Journal of King Saud University – Computer and Information Sciences 34 (2022) 4191–4208

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

Resource identifier interoperability among heterogeneous IoT platforms

Jahoon Koo*, Young-Gab Kim*

Department of Computer and Information Security, and Convergence Engineering for Intelligent Drone, Sejong University, Seoul, Republic of Korea

A R T I C L E   I N F O

Article history:
Received 1 November 2021
Revised 3 April 2022
Accepted 4 May 2022
Available online 8 May 2022

Keywords:
Internet of Things
Interoperability
IoT platform
Resource identifier
Smart city

A B S T R A C T

Many standards, projects, and platforms are being developed as the Internet of Things (IoT) is adopted in a wide range of fields. However, because each IoT platform is based on a different resource identifier (ID), it is difficult to identify each device and use the service among heterogeneous IoT platforms. To solve this problem, we propose an interoperability framework that includes an IoT resource name system (RNS) based on analysis of the resource IDs (i.e., device ID and resource request formats) of five selected IoT platforms: oneM2M, Oliot, Watson IoT, IoTivity, and FIWARE. The IoT RNS converts a specific resource path into a resource request format for each platform. The converted resource path is shared among IoT RNSs for each platform, and users can request services from other platforms using converted resource paths. We also present an example of interoperability scenario among heterogeneous IoT platforms using the proposed IoT RNS in a smart city. The scenario includes each stage, such as resource registration and deletion, sharing mapping tables, converting resource addresses, and service requests. Furthermore, to prove the aims of the proposed approach, we implemented the resource interoperability scenario between oneM2M and FIWARE. In the experiments, resources can interwork in the two platforms through resource path conversion. Based on the results, we performed a qualitative evaluation of the IoT RNS with the current studies. In conclusion, our proposal overcomes the issues of building an existing integrated platform or specific central ontology and duplicating resources inside the platform. In addition, we separate the functions of the root and local IoT RNSs to solve communication traffic and memory capability issues.

1. Introduction

Internet of Things (IoT) technology is developing and expanding in various fields such as smart homes, healthcare, smart cities, logistics, and smart car. The international standard, "ISO/IEC 20924:2018 Information technology—Internet of Things (IoT)—Vocabulary" (ISO/IEC, 2021), defines the IoT as the "infrastructure of interconnected entities, people systems, and information resources together with services that process and react to information from the physical and virtual world." In other words, the IoT is hyper-connectivity among smart things, services, and humans to provide useful and seamless services regardless of the types of networks, devices, and platforms and with minimum human involvement. These technologies, related standards, projects, and platforms are continuously being developed (Lee et al., 2021). In particular, IoT platforms are an essential factor in providing interoperability because they support the network connection to various devices (e.g., sensors and access points) and provide services to users. According to the IoT Platform Companies Landscape & Database 2020, the official number of IoT platform companies in the open market is more than 620, up from 450 in 2017. For example, many platforms (e.g., AllSeen Alliance AllJoyn, Apple HomeKit, oneM2M, FIWARE, Google Cloud IoT, GS1 Oliot, IBM Watson IoT, Microsoft Azure, and OCF IoTivity) are being developed to provide various services. Therefore, interoperability, such as requesting services and sharing resources among diverse IoT platforms, is important, and it is an essential factor for building a real IoT environment that provides seamless services regardless of the type of IoT platform.

Platform interoperability is challenging to resolve due to various issues such as support for diverse protocols, device discovery,

* Corresponding authors at: Department of Computer and Information Security, and Convergence Engineering for Intelligent Drone, Sejong University, Seoul 05006, Republic of Korea.

E-mail addresses: sigmao91@sju.ac.kr (J. Koo), alwaysgabi@sejong.ac.kr (Y.-G. Kim).

Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

well-defined semantic management, and processing of data formats in heterogeneous platforms. However, current diverse platforms and related standards make it difficult to achieve interoperability and collaboration among heterogeneous IoT platforms. Each IoT platform has been developed using a specific and unique device identifier and resource-request format. These unique formats cause difficulty in identifying each resource among heterogeneous IoT platforms. For example, as depicted in Fig. 1, suppose that four IoT platforms (i.e., oneM2M, Watson IoT, FIWARE, and IoTivity) are connected to the same middleware. It will cause interoperability issues in resource discovery and resource requests due to the differences in resource discovery and request format of each platform.

OneM2M users check the available services using the resource discovery provided by oneM2M, but because resource discovery is based on identifiers unique to each platform, resources of heterogeneous IoT platforms are not discovered. In addition, even if a oneM2M user sends a request in the oneM2M format, the meaning of the request and the included resource identifier cannot be understood because each platform uses a different identifier and request format. Therefore, even if multiple platform resources are connected to the same middleware, the user is provided with only limited services. To solve these resource interoperability issues, existing research focuses mainly on integrating the ontology of each platform or duplicating the resources of other platforms on the corresponding platform. Furthermore, the existing interoperability schemes need to construct specific ontologies (Thiéblin et al., 2020; Ranpara and Kumbharana, 2021) and consume a significant amount of memory (Li et al., 2019).

Therefore, to overcome the limitations of the existing approaches, we propose an IoT resource name system (RNS), which focuses on mapping and converting the format of a resource identifier (ID) among diverse IoT platforms. We concentrate on converting resource paths (e.g., device ID, resource-request format) used in a specific IoT platform to the target IoT platform, but data transmission issues (e.g., data format, supported network protocols) are not addressed. Koo et al. (Koo and Kim, 2017; Koo et al., 2019; Koo and Kim, 2021) proposed an architecture for semantic interoperability and implemented a simple testbed for interoperability based on resource path conversion by mapping the IDs in heterogeneous IoT platforms. We use this approach as the initial model for the current study. In particular, we clearly distinguish and describe procedures such as creating, deleting, and converting resources for interoperability and service requests in a smart city. The proposed IoT RNS analyzes and converts resource IDs into the desired request formats, including reconfiguring requests among heterogeneous IoT platforms as appropriate for the requested resources. The contributions of this paper include the following:

(1) The proposed IoT RNS focuses on the resource IDs of the IoT platforms. We analyzed and mapped the format of resource IDs and service requests used by five representative IoT platforms. The IoT RNS uses the mapped information to convert the resource IDs.
(2) We also propose a conceptual framework and architecture to convert resource IDs among heterogeneous IoT platforms. The resource registration, discovery, path conversion, and service request processes are proposed, and examples of each step are presented through resource interoperability scenarios in a smart city.
(3) Furthermore, to prove the aims of the proposed approach, we created an implementation scenario for two IoT platforms (i.e., oneM2M and FIWARE). Experimental results indicate that resources can interwork in the two platforms through resource path conversion. Based on the results, we performed a qualitative evaluation of our proposal with the current studies.

The remainder of this paper is organized as follows. Section 2 analyzes background knowledge and summarizes related works regarding interoperability in IoT environments. Section 3 proposes an IoT RNS architecture, including a scenario in the smart city environment. Section 4 implements an interoperability scenario of our proposal. Section 5 explains the comparative evaluation and discusses the limitations. Finally, Section 6 summarizes this study as a conclusion and future work.

2. Background and related work

Section 2.1 describes a taxonomy for interoperability in the IoT environment. Section 2.2 compares resource IDs (i.e., the resource request format and the device ID) for the five selected IoT platforms. Section 2.3 describes and summarizes IoT interoperability studies.

2.1. Taxonomy for interoperability in the IoT environments

The IoT environment can be divided into diverse layers such as device, network, middleware, and application, as shown in Fig. 2. Each layer includes various elements. For example, the device layer includes heterogeneous sensors and actuators such as drones, surveillance cameras, and smart cars in different domains used in

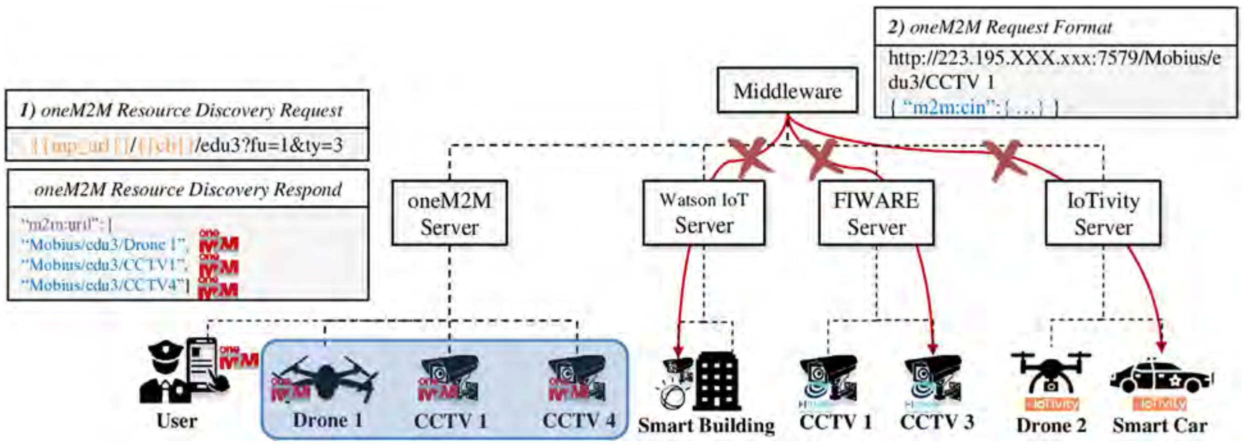


Fig. 1. Problems in resource discovery and resource request.

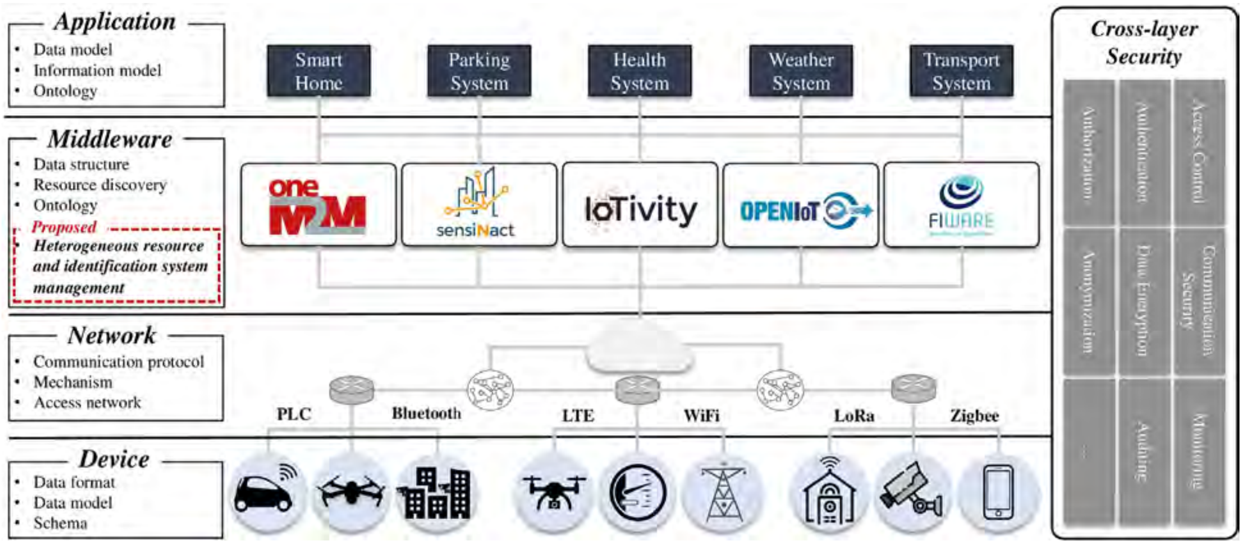


Fig. 2. Interoperability issues for multi-layers in IoT.

IoT environments. The network layer consists of the communication protocols used by heterogeneous devices (e.g., power line communication (PLC), Bluetooth, wireless fidelity (WiFi), long-term evolution (LTE), Zigbee, and long-range (LoRa)). The middleware layer includes various platforms that provide services and infrastructure in the IoT environment, and numerous platforms are currently being developed, such as oneM2M, sensiNact, IoTivity, OpenIoT, FIWARE, and WSO2. The application layer includes services and data of various domains provided through the IoT platform. IoT requires hyper-connectivity among different objects (e.g., software or hardware platforms, network, resource (service or device)) in the various layers.

Therefore, interoperability is a key factor in building a real IoT environment, especially sharing the resource among heterogeneous IoT platforms. The diversity of each layer has several issues and requirements to build interoperability in an IoT environment. In addition, these issues and requirements can be classified according to the interoperability types. In the international standard, "ISO/IEC 21823-1, Internet of things (IoT) – interoperability for IoT systems – part 1: framework" (ISO/IEC, 2019), IoT interoperability is divided into five types (transport, syntactic, semantic, behavioral, and policy). Similar to types of standards, some papers (Bröring et al., 2017; Bröring et al., 2018; Ganzha et al., 2017) include semantic, syntactic, middleware, and networks in the classification of interoperability in IoT systems, although there are some differences in configuration and expression. Based on it, we classify the interoperability issues and requirements in IoT systems as middleware, network, syntax, semantic, and security.

(1) Middleware interoperability refers to interoperability among IoT platforms and includes factors supported by the platform (i.e., data structures, resource discovery, heterogeneous resource, and ID management).
(2) Network interoperability includes network functions for each end device (i.e., communication protocols, network mechanisms, and access networks).
(3) Syntactic interoperability includes format, schema, and interface for data shared among IoT platforms.
(4) Semantic interoperability includes factors representing the same data in heterogeneous platforms, such as data model, information model, and ontology.
(5) Security factors commonly required for interoperability include communication security, data encryption, authentication, authorization, access control, identity management, monitoring, auditing, and anonymization.

2.2. Resource identifiers in the IoT platforms

As mentioned previously, the official number of IoT platform companies in the open market is more than 620. In this paper,

Table 1
Resource identifier of five IoT platforms.

Platform	Type	Device ID Format Resource-Request Format
oneM2M	OID based	[OID(Higher Arc)].[ManufacturerID].[DeviceTypeID].[DeviceSerialNo] (HTTP) [Server_IP_address]/[CSEBase_Name]/[cse_name]{n}/[ae_name]/[cnt_name]
GS1 Oliot	OID, GS1 ID Key	OID = GS1 OID(2.51).Identification Keys(1).[ID Key Type], GS1 ID Key = [GS1prefix].[CompanyNo].[ReferenceNo].[Serial/ExtensionNo] (HTTP) urn:epc:id:[ID Key Type]:[GS1 ID Key]
IBM Watson IoT	Client ID	d:[orgID]:[deviceType]:[deviceID] (HTTP) GET /device/types/[typeId]/devices/[deviceId]/state/[logicalInterfaceId](MQTT) iot-2/type/$[typeId]/id/$[deviceId]/intf/$[logicalInterfaceId]/evt/state
OCF IoTivity	Resource type, device id	[di], rt:oic.wk.d, oic.d.[*] (coap)://[IP_address]/[path]?[Query]
FIWARE	Entity type, Entity ID	No specific restriction except some characters (e.g., <, >, etc.) (HTTP) [ip address]: [port]/v2/entities/[id] or [type]

we selected the five IoT platforms (i.e., oneM2M, GS1 Oliot, IBM Watson IoT, OCF IoTivity, and FIWARE), which have important contributions to the IoT industrial environment. In this section, we describe the contributions of the five selected IoT platforms and compare the resource IDs. Table 1 depicts the device ID and resource request format of the five selected IoT platforms.

(1) oneM2M is an international Machine-to-Machine (M2M) standard organization that provides IoT architectures, platforms, application programming interfaces (API), and security solutions (oneM2M, 2021). oneM2M has a tree structure jointly developed by ITU-T and ISO/IEC and identifies devices using object identifiers (OID). The resource structure of oneM2M consists of an infrastructure node (IN), a middle node (MN), a common service entity (CSE), an application service node (ASN), and an application-dedicated node (ADN). In addition, when requesting service in oneM2M, it can have several CSEs attached depending on the structure. In these cases, the number of CSEs in the request formats may be different.
(2) GS1 is a private international organization for developing standards for barcode IDs and electronic catalogs of products used in all industries (e.g., distribution and logistics). GS1 is developing an IoT platform named Oliot (2021). GS1 Oliot uses an ID Key based on the OID system to identify various resources (e.g., devices and events). Oliot stores and manages all resources in event format through electronic product code information services (EPCIS).
(3) IBM is developing the Watson IoT platform to connect and analyze IoT data securely (Watson IoT, 2021). IBM Watson IoT uses a client ID to identify each device, and the format for HTTP requesting includes these IDs.
(4) OCF is an international organization for developing the IoTivity platform as an open-source framework and aims to provide easy and secure communication among IoT devices (IoTivity, 2021). OCF IoTivity identifies all resources using resource type (rt) and a device identifier (di). The IoTivity format for resource requests includes the ID is used as a component of [path].
(5) FIWARE is a framework of open-source platform components to accelerate intelligent solutions (e.g., smart cities, smart agrifood, smart energy, and smart industry) developed by the future internet public–private partnership (FI-PPP) within Europe (FIWARE, 2021). The FIWARE device ID is based on the ETSI NGSI-LD standard and identifies all entities using entity type and entity id. Three brokers (i.e., Orion-LD, Scorpio, and Stellio) are being developed in FIWARE, and Orion-LD is used in this paper. In addition, this paper analyzes the HTTP resource request format based on ETSI NGSI-LD.

The five analyzed IoT platforms use different structures of a device ID and request formats. Due to different request formats, requests among different platforms cannot be understood, and it is challenging to discover and request resources of different platforms. Therefore, it is necessary for a system that maps and converts different IDs.

2.3. Related work

Related work includes research proposing solutions for the types such as syntactic, semantic, and middleware interoperability. In addition, existing interoperability studies can be divided into two types (i.e., building an integrated platform and duplicated resources).

2.3.1. Integrated platform

The IoT European platforms initiative (IoT-EPI) (Bröring et al., 2018) was formed to increase innovation in IoT research in Europe. The primary purpose of the project is to develop an integrated platform that dynamically composes several technologies. The IoT-EPI project is considering a platform for providing IoT with various connected devices into the web to users in a smart environment. Major projects include AGILE, BIG IoT, INTER-IoT, VICINITY, SymbIoTe, bIoTope, and TagItSmart. Bröring et al. (2017) investigated IoT ecosystem interoperability through the BIG IoT project. They devised an IoT ecosystem architecture using common information models to bridge interoperability gaps. This architecture uses the open framework BIG IoT API and BIG IoT Marketplace to solve the interoperability issues among heterogeneous platforms, standards, and domains. Providers use APIs to implement registries and provide services and applications through marketplaces to consumers. Ganzha et al. (2017) introduced use case scenarios highlighting INTER-IoT projects and summarized the necessary steps to construct common semantics among various IoT platforms. The approach employed a modularized central ontology to make semantic interoperability among IoT platforms. The purpose was a semantic-driven translation for messages, and the core factor was the IoT platform semantic mediator (IPSM) to address message translation by applying ontology alignment. Zarko et al. (2017) presented symbIoTe, an interoperability approach from the H2020 project to create an interoperability framework providing semantic interoperability. The symbIoTe is a hierarchical stack-based architecture of application, cloud, and smart environments to achieve interoperability among heterogeneous resources. They described in detail the role of IoT platform federations and the symbIoTe architecture and defined four compliance levels for IoT platforms.

2.3.2. Duplicated resources

oneM2M has developed an interworking proxy entity (IPE) (oneM2M, 2019b) that focuses on interworking with different IoT systems. IPE converts resources from other systems to the oneM2M resource format and creates redundant resources on the oneM2M gateway. All devices connected to the oneM2M gateway provide resource discovery results at regular intervals. As an example of using duplicated resources, the oneM2M device can use the converted OCF IoTivity resource. In addition, oneM2M provides oneM2M base ontology (oneM2M, 2019a) to provide semantic and syntactic interoperability through the mapping ontology method with other platforms and organizations. Kang and Chung (2018) proposed the IoT framework based on the oneM2M standard and ontology to achieve interoperability among heterogeneous IoT platforms. When a new resource is connected to the IoT platform through a network, the IoT platform server registers the resource as a oneM2M platform and performs a health check to managing devices dynamically. Data generated by registered devices are integrated into the oneM2M gateway by discovering, and data events generated by middleware are reported. Tao et al. (2017, 2018) applied ontology representation in the public cloud to satisfy interoperability among heterogeneous platforms. They proposed a multilayer cloud architectural model. In addition, ontology has been used to address heterogeneous data representation and application. Wu et al. (2017) proposed the IPE-based integration architecture to achieve the interworking between oneM2M and IoTivity. This IPE is a middleware that supports the mapping of device management functions. They presented mapping IoTivity resources (i.e., di) to the container (i.e., AE) under the MN-CSE tree in oneM2M and evaluated the proposal by testing interworking cases. Carrez et al. (2017) described the FIESTA-IoT platform system architecture in detail. The main purpose is to federate various testbeds and offer researchers to address various semantically interoperable resources. The FIESTA-IoT platform is enabled to integrate the semantics, and various mechanisms are required semantics such as languages and ontologies. An et al. (2019) designed and implemented a novel IoT interworking architecture by taking the example of two standards (i.e., oneM2M and FIWARE) to provide a semantic integration framework in a smart city. They approached the transforming data format between oneM2M and FIWARE to address an interworking proxy that performs a static mapping of sensor data. Yang and Wei (2018) addressed the user-device interoperability framework to implement semantic interoperability. They approached the cross-context semantic document exchange consistently interpreted between heterogeneous devices and user applications. They also proposed a semantic extraction and interpretation algorithm to implement an automatic sharing message. Ahmed et al. (2018) proposed an IoT Hub based on modeling principles to offer syntactic, semantic, communication interoperability. It aims at system integration and data exchange between heterogeneous systems in a smart gas network. They presented the common data model that aggregates the transformed data from each data producer. In addition, they implemented software and described the approach of use cases. Antoniazzi and Viola (2019) designed the concept of dynamic ontology, including the patterns of interaction in heterogeneous devices. The dynamic ontology was used to describe the device semantics incorporated in the web of things (WoT) and enabled the dynamic interactions among devices. They presented an implementation of the WoT using the common ontology for describing devices. Tolcha et al. (2021) focused on interoperability between an event-based GS1 EPCIS and an entity-based FIWARE NGSI. They proposed the Oliot Mediation Gateway and showed its feasibility by applying it to an actual case study. When the Oliot Mediation Gateway receives NGSI entity data, it generates EPCIS events based on it. EPCIS stores the newly generated events and enables interoperability by allowing any application to access events via the EPCIS standardized query interface. Cavalieri (2021) proposed the OPCUA-IPE that enables Open Platform Communications Unified Architecture (OPC-UA), the main reference communication standard among industrial applications, to consume information generated by oneM2M. OPCUA-IPE maps and

Table 2
Interoperability classification and summary of IoT RNS and related studies.

Paper	Interoperability Types					Description
	Syntactic	Semantic	Network	Middleware	Security	
Bröring et al., 2018	✓	✓		✓		They devised an IoT ecosystem architecture that uses the BIG IoT API and BIG IoT Marketplace among heterogeneous platforms, standards, and domains. In addition, they register redeveloped resources based on a common API and share them with providers and consumers.
Ganzha et al., 2017		✓		✓		The approach was a semantic-driven translation for messages, and the core factor was the IPSM to address message translation by applying ontology alignment.
Zarko et al., 2017	✓	✓		✓		The symbIoTe is a hierarchical stack-based architecture of application, cloud, and smart environments to achieve interoperability among heterogeneous resources. They built a core information model (CIM) ontology in which various platforms participate for semantic and syntactic interoperability with heterogeneous information models, and CIM acts as an intermediary between applications and platforms.
Kang and Chung, 2018		✓		✓		They proposed the oneM2M ontology based IoT framework that the IoT platform server registers the new resource as a oneM2M platform and performs a check to managing devices dynamically.
Tao et al., 2017, 2018		✓		✓		They proposed a multilayer cloud architectural model and applied ontology representation in the public cloud.
Wu et al., 2017	✓			✓		They proposed the IPE-based integration architecture mapping IoTivity resource to the container under the MN-CSE tree in oneM2M.
Carrez et al., 2017		✓		✓		They described the FIESTA-IoT platform system architecture that federates various testbeds to integrate the semantics. This system architecture semantically transforms other information model testbeds and replicates them to the local database of FIESTA-IoT.
An et al., 2019	✓			✓		They designed a novel IoT interworking architecture transforming data format between oneM2M and FIWARE using a static mapping of data.
Yang and Wei, 2018		✓		✓		They designed the semantic extraction and interpretation algorithm to handle cross-context semantic document exchange consistently interpreted between heterogeneous devices and user applications.
Ahmed et al., 2018	✓	✓	✓	✓		They proposed an IoT Hub that aggregates the transformed data from each data producer based on modeling. IoT Hub supports communication protocols for OPCUA, Rest web service, and file sharing and centrally manages data by converting it into a common format for semantic/syntactic interoperability of data.
Antoniazzi and Viola, 2019		✓		✓		They presented an implementation of the WoT using the common ontology that enabled the dynamic interactions among devices for describing devices.
Tolcha et al., 2021	✓			✓		They proposed the Oliot Mediation Gateway that generates EPCIS events based on received NGSI entity data. EPCIS stores the newly generated events and users can access the events via the EPCIS standardized query interface.
Cavalieri, 2021	✓			✓		They proposed the OPCUA-IPE that maps and transforms the information model of OPC-UA and oneM2M to use the oneM2M resource by OPC-UA users.
Zyrianoff et al., 2021	✓			✓		They proposed an adapter that converts W3C WoT data into FIWARE NGSI format by subscribing to WoT Thing Description events and mapping it to a FIWARE Orion entity.
Proposed	✓			✓		IoT RNS converts heterogeneous platform resource addresses to each connected platform's format using a mapping table and connected platforms can request resources of heterogeneous platforms using converted addresses.

transforms the information model of OPC-UA and oneM2M based on IPE, an interoperability solution between oneM2M and non-oneM2M, so that OPC-UA users can use the converted resources of other standards through the server. Zyrianoff (2021) proposed an adapter that connects the Web of Things (WoT) architecture proposed by the W3C Consortium to the FIWARE. This adapter is a general mashup application that converts WoT data into NGSI format by subscribing to WoT Thing Description events and mapping it to a FIWARE Orion entity. If this entity does not exist in Orion, the adapter creates it, and Orion stores this entity information in the FIWARE database.

Table 2 shows a comparison of related work in IoT and represents the type of interoperability. Building an integrated platform requires an accurate ontology and a relationship with the central platform. In addition, resource duplication has memory and format conversion issues. In order to solve these issues, we propose an approach that converts resource paths of heterogeneous platforms and details are described in Section 3.

3. Proposed IoT RNS

This section describes the proposed IoT RNS architecture and scenario. Section 3.1 presents the assumption of the environment. Section 3.2 describes the IoT RNS functions and presents the shared mapping table components. Section 3.3 describes scenarios for sharing resources among heterogeneous IoT platforms in a smart city and details the process for each stage.

3.1. Assumption

To illustrate our IoT RNS, we take into consideration some prerequisites and assumptions as follows:

(1) In numerous IoT platforms, various devices are abstracted to different levels. For example, some devices may be core devices performing data collection, calculation, and processing. Some may be sensors that only measure specific data or devices that perform simple services. These low performance sensors have limitations for requesting and processing resources. Since there are many different types of devices in the IoT environment, it is difficult to generalize. Therefore, we assumed that service requests using the converted resource table are not sent directly from the client (i.e., the end device) but are sent through the server of a specific platform.

(2) Some IoT platforms (e.g., oneM2M and IoTivity) only provide resource discovery functions to users. In our scenario, the root IoT RNS must manage the resources connected to each IoT platform (e.g., whether the resources are connected). Therefore, we assumed that each platform could provide a resource discovery function. Furthermore, the local IoT RNS stores the resource information (e.g., service list) discovered in each platform in the resource table. In addition, the resource information stored in the local IoT RNS can be shared with the root IoT RNS periodically.

(3) This study defines a resource as the services used between devices based on various IoT platforms. Therefore, the proposed interoperability framework concentrates on converting resource paths (e.g., uniform resource identifier (URI) used in heterogeneous IoT platforms) in a specific IoT platform to the target IoT platform. However, in some service centric IoT platforms (e.g., oneM2M and FIWARE), a user should input data as a parameter to get a service. We do not consider it in this study.

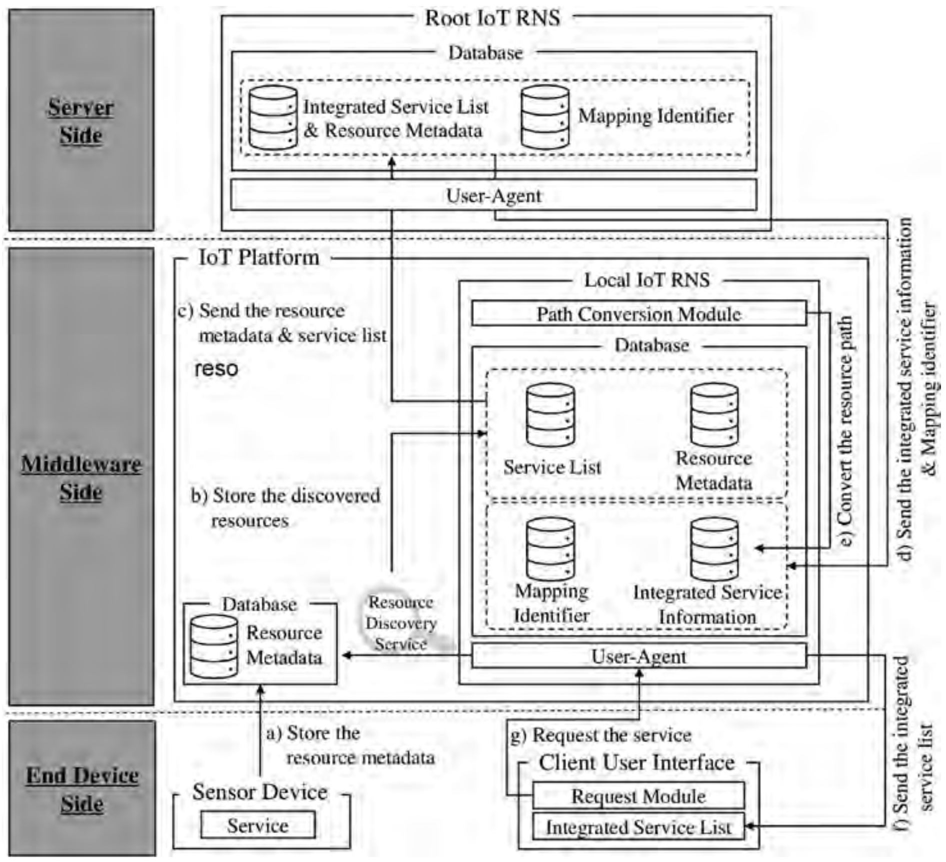

Server Side
Root IoT RNS
Database
Integrated Service List & Resource Metadata
Mapping Identifier
User-Agent
Middleware Side
IoT Platform
c) Send the resource metadata & service list
reso
Local IoT RNS
Path Conversion Module
Database
Service List
Resource Metadata
Mapping Identifier
Integrated Service Information
User-Agent
b) Store the discovered resources
Database
Resource Metadata
Resource Discovery Service
e) Convert the resource path
d) Send the integrated service information & Mapping identifier
End Device Side
a) Store the resource metadata
Sensor Device
Service
g) Request the service
Client User Interface
Request Module
Integrated Service List
f) Send the integrated service list

Fig. 3. Overview of IoT RNS – system structure and components.

(4) In general, the FIWARE platform uses the expression as an entity, including various entities such as devices, services, and data. Therefore, in our study, the scope of entities to be devices typically used in NGSI-LD, such as cars and buildings, to reduce confusion with other platforms. In addition, in the oneM2M platform, we use the hierarchical CSEBase relative representation although there are various representations for a specific resource.

3.2. Architecture and algorithms

Fig. 3 represents the system structure and components of the proposed IoT RNS. It is divided into three sides (i.e., server, middleware, and end device side).

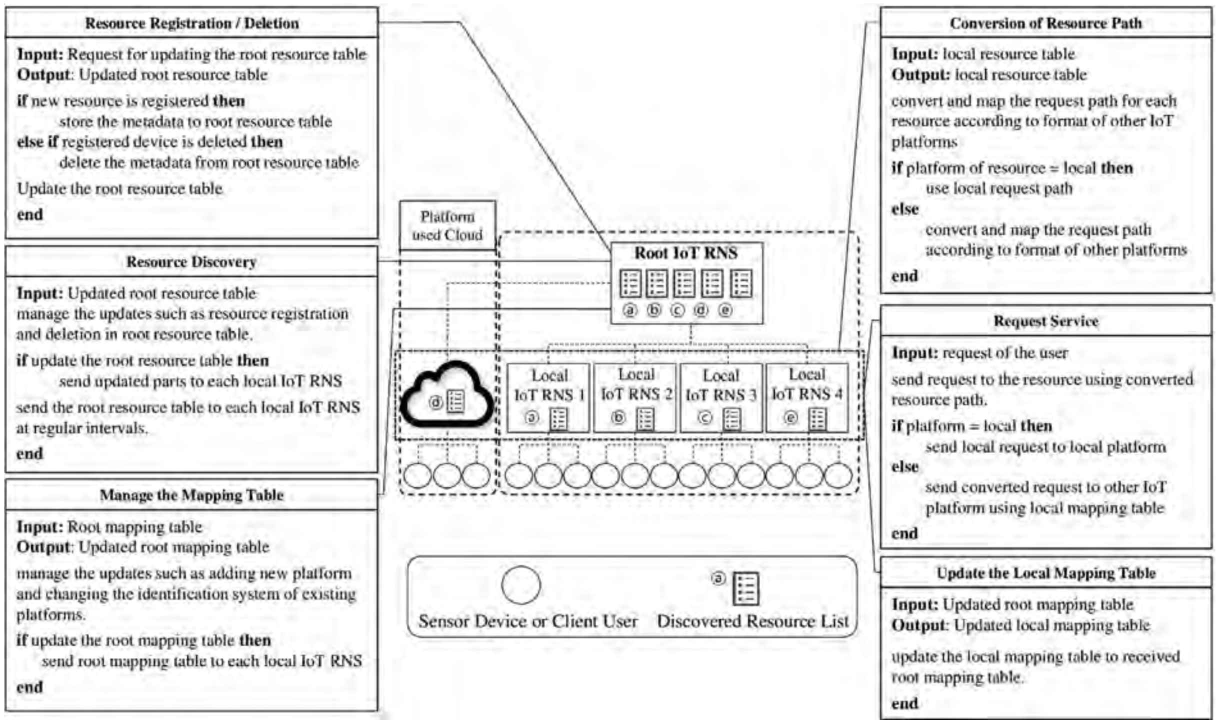
The server-side includes the root IoT RNS that manages the connected resources of various IoT platforms. The root IoT RNS has a database that manages the integrated service list, mapping information, and resource metadata. In our proposal, an integrated service list means the discovered resources (i.e., service name) from the diverse IoT platforms. The mapping information refers to a mapping table in which ID formats among heterogeneous IoT platforms are mapped. In addition, resource metadata contains resource information such as a resource name, platform type, device type, device ID, IP address, resource path, and service parameter. The service parameter is required data when a user requests a resource in a service-centric platform (e.g., oneM2M and FIWARE). The local IoT RNS is modularized in the IoT platform and has the service information (i.e., service list, service parameters, resource metadata, and integrated service list) and the path conversion module. The end device side includes the sensor devices that provide the service and the client user interface with the service list and the request module. The overall flow of IoT RNS is as follows:

(1) When an administrator registers a new sensor device, the resource metadata is stored in the IoT platform server.
(2) Resource metadata through resource discovery of the IoT platform server is stored periodically in local IoT RNS.
(3) The local IoT RNS transfers resource metadata to the root IoT RNS periodically.
(4) The root IoT RNS integrates the resource metadata received from all connected local IoT RNSs and sends them to each local IoT RNS.
(5) Each local IoT RNS converts the resource path of integrated service information.
(6) The local IoT RNS sends the integrated service list to the client user interface.
(7) A user requests a service, and the local IoT RNS uses the converted path to use the service on another platform.

Fig. 4 represents the overall function and algorithm of IoT RNS in the IoT environment. It represents IoT RNSs, including local IoT RNSs that modules on each server device and a root IoT RNS connected to the local IoT RNSs and managing the entire resource table.

The root IoT RNS manages the metadata for connected and disconnected resources. Each local IoT RNS sends related metadata (i.e., device type, device ID) to the root IoT RNS when new devices and services are registered and informs the root IoT RNS when devices and services are deleted or disconnected. The root IoT RNS updates the resource table with metadata received from each local IoT RNS and then transmits the integrated resource table to each local IoT RNS. Regardless of updates, the integrated resource table is sent to each local IoT RNS at regular intervals. Local IoT RNSs convert and store the resource path in the resource table received from the root RNS, depending on the format requested by each platform using the mapping table as shown in Table 3.

The mapping table includes the root and local mapping table. The root mapping table manages the local mapping tables in various IoT platforms. If a new IoT platform is added or the existing ID format changes, the root mapping table is updated and sent to the



```
Resource Registration / Deletion
Input: Request for updating the root resource table
Output: Updated root resource table
if new resource is registered then
    store the metadata to root resource table
else if registered device is deleted then
    delete the metadata from root resource table
Update the root resource table
end
```

```
Resource Discovery
Input: Updated root resource table
manage the updates such as resource registration
and deletion in root resource table.
if update the root resource table then
    send updated parts to each local IoT RNS
send the root resource table to each local IoT RNS
at regular intervals.
end
```

```
Manage the Mapping Table
Input: Root mapping table
Output: Updated root mapping table
manage the updates such as adding new platform
and changing the identification system of existing
platforms.
if update the root mapping table then
    send root mapping table to each local IoT RNS
end
```

```
Conversion of Resource Path
Input: local resource table
Output: local resource table
convert and map the request path for each
resource according to format of other IoT
platforms
if platform of resource = local then
    use local request path
else
    convert and map the request path
    according to format of other platforms
end
```

```
Request Service
Input: request of the user
send request to the resource using converted
resource path.
if platform = local then
    send local request to local platform
else
    send converted request to other IoT
    platform using local mapping table
end
```

```
Update the Local Mapping Table
Input: Updated root mapping table
Output: Updated local mapping table
update the local mapping table to received
root mapping table.
end
```

Fig. 4. Overall function and algorithm of proposed IoT RNS.

Table 3
Proposed mapping table example of IoT RNS.

Platform	Device Type	Device ID	IP	Path
oneM2M	Device Type	Device ID	IP	[Server_IP_address]/ [CSEBase_Name]/[cse_name]{n}/[ae_name]/[cnt_name]
GS1 Oliot	n/a	GS1 ID Key	IP	urn:epc:id:[ID Key Type]:[GS1 ID Key]
IBM Watson IoT	Type ID	Device ID	IP	/device/types/[typeId]/devices/[deviceId]
OCF IoTivity	Resource type (rt)	Device ID (di)	IP	[ip address]/[URL path]
FIWARE	n/a	Entity ID	IP	[ip address]:[port]/v2/entities/[id] or [type]

local IoT RNS. Then each local IoT RNS updates their local mapping table received from the root mapping table. The mapping table contains resource information for IoT platforms, including attributes such as platform type, device type, device ID, IP address, and resource path. The device ID and request format for each platform differs for each IoT platform and is used by the local IoT RNS to convert into the appropriate request format. Fig. 5 represents the metamodel of the IoT RNS architecture using the class diagram. The root IoT RNS has four functions:

(1) Store connected and registered resource metadata.
(2) Send updated root resource table components to the local IoT RNSs.
(3) Send the integrated root resource table to the local IoT RNSs at regular intervals.
(4) Send updated root mapping tables to the local IoT RNSs.

The local IoT RNS has five functions:

(1) Send newly registered resource metadata to the root IoT RNS.
(2) Inform the root IoT RNS of disconnected resource metadata.
(3) Send updated resource metadata to the root IoT RNS.
(4) Convert resource paths using the local mapping table.
(5) Update the local mapping table when a root mapping table is received.

When the resource is updated, the root IoT RNS only sends the updated part to the local IoT RNS. Thus, it can reduce traffic issues between the root and local IoT RNS. In addition, since the entire resource table is sent at regular intervals, it is possible to check all connected resources. The root resource table includes resource name, platform name, device ID, device type, IP address, resource path, and service parameter, and the local resource table includes resource name, platform name, device ID, device type, IP address, original path, converted path, and service parameter. Root and local mapping tables include platform name, device ID, device type, IP address, and resource path format. Each platform contains registered resource metadata. The root IoT RNS stores and manages registered resource metadata for each platform and employs default or arbitrary values for any missing parameter when mapping the metadata by data type. Fig. 6 represents how the local IoT RNS converts the resource path of a newly registered resource. The local IoT RNS converts and maps the resource path from the root IoT RNS into the request format of its corresponding platform, as follows:

(1) Input updated resource table from the root IoT RNS.
(2) Check if the resource is newly registered.
  - If the resource has disconnected: delete the related metadata.
  - If the resource is newly registered: check if it is a local platform.
    - If it is a local platform, use the original path.
    - If it is another platform, convert the path using the requested format in the mapping table and store it in the resource table.
(3) Update the resource table.

Based on the converted path stored on the platform server, the end device for each platform can request the service to the end device of any other platform.

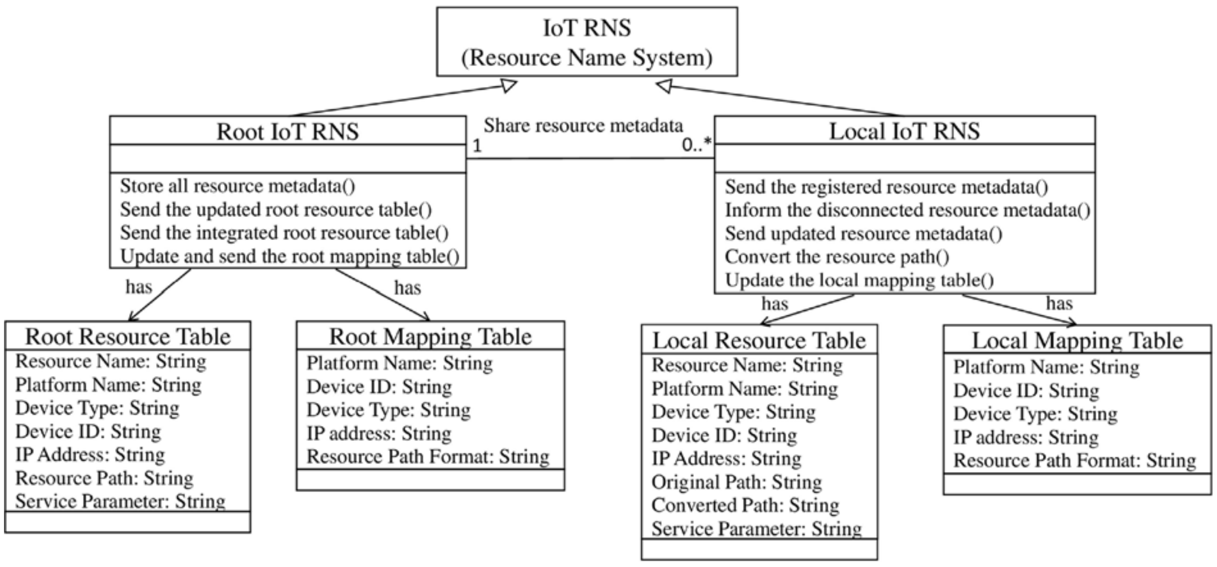


Fig. 5. Proposed IoT RNS metamodel.

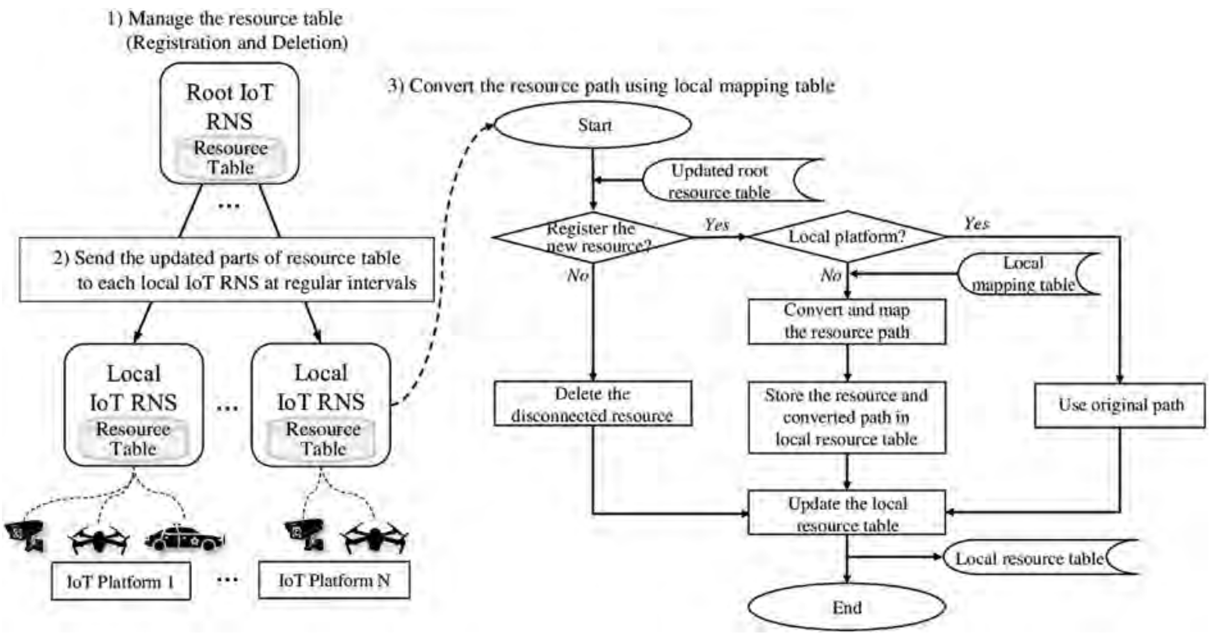

1) Manage the resource table (Registration and Deletion)
Root IoT RNS
Resource Table
2) Send the updated parts of resource table to each local IoT RNS at regular intervals
Local IoT RNS
Resource Table
IoT Platform 1
IoT Platform N
3) Convert the resource path using local mapping table
Start
Updated root resource table
Register the new resource?
Yes
No
Local platform?
Local mapping table
Convert and map the resource path
Delete the disconnected resource
Store the resource and converted path in local resource table
Use original path
Update the local resource table
Local resource table
End

Fig. 6. Flowchart for resource path conversion.

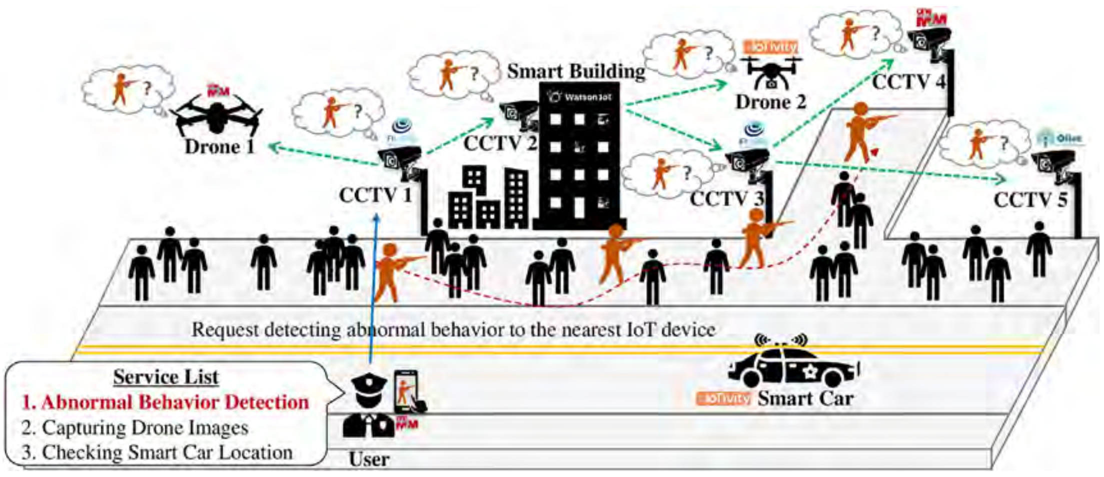

Smart Building
Drone 1
Drone 2
CCTV 1
CCTV 2
CCTV 3
CCTV 4
CCTV 5
Request detecting abnormal behavior to the nearest IoT device
Service List
1. Abnormal Behavior Detection
2. Capturing Drone Images
3. Checking Smart Car Location
User
Smart Car

Fig. 7. Proposed IoT RSN interoperability scenario in a smart city.

3.3. Scenario

Fig. 7 represents the considered smart city scenario in this research. A smart city is an urban area that provides the information to efficiently manage assets and resources using various electronic data collection sensors. Smart cities connect various resources within the city through a network and optimize city operations by introducing IoT, artificial intelligence, big data technologies. The current study represented an example scenario that requests resources among heterogeneous platform's devices in a smart city. This scenario is a limited smart city environment, which considered only five IoT platforms (i.e., oneM2M, GS1 Oliot, IBM Watson IoT, OCF IoTivity, and FIWARE). In this scenario, the root IoT RNS is assumed to be managed by a trusted organization such as a trusted third party (TTP), including the government. It has high performance and securely manages all platform's resources existing in the smart city. In addition, it is assumed that servers or gateways of all platforms are connected to the root IoT RNS and share the list of available resources in advance. The scenario supposes that a user connected to oneM2M sends a service request to the nearest device connected to FIWARE. In addition, it is assumed that Interworking between the user platform (i.e., oneM2M) and the nearest IoT device platform (i.e., FIWARE) is enabled. The overall flow of the scenario is as follows:

(1) After the user selects the service (i.e., Abnormal Behavior Detection), the request is sent to the nearest IoT device (i.e., CCTV 1).
(2) The request is shared with other IoT devices (i.e., smart buildings, CCTVs, and drones) without requiring human intervention.
(3) Every IoT device that detects abnormal activity sends the results to the user (i.e., locations and images).
(4) Results could also be sent to other smart objects (i.e., smart cars) if necessary.

We presented a scenario-based sequence diagram of an example that converts the resource path among heterogeneous IoT platforms, as shown in Fig. 8. It represents the sequence in which a new device is registered in the FIWARE platform, and the resource

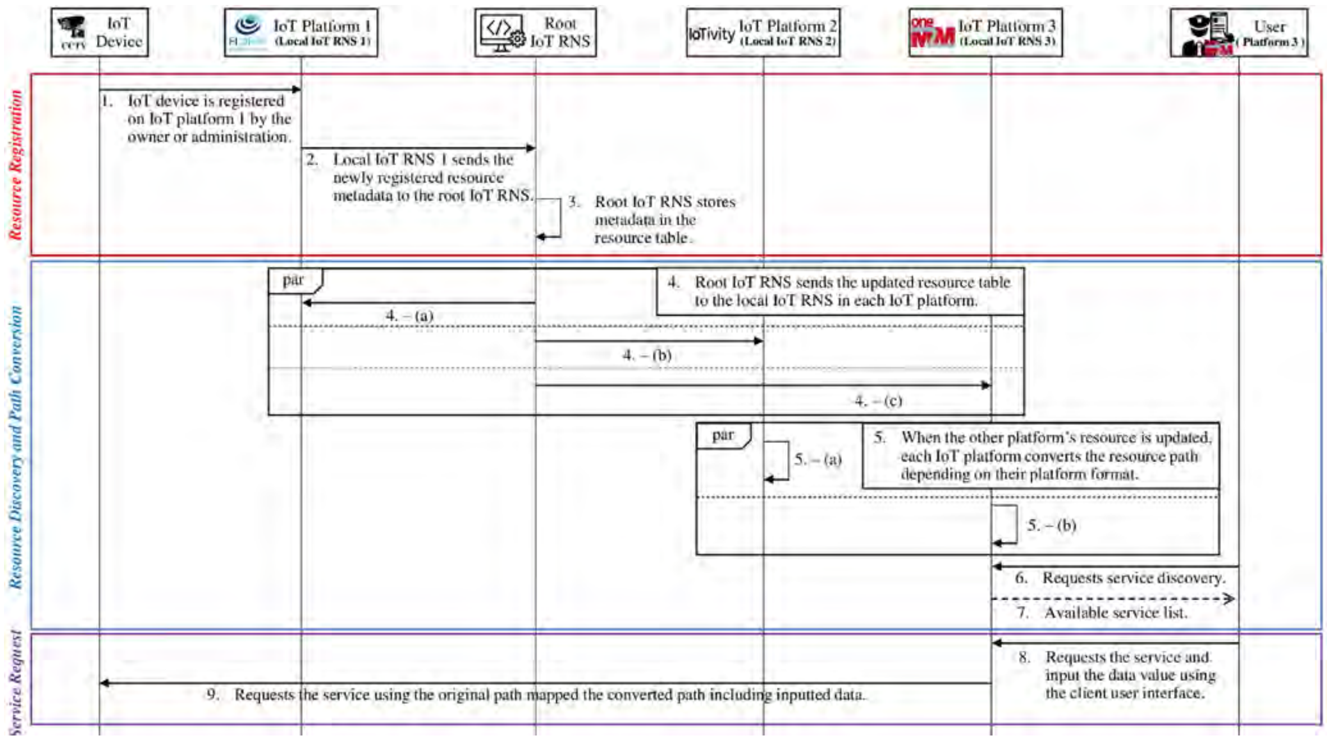

IoT Device
IoT Platform 1 (Local IoT RNS 1)
Root IoT RNS
IoT Platform 2 (Local IoT RNS 2)
IoT Platform 3 (Local IoT RNS 3)
User (Platform 3)
Resource Registration
1. IoT device is registered on IoT platform 1 by the owner or administration.
2. Local IoT RNS 1 sends the newly registered resource metadata to the root IoT RNS.
3. Root IoT RNS stores metadata in the resource table.
Resource Discovery and Path Conversion
par
4. Root IoT RNS sends the updated resource table to the local IoT RNS in each IoT platform.
4. – (a)
4. – (b)
4. – (c)
par
5. – (a)
5. When the other platform's resource is updated, each IoT platform converts the resource path depending on their platform format.
5. – (b)
6. Requests service discovery.
7. Available service list.
Service Request
8. Requests the service and input the data value using the client user interface.
9. Requests the service using the original path mapped the converted path including inputted data.

Fig. 8. Scenario-based sequence diagram that converts the resource path among heterogeneous IoT platforms.

path for this device is converted and used by each heterogeneous IoT platform.

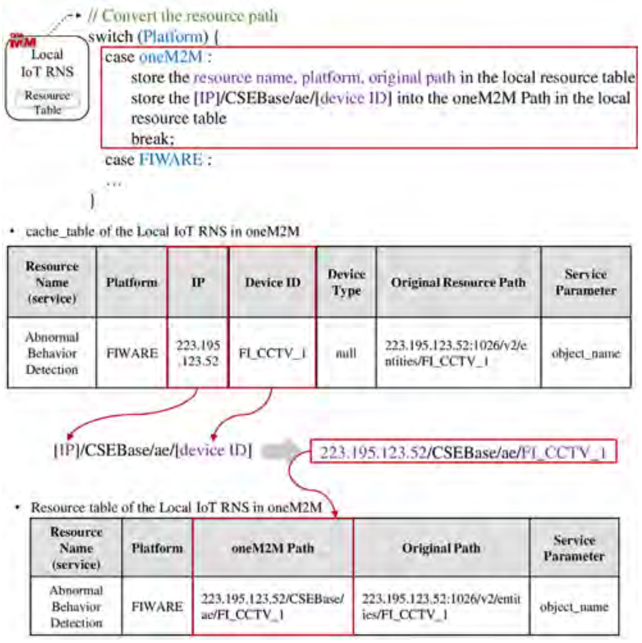
Firstly, a new IoT device (i.e., CCTV) is registered on the FIWARE platform by the owner or administrator. The local IoT RNS modularized in the FIWARE server sends the metadata of newly registered CCTV (e.g., device ID, IP, and platform) to root IoT RNS. Then, the root IoT RNS stores the metadata of the newly registered device in the resource table and sends the newly added part to the local IoT RNS modularized to each platform server. Since different platform's resources have been registered, local IoT RNSs modularized in IoTivity, and oneM2M server converts and stores the resource path according to the ID format used by its own platform. When a user with a oneM2M's device checks the available list using service discovery and requests another platform service, the user's device requests a service using the converted path. Then, the local IoT RNS modularized in the oneM2M server can request the service using the original path mapped to the request. The procedure for a user to use a newly registered resource on another platform is divided into three stages: resource registration and deletion, resource discovery and path conversion, and service request. Details of each stage are as follows.

3.3.1. Resource registration and deletion

The FIWARE server sends the newly registered resource metadata to the root IoT RNS, including resource name, platform type, device type, device ID, IP address, original resource path, and service parameter which is stored in the root IoT RNS's resource table. The root IoT RNS stored the metadata (i.e., platform: FIWARE, device type: CCTV, device ID: FI_CCTV_1, IP:223.195.123.52, original resource path, service parameter: object_name) of the service (i.e., Abnormal Behavior Detection) in the resource table.

3.3.2. Resource discovery and path conversion

When new resources are registered, each local IoT RNS receives the updated table from the root IoT RNS and checks the platform type. Local IoT RNS converts the resource paths depending on their platform format if the resource is registered from another platform, referring to the mapping table. In this scenario, in the root IoT RNS, the new resource (i.e., Abnormal Behavior Detection) has been registered, and the root resource table updated. And then, the root IoT RNS sends the updated resource table to the local IoT RNSs in all IoT platforms. Fig. 9 represents how the local IoT RNS in the oneM2M platform converts the resource path of the FIWARE platform:


Local IoT RNS
Resource Table
[IP]/CSEBase/ae/[device ID]
223.195.123.52/CSEBase/ae/FI_CCTV_1

```
// Convert the resource path
switch (Platform) {
    case oneM2M :
        store the resource name, platform, original path in the local resource table
        store the [IP]/CSEBase/ae/[device ID] into the oneM2M Path in the local
        resource table
        break;
    case FIWARE :
        ...
}
```

• cache_table of the Local IoT RNS in oneM2M

Resource Name (service)	Platform	IP	Device ID	Device Type	Original Resource Path	Service Parameter
Abnormal Behavior Detection	FIWARE	223.195.123.52	FI_CCTV_1	null	223.195.123.52:1026/v2/entities/FI_CCTV_1	object_name

• Resource table of the Local IoT RNS in oneM2M

Resource Name (service)	Platform	oneM2M Path	Original Path	Service Parameter
Abnormal Behavior Detection	FIWARE	223.195.123.52/CSEBase/ae/FI_CCTV_1	223.195.123.52:1026/v2/entities/FI_CCTV_1	object_name

Fig. 9. Example of the proposed path conversion in IoT RNS.

(1) Create a new path by inputting the IP address and entity ID into the oneM2M platform's request format.
(2) Store the oneM2M path of the new resource in the oneM2M resource table.
(3) When the conversion of the resource path is completed, the user can receive a list of available services newly.

3.3.3. Service request

As mentioned previously, additional data input is required when the user requests a service in the FIWARE platform. In our scenario, when the oneM2M user uses a FIWARE platform's service (i.e., converted 'Abnormal Behavior Detection' service), the oneM2M user should input the service parameter and send it to the oneM2M server. Then, the oneM2M server deals with the oneM2M user's request using the original resource path, including inputted data (e.g., object_name: gun).

4. Proof of concept

This section designs and implements a feasible scenario to represent the proof of concept for the proposed IoT RNS. The scenario uses oneM2M and FIWARE, which are open source IoT platforms, and includes detailed examples and sequences for using other platform resources.

4.1. Feasible scenario

Fig. 10 presents an overview of ID interoperability scenarios between IoT platforms based on different standards (i.e., oneM2M and FIWARE). This scenario does not include the resource registering process because it focuses on the path conversion and service request for the resource in oneM2M and FIWARE. The scenario can be divided into three phases (i.e., resource discovery, path conversion, and service request). Sequences 1 to 3 are resource discovery phases that share the metadata of resources registered to all platforms through the root IoT RNS. Sequence 4 and its subsequence are the phases in which the IoT RNSs of oneM2M and FIWARE convert the resource paths of each other. Sequences 5 and 6 show that when a user requests a service of another platform using the request module of the Client User Interface, the local IoT RNS requests the service using the original path mapped for the converted request.

4.2. Implementation

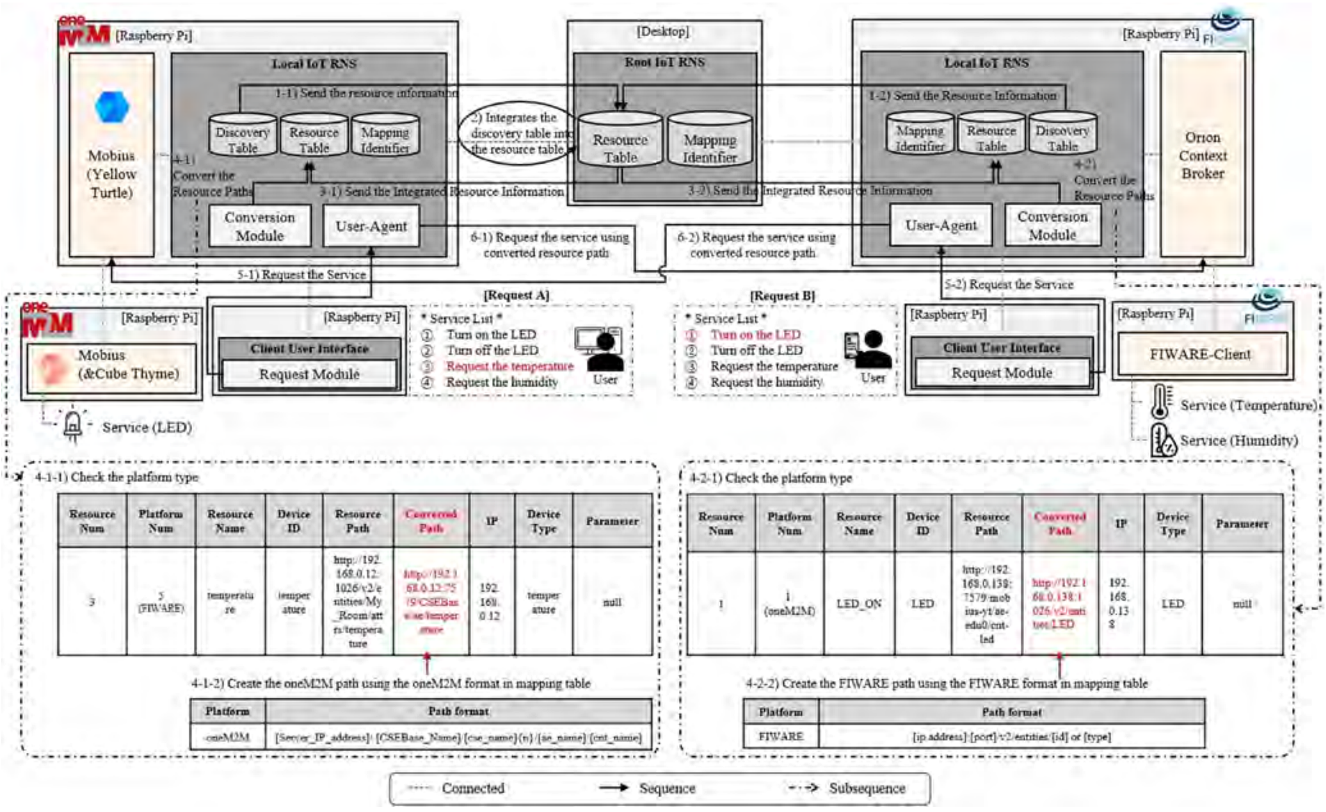
Implementing the feasible interoperability scenario illustrated in Fig. 10 requires the oneM2M and FIWARE platforms (i.e., server, client, and sensor), root IoT RNS, and local IoT RNS in oneM2M and FIWARE. This section describes the implementation environments and details.

4.2.1. Environment

All modules use Python code, oneM2M uses JavaScript, and FIWARE uses C++. The modules used are a Python request module, which sends the requests in Python, and a MySQL connector module named PyMySQL, which connects MySQL and Python. In addition, all modules use Raspberry Pi OS, except the FIWARE server, which uses Ubuntu on a Raspberry Pi, as shown in Table 4.

In the oneM2M environment, Mobius and &Cube were used, and a service to control the LED was configured. Fig. 11 shows the resources of the oneM2M server and sensor in a tree structure.

The resource structure has a container called cnt-led under CSE-Base, called mobius-yt. In addition, services can be requested when creating content instances called led_on and led_off under this container. The URL path to control this LED is https://192.168.0.



4-1-1) Check the platform type

Resource Num	Platform Num	Resource Name	Device ID	Resource Path	Converted Path	IP	Device Type	Parameter
3	5 (FIWARE)	temperature	temperature	http://192.168.0.12:1026/v2/entities/My_Room/attrs/temperature	http://192.168.0.12:7579/CSEBase/ae/temperature	192.168.0.12	temperature	null

4-1-2) Create the oneM2M path using the oneM2M format in mapping table

Platform	Path format
oneM2M	[Server_IP_address]:[CSEBase_Name]/[cse_name](n)/[ae_name]/[cnt_name]

4-2-1) Check the platform type

Resource Num	Platform Num	Resource Name	Device ID	Resource Path	Converted Path	IP	Device Type	Parameter
1	1 (oneM2M)	LED_ON	LED	http://192.168.0.138:7579/mobius-yt/ae-edu0/cnt-led	http://192.168.0.138:1026/v2/entities/LED	192.168.0.138	LED	null

4-2-2) Create the FIWARE path using the FIWARE format in mapping table

Platform	Path format
FIWARE	[ip address]:[port]/v2/entities/[id] or [type]

Fig. 10. Overview of a feasible scenario for resource ID interoperability between oneM2M and FIWARE.

Table 4
Specifications for each module in the implementation environment.

List		Root IoT RNS	oneM2M Client	oneM2M Sensor	oneM2M Server	FIWARE Client	FIWARE Sensor	FIWARE Server
Source Language		Python	Python JavaScript	Python JavaScript	Python JavaScript	Python C++	Python C++	Python C++
Module & Tool				Python request module: Python requests version: 2.23.0 MySQL connector module: PyMySQL version: 0.9.3 &Cube	Mobius		Orion_client	Orion Context Broker
Device	Name	Desktop		Raspberry Pi 3 Model B+				
	CPU	1.4 GHz ARM Cortex-A53 MP4			1.4 GHz ARM Cortex-A53 MP4			
	OS	Windows 10		Raspberry Pi OS Stretch with desktop version: February 5th 2020			Ubuntu MATE 18.04	
Sensor Device				LED			DHT11 (Temperature &Humidity)	

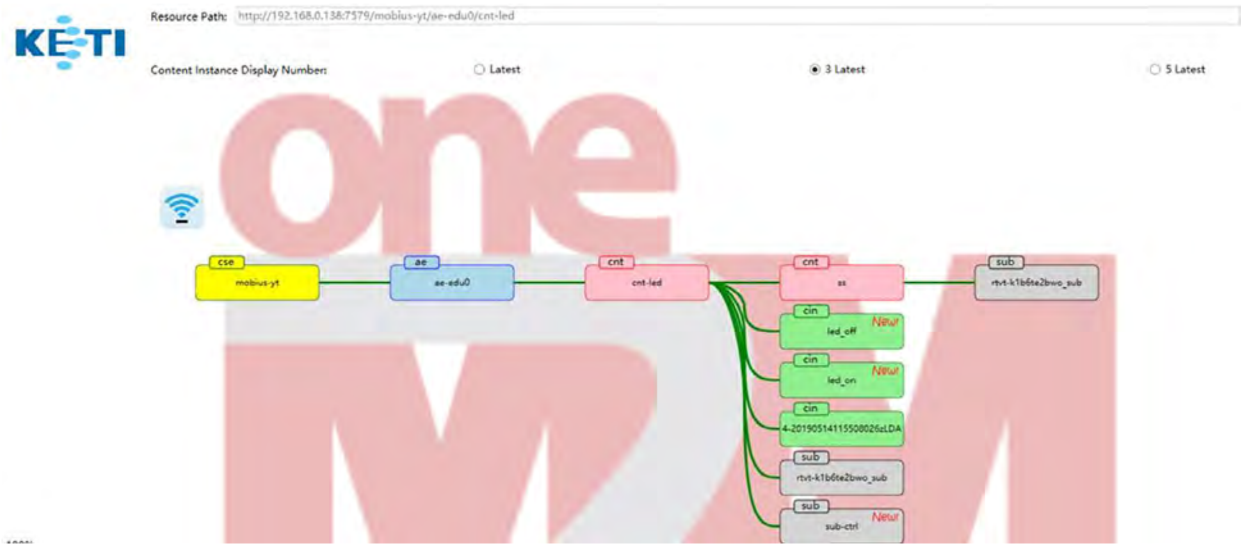
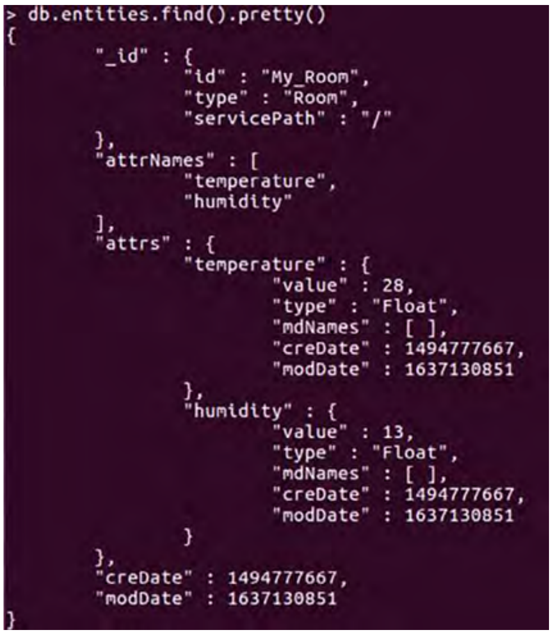


Fig. 11. oneM2M resource structure for LED control.

138:7579/mobius-yt/ae-edu0/cnt-led. If the oneM2M client requests the URL including the BODY statement, a content instance that controls the LED is created.

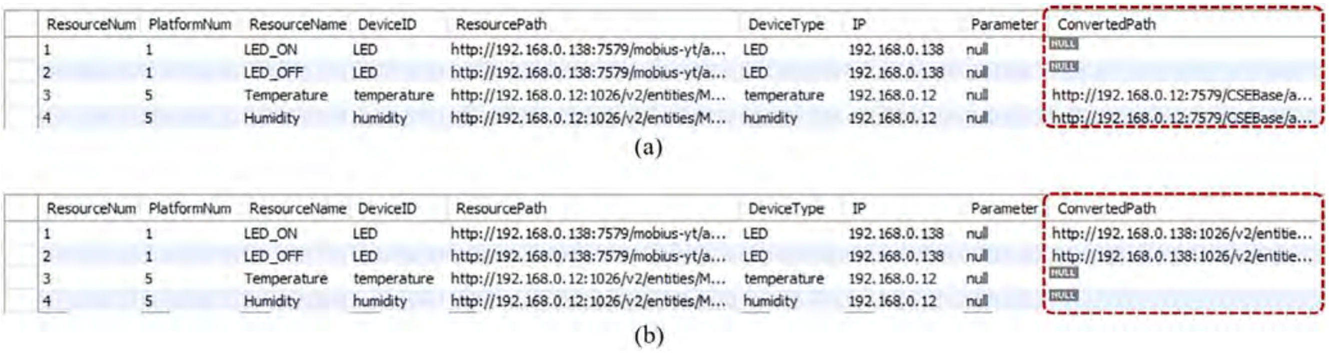
To build the FIWARE environment, we use the Orion Context Broker, which the user can query to register and manage the context elements. In addition, an Orion_client with a dht11 sensor is used to measure temperature and humidity, and the client device is used for the user request. We create an entity with an id status of My_Room, containing temperature and humidity attribute values, in the Orion Context Broker database. The temperature and humidity values are measured regularly and updated in the Orion Context Broker database, as shown in Fig. 12.

In this implementation, we modularize the root IoT RNS to manage various resources on the desktop, which perform better than a Raspberry Pi. The root IoT RNS integrates and stores the resource information received from oneM2M and FIWARE, including the resource number, platform number (i.e., platform type), resource name, device ID, resource path, device type, IP, and parameter. The integrated resource list is sent to and stored in the local IoT RNS in oneM2M and FIWARE at regular intervals. The local IoT RNS has a resource table that stores the list of integrated resources received from the root IoT RNS, and the resource table of the local IoT RNS has an additional property where the converted paths are stored, as shown in Fig. 13.



```
> db.entities.find().pretty()
{
        "_id" : {
                "id" : "My_Room",
                "type" : "Room",
                "servicePath" : "/"
        },
        "attrNames" : [
                "temperature",
                "humidity"
        ],
        "attrs" : {
                "temperature" : {
                        "value" : 28,
                        "type" : "Float",
                        "mdNames" : [ ],
                        "creDate" : 1494777667,
                        "modDate" : 1637130851
                },
                "humidity" : {
                        "value" : 13,
                        "type" : "Float",
                        "mdNames" : [ ],
                        "creDate" : 1494777667,
                        "modDate" : 1637130851
                }
        },
        "creDate" : 1494777667,
        "modDate" : 1637130851
}
```

Fig. 12. FIWARE resource structure for measuring the temperature and humidity.



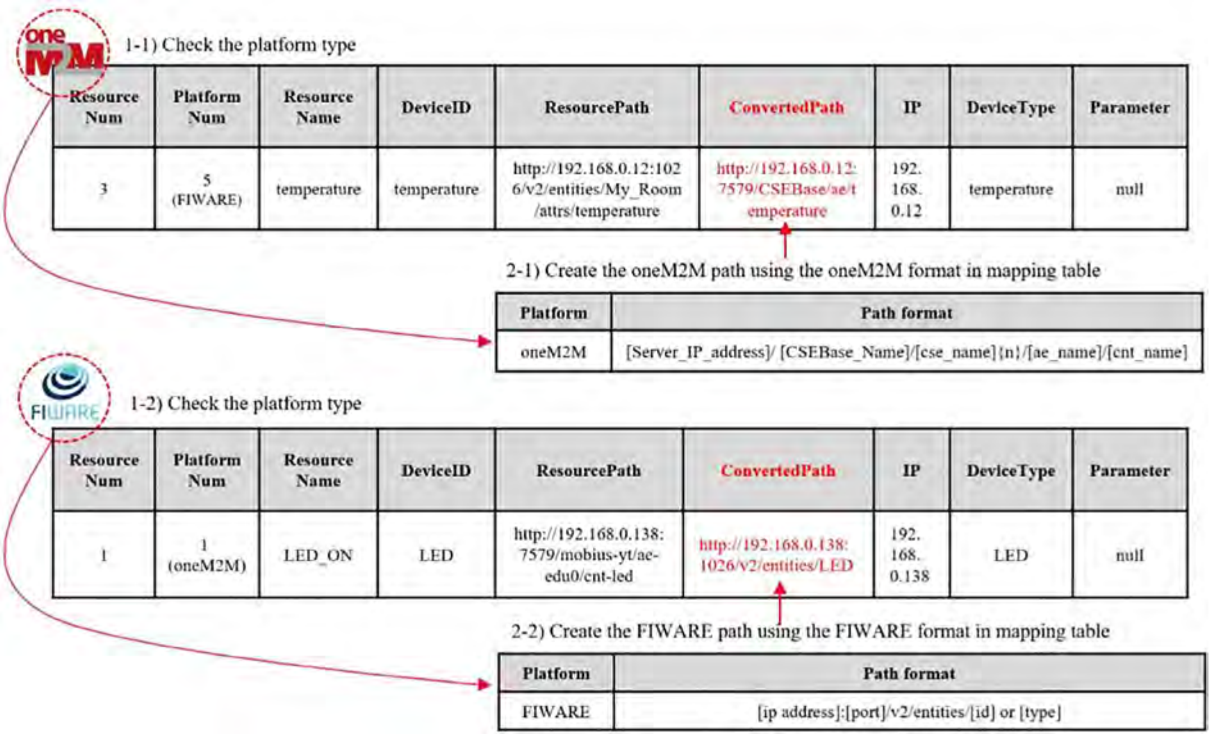
	ResourceNum	PlatformNum	ResourceName	DeviceID	ResourcePath	DeviceType	IP	Parameter	ConvertedPath
	1	1	LED_ON	LED	http://192.168.0.138:7579/mobius-yt/a...	LED	192.168.0.138	null	NULL
	2	1	LED_OFF	LED	http://192.168.0.138:7579/mobius-yt/a...	LED	192.168.0.138	null	NULL
	3	5	Temperature	temperature	http://192.168.0.12:1026/v2/entities/M...	temperature	192.168.0.12	null	http://192.168.0.12:7579/CSEBase/a...
	4	5	Humidity	humidity	http://192.168.0.12:1026/v2/entities/M...	humidity	192.168.0.12	null	http://192.168.0.12:7579/CSEBase/a...

(a)

	ResourceNum	PlatformNum	ResourceName	DeviceID	ResourcePath	DeviceType	IP	Parameter	ConvertedPath
	1	1	LED_ON	LED	http://192.168.0.138:7579/mobius-yt/a...	LED	192.168.0.138	null	http://192.168.0.138:1026/v2/entitie...
	2	1	LED_OFF	LED	http://192.168.0.138:7579/mobius-yt/a...	LED	192.168.0.138	null	http://192.168.0.138:1026/v2/entitie...
	3	5	Temperature	temperature	http://192.168.0.12:1026/v2/entities/M...	temperature	192.168.0.12	null	NULL
	4	5	Humidity	humidity	http://192.168.0.12:1026/v2/entities/M...	humidity	192.168.0.12	null	NULL

(b)

Fig. 13. Resource table in local IoT RNSs. (a) is the resource table in the oneM2M IoT RNS. (b) is the resource table in the FIWARE IoT RNS.



1-1) Check the platform type

Resource Num	Platform Num	Resource Name	DeviceID	ResourcePath	ConvertedPath	IP	DeviceType	Parameter
3	5 (FIWARE)	temperature	temperature	http://192.168.0.12:1026/v2/entities/My_Room/attrs/temperature	http://192.168.0.12:7579/CSEBase/ae/temperature	192.168.0.12	temperature	null

2-1) Create the oneM2M path using the oneM2M format in mapping table

Platform	Path format
oneM2M	[Server_IP_address]/ [CSEBase_Name]/[cse_name]{n}/[ae_name]/[cnt_name]

1-2) Check the platform type

Resource Num	Platform Num	Resource Name	DeviceID	ResourcePath	ConvertedPath	IP	DeviceType	Parameter
1	1 (oneM2M)	LED_ON	LED	http://192.168.0.138:7579/mobius-yt/ae-edu0/cnt-led	http://192.168.0.138:1026/v2/entities/LED	192.168.0.138	LED	null

2-2) Create the FIWARE path using the FIWARE format in mapping table

Platform	Path format
FIWARE	[ip address]:[port]/v2/entities/[id] or [type]

Fig. 14. Example of path conversion in local IoT RNSs.

Fig. 14 shows that if the platform type is oneM2M, a new request statement is created using the oneM2M format and stored as ConvertedPath in the FIWARE IoT RNS. In contrast, if the platform type is FIWARE, a new request statement is created using the FIWARE format and stored as ConvertedPath in the oneM2M IoT RNS. The path conversion module of the local IoT RNS identifies the platform type by checking the platform number in the integrated resource list and converts it into the form of its own resource path type.

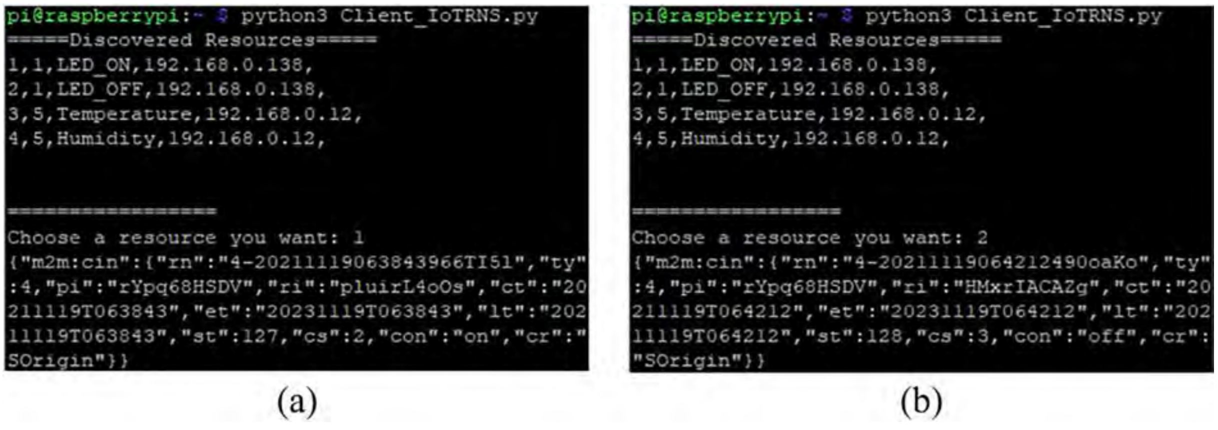
4.2.2. Demonstration

As described previously, if the original resource path is converted in the local IoT RNS of each IoT platform, users can employ services based on other IoT platforms using the converted path. This section demonstrates using the oneM2M service at the request of the FIWARE client and using the FIWARE service at the request of the oneM2M client. As depicted in Fig. 15, when the FIWARE Client User Interface is executed and connected to the User-Agent of the local IoT RNS, the FIWARE user can check the list of available services.

The first number in the service list is the resource number, and a request can be made by inputting that number. The second number is the platform number used to identify each platform type. In this example, the number 1 means the service of the oneM2M platform, and the number 5 means the service of the FIWARE platform. To test the request of the oneM2M service from the FIWARE client, service number 1 and service number 2 (i.e., platform number 1, LED _ON and LED_OFF) are input.

When the user inputs the resource number, the request module sends a request using the converted path, and the User-Agent of the local IoT RNS requests the oneM2M service using the original path mapped to the corresponding converted path. If the request through the mapped original path is successful, a new content instance is created in the existing oneM2M resource structure, as shown in Fig. 16.

A FIWARE service request from the oneM2M client is similar to that described above. Through the FIWARE client, the user



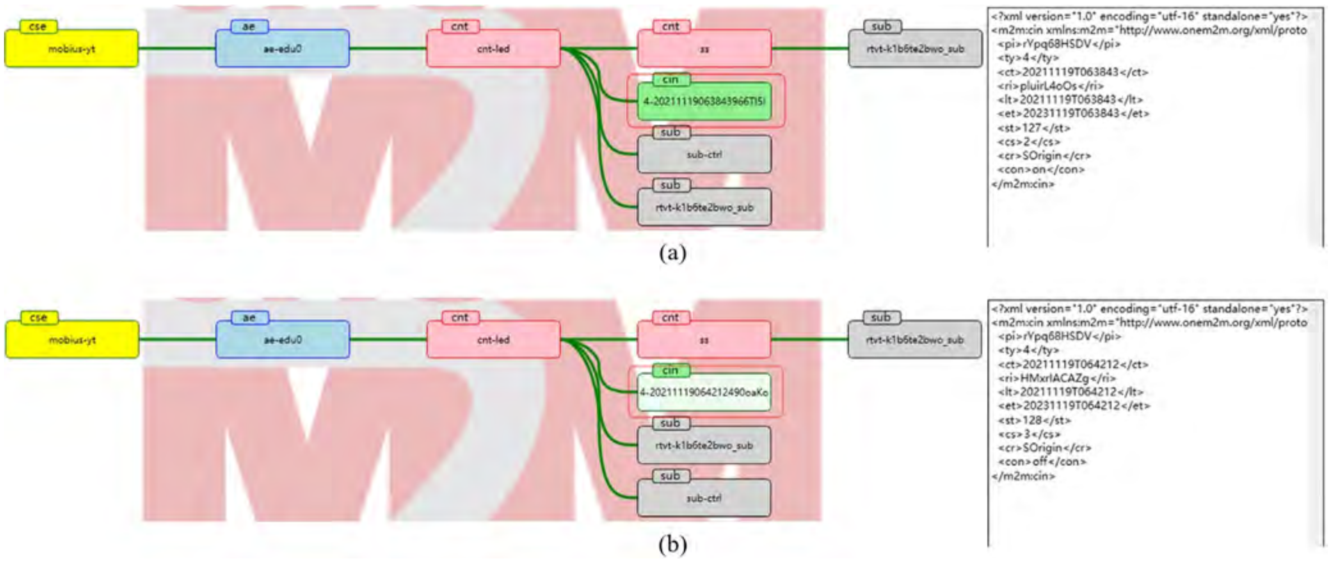
(a)

```
pi@raspberrypi:~ $ python3 Client_IoTRNS.py
=====Discovered Resources=====
1,1,LED_ON,192.168.0.138,
2,1,LED_OFF,192.168.0.138,
3,5,Temperature,192.168.0.12,
4,5,Humidity,192.168.0.12,


==================
Choose a resource you want: 1
{"m2m:cin":{"rn":"4-20211119063843966TI5l","ty"
:4,"pi":"rYpq68HSDV","ri":"pluirL4oOs","ct":"20
211119T063843","et":"20231119T063843","lt":"202
11119T063843","st":127,"cs":2,"con":"on","cr":"
SOrigin"}}
```

(b)

```
pi@raspberrypi:~ $ python3 Client_IoTRNS.py
=====Discovered Resources=====
1,1,LED_ON,192.168.0.138,
2,1,LED_OFF,192.168.0.138,
3,5,Temperature,192.168.0.12,
4,5,Humidity,192.168.0.12,


==================
Choose a resource you want: 2
{"m2m:cin":{"rn":"4-20211119064212490oaKo","ty"
:4,"pi":"rYpq68HSDV","ri":"HMxrIACAZg","ct":"20
211119T064212","et":"20231119T064212","lt":"202
11119T064212","st":128,"cs":3,"con":"off","cr":
"SOrigin"}}
```

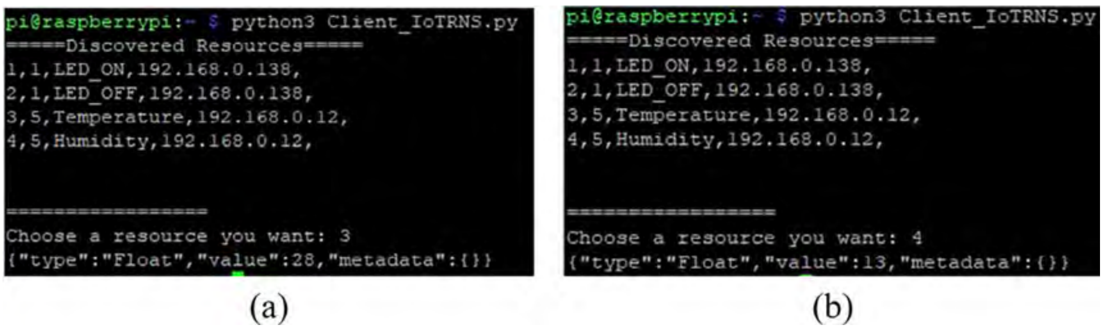
Fig. 15. The request of oneM2M service from FIWARE client. (a) is the oneM2M request for LED_ON, and (b) is the oneM2M request for LED_OFF.



```
<?xml version="1.0" encoding="utf-16" standalone="yes"?>
<m2m:cin xmlns:m2m="http://www.onem2m.org/xml/proto
  <pi>rYpq68HSDV</pi>
  <ty>4</ty>
  <ct>20211119T063843</ct>
  <ri>pluirL4oOs</ri>
  <lt>20211119T063843</lt>
  <et>20231119T063843</et>
  <st>127</st>
  <cs>2</cs>
  <cr>SOrigin</cr>
  <con>on</con>
</m2m:cin>
```

```
<?xml version="1.0" encoding="utf-16" standalone="yes"?>
<m2m:cin xmlns:m2m="http://www.onem2m.org/xml/proto
  <pi>rYpq68HSDV</pi>
  <ty>4</ty>
  <ct>20211119T064212</ct>
  <ri>HMxrIACAZg</ri>
  <lt>20211119T064212</lt>
  <et>20231119T064212</et>
  <st>128</st>
  <cs>3</cs>
  <cr>SOrigin</cr>
  <con>off</con>
</m2m:cin>
```

Fig. 16. Example of the successful creation of a content instance in the oneM2M resource structure. (a) is the result of the request for LED_ON, and (b) is the result of the request for LED_OFF.



(a)

```
pi@raspberrypi:~ $ python3 Client_IoTRNS.py
=====Discovered Resources=====
1,1,LED_ON,192.168.0.138,
2,1,LED_OFF,192.168.0.138,
3,5,Temperature,192.168.0.12,
4,5,Humidity,192.168.0.12,


==================
Choose a resource you want: 3
{"type":"Float","value":28,"metadata":{}}
```

(b)

```
pi@raspberrypi:~ $ python3 Client_IoTRNS.py
=====Discovered Resources=====
1,1,LED_ON,192.168.0.138,
2,1,LED_OFF,192.168.0.138,
3,5,Temperature,192.168.0.12,
4,5,Humidity,192.168.0.12,


==================
Choose a resource you want: 4
{"type":"Float","value":13,"metadata":{}}
```

Fig. 17. Example of a successful request for the FIWARE service. (a) is the result of a temperature request, and (b) is the result of a humidity request.

inputs service numbers 3 and 4 (i.e., temperature and humidity measurements), as shown in Fig. 17. The converted path is used through the request module of the oneM2M client, and the local IoT RNS uses the original path mapped to the converted path to request a service. If the service request on the FIWARE server-side is successful, request-statement can access the database, inquire about the measured value, and respond to the oneM2M client.

5. Evaluation and discussion

Section 5 compares our proposal with other projects and related studies through qualitative evaluation. In addition, we describe IoT RNS limitations and necessary factors for future study.

5.1. Qualitative evaluation

Existing research can be divided into two approaches regarding resource ID interoperability: integrated platform and duplicated resources, and the detailed features and limitations are described as follows.

5.1.1. Integrated platform

An integrated platform is a common central platform that requires creating resources newly with an API or constructing ontology for each platform based on a core ontology. The method requires a platform to participate in the integrated platform or construct a relationship with the central ontology, which requires newly creating resources with an API or constructing ontology for each platform based on a core ontology. However, existing methods require the construction of a precise ontology and a relationship with the central platform. BIG IoT provides integrated platform functions through BIG IoT Marketplace and BIG IoT API. However, in order to use resources of heterogeneous platforms, resources created through the BIG IoT API and registered in the BIG IoT Marketplace can be only shared. INTER-IoT uses global ontology for IoT platform (GOIoTP), a modular central ontology. Each platform constitutes an IoT platform federation, constructing an ontology with matching, merging, and alignment processes using IPSM with the central ontology. However, each platform's ontology should be constructed to participate in the IoT platform federation, and it is difficult to define the relationship between the central ontology and each platform's ontology. SymbIoTe uses an integrated ontology approach. Each platform constructs an ontology to share resources based on the ontology for the designated CIM platform. However, it is difficult to adapt CIM ontology construction rules for the other platforms accurately.

5.1.2. Duplicated resources

Duplicating resources is a conversion method and storing heterogeneous platforms' resources into their own platform format. Duplicating and storing resources raise memory capability issues when connecting many resources and require format conversion before storing the duplicated resources. oneM2M uses an IPE to convert other platform resources into oneM2M format and then stores that. However, it is inefficient to duplicate and use all resources when diverse platforms and resources are connected. In addition, it is difficult to map resources from other platforms onto the oneM2M resource format and structure for resource duplication. FIESTA-IoT constructs and uses its own ontology. The nonFIESTA-IoT platform resources are duplicated in the FIESTA-IoT ontology and stored. However, similar to oneM2M IPE, memory limitations become essential when there are many platforms and resources. In addition, it is impossible to use and share the resources for platform types that do not accurately map onto the FIESTA-IoT ontology.

5.1.3. Path conversion in the IoT RNS

In contrast, the proposed IoT RNS is similar to the role of the existing integration platform and central ontology, but it is not necessary to construct the ontology of each platform. The IoT RNS approach focused on path conversion, such as URI of services provided by each IoT platform. The IoT RNS is divided into root and local to separate roles. The root IoT RNS stores resources of all platforms and sends updates related resources to the local IoT RNS. Metadata for the available resources of all connected platforms be stored, but the local IoT RNS plays a role in resource path conversion, reducing the centralization of the root IoT RNS. In addition, when the local IoT RNS converts the path, it uses a switch-case statement to reduce the processing speed. Thus, communication traffic and memory issues can be solved by separating the functions of root and local IoT RNS without duplicating resources inside the platform. In other words, IoT RNS has fewer memory capability issues than duplicating all resources. Table 5 represents the comparison of existing research and projects with the proposed IoT RNS. In addition, Table 6 represents the qualitative evaluation of existing research and proposed IoT RNS for the identified issues.

5.2. Discussion

The proposed IoT RNS approach has several issues to be considered and studied in the future.

5.2.1. Evaluation method and implementation

Currently, interoperability technologies among heterogeneous IoT platforms are being developed, and there is no clear solution. In addition, it is limited to implement such an actual environment, including numerous resources. There are few studies that test semantic, syntactic, and middleware interoperability in a smart city environment where numerous real resources are connected, and quantitative comparative evaluation is limited. Therefore, this study focused on the proposal of the framework to provide interoperability among heterogeneous IoT platforms. To prove the aims of the proposed approach, we implemented the resource interoperability scenario between oneM2M and FIWARE. Furthermore, we performed a qualitative evaluation of our proposal with the current studies and did not provide a quantitative evaluation of its implementation.

5.2.2. Scope of scenario

Related to the scenario of IoT RNS, there are some limitations as follows:

(1) In our scenario, each IoT platform provides a resource discovery function, and the discovered resource information is updated in the resource table of the local IoT RNS. In addition, the updated resource table is periodically shared with the root IoT RNS, and the root IoT RNS sends the integrated resource table to each local IoT RNS. In order for a user to check a list of all connected services, a client user interface that can communicate with the local IoT RNS is required. The client user interface should be modularized on the user terminal and can support resource request and resource discovery functions.

(2) Data-centric IoT platforms (i.e., oneM2M and FIWARE) cannot use the service directly and should update data to provide the service. However, since our proposal focuses on service interworking, data-centric platforms require additional data input from the user to request the service. Therefore, there is a limitation that data input should be provided through the client user interface.

(3) In our scenario, we selected only the five platforms (i.e., oneM2M, Oliot, Watson IoT, IoTivity, and FIWARE). However, other IoT platforms can be easily added if the ID format of another platform is analyzed and inputted into the mapping table.

(4) As analyzed in Section 2.2, some IoT platforms have multiple ID formats. For example, in the IBM Watson IoT, HTTP requests (i.e., GET /device/types/[typeId]/devices/[devi

Table 5
Comparison of existing projects and proposed IoT RNS.

Project/architecture	Type	Feature	Limitation
BIG IoT (Bröring et al., 2018)	Integrated platform	• Using an open BIG IoT API and BIG IoT Marketplace enables semantic interoperability on cross-platform. The architecture is based on a common interface, and providers and consumers share resources using the BIG IoT Marketplace, a form of an integrated platform. Service providers in heterogeneous platforms can register resources on the BIG IoT Marketplace using the BIG IoT API.	• The resources and services created through the BIG IoT API can be only shared between platforms.It is difficult to interoperate services from heterogeneous IoT platforms.
INTER-IoT (Ganzha et al., 2017)	Integrated platform	• INTER-IoT provides the common interpretation of data and information among heterogeneous IoT platforms and data sources that cannot share information with each other due to heterogeneous data formats and ontology. INTER-IoT is based on the semantic translation among a common central ontology, global ontology for IoT platforms, and heterogeneous IoT platform ontologies. INTER-IoT developed an inter platform semantic mediator that applies ontology alignment based on message semantics to solve message translation among heterogeneous IoT platforms.	• For an IoT platform to participate in a federation, the ontology of each platform must be constructed.It is difficult to automatically update and clearly define the relationship between the central ontology and the ontology for each platform.
SymbIoTe (Zarko et al., 2017)	Integrated platform	• The CIM is a central model for describing resources registered with symbIoTe and is shared among all platforms participating in the federation. It is an ontology providing information related to sensors, actuators, applications, and services. The CIM performs the intermediary between IoT platforms and applications, retrieves registered resources from heterogeneous platforms and provides them to users.	• Requires constructing an accurate ontology for the main model (i.e., the CIM). Other platforms must be accurately constructed using the main ontology to be interoperable.
Smart hub (Ahmed et al., 2018)	Integrated platform	• Aggregates the transformed data from each data producer based on modeling principles. Interoperability is satisfied by converting the data format into a predetermined common format and classifying it into a predetermined meaning.	• There is a limitation in that heterogeneous IoT platform-based resources must be converted to a specified modeling method.
oneM2M (Kang and Chung, 2018; Tao et al., 2017, 2018; Wu et al., 2017; An et al., 2019)	Duplicated Resources	• oneM2M is developing an IPE for interworking with heterogeneous IoT platforms. The IPE converts resources of other platforms into oneM2M resource structure and creates new resources in oneM2M gateway. All devices connected to the oneM2M gateway periodically transmit the discovery results to the gateway and convert the results into the oneM2M resource tree structure.	• IPE converts resources of other platforms to oneM2M resource structures and creates duplicated resources in the oneM2M gateways. Inefficient memory usage by gateways due to resources being only created by changing the structure.
FIESTA-IoT (Carrez et al., 2017)	Duplicated resources	• Classifies various testbeds by type. Testbeds that use FIESTA-IoT compliant and semantic annotated data together store the data locally. Uses endpoints and queries directly for services and resources. Testbeds that use other annotated data replicate the data according to the FIESTA-IoT ontology.	• Huge memory requirement due to storing resource replicas in the FIESTA-IoT registry. Difficult to accurately construct the FIESTA-IoT registry because it is converted and stored based on the FIESTA-IoT ontology.
Dynamic ontology (Antoniazzi and Viola, 2019)	Common ontology	• Defines the patterns for dynamic interactions among devices as a dynamic ontology and implements the WoT using the common ontology.	• The ontology of each platform must be constructed. Other platforms must be accurately constructed using the common ontology to be interoperable.
Oliot Mediation Gateway (Tolcha et al., 2021)	Duplicated Resources	• For syntactic interoperability between the event-based information model and the entity-based information model, the information model of FIWARE NGSI and GS1 EPCIS is mapped, and the data format is converted.	• A specific mapping between information models is required. In the example of the proposal, only limited domains and platforms are possible.
OPCUA-IPE (Cavalieri, 2021)	Duplicated Resources	• A syntactic interoperability solution performs information model mapping and converting between OPC-UA and oneM2M based on oneM2M IPE. OneM2M resources converted through the interworking manager can be used by OPC-UA users.	• A clear and specific mapping between the information model of oneM2M and OPC-UA is required for format conversion of existing resources. One-way solution for OPC-UA users to use oneM2M resources.
WoT-FIWARE Adapter (Zyrianoff et al., 2021)	Duplicated Resources	• An adapter maps WoT thing description events to FIWARE Orion entities and converts them to NGSI format.	• As with conventional IPE, accurate conversion between resources is required, and synchronization of short time lags is required.
Proposed IoT RNS	Converts and maps the resource address	• It is not necessary to construct the ontology of each platform, and it enables interoperability among heterogeneous platforms through mapping and conversion of IDs. Converts the resource path to each platform's format using a mapping table, and each platform can request to other platforms using the converted path. Communication traffic and memory issues can be solved by separating the functions of root and local IoT RNS without duplicating resources inside the platform.	• To add a new IoT platform, the ID format must be manually registered in the IoT RNS in advance.

Table 6
Qualitative evaluation between existing approach and the proposed IoT RNS.

Project/architecture	**Integrated platform**		**Duplicated resources**	
	Accurate ontology	**Mapping with the central platform**	**High memory**	**Format conversion**
BIG IoT (Bröring et al., 2018)	✗	✓	✗	✗
INTER-IoT (Ganzha et al., 2017)	✓	✓	✗	✗
SymbIoTe (Zarko et al., 2017)	✓	✓	✗	✗
oneM2M IPE (oneM2M, 2019b)	✗	✗	✓	✓
oneM2M base ontology (oneM2M, 2019a)	✓	✓	✗	✗
Smart hub (Ahmed et al., 2018)	✗	✓	✗	✗
FIESTA-IoT (Carrez et al., 2017)	✗	✗	✓	✓
Dynamic ontology (Antoniazzi and Viola, 2019)	✓	✓	✗	✗
Oliot Mediation Gateway (Tolcha et al., 2021)	✗	✗	✓	✓
OPCUA-IPE (Cavalieri, 2021)	✗	✗	✓	✓
WoT-FIWARE Adapter (Zyrianoff et al., 2021)	✗	✗	✓	✓
Proposed IoT RNS	✗	✗	✗	✗

Note that symbols in the interoperability types denote the following: (✓) is required and (✗) is not required qualitative factor.

ceId]/st-ate/[logicalInterfaceId]) and MQTT requests (i.e., iot-2/type/${typeId}/id/${deviceId}/intf/${logicalInterfaceId}/evt/state) format is different. In our scenario, a single ID format is only used in each platform. Therefore, a scenario using various ID formats are considered as future work.

(5) As mentioned previously, five platforms were only targeted in this scenario, but in an actual IoT environment (e.g., smart cities), various platforms and devices can be connected. Therefore, the high performance of the root IoT RNS is indispensable for manage the local IoT RNSs in various IoT platforms. In addition, root IoT RNS should be managed by a trusted organization (e.g., TTP).

(6) In this scenario, service requests are restricted to one direction. For example, oneM2M users request a FIWARE resource, and a specific service is conducted. However, additional data exchanges following the performance of the request are not considered. Therefore, the format and meaning for exchanging data should be further considered in future work.

(7) Security issues have not yet been fully considered for the proposed IoT RNS. The authenticated users should only access the mapping table in root IoT RNS, and security policies must be defined. In addition, users authenticated on their platforms need to control access to specific resources on other platforms (Oh et al., 2019). For example, how to authorize users authenticated on platform *A* to use resources on platform *B* should be considered.

6. Conclusion

IoT technology is rapidly expanding in many fields, including smart homes, logistics, automobiles, healthcare, and smart cities, and related standards, projects, and IoT platforms are constantly being developed and improved. However, the numerous platforms and related standards make it difficult to achieve interoperability and collaboration among platforms. In particular, identifying each resource among heterogeneous IoT platforms is challenging due to various ID formats.

In order to solve this problem, we firstly classified interoperability taxonomy in IoT environments into middleware, network, syntactic, and semantic interoperability, with common security factors for each case. In addition, we proposed an IoT RNS architecture and scenario in a smart city to convert IDs between heterogeneous IoT platforms. Finally, we compared the proposed IoT RNS with existing projects under development and showed it satisfies the interoperability in the heterogeneous IoT platforms.

Future studies will expand interoperability by considering security and implementation-related issues in a real environment. Device authentication and authorization will be added to resource sharing among heterogeneous platforms to address particular security issues. Defining these security policies and applying them to IoT RNS will allow restrictions to be applied for unauthorized access to heterogeneous platform resources.

Funding

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1A2C2012635).

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Ahmed, A., Kleiner, M., Roucoules, L., 2018. Model-based interoperability IoT hub for the supervision of smart gas distribution network. IEEE Syst. J. 13 (2), 1526–1533. https://doi.org/10.1109/JSYST.2018.2851663.

An, J., Le Gall, F., Kim, J., Yun, J., Hwang, J., Bauer, M., Zhao, M., Song, J., 2019. Toward global IoT-enabled smart cities interworking using adaptive semantic adapter. IEEE Internet Things J. 6 (3), 5753–5765. https://doi.org/10.1109/JIOT.2019.2905275.

Antoniazzi, F., Viola, F., 2019. Building the semantic web of things through a dynamic ontology. IEEE Internet Things J. 6 (6), 10560–10579. https://doi.org/10.1109/JIOT.2019.2939882.

Bröring, A., Schmid, S., Schindhelm, C.-K., Khelil, A., Käbisch, S., Kramer, D., Le Phuoc, D., Mitic, J., Anicic, D., Teniente, E., 2017. Enabling IoT ecosystems through platform interoperability. IEEE Softw. 34 (1), 54–61. https://doi.org/10.1109/MS.2017.2.

Bröring, A., Zappa, A., Vermesan, O., Framling, K., Zaslavsky, A., Gonzalez-Usach, R., Kiraly, C., 2018. Advancing IoT Platforms Interoperability. River Publishers.

Carrez, F., Elsaleh, T., Gomez, D., Sanchez, L., Lanza, J., Grace, P., 2017. A reference architecture for federating IoT infrastructures supporting semantic interoperability. In: European Conference on Networks and Communications (EuCNC), pp. 1–6. https://doi.org/10.1109/EuCNC.2017.7980765.

Cavalieri, S., 2021. A proposal to improve interoperability in the industry 4.0 based on the open platform communications unified architecture standard. Computers 10 (6), 1–24. https://doi.org/10.3390/computers10060070.

FIWARE, 2021. https://www.fiware.org/about-us/. (accessed 27 October 2021).

Ganzha, M., Paprzycki, M., Pawłowski, W., Szmeja, P., Wasielewska, K., 2017. Towards semantic interoperability between internet of things platforms. In: Integration, Interconnection, and Interoperability of IoT Systems, pp. 103–127. https://doi.org/10.1007/978-3-319-61300-0_6.

IoTivity, 2021. https://iotivity.org/about. (accessed 27 October 2021).

ISO/IEC, 2019. ISO/IEC 21823-1:2019 Internet of things (IoT) — Interoperability for IoT systems — Part 1: Framework.

ISO/IEC, 2021. ISO/IEC 20924:2021 Information technology — Internet of Things (IoT) — Vocabulary.

Kang, S., Chung, K., 2018. IoT framework for interworking and autonomous interaction between heterogeneous IoT platforms. In: International Conference on Smart Computing and Communication, pp. 217–225. https://doi.org/10.1007/978-3-030-05755-8_22.

Koo, J., Kim, Y.-G., 2017. Interoperability of device identification in heterogeneous IoT platforms. In: 13th International Computer Engineering Conference (ICENCO), pp. 26–29. https://doi.org/10.1109/ICENCO.2017.8289757.

Koo, J., Kim, Y.-G., 2021. Interoperability Requirements for a Smart City. 36th ACM Symposium on Applied Computing (ACM SAC 2021). 690–698. https://doi.org/10.1145/3412841.3441948.

Koo, J., Oh, S.-R., Kim, Y.-G., 2019. Device identification interoperability in heterogeneous IoT platforms. Sensors 19, 1–16. https://doi.org/10.3390/s19061433.

Lee, E., Seo, Y.-D., Oh, S.-R., Kim, Y.-G., 2021. A survey on standards for interoperability and security in internet of things. IEEE Commun. Surv. Tutorials 23 (2), 1020–1047. https://doi.org/10.1109/COMST.2021.3067354.

Li, Y., Zheng, C., Wang, S., 2019. Efficient oneM2M protocol conversion platform based on NB-IoT access. China Commun. 16 (11), 56–69. https://doi.org/10.23919/JCC.2019.11.005.

Oh, S.-R., Kim, Y.-G., Cho, S., 2019. An interoperable access control framework for diverse IoT platforms based on OAuth and role. Sensors 19, 1–17. https://doi.org/10.3390/s19081884.

Oliot, 2021. https://gs1oliot.github.io/oliot/. (accessed 27 October 2021).

oneM2M, 2019. TS-0012-V3.7.3 Base Ontology.

oneM2M, 2019. TS-0024-V3.2.2 OCF Interworking.

oneM2M, 2021. https://www.onem2m.org/harmonization-m2m. (accessed 27 October 2021).

Ranpara, R., Kumbharana, C.K., 2021. Challenges and issues in the existing methodology for dynamic data capturing of ontology. In: Rising Threats in Expert Applications and Solutions, pp. 263–268. https://doi.org/10.1007/978-981-15-6014-9_30.

Tao, M., Ota, K., Dong, M., 2017. Ontology-based data semantic management and application in IoT-and cloud-enabled smart homes. Future Generation Comput. Syst. 76, 528–539. https://doi.org/10.1016/j.future.2016.11.012.

Tao, M., Zuo, J., Liu, Z., Castiglione, A., Palmieri, F., 2018. Multilayer cloud architectural model and ontology-based security service framework for IoT-based smart homes. Future Generation Comput. Syst. 78, 1040–1051. https://doi.org/10.1016/j.future.2016.11.011.

Thiéblin, E., Haemmerlé, O., Hernandez, N., Trojahn, C., Sabou, M., 2020. Survey on complex ontology matching. Semantic Web. 11 (4), 689–727.

Tolcha, Y., Kassahun, A., Montanaro, T., Conzon, D., Schwering, G., Maselyne, J., Kim, D., 2021. Towards interoperability of entity-based and event-based iot platforms: the case of ngsi and epcis standards. IEEE Access 9, 49868–49880. https://doi.org/10.1109/ACCESS.2021.3069194.

Watson IoT, 2021. https://www.ibm.com/docs/en/watson-iot-platform. (accessed 27 October 2021).

Wu, C.-W., Lin, F.J., Wang, C.-H., Chang, N., 2017. OneM2M-based IoT protocol integration. In: IEEE Conference on Standards for Communications and Networking (CSCN), pp. 252–257. https://doi.org/10.1109/CSCN.2017.8088630.

Yang, S., Wei, R., 2018. Tabdoc approach: an information fusion method to implement semantic interoperability between IoT devices and users. IEEE Internet Things J. 6 (2), 1972–1986. https://doi.org/10.1109/JIOT.2018.2871274.

Zarko, I.P., Soursos, S., Gojmerac, I., Ostermann, E.G., Insolvibile, G., Plociennik, M., Reichl, P., Bianchi, G., 2017. Towards an IoT framework for semantic and organizational interoperability. Global Internet of Things Summit (GIoTS) 1–6. https://doi.org/10.1109/GIOTS.2017.8016253.

Zyrianoff, I., Heideker, A., Sciullo, L., Kamienski, C., Di Felice, M., 2021. Interoperability in open IoT platforms: WoT-FIWARE comparison and integration. In: 2021 IEEE International Conference on Smart Computing (SMARTCOMP), pp. 169–174. https://doi.org/10.1109/SMARTCOMP52413.2021.00043.