IoT Framework for Interoperability in the oneM2M Architecture

Seongju KANG, Kwangsue CHUNG
Department of Electronics and Communications Engineering,
Kwangwoon University, Seoul, South Korea
kchung@kw.ac.kr

***Abstract*—The IoT is expected that many devices and sensors can be interconnected and interact over the Internet. Conventional IoT solutions rely on vertically developed machine-to-machine solutions that yield limited interoperability. To ensure interoperability between IoT solutions, the oneM2M global initiative defines a horizontal M2M service layer. To provide more intelligent services, such as autonomous interaction services, semantic-level interoperability should be ensured. Previous studies have proposed solutions based on ontologies to realize semantic level interoperability. However, in dynamic environments such as IoT, where data generated by many devices must be processed, an ontology leads to a system performance degradation owing the overhead of the resource mapping mechanism. In this study, we propose a semantic IoT framework based on the Resource Description Framework graph extension scheme. We utilize an aggregator based on the oneM2M standard platform. All data are represented as an RDF graph, and reconfigured dynamically through semantic queries. The proposed semantic IoT gateway provides a user-based rule management mechanism via the Web, thereby enabling rule configuration to be dynamically tailored to user requirements. Finally, the performance is evaluated compared with a solution that utilizes an ontology in a real IoT system.**

***Index Terms*—inference mechanisms, information science, internet of things, semantic web, standardization.**

I. INTRODUCTION

The Internet of Things (IoT) provides various solutions by interconnecting “things” such as sensors, devices, and gateways. The conventional IoT relies on vertically developed machine-to-machine (M2M) solutions, optimized for data processing and communications between applications and devices. These solutions allow limited interactions between IoT domains, because of the various protocols, data types, and data formats that they utilize. To overcome the problem of the fragmented IoT market, seven Standards Development Organizations (SDOs) have established the oneM2M global initiative [1]. oneM2M defines the Common Service Functions (CSFs) for the horizontal service layer, and supports them with the RESTful Application Programming Interface (API) of the oneM2M platform. The oneM2M service layer guarantees communication-level interoperability between IoT service domains on a global scale [2].

To provide intelligent services, IoT solutions should be able to interact with both people and things [3]. Semantic-level interoperability is necessary to achieve interoperability between different data schemes and convert them to a common vocabulary to be interpreted by machines. In current IoT solutions, the server or gateway only receives syntactic data for events that occur on the device or end products. That is, the service platform knows whether events have occurred, but does not understand what the events mean. Using semantic technology, syntactic data can be represented as semantic data that can be understood by machines. The service platform can understand the meaning of contextual data through semantic data, which enables a suitable processing of data and facilitates decision-making.

Previous studies have utilized ontologies as IoT solutions for semantic interoperability. An ontology is a tool for implementing a semantic Web for organizing and representing information, and has advantages in terms of information integration, information retrieval, and knowledge management [4]. Many ontologies are available in IoT, such as the Smart Application REFerence ontology (SAREF), Semantic Sensor Network (SSN), and Sensor Observation Sampling Actuator ontology (SOSA) [5-7]. An ontology provides a metadata schema based on explicitly defined semantic vocabularies. This schema allows the service platform to interpret the meanings of events or data that occur. For example, a movie ontology can be employed in a movie information service platform to manage related information, such as movie titles, directors, and actors. In addition, the service provider can recommend movies of similar genres and related queries to the user. Ontologies provide efficient querying and reasoning in static environments, where data creation and update intervals are infrequent. However, in dynamic environments such as IoT, where data generated by many devices must be processed, ontologies are not suitable, owing the overhead of the resource mapping process and query rewriting, which leads to a system performance degradation.

In IoT, an ontology mapping process should be performed to understand all the syntactic data generated by many devices and sensors. In addition, because not all devices and sensors can be described with a single ontology, integration with other ontologies is necessary. The ontology integration process requires ontology alignment via a matching algorithm to associate all ontologies with the same semantic vocabulary [8-10]. Ontologies are efficient for resource sharing and management, but have limitations in describing and managing all the data generated by devices and sensors as semantic data. In addition, because the range of semantic interoperability is limited to platforms’ own ontologies, it is

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.2017-0-00167, Development of Human Implicit/Explicit Intention Recognition Technologies for Autonomous Human Interaction).

Digital Object Identifier 10.4316/AECE.2020.02002

necessary to integrate ontologies to enable interactions with other platforms.

In this study, we propose an IoT framework based on the Resource Description Framework (RDF) graph extension scheme. Data are integrated via an aggregator based on the oneM2M-based platform, to enable interoperability of communication levels in the vertical IoT domains. The integrated data are represented by an RDF graph on the semantic IoT gateway and autonomously managed through a selective semantic query-based mechanism. In addition, the gateway provides a user-based rule management mechanism that allows users to dynamically create or modify rules. The proposed IoT framework achieves semantic level interoperability in a dynamic environment, such as IoT, via these mechanisms.

The remainder of this paper is organized as follows. In Section 2, we discuss how to ensure full interoperability in IoT, and review related work. In Section 3, we describe the RDF graph extension scheme, and semantic data management mechanism for guaranteeing semantic interoperability in the proposed semantic IoT framework. In Section 4, we evaluate the performance of the proposed scheme in a real IoT environment by comparing it with an existing ontology-based solution. Finally, we present the conclusions of this study and discuss future work in Section 5.

II. Related Work

In this section, we discuss how to ensure interoperability and review related work. Moreover, we discuss why ontologies are not suitable for IoT, and present conditions for semantic interoperability in IoT.

A. Interoperability

Interoperability is the property that one system can be utilized compatibly with other systems of the same or different kinds. There are many layers of interoperability, such as business process, semantic, organizational, and communication [11-12]. To provide intelligent and autonomous services, IoT solutions should ensure various levels of interoperability. For interoperability in IoT environments, communication-level interoperability should be ensured to allow each device to share and integrate resources with other devices. In addition, to process data autonomously, resources should be represented through explicitly defined vocabularies that can be shared and reused. Therefore, semantic-level interoperability should also be ensured, to make service platforms able to understand the meanings of occurred events and exchanged resources. We focus on guaranteeing these two levels of interoperability in IoT systems.

Communication-level interoperability is a substantial problem in vertical silo domains. Vertically developed M2M solutions that employ heterogeneous protocols, data types, and data formats cannot communicate with each other without a proxy or human intervention. To solve this problem, seven SDOs, in Korea (TTA), Europe (ETSI), China (CCSA), USA (ATIS/TIA), and Japan (TTC/ARIB), have established the oneM2M global initiative. To minimize fragmentation in the M2M service layer, oneM2M defines IoT CSFs.

Fig. 1 illustrates the oneM2M system architecture, where systems are divided into four functional node types: Infrastructure Node (IN), Middle Node (MN), Application Service Node (ASN), and Application Dedicated Node (ADN). Each functional node has a Common Service Entity (CSE) or Application Entity (AE) [13]. A CSE is a logical entity that is instantiated in each functional node and consists of a CSF. A CSE includes the defined IoT CSFs, such as device management, registration, and resource management. An AE is an entity that provides application logic to facilitate M2M solutions such as remote control and monitoring. Services provided by a CSE and AE are represented in resource form. The oneM2M resource is uniquely specified by a Uniform Resource Identifier (URI). oneM2M provides a horizontal service layer based on CSFs and the application logic used in functional nodes. Through the represented URI, the common service layer performs Create, Retrieve, Update, Delete, and Notify (CRUDN) using a RESTful API. In addition, it supports protocol bindings including Constrained Application Protocol (CoAP), Message Queue Telemetry Transport (MQTT), and Hyper Text Transfer Protocol (HTTP), and interworking with technologies such as 3rd Generation Partnership Project (3GPP), Open Connectivity Foundation (OCF), and Lightweight M2M (LwM2M). The oneM2M is a global initiative to ensure the interoperability of communication levels.

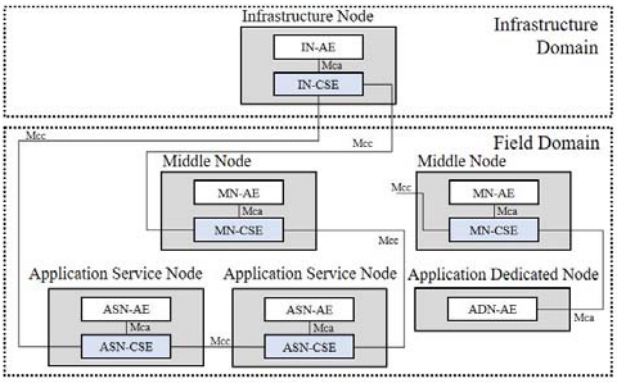


Figure 1. oneM2M standard architecture

The semantic Web is an extended Web technology that enables machines to integrate and exchange information through machine-readable vocabularies [14]. Semantic technology enables machines to understand the meaning of data and infer new knowledge through relationships between data. Linked open data (LOD) is the most fundamental concept for realizing semantic Web technologies [15]. Fig. 2 shows the building blocks of the semantic Web technologies. The RDF for representing the LOD concept was standardized by the World Wide Web Consortium (W3C). This RDF enables the association and integration of the relationships between multiple resources through standardized identifiers [16]. An RDF graph consists of RDF triples, which each consists of a subject, a predicate, and an object. Fig. 3 presents an example of an RDF graph. The subject and each object are connected through a predicate, which indicates the relationship. The object can also be a subject, and connected with an RDF triple or object.

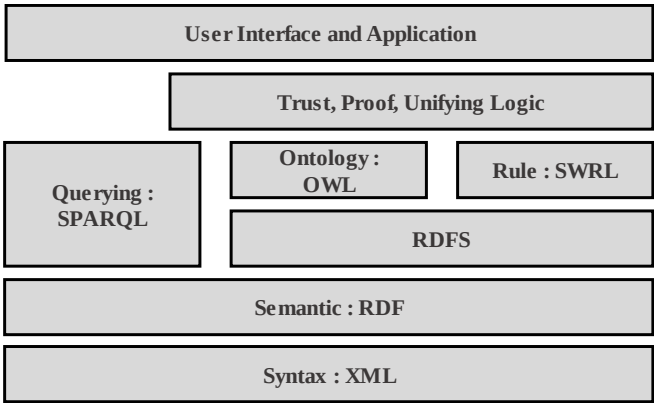


Figure 2. Building blocks of semantic Web technologies

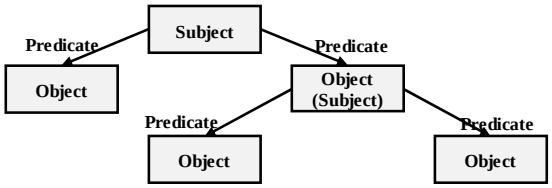


Figure 3. Example of RDF graphs

The data storage and management schema in the DataBase Management System (DBMS) must define the relationships between the tuples to be connected in the table configuration step, and further work is required on related tuples to add relations with new tuples. To search related data, the DBMS requires complex query statements. In contrast, the RDF can use predicates to represent resource relationships. Moreover, an RDF graph can be searched using a semantic query statement, and it is easy to search for related resources. Compared to a DBMS, RDF graphs have advantages in terms of data management and expansion.

Ontology Web language (OWL) and RDF schema (RDFS) are machine interpretable knowledge representation languages standardized by the W3C for sharing and providing knowledge. RDFS provides a vocabulary for structuring RDF resources and representing relationships among resources. OWL provides a richer vocabulary than RDF and RDFS, providing a variety of sublanguages with properties of reasoning completeness and time complexity. Semantic protocol and RDF query language (SPARQL) supports queries composed of RDF triples. Using SPARQL, semantic queries on rules and knowledge can be performed on ontologies that are designed for OWL and RDFS. Semantic Web rule language (SWRL) is an OWL-based rule language that enables a high level of matching and reasoning to better query OWL vocabularies and relationships compared to SPARQL. Ontologies can be shared and reused through integration with other ontologies via ontology matching algorithms based on semantic annotations. In an IoT system, a machine can understand resources based on an ontology and can infer relationships between resources through the object properties or predicates of resources.

B. Related work

In [17], the Semantic Rule Engine (SRE) was proposed for industrial IoT gateways. The SRE consists of a semantic engine and a rule engine, and executes rules based on the Lua script language, which is a lightweight language that is available in the Java and C platforms. The rules rely on a device's semantic annotations, and are not tied to the unique identifier or any other binding. Hence, even if the topology changes flexible and independent semantic functions can be performed. The SRE also provides life-cycle management of its rules, which can be managed in real time. Based on semantically annotated data, the rule engine compares a value with the threshold value and performs the operation according to the logic specified in the rule. The SRE provides a dynamic mechanism for a rule and matching via the rule engine and semantic engine. However, it does not address a dynamic mechanism for connecting devices or sensors.

In [18], IoT-O was proposed to ensure semantic interoperability in the oneM2M architecture. IoT-O is a new ontology, which merges five ontologies for sensors, observations, actuators, actuations, and service models respectively. The main objective of the IoT-O ontology is a dynamic reconfiguration of CSE resources to interconnect applications according to semantic descriptions. Instances of IoT-O are generated based on the locations of oneM2M resources. However, IoT-O provides insufficient rules and resource management mechanisms, and requires predefined properties such as temperature, luminosity, and humidity to generate automated rules.

The authors of [19] proposed IoT-Lite, which is a lightweight IoT model based on the SSN ontology. The authors noted that the detailed descriptions for facilitating a flexible description of the ontologies require heavy queries to perform semantic functions. IoT-Lite is designed as a core lightweight ontology, which excludes the nonessential components of SSN. Because IoT-Lite focuses on the semantic interoperability of multiple sensors and the processing of data generated by sensors, integration with other ontologies is required for use in a real IoT scenario. In addition, it does not address management mechanisms for dynamic configuration and semantic functions.

In [20], the authors proposed a methodology for agnostic endpoints, which can provide interoperability through devices' RESTful APIs. The endpoint analyzes the machine-readable interface file and creates a semantic model of the service input/output of the vendor. The endpoint can create a unified model by defining common vocabularies for service models from multiple vendors, and users can send requests to a device through a generic input model and receive services in an output model. The detailed implementation and mechanism of the proposed methodology were not described in that work. However, the methodology, which manages the agnostic endpoint vendor API as an integrated model and represents it through semantic data, can solve the fundamental problem of semantic vocabulary definition.

Several other studies have considered interoperability issues. In [21], a vision of how data and knowledge levels of interoperability can be resolved in IoT was presented. The authors of [22] discussed how to employ an ontology and process semantic data to facilitate interoperability in IoT environments. An ontology analyzes semantic annotations, and can describe them as semantic data. Semantic interoperability between vertical silos can be provided based on the semantic annotation of IoT and legacy devices. However, it is difficult to semantically annotate all data, and

semantic annotations do not include knowledge of M2M interactions. Therefore, ontologies that use semantic annotations are insufficient for handling large amounts of data distributed across multiple IoT silos. In addition, for ontology sharing and reuse, a semantic vocabulary matching process and a new resource mapping mechanism of the integrated ontology is needed are required.

Here, we identify a further problem: To understand the data generated by all devices and sensors, we require an ontology mapping process to describe syntactic data as semantic data. IoT things generate a large amount of data. A single mapping process has a small overhead, but it can lead to a system performance degradation. In summary, an ontology suffers from large overhead caused by the ontology mapping process and the additional work required to share resources between ontologies. For this reason, conventional solutions based on ontologies have many limitations in achieving semantic interoperability in IoT environments.

Previous research has focused on autonomous inference and resource management in a specified environment based on its ontology and interoperability is only ensured for limited resources. We propose the following conditions for achieving enhanced communication-level and semantic-level interoperability in IoT. First, generated data should be integrated through an aggregator based on the IoT standard platform. Conventional approaches to defining semantic vocabularies for all devices and sensors have problems with reprogramming and hard coding. Second, semantic data should be represented based on a lightweight resource representation model. A resource model that includes detailed descriptions requires heavy queries to perform semantic functions. Third, semantic data management should be dynamically configurable according to the states of devices and sensors and the data generated by IoT things should be updated in real time to perform proper semantic functions. Finally, a knowledge management mechanism should be provided to enable rule reconfiguration according to user requirements. The user can manage the rules through the user interface of the Web or the application.

All authors have to personally sign the copyright transfer form - with no exceptions. The signed copyright form has to be scanned and uploaded by using the corresponding interface on the website. There are some restrictions regarding the length of the copyright file, so please read carefully the instruction provided in the copyright file uploading interface.

III. Proposed IoT Framework

Communication-level interoperability can be ensured via IoT standard platforms, such as OCEAN and IoTivity, which provide many protocol bindings and equivalent data formats. However, using ontologies to ensure semantic-level interoperability in IoT environments has many limitations. There is no ontology that provides a complete solution for semantic vocabulary matching and resource mapping for interactions on heterogeneous platforms, and it is difficult to select an ontology that matches the target environments among various existing ontologies.

A. Aggregator

In this study, we propose an IoT framework based on an RDF graph extension scheme for communication-level and semantic-level interoperability. In addition, to solve the limitations of ontologies in IoT, we design a semantic data model based on the oneM2M resource architecture. Fig. 4 illustrates the architecture of the proposed IoT framework, which consists of an aggregator, a semantic IoT gateway, and a Web client. To describe syntactic data in terms of semantic data, a semantic vocabulary should be defined by analyzing communication protocols and message formats. The semantic vocabulary provides an equivalent representation of the data generated by vertical silos. However, defining an entire vocabulary requires expert knowledge and skills, and problems of hard coding and reprogramming arise when interconnecting new devices. In addition, an ontology that represents all IoT devices and sensors requires heavy queries to perform semantic functions.

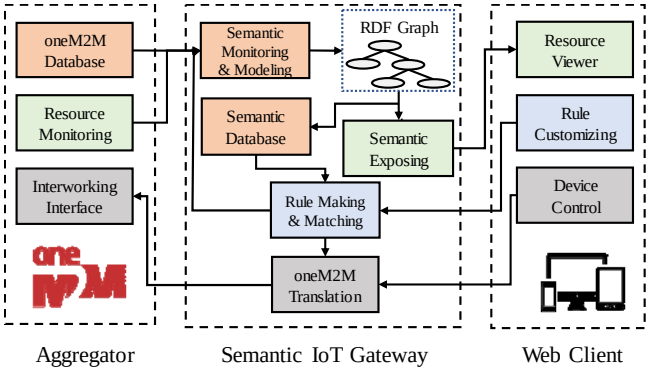


Figure 4. Semantic IoT framework architecture

To overcome this problem, it is necessary to define a minimal semantic vocabulary to represent all data. An aggregator based on the oneM2M standard provides a solution for integrating data generated from vertical silos. oneM2M resources are organized in a hierarchical structure and identified by resource parameters. These resources can be described as RDF triples by analyzing the parameters. Because there is no need to define a semantic vocabulary for all data generated by devices and sensors, hard coding and reprogramming problems are prevented. In addition, the oneM2M standard supports various protocol bindings to perform interactions between heterogeneous platforms. The aggregator ensures communication-level interoperability and addresses fundamental problems for semantic-level interoperability. The aggregated data are translated into oneM2M resources, stored in the database, and managed. Events for the creation, deletion, and updating of resources generated by a device or sensor are reported to the gateway in real time in the resource monitoring module. Because IoT interconnects many devices and creates a large amount of data, it is necessary to reduce the delays that occur in all communications as much as possible. MQTT has a small communication overhead compared to HTTP and a lower delay compared to CoAP in a low-packet-loss environment [23-24]. Therefore, the aggregator reports all events that occur to the gateway using MQTT. The interworking interface module modularizes the information on the API of the vertical silos, and executes the device control command.

B. Semantic IoT gateway

We propose an RDF graph extension scheme that can dynamically manage resources in an IoT environment. The semantic IoT gateway represents the integrated oneM2M resource in the aggregator as an RDF graph. Conventional IoT solutions based on ontologies focus on rule design and matching in a static environment with semantic modeling of all devices and sensors. However, they do not address a mechanism for dynamically managing connected devices. For example, if a sensor device that measures temperature is disconnected and there is no device to replace it, then the consequences of inference become meaningless. Therefore, it is necessary to have a mechanism that can dynamically manage devices based on the health of the connection states of the devices. The proposed semantic IoT framework for achieving semantic interoperability provides a dynamic configuration mechanism for connected devices. Fig. 5 illustrates the proposed RDF graph model based on the oneM2M architecture. Various parameters are defined in oneM2M [13]. Semantic IoT gateways use *ty* (type), *pi* (parentId), *ri* (resourceId), and *con* (content) parameters and a URI to represent hierarchical resource structures as semantic data. The URI of the device and service graph refers to the URI of the oneM2M RESTful API, which represents the resource path within the oneM2M system. The *ty* parameter represents the entity to which the resource belongs. The gateway classifies the properties of resources based on *ty* values. The *ri* parameter is the identifier of a resource, and the *pi* parameter is the identifier of the parent resource. The gateway associates the relationship between the device and the service graph based on the *ri* and *pi* parameters. The *con* parameter represents a service's status or value, such as temperature, humidity, or door status. The semantic IoT gateway describes the oneM2M resource in terms of semantic data based on the oneM2M resource parameters. In addition, the gateway understands an occurred event based on the *op* (operation), *to* (To), and *fr* (From) parameters of the oneM2M standard message specification. The op parameter is a value of post, get, put, or delete method. For example, when the *op* parameter is 1, this represents a message for a post method. The *to* and *fr* parameters indicate the receiver and originator of a oneM2M request message.

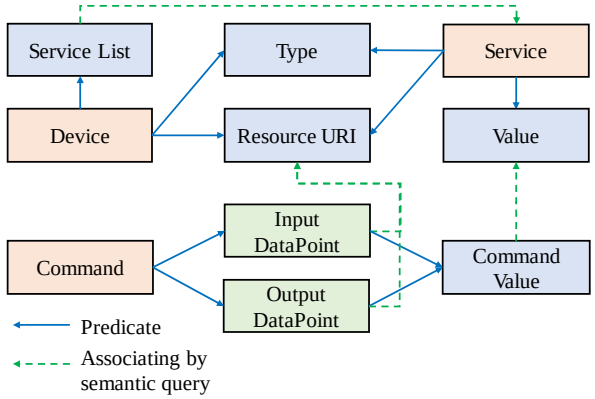


Figure 5. Proposed RDF graph model

All oneM2M resources and events can be described by semantic vocabularies, and the gateway can understand the meanings of resources and generated events and dynamically generate semantic queries. Therefore, users can create or delete interaction rules between devices for services represented by semantic data. The command graph consists of the input and output datapoint, which refer to the resource URI of the service graph. Through semantic queries, the command value is associated with a value for the service graph as a condition of rule execution.

Fig. 6 illustrates the RDF graph creation, updating, and deletion processes. The gateway searches all resources stored in the oneM2M database, and the semantic monitoring and modeling module creates RDF graphs. When a new device or sensor is registered as a oneM2M resource, the gateway performs a semantic query based on the reported syntactic data and creates an RDF graph using that query. If the device is disconnected, then the gateway deletes the RDF graph by performing a delete query via the same mechanism. When an updating event occurs for a service state, the semantic monitoring and modeling module deletes the value object from the service graph and regenerates the value object to update the graph. All RDF graphs are stored in the semantic database and are provided to the user via the Web client through the semantic exposing module.

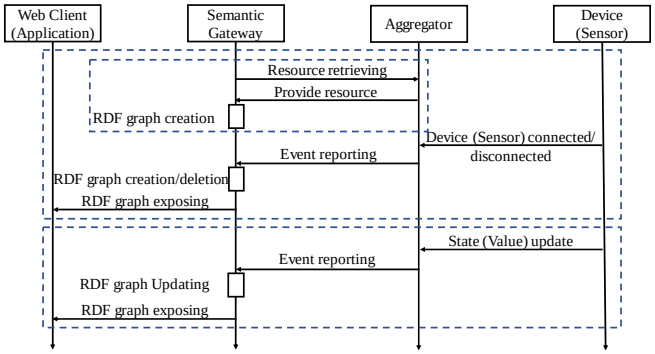


Figure 6. RDF graph creation, deletion, and updating processes for proposed framework

Fig. 7 illustrates the process of user-based rule creation and matching. The user sends a creation rule request to the rule making and matching module through the Web client's rule customizing module. The rule making and matching module consists of a rule validator, rule manager, rule matcher, and comparator. The rule validator checks the validity of the rule requested by the user. For example, the rule *"set the room temperature to 23 degrees when the temperature is below 25 degrees"* causes the system to enter an infinite loop. The rule manager creates a command graph for rule creation requests that have passed validation. The command graph consists of input datapoint and output datapoint objects. The input datapoint is a condition object of the rule that executes the rule and is connected to the resource URI and the value object of the service graph through a semantic query. The output datapoint is connected to the resource URI and the value object of the actuator that executes the rule through a semantic query. When the resource monitoring module reports to the gateway on events that occur on devices and sensors, the rule making and matching module updates the RDF graph and searches for rules using the rule matcher. If there is an input datapoint for the resource URI of the updated service graph, the comparator evaluates the updated value with the value of the

input datapoint. Based on the comparison result, the oneM2M translator module composes a oneM2M message to updating the resource URI object of the output datapoint and sends it to the aggregator. The aggregator then performs the service by calling the API of the actuator managed by the interworking interface module. The rule manager reduces unnecessary system behavior by deactivating the rules of disconnected devices.

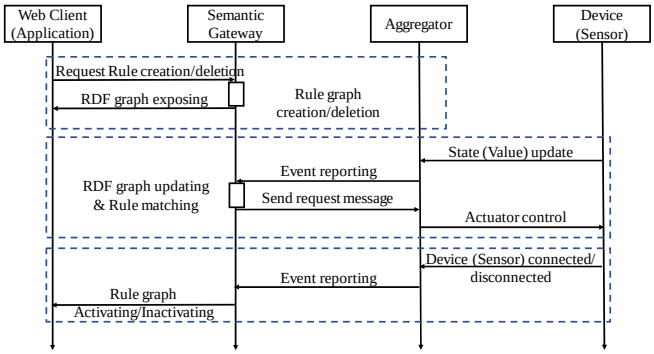


Figure 7. Semantic rule creation and matching process of the proposed framework

The RDF graph stored in the gateway's semantic database is provided to the user via the resource viewer module. The Web client provides all the mechanisms for rule management. The user can easily create rules for all devices or sensors represented by RDF graphs via the Web UI, rather than creating rules through hard coding at the code level. Many previous studies have designed rules that meet the user's requirements by hard-coding through rich expressions and conditions. The hard-coding method does not allow the user to dynamically manage the created rules, and additional work is required to create new rules. Through the rule customizing module of the proposed framework, the user can manage rules dynamically. In addition, the user can control the device through the gateway by requesting an update query for the device service status with the device control module. The Web client provides all the semantic functions through the Web UI, thereby eliminating the need for expert knowledge and skills and providing an endpoint that can satisfy the user requirements.

C. Difference from conventional solution

An ontology mapping algorithm should be utilized for the server or gateway to understanding the meaning of data. In the proposed scheme, the oneM2M standards-based aggregator integrates data from vertical silos. The gateway analyzes the oneM2M resource parameters and describes them in an RDF graph. Therefore, the gateway does not have to define a semantic vocabulary for all devices and can manage resources by performing semantic queries dynamically. The gateway performs semantic functions to minimize the mapping process for the semantic vocabulary through the lightweight RDF graph model illustrated in Fig. 5. Therefore, the semantic execution is faster compared to ontology-based solutions and more suitable for the IoT environment. In addition, the RDF graph extension scheme provides a mechanism for dynamically managing semantic data for M2M interactions through RDF graphs and semantic queries.

IV. Performance Evaluation

The aggregator is implemented based on Mobius, which is an opensource platform provided by OCEAN. We utilize the Jena Framework to develop semantic functions, which is a library for Java that supports RDF, OWL, Database, and SPARQL [25]. We employ the Mosquitto MQTT broker, and the MQTT client is implemented based on the Eclipse Paho library. The aggregator operates on Linux OS with Ubuntu, and the semantic IoT gateway operates on Raspberry Pi 3 B+. We also implement an ontology-based IoT framework for comparison experiments with the proposed framework.

Fig. 8 shows the classes, object properties, and data properties of the ontology, which was designed using Protege [26]. We designed the ontology based on the RDF graph model shown in Fig. 5, to perform the experiment under similar conditions. In addition, all resource and rule management are performed using the same mechanism for both the ontology-based framework and the proposed framework. We perform three comparative experiments to evaluate the proposed IoT framework and the ontology-based framework. All experiments are performed in a real IoT environment and are used to evaluate the latency of performing semantic functions. The latency is measured for each module from the time at which the report on the event is received from the aggregator to that at which the function is completed.

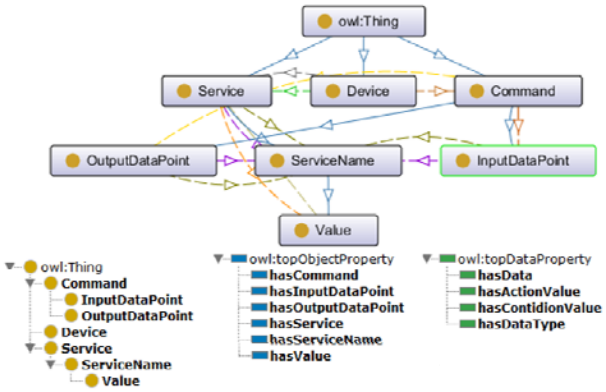
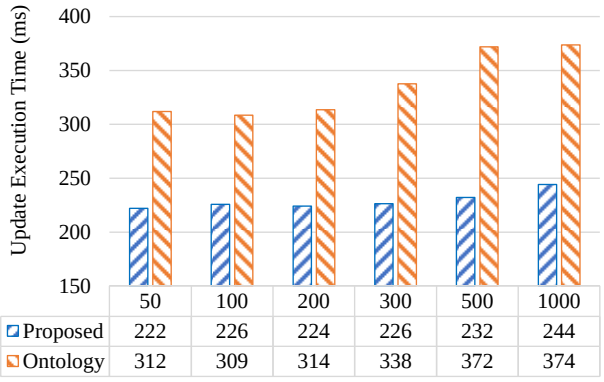


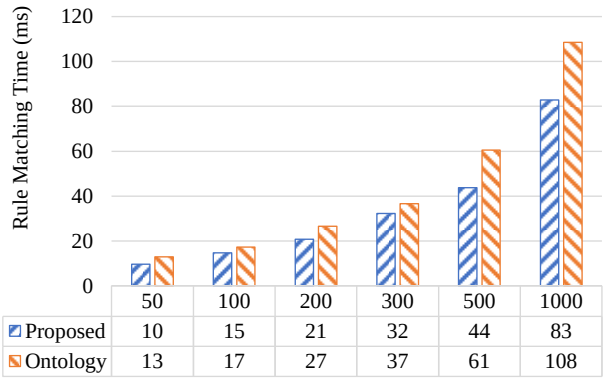
Figure 8. Designed ontology based on RDF graph model in Figure 5

In the first experiment, we evaluate the resource update function according to the number of devices. We measure the execution time for resource updating by increasing the number of devices from 50 to 1000. The average latency is calculated based on 200 measurements in every case. Fig. 9 illustrate the results for latency of the update process as a function of the number of devices. In experiments with less than 200 devices, there is little change in the latency for either framework. However, in experiments with over 300 devices, the latency of the proposed framework increased by approximately 20 ms, whereas that of the ontology-based framework increased by approximately 66 ms. The latency difference between the two frameworks results from the overhead in the resource mapping process of the ontology-based framework. As the number of RDF triples increases, the overhead of the resource mapping process increases the semantic function time of the ontology-based framework. The proposed framework demonstrates a performance improvement of between 26.8% and 40.1% in all experiments compared to the ontology-based framework.



	50	100	200	300	500	1000
Proposed	222	226	224	226	232	244
Ontology	312	309	314	338	372	374

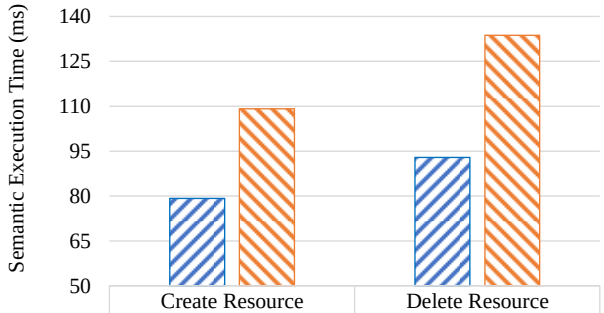
Figure 9. Execution time for resource updating according to the number of devices



	50	100	200	300	500	1000
Proposed	10	15	21	32	44	83
Ontology	13	17	27	37	61	108

Figure 11. Execution time for rule matching according to the number of rules

In the second experiment, we measure the latency of resource creation and deletion with 200 devices. We assume that each device has one service. Fig. 10 presents the execution time results for each resource creation and deletion process. The proposed framework generates six triples to represent each device. In contrast, the ontology-based framework uses 97 triples to represent an ontology, as illustrated in Fig. 8, and generates eight triples to represent a single device. The proposed framework generates 1200 triples to represent 200 devices and the ontology-based framework generates 1697 triples. In Fig. 10, the sums of the latencies for generation and deletion differ by approximately 52 ms and 71 ms, respectively, from the results in Fig. 9. These differences in latency represent the overheads that result from storing and managing the RDF triples at the gateway. In this experiment, this involves a tiny overhead of 0.04 ms per RDF triple. From the results of the experiment, we can conclude that the number of RDF triples does not have a noticeable effect on the latency of performing semantic functions.



	Create Resource	Delete Resource
Proposed	79	93
Ontology	109	134

Figure 10. Execution time for semantic functions when 200 devices connected

In the final experiment, we evaluate the rule matching execution time as a function of the number of rules. We perform the experiment by increasing the number of rules from 50 to 1000. The average latency is calculated from 50 measurements in each case. Fig. 11 shows the execution time for matching according to the number of rules. As observed in the previous two experiments, there is a difference in latency owing to resource mapping and triple management overhead in the ontology and the proposed framework improves performance by up to 27%.

As the number of RDF triples increases, the latency increases in the ontology-based solution owing to the overhead of the resource mapping process. However, because the proposed solution minimizes overheads of a mapping mechanism, the latency for semantic function execution is less affected by the number of triples. In IoT, the number of RDF triples can increase exponentially, as many devices are connected and devices have multiple services. The proposed solution is more efficient in processing all the data generated by devices or sensors than the solution that utilizing the ontology.

We constructed an IoT environment as illustrated in Fig. 12 for rule-based interactions between heterogeneous platform devices. The experimental environment consists of five IoT platforms, namely Nest, Foobot, SmartThings, Alljoyn and Philips, and a sensor that was developed with Raspberry Pi 3B+. Each platform device performs device registration, service registration, and resource creation based on the oneM2M standard, without knowledge of the semantic vocabulary used in the semantic gateway.

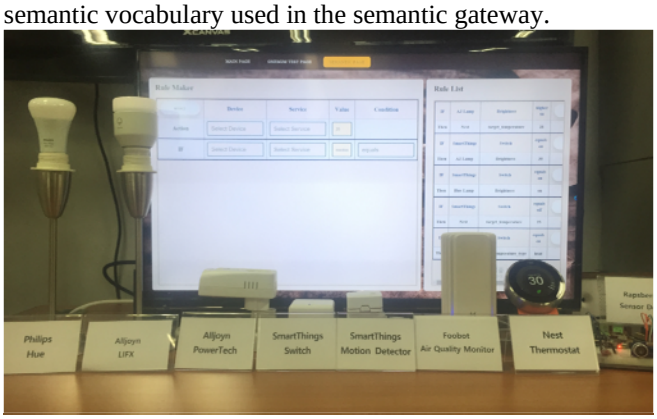


Figure 12. IoT environment for interaction between heterogeneous platforms

Fig. 13 shows the results for rule-based interaction between the Thermostat device of the Nest platform and the LIFX device of the Alljoyn platform. We previously created the following rule: "If the brightness of LIFX exceeds a specified level, then the thermostat's target temperature service controls the temperature to 28 degrees". When the gateway receives a report concerning an event occurring in LIFX, it searches the rule by understanding the meaning of the event. If there is a rule as shown in Fig. 13, then the comparator of the gateway checks the rule condition value and executes the service defined in the output datapoint.

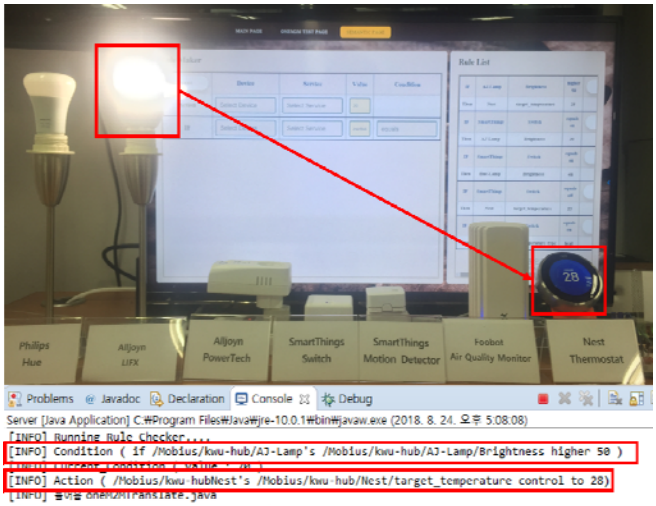


Figure 13. Rule-based interaction between Nest and Alljoyn platform

V. Conclusion

In this study, we proposed an IoT framework based on an RDF graph extension schema to achieve enhanced interoperability in IoT. We noted the following main shortcoming of ontologies: To interact with new devices or sensors, ontology-based solutions must define a semantic vocabulary or utilize ontology integration mechanisms. Defining semantic vocabularies for all IoT resource requires human intervention and can lead to reprogramming and hard coding problems. In addition, an ontology matching process is required for all data for a machine to understand the meaning of the data. We designed a lightweight RDF graph model for M2M interactions that solves the ontology problem in IoT. We integrate the data using the oneM2M standard aggregator. The semantic gateway analyzes the oneM2M resource parameters of the integrated data and represents them as semantic data. We conducted a comparative experiment with an ontology-based solution. In terms of the experiment on the semantic function execution time, the proposed framework outperformed the ontology-based framework by up to 40%, and by up to 27% in terms of rule matching. We have demonstrated experimentally that the proposed framework is more suitable for IoT than an ontology-based framework.

References

[1] J. Yun, I. Ahn, J. Song, and J. Kim, “Implementation of Sensing and Actuation Capabilities for IoT Devices Using oneM2M Platforms”, Sensors, vol. 19, no. 20, pp. 1-18, 2019, doi:10.3390/s19204567.

[2] J. Kim, J. Yun, S. Choi, D. N. Seed, G. Lu, M. Bauer, A. Al-Hezmi, K. Campowsky, and J. Song, “Standard-based IoT Platforms Interworking: Implementation, Experiences, and Lessons Learned”, IEEE Communications Magazine, vol. 54, pp. 48-54, 2016, doi:10.1109/MCOM.2016.7514163.

[3] J. Miranda, N. Makitalo, J. Garcia-Alonso, J. Berrocal, T. Mikkonen, C. Canal, and J. M. Murillo, "From the Internet of Things to the Internet of People", IEEE Internet Computing, vol. 19, pp. 40-47, 2015, doi:10.1109/MIC.2015.24.

[4] A. I. Maarala, X. S, and J. R, “Semantic matching for context-aware Internet of Things applications”, IEEE Internet of Things Journal, vol. 4, pp. 461-473, 2017, doi:10.1109/JIOT.2016.2587060.

[5] L. Daniele, F. D. Hartog, and J. Rose, “Created in Close Interaction with the Industry: The Smart Appliances REFerence (SAREF) Ontology”, International Workshop Formal Ontologies Meet Industries, vol. 255, pp. 100-112, 2015, doi:10.1007/978-3-319-21545-7_9.

[6] “W3C SSN Incubator Group Report”, 2011. [Online] Available: https://www.w3.org/2005/Incubator/ssn/

[7] K. Janowicz, A. Haller, S. J. D. Cox, D. L. huoc, and M. Lefrancois, “SOSA: A lightweight ontology for sensors, observations, samples, and actuators”, Journal of Web Semantics, vo 56, pp. 1-10, 2019, doi:10.1016/j.websem.2018.06.003.

[8] J. Kiljander, A. D’elia, F. Morandi, P. Hyttinen, J. Taskalo-mattila, A. Ylisaukko-oja, J. Soininen, and T. S. Cinotti, “Semantic Interoperability Architecture for Pervasive Computing and Internet of Things”, IEEE Access, vol. 2, pp. 856-873, 2014, doi:10.1109/ACCESS.2014.2347992.

[9] G. Stoilos, D. Geleta, J. Shamdasani, and M. Khodadadi, “A Novel Approach and Practical Algorithms for Ontology Integration”, in Proc. International Semantic Web Conference, vol. 11136, pp. 458-476, 2018, doi:10.1007/978-3-030-00671-6_27.

[10] G. Xiao, D. Hovland, D. Bilidas, M. Rezk, M. Giese, and D. Calvanese, “Efficient Ontology-Based Data Integration with Canonical IRIs”, in Proc. European Semantic Web Conference, vol. 10843, pp. 697-713, 2018, doi:10.1007/978-3-319-93417-4_45.

[11] H. Kubicek, R. Cimander, H. J. Scholl, “Organizational Interoperability in E-government”, Springer Verlag, 2011.

[12] M. Ganzha, M. Paprzycki, W. Pawlowski, P. Szmeja, and K. Wasielewska, “Towards Semantic Interoperability Between Internet of Things Platforms”, Integration, Interconnection, and Interoperability of IoT Systems, Springer, pp 103-127, 2017, doi:10.1007/978-3-319-61300-0_6.

[13] oneM2M-TS-0001, “Functional architecture”, v3.11.0, 2018.

[14] K. Gilani, J. Kim, J. Song, D. Seed, and C. Wang, “Semantic Enablement in IoT Service Layers-Standard Progress and Challenges”, IEEE Internet Computing, vol. 22, pp. 56-63, 2018, doi:10.1109/MIC.2018.043051465.

[15] P. Jain, P. Hitzler, A. P. Sheth, K. Verma, and P. Z. Yeh, “Ontology Alignment for Linked Open Data”, in Proc. International Semantic Web Conference, Springer, Berlin, Heidelberg, vol 6496. pp. 402-417, 2010, doi:10.1007/978-3-642-17746-0_26.

[16] E. Miller, “An Introduction to the Resource Description Framework”, Bulletin of the American Society for Information Science and Technology, vol. 25, no. 1, pp. 15-19, 2005, doi:10.1002/bult.105.

[17] 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 Gateways”, IEEE Transactions on Industrial Informatics, vol. 14, pp. 715-724, 2018, doi:10.1109/TII.2017.2769001.

[18] M. B. Alaya, S. Medjiah, T. Monteil, and K. Drira, “Toward Semantic Interoperability in oneM2M Architecture”, IEEE Communications Magazine, vol. 53, pp. 35-41, 2015, doi:10.1109/MCOM.2015.7355582.

[19] M. Bermudez-Edo, T. Elsaleh, P. Barnaghi, and K. Taylor, “IoT-Lite: a Lightweight Semantic Model for the Internet of Things and Its Use with Dynamic Semantics”, Personal and Ubiquitous Computing, vol. 21, no. 3, pp. 475-487, 2017, doi:10.1007/s00779-017-1010-8.

[20] D. D. Marino, A. Esposito, S. A. Maisto, and S. Nacchia, “A Semantic IoT Framework to Support RESTful Devices’ API Interoperability”, in Proc. IEEE International Conference on Networking, Sensing and Control, pp. 78-83, 2017, doi:10.1109/ICNSC.2017.8000071.

[21] 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, doi:10.1109/ACCESS.2017.2763425.

[22] M. Ganzha, M. Paprzycki, W. Pawlowski, P. Szmeja, and K. Wasielewska, “Semantic Interoperability in the Internet of Things: an Overview from the INTER-IoT Perspective”, Journal of Network and Computer Applications, vol. 81, pp. 111-124, 2017, doi:10.1016/j.jnca.2016.08.007.

[23] T. Yokotani, and Y. Sasaki, “Comparison with HTTP and MQTT on Required Network Resources for IoT”, in Proc. IEEE International Conference on Control, Electronics, Renewable Energy and Communications, pp. 1-6, 2016, doi:10.1109/ICCEREC.2016.7814989.

[24] D. Thangavel, X. Ma, A. Valera, H. X. Tan, and C. K. Y. Tan, “Performance Evaluation of MQTT and CoAP via a Common Middleware”, in Proc. IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing Conference, pp. 1-6, 2014, doi:10.1109/ISSNIP.2014.6827678.

[25] B. McBride, “Jena: a semantic Web toolkit”, IEEE Internet Computing, vol. 6, no. 6, pp. 55-59, 2002, doi:10.1109/MIC.2002.1067737.

[26] N. F. Noy, M. Sintek, S. Decker, M. Crubezy, R. W. Fergerson, and M. A. Musen, “Creating Semantic Web contents with Protégé-2000”, IEEE Intelligent Systems, vol. 16, no. 2, pp. 60-71, 2001, doi:10.1109/5254.920601.