Enhancing Semantic Discovery in oneM2M with Direct Query

Setiawan Wibowo Purnomo
EECS International Graduate Program
National Chiao Tung University
Hsinchu, Taiwan
setiawantong.cs05g@nctu.edu.tw

Fuchun Joseph Lin
Department of Computer Science
College of Computer Science
National Chiao Tung University
Hsinchu, Taiwan
fjlin@nctu.edu.tw

***Abstract*—Semantic information has been proven to be necessary in order to increase IoT interoperability by adding meaningful annotations to the data under exchange. The oneM2M as a global standard for IoT middleware has already supported semantic capabilities and allows semantic information to be annotated in its resources. Based on the added semantic information, oneM2M can support more effective resource discovery with semantic discovery. However, the oneM2M approach for semantic discovery is based on indirect query that requires pre-collection of all semantic information distributed in the resource tree while performing the discovery, thus results in very slow response. In this research, we propose a method of direct query to expedite the function of semantic discovery in oneM2M. In our approach, instead of storing the semantic information in the resource tree, we store the semantic information separately and centrally in a permanent RDF store. Our method significantly reduces the response time when performing semantic querying.**

Keywords—Internet of Things; oneM2M; semantic discovery

I. INTRODUCTION

As predicted by analysts [1], there will be at least 20 billion devices deployed in the cyber-physical world that blends together our physical environment and cyber virtual world, known as the Internet of Things (IoT) [2]. These devices are connected and thus required to interwork with each other, regardless their differences in type, purpose and model [3]. However, due to the heterogeneity of these devices they often have troubles in understanding the data exchanged by each other [4].

Semantic information helps to solve this problem by providing the meaning of exchanged data with additional annotations [5]. This adds a new level of interoperability without having to memorize the syntactic structure of the exchanged data [6][7]. Semantic information enables the transmission of the knowledge regarding the exchanged data.

The oneM2M is a global middleware standard for the machine-to-machine communications (M2M) [8]. It provides a platform, called Common Service Entity (CSE), for IoT devices to connect to each other with the support of common service functions (CSFs) [9] such as data management and repository (DMR), discovery (DIS), communication management, subscription & notification and registration. The oneM2M Release 2 specification added a semantic engine (SEM) CSF [10] in its CSE, which provides a temporary RDF store and an ability to execute a given semantic query on the RDF store. In addition, its DMR CSF is enhanced to store the semantic information, and its DIS CSF is enabled to collect the semantic information stored inside DMR CSF.

The current oneM2M adopts an approach we refer to as indirect query for semantic operations in oneM2M including (1) manipulating semantic information and (2) discovering resources based on semantic query, which is also referred as semantic discovery. In this approach, DIS CSF has to explore the semantic information of each resource in the DMR CSF every time the semantic discovery is executed.

However, this approach is very time consuming and cannot meet the low latency requirement of time-critical applications. To solve this issue, we propose an approach of direct query to improve the response time of semantic discovery. In ou approach, the need for DIS CSF to collect semantic information from DMR CSF is eliminated. The test result shows that our method significantly shortens the response time of performing semantic discovery and reduces the complexity of implementation required.

The rest of this paper is organized as follows. Section II focuses on the semantic information annotation, the background of oneM2M and the indirect query. Section III explains our proposed method of direct query. Section IV shows the response time comparison of indirect query and direct query in a smart-home environment. Finally, we present our conclusion and future work in Section V.

II. BACKGROUND

This section briefly introduces the fundamental of annotating semantic information, the functional architecture of oneM2M and the indirect approach supported by oneM2M to handle semantic discovery.

A. *Semantic Information Annotation*

Annotating resources with semantic information can be done by describing the resource semantically in the form of

Resource Description Framework (RDF). The resource description in RDF is expressed in a *triple*, which contains:

1. "Subject" (S): the annotated resource.
2. "Predicate" (P): an identifier that specifies the relationship between the subject and the object.
3. "Object" (O): a resource or literal that has the relation with the subject.

An example is illustrated Figure 1(a) where two *triple*s are used to describe an air conditioner located in a kitchen: "Kitchen has an air conditioner" and "Air Conditioner is an actuator". In this example, the air conditioner can act as a subject and an object, thus creating an information chain. These *triple*s need to be serialized in other formats, such as XML (See Figure 1(b)) to enable distribution from one entity to another.

B. *oneM2M*

The oneM2M CSE consists of 13 CSFs and provides the Mca reference point for an Application Entity (AE) to access these CSFs. This Mca reference point currently supports five operations: (1) CREATE, (2) RETRIEVE, (3) UPDATE, (4) DELETE, and (5) NOTIFY [11], over several communication protocols including HTTP.

In particular, the DMR (Data Management & Repository) CSF is responsible for storing data from applications in a resource tree as depicted in Figure 2. The root of the resource tree is the <CSEBase> resource that consists of children resources such as <remoteCSE>, <AE>, <container>, <group>. Under an <AE> resource, there can be multiple <container> resources that are used to store data. Both <AE> and <container> resources can have semantic information stored in the <semanticDescriptor> resource that contains several important attributes (see Figure 3) as follows.

- "descriptorRepresentation" contains the serialization format of the semantic information.
- "semanticOpExec" is for placing a SPARQL query to update the semantic information.
- "descriptor" contains the semantic information of the parent resource in a form of RDF graph data model.
- "ontologyRef" contains a URI that describes an ontology used in the descriptor.
- "relatedSemantics" contains a list of URIs pointing to other <semanticDescriptor> resources that have relation with the current <semanticDescriptor> resource.

Kitchen (S) has (P) an air conditioner (O)

Air conditioner (S) is (P) an actuator (O)

(a)

```
<rdf:Description rdf:about="Kitchen">
  <has>
    <Actuator rdf:about="Air-Conditioner"/>
  </:has>
</rdf:Description>
```

(b)

Figure 1. Example of semantic information in (a) triple and (b) XML

C. *Indirect Query Supported by oneM2M*

Semantic discovery in indirect query relies on three CSFs:

- DMR CSF to provide the <semanticDescriptor> resources to store the semantic information that is annotated on a resource.
- DIS (Discovery) CSF to collect semantic information from each <semanticDescriptor> resource in the resource tree.
- SEM (Semantic Engine) CSF to execute the given semantic query on the extracted semantic information.

The process of semantic discovery in indirect query is depicted in Figure 4, with the explanation as follows:

1. The requester sends an HTTP GET request with the format as illustrated in Table I. The URI points to the root of searching in the resource tree, and encodes two parameters: a SPARQL query and a filterUsage "fu=1" indicating a discovery request [12]. The header contains the identity of the requester ("X-M2M-Origin" → "user:password").
2. The Mca reference point receives the request, marks it as a RETRIEVE operation with discovery request, and forwards it to the DIS CSF for further operation.
3. The DIS CSF explores <semanticDescriptor> resources by performing breadth-first-search to traverse the resource tree starting from the specified search-root resource.

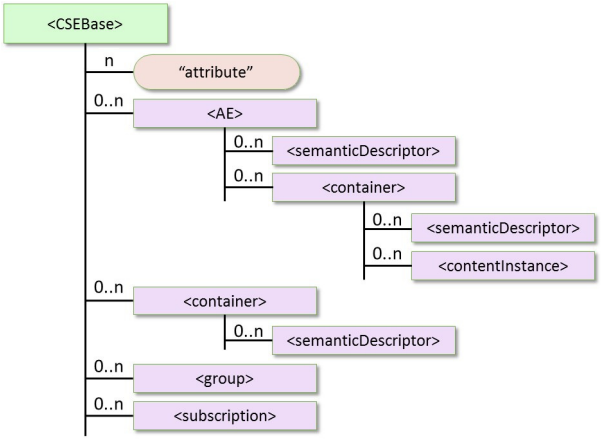

Figure 2. Resource tree in DMR CSF

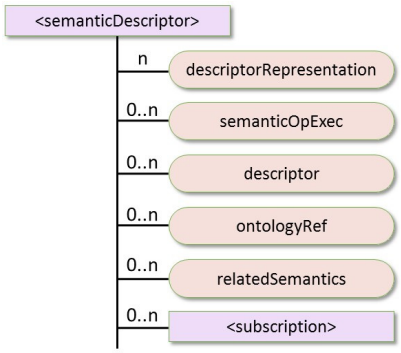

Figure 3. <semanticDescriptor> resource

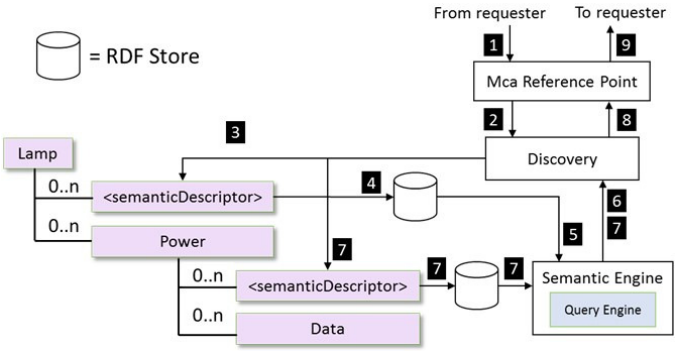

Figure 4. Process flow of semantic discovery

TABLE I. HTTP REQUEST PARAMETER IN INDIRECT QUERY

Description	Content
Operation	HTTP GET
URI	http://<CSEAddress>/~/<searchRoot> ?smd=<SPARQLQuery>&fu=1
Header	"X-M2M-Origin" → "user:password"
Body	-

4. The semantic information of each <semanticDescriptor> resource would be extracted and stored in a temporary RDF store.
5. The SEM CSF will execute the given SPARQL query on the temporary RDF store.
6. If the query returns any result, the URI of the resource with the matched semantic information will be saved and sent back to the DIS CSF.
7. The DIS CSF continues traversing the resource tree and repeats Steps 4-6 until no more <semanticDescriptor> resource can be found.
8. A response message is created containing all the saved URIs. This response message is sent to the Mca reference point to be forwarded to the requester.
9. The Mca reference point forwards the response message to the requester.

Furthermore, extracting semantic information from a <semanticDescriptor> resource requires more steps which are depicted in Figure 5 and explained as follows:

1. The DIS CSF finds a <semanticDescriptor> resource.
2. The semantic information from the *descriptor* attribute is extracted and put into a RDF Store.
3. If there is any URI in the relatedSemantics attribute, If there is any URI in the relatedSemantics attribute, the DIS CSF will proceed to analyze the <semanticDescriptor> resource mentioned in the URI.

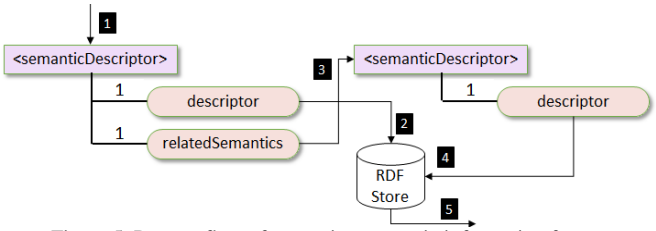

Figure 5. Process flow of extracting semantic information from a <semanticDescriptor> resource

4. The semantic information from the *descriptor* attribute on another <semanticDescriptor> resource is extracted, and put in the same RDF store.
5. If there is no more URI in the *relatedSemantics* attribute, the process of extracting semantic information is considered finished. The RDF store is now ready for the SEM CSF to execute the semantic query.

III. PROPOSED METHOD

In this research, we propose a new approach of "direct query" for the semantic discovery in oneM2M. The idea is to allow the semantic information to be stored in a permanent RDF store residing in the SEM CSF than in the <semanticDescriptor> resources. Our approach has the following benefits:

1. Reducing the coding complexity by unifying the HTTP request parameters on both semantic information manipulation and semantic discovery.
2. Faster processing time on semantic discovery.

Since the <semanticDescriptor> resource is no longer used, the need for DIS CSF to collect and extract semantic information from the <semanticDescriptor> resources can be eliminated to simplify the process of semantic discovery. The process would no longer be triggered by a RETRIEVE operation of discovery request in indirect query. Instead, it requires the support of a new operation type "QUERY" at the Mca reference point, that redirects the request to the SEM CSF for executing the given SPARQL query. This QUERY operation type is mapped to an HTTP POST with a Content-Type "query".

In a direct query, the process flow of manipulating semantic information and discovering resources with semantic query are the same. They are all accomplished by SPARQL with different types: SPARQL INSERT DATA to create new semantic information, SPARQL DELETE to delete the existing semantic information, and SPARQL SELECT to perform semantic discovery.

The whole process of direct query is depicted in Figure 6, with the explanation as follows:

1. The requester sends an HTTP POST request (see the format in Table II) with a URI pointing to the CSE. The header contains the identity of the requester and the type of the content "query" to indicate a semantic query request. The SPARQL query is carried in the HTTP body without any encoding format.
2. The Mca reference point receives the request, marks it as a QUERY operation, and forward it to the SEM CSF for further operation.
3. The query engine in the SEM CSF executes the given SPARQL query on the RDF store.
4. The SEM CSF creates a response message containing the execution result from executing SPARQL query. For SPARQL SELECT query, the response message is expanded with the result from the execution if any. This response message is sent to the Mca reference point to be forwarded to the requester.
5. The Mca reference point forwards the response message to the requester.

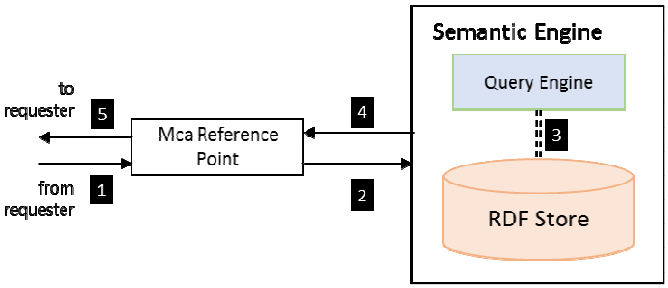

Figure 6. Process flow in direct query

TABLE II. HTTP REQUEST PARAMETER IN DIRECT QUERY

Description	Content
Operation	HTTP POST
URI	http://<CSEAddress>/~/in-cse/
Header	"X-M2M-Origin" → "user:password" "Content-Type" → "query"
Body	<SPARQLQuery>

However, removing the <semanticDescriptor> resource brings in two new issues: First, the relation between the semantic information and the annotated resource disappears. Second, it becomes incapable of sharing semantic information between resources. To overcome the first issue, the URI of the annotated resource is used as the subject in the RDF triple as illustrated in Figure 7. Note that the semantic information in direct query is exactly the same as that in the <semanticDescriptor> resource in indirect query, except that the subject in the RDF triple is replaced with the URI of the annotated resource "AC".

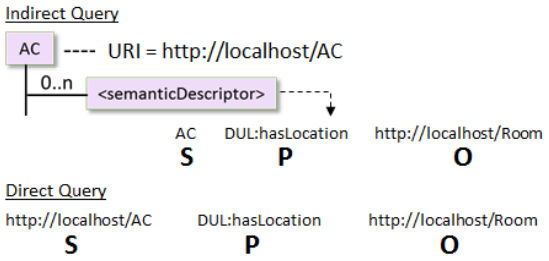

Figure 7. Example of using the URI of a resource as the subject in direct query

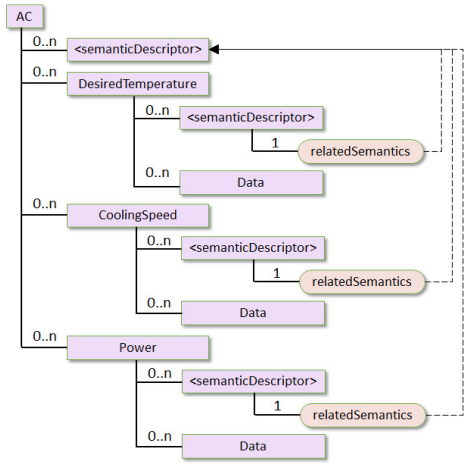

Figure 8. An example of *relatedSemantics* attribute containing a URI to another <semanticDescriptor> resource

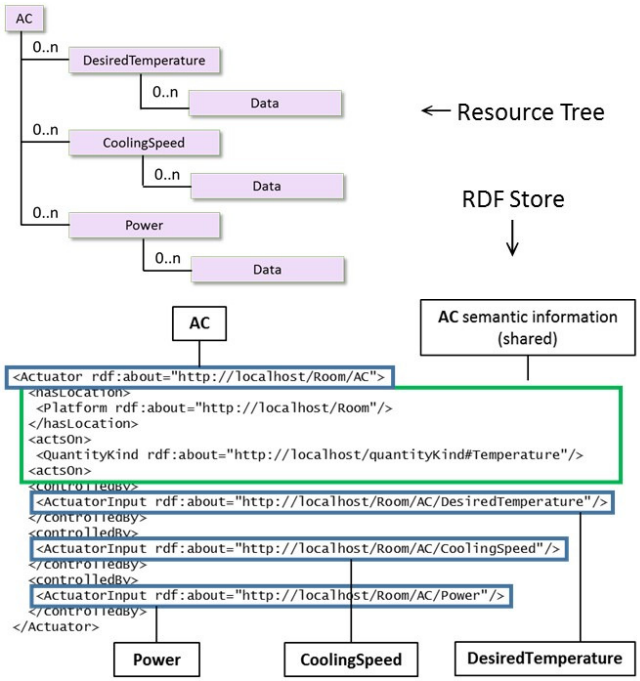

Figure 9. Sharing semantic information in direct query

As an example of the second issue, note that the <semanticDescriptor> resource has the relatedSemantics attribute to allow including some existing external semantic information as a part of the semantic information of the annotated resource. With this resource, semantic information can be shared between different resources. For example, Figure 8 shows that an air conditioner "AC" contains three properties: "DesiredTemperature", "CoolingSpeed", and "Power". Each property contains <semanticDescriptor> resource with a relatedSemantics attribute referring to the <semanticDescriptor> resource of its parent "AC".

The direct query approach disables the sharing of related semantic information due to the removal of the <semanticDescriptor> resource. To solve this issue, we put the resource with the semantic information to be shared as the parent in the RDF store, and other resources that depend on the shared semantic information as the children. An example is illustrated in Figure 9 where the "AC" has semantic information to be shared with its properties. As these properties of the "AC" depend on the semantic information of the "AC", they are put as the children of "AC" so that they can share the semantic information of "AC".

A. *Testing Scenario*

Inspired from ADREAM Smart Building Use Case [13], we assume a smart home with multiple rooms, where each room consists of various kinds of actuators and sensors including lamp, air conditioner, humidifier, luminosity sensor, temperature sensor, and humidity sensor. Each device is connected to a CSE through the Mca reference point.

We designed an ontology as depicted in Figure 10 for the

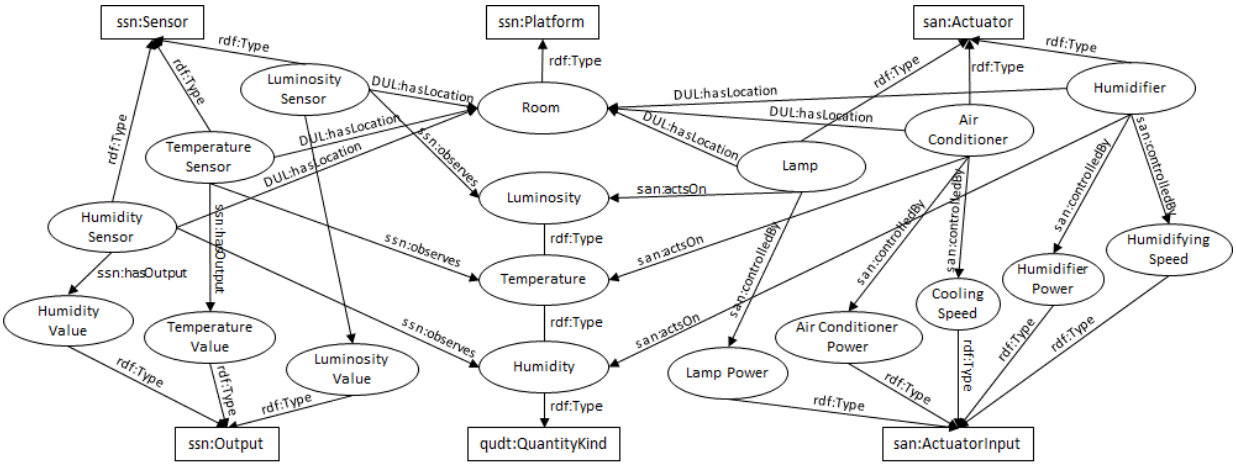

Figure 10. Ontology for the smart-home environment

TABLE III. TESTBED HARDWARE CONFIGURATION

Type	Value
Processor	Intel i5-6200U clocked to 2.08 GHz
RAM	8 GB DDR3L 1.6 GHz
Storage	256 GB SSD – R/W up to 560MBps/320MBps
Operating System	Microsoft Windows 8.1 x64

smart home environment. This ontology provides the guidance for annotating semantic information on smart home resources. It combines existing ontologies such as SAN [14], SSN [15], and QUDT [16]. Each actuator (Lamp, Air Conditioner, and Humidifer) in Figure 10 is of (rdf:Type) SAN:Actuator and controlled by its own unique properties (SAN:controlledBy); these properties are of (rdf:Type) SAN:ActuatorInput. The actuators act on a specific quantity kind (Lamp, Temperature, and Humidity) (SAN:actsOn), which is of qudt:QuantityKind. Similarly for sensors (Luminosity Sensor, Temperature Sensor, and Humidity Sensor) in Figure 10, they are of (rdf:Type) ssn:Sensor, which senses a specific quantity kind in the environment (ssn:observes), and generates an output value (Luminosity Value, Temperature Value, and Humidity Value) (ssn:hasOutput), which is of (rdf:Type) ssn:Output.

The testing begins with the installation of three sensors in multiple rooms. The sensors start their operations by periodically sensing the Assuming that the CSE finds the resources that match with the request criteria, the CSE will respond with the URIs of the resources containing the sensed values of the corresponding sensors. Then, the actuator would update its status based on the value of the corresponding sensor. Lastly, the actuators are removed from the environment, followed by removing the sensors.

B. *Testing Method*

To compare the response time of the direct query with that of the indirect query in performing semantic query request, we constructed two systems based on OM2M, which is a oneM2M open source project developed by LAAS-CNRS and managed by the Eclipse Foundation.environment and uploading the sensed value to the CSE. Then, three actuators are also installed in the same rooms. The actuators start their operations by periodically sending a semantic discovery request to the CSE in order to find its corresponding sensors in the same room and discover the resources where the sensed value from these sensors are stored.

The response time test is done by running OM2M as the oneM2M CSE, and a smart-home simulator as an AE over the CSE, in a hardware configuration as specified in Table III. The response time is measured on 3 operations:

1. Creating semantic information, when a device is installed.
2. Performing semantic discovery, when an actuator finds the sensed value of the corresponding sensor.
3. Deleting semantic information, when a device is removed.

The measurement is also done on 3 different environments:

1. Environment A: One device of each type (See Figure 10) installed in a room (total 6 devices)
2. Environment B: One device of each type installed in two rooms (total 12 devices)
3. Environment C: One device of each type installed in five rooms (total 30 devices)

C. *Testing Result*

The response time in Figure 11 is obtained by calculating the average of all response times collected from 100 executions of each operation in each environment. The results show that the direct query performs significantly faster than the indirect query in performing semantic discovery. This improvement is because the direct query does not traverse the resource tree to explore and extract the semantic information from <semanticDescriptor> resources as in the indirect query.

On the other hand, the direct query requires more time to create or delete semantic information when compared to indirect query. As the amount of the semantic information in RDF store is increasing, the direct query requires more time to perform these operations. In direct query, the new semantic information is inserted in a specific element according to its

relation with other semantic information. Thus deleting semantic information requires searching a specific information to be deleted among numerous of semantic information in the RDF store. These operations are more time-consuming than in the case of indirect query where inserting and deleting semantic information are treated as creating or deleting a <semanticDescriptor> resource that is a fast operation.

In summary, the direct query should be preferred in a system that highly depends on semantic discovery than on semantic information manipulation. However, the indirect query is still considered a better solution in a dynamic system that requires lots of semantic information manipulation than semantic discovery.

IV. Conclusion and Future Work

In the direct query approach, the semantic information is stored in a permanent RDF store instead of in <semanticDescriptor> resources. This eliminates the need of DIS CSF to discover the <semanticDescriptor> resources and extract their semantic information. On the other hand, it requires the support of a new operation type "QUERY" that would forward the request directly to the SEM CSF.

When compared the response time of the direct query with that of indirect query in a smart-home environment, the result shows that the direct query performs significantly faster than indirect query in discovering resources with semantic query.

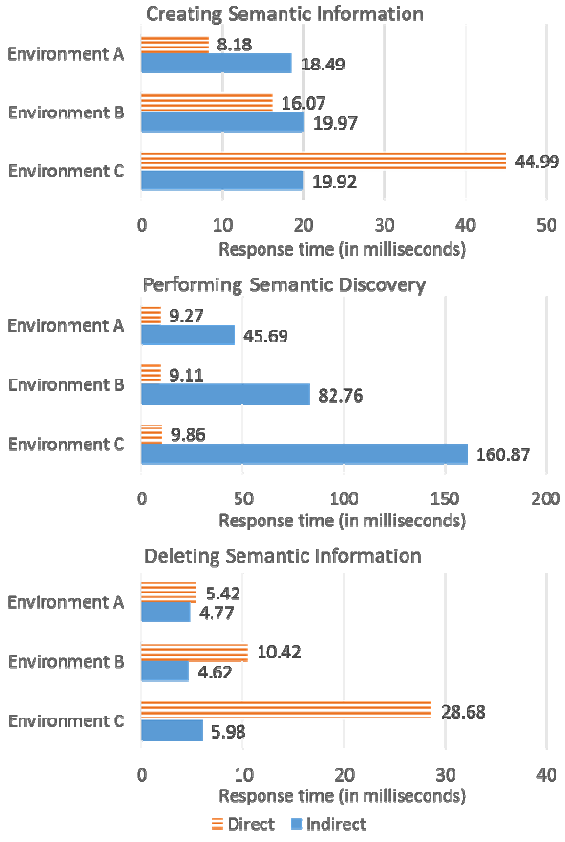

Figure 11. Response time benchmark result

In the future, we plan to design a new hybrid method that combines both indirect query and direct query to achieve optimal performance.

Acknowledgement

The project reported in this paper is sponsored by Institute for Information Industry in Taiwan under the 2017 Initiative on *Docker-based OpenFog IoT Technologies* and the 2018 Initiative on *Gateway Site Fog Computing*.

References

[1] A. Nordrum, "The internet of fewer things [News]," in *IEEE Spectrum*, vol. 53, no. 10, pp. 12-13, October 2016.

[2] V. Jirkovský, M. Obitko and V. Mařík, "Understanding Data Heterogeneity in the Context of Cyber-Physical Systems Integration," in *IEEE Transactions on Industrial Informatics*, vol. 13, no. 2, pp. 660-667, April 2017.

[3] I. Yaqoob *et al.*, "Internet of Things Architecture: Recent Advances, Taxonomy, Requirements, and Open Challenges," in *IEEE Wireless Communications*, vol. 24, no. 3, pp. 10-16, June 2017.

[4] P. Desai, A. Sheth and P. Anantharam, "Semantic Gateway as a Service Architecture for IoT Interoperability," *2015 IEEE International Conference on Mobile Services*, New York, NY, 2015, pp. 313-319.

[5] W. Li, G. Privat and F. Le Gall, "Towards a semantics extractor for interoperability of IoT platforms," *2017 Global Internet of Things Summit (GIoTS)*, Geneva, 2017, pp. 1-6.

[6] H. Dibowski, "Semantic interoperability evaluation model for devices in automation systems," 2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Limassol, 2017, pp. 1-6.

[7] Fortino, Giancarlo, et al. "Towards multi-layer interoperability of heterogeneous IoT platforms: the INTER-IoT approach." Integration, Interconnection, and Interoperability of IoT Systems. Springer, Cham, 2018, pp. 199-232.

[8] V. Gazis, "A Survey of Standards for Machine-to-Machine and the Internet of Things," in *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 482-511, Firstquarter 2017.

[9] oneM2M. *TS 0001 Functional Architecture v2.10.0*. [Online]. Available: http://www.onem2m.org/images/files/deliverables/Release2/ TS-0001-%20Functional_Architecture-V2_10_0.pdf

[10] oneM2M. *TR 0007 Study of Abstraction and Semantics Enablements v2.11.1*. [Online]. Available: http://www.onem2m.org/images/files/deliverables/Release2/TR-0007-Study_on_Abstraction_and_Semantics_Enablement-V2_11_1.pdf

[11] oneM2M. *TS 0004 Service Layer Core Protocol Specification v2.7.1*. [Online]. Available: http://www.onem2m.org/images/files/deliverables/Release2/TS-0004_Service_Layer_Core_Protocol_V2_7_1.zip

[12] oneM2M. *TS 0009 HTTP Protocol Binding v2.6.1*. [Online]. Available: http://www.onem2m.org/images/files/deliverables/Release2/TS-0009-HTTP_Protocol_Binding-V2_6_1.pdf

[13] M. B. Alaya, S. Medjiah, T. Monteil and K. Drira, "Toward semantic interoperability in oneM2M architecture," in *IEEE Communications Magazine*, vol. 53, no. 12, pp. 35-41, Dec. 2015.

[14] LAAS-CRNS and IRIT. *SAN* [Online]. Available: https://www.irit.fr/recherches/MELODI/ontologies/SAN.html

[15] W3C Semantic Sensor Network Incubator Group. *Semantic Sensor Network Ontology* [Online]. Available: https://www.w3.org/2005/Incubator/ssn/ssnx/ssn

[16] Linked Models. *Quantities, Units, Dimensions and Types Catalog* [Online]. Available: http://www.linkedmodel.org/catalog/qudt/1.1/index.html