, no. 3, pp. 558–571, Mar. 2016.
[28] M. Ruta, F. Scioscia, G. Loseto, and E. Di Sciascio, "A semantic enabled social network of devices for building automation," *IEEE Trans. Ind. Informat.*, vol. 13, no. 6, pp. 3379–3388, Dec. 2017.
[29] J. Park, H. S. Kim, and W. Kim, "DM-MQTT: An efficient MQTT based on SDN multicast for massive IoT communications," *Sensors (Basel)*, vol. 18, no. 9, pp. 1–15, Sep. 2018.
[30] M. Compton *et al.*, "The SSN ontology of the W3C semantic sensor network incubator group," *J. Web Semantics*, vol. 17, pp. 25–32, Dec. 2012.
[31] *oneM2M Ontologies*. Accessed: Jan. 27, 2018. [Online]. Available: http://www.onem2m.org/technical/onem2m-ontologies
[32] *SPARQL 1.1 Update*. Accessed: Jan. 26, 2018. [Online]. Available: https://www.w3.org/TR/2013/REC-sparql11-update-20130321/#update Language
[33] *Ocean*. Accessed: Jan. 27, 2018. [Online]. Available: http://developers.iotocean.org/opensource
[34] *Mosquitto*. Accessed: Jan. 27, 2018. [Online]. Available: http://mosquitto.org
[35] *Node-mDNS*. Accessed: Jan. 28, 2018. [Online]. Available: http://npmjs.com/package/mdns
[36] *Apache Jena*. Accessed: Jan. 28, 2018. [Online]. Available: http://jena.apache.org
[37] *D3.js*. Accessed: Feb. 11, 2018. [Online]. Available: http://d3js.org

Geonwoo Kim received the B.S. degree in electronics and communications engineering from Kwangwoon University, Seoul, South Korea, in 2017, where he is currently pursuing the M.S. degree in electronics and communications engineering.

His current research interests include network protocols, Internet of Things—in particular, and QoS support in IoT.

Seongju Kang received the B.S. degree from the Electronics and Communications Engineering Department, Kwangwoon University, Seoul, South Korea, in 2017, where he is currently pursuing the Ph.D. degree.

His current research interests include Internet of Things, oneM2M standards, and semantic Web.

Jiwoo Park received the B.S. degree in electronics and communications engineering from Kwangwoon University, Seoul, South Korea, in 2013, where he is currently pursuing the Ph.D. degree in electronics and communications engineering.

His current research interests include network protocols, multimedia systems, Internet of Things, and video communications—in particular, QoS support in adaptive bitrate streaming.

Kwangsue Chung (M'93–SM'01) received the B.S. degree from the Electrical Engineering Department, Hanyang University, Seoul, South Korea, the M.S. degree from the Electrical Engineering Department, Korea Advanced Institute of Science and Technology (KAIST), Seoul, and the Ph.D. degree from the Electrical Engineering Department, University of Florida, Gainesville, FL, USA.

He was a Research Staff with Electronics and Telecommunications Research Institute, Daejeon, South Korea, for ten years. He was also an Adjunct Professor of KAIST from 1991 to 1992 and a Visiting Scholar with the University of California at Irvine, Irvine, CA, USA, from 2003 to 2004. In 1993, he joined Kwangwoon University, Seoul, where he is currently a Professor with the Department of Electronics and Communications Engineering. His current research interests include communication protocols and networks, QoS mechanism, Internet of Things, and video streaming.