OneM2M based-Interworking Architecture for Heterogeneous Devices Interoperability in IoT

Diana Yacchirema [a b,*]
[a]*Escuela Politécnica Nacional,*
Ladrón de Guevara E11-253, Quito
17-01-2759, Ecuador

Andreu Belsa Pellicer[b],
Carlos Palau [b], *Manuel Esteve* [b]
[b]*Universitat Politècnica de València,*
Camino de Vera S/N, Valencia,
46022, Spain

***Abstract*— The Internet of things (IoT) fosters a hyper-connected world in which billions of devices that range from higher-grade intelligent mobile terminals to resource-constrained sensors will be connected to the Internet anytime and anywhere. Nevertheless, one of the major obstacles facing the Internet of things is the high diversity of communication capabilities (protocols, technologies and hardware) of the IoT devices. This diversity leads to a highly fragmented IoT market, where various IoT solutions have been developed independently and separately to be used in legacy deployment, which prioritize the vertical optimization instead of the horizontal. Therefore, in this research, in order to address this issue, we analyse the oneM2M specifications and propose an interworking architecture based on such specifications to support both the seamless interoperability of heterogeneous IoT devices and their integration with oneM2M ecosystem. We evaluate the feasibility of this architecture by a use case applied to under real scenario, which derives from an ongoing project.**

Keywords—Internet of things, oneM2M, interworking, interoperability, heterogeneous devices.

I. Introduction and Related Work

Recent advances in communication technologies, and in the capabilities of devices as well as their low cost furnish a great opportunity for the deployment and developed of IoT solutions that can be beneficially applied in various domains from intelligent vehicles, smart cities, smart grid, e-health/m-health to industry control. The rapid rise of this ecosystem is leading IoT towards a hyper-connected world, in which billions of devices that surround us will be connected to the Internet anytime and anywhere. Nevertheless, one of the major obstacles facing IoT is the high degree of diversity of such devices in terms of hardware, software, and communication protocols and technologies. This diversity leads to a highly fragmented IoT market, where various IoT solutions have been developed independently and separately focusing on a specific purpose and being isolated from the rest of the world. In particular, the interoperability of devices is one of the major challenges that must be achieved for facilitating the integration and development of services and IoT applications [1] [2] creating an ecosystem of interoperable IoT solutions. Indeed, the realization of 40% of the potential benefits of IoT depends upon Interoperability (protocols, data formats, content) [3]. However, achieving the interoperability is not straightforward. Toward this end, standards development organizations (e.g., oneM2M, ITU), research projects (e.g., Inter-IoT) and industrial consortiums (e.g., AllJoyn) have been actively conducted activities towards to achieve global interoperability in IoT. In particular, the global standards initiative-oneM2M, an international partnership project launched in 2012, by seven of the world's major standards development bodies of Europe (ETSI), Japan (ARIB and TTC), USA (ATIS and TIA), Korea (TTA), and China (CCSA) have gathered their endeavours to minimize the current fragmentation of proprietary solutions present in the IoT market and in M2M communications through drawing up of broadly applicable interworking specifications independent of underlying access and network transmission technologies [4]. This interworking specifications seeking to ensure that IoT devices seamless interoperate between them on a global scale.

Supporting the IoT devices interoperability using international standards has been research of particular relevance in IoT. Few studies have exploited the technical specifications provided by the oneM2M standard, which define how oneM2M can be used for interworking with legacy systems (i.e., non-oneM2M compliant systems) via specialized interworking proxy entities (IPEs). In this sense, Yun *et al.* [5] introduced an IPE for interworking oneM2M systems with legacy IoT consumer products from Nest, Jawbone, and Withings. The proposed IPE translates the protocol messages between oneM2M's request/response and binding target protocol's message of the servers that store the data of the devices rather than directly translates to protocol's message of the devices. Therefore, this solution requires that the web servers of each device app is in operation otherwise this solution will not work. Likewise, other research proposed by Chia-Wei Wu *et al.* [6] designed an integration architecture based on IPE to address the interworking of specific IoT platforms (i.e., AllJoyn/IoTivity platforms) with oneM2M system. This IPE acts as middleware, which supports the mapping of device management functions among these platforms. They evaluate this solution by two interworking test cases. Although these integration designs are disclosed, there is no evidence that these designs provide the bidirectional communication between devices. Finally, Kim *et al.* [7] demonstrated how the interworking procedures provided by IPE could be applied under real conditions such as smart cities for interworking multiple IoT services platforms. Through the experiences, and lessons learned, the authors emphasized the advantages of the interworking feature.

The main goal of this research consisting of the design and implementation of an interworking architecture to enable both

the seamless interoperability of heterogeneous IoT devices and their integration with oneM2M ecosystem by implementing of an IPE, which runs on a low-cost resource-constrained device. Toward this direction, we extend the contents of our preliminary work presented in [8] by adding (i) new discussions on the internetworking of non-oneM2M compliant heterogeneous IoT devices with oneM2M platform, (ii) an IPE which performs resource mapping, (iii) integration with oneM2M platform, (iv) more details about the architecture developed, and (v) new results obtained as a key component of the proposed solution.

This work continues with an overview of oneM2M functional architecture and interworking specifications (Section II) followed by the description of the proposed interworking architecture and its constituent components for interoperability of IoT devices and integration them with oneM2M ecosystem. After that, we explain the implementation of interworking architecture and evaluate its feasibility by a real use case in which several services are implemented as a result of the interoperability. Finally, we conclude the work and provide future research directions.

II. BACKGROUND

Toward a better understanding, in this section, we briefly introduce the oneM2M functional architecture and the oneM2M Interworking specifications based on IPE.

A. OneM2M functional architecture

The oneM2M defines a functional architecture consisting of two domains (infrastructure and field domain) as shown in Fig. 1. The infrastructure domain consists of an infrastructure Node (IN), which is a server housed on the transmission network side. IN can be connected with INs of another oneM2M service providers. On the other hand, in the field domain, M2M nodes such as middle nodes (MNs) and M2M devices are included. The different M2M devices can be located at different points of the M2M network and according to this can be called Application Service Nodes (ASN) or application dedicated Nodes (AND). The oneM2M defines two basic entities, which can be implemented as software functions within of each node.

A logic node Common Services Entity (CSE) that supports a set of service middleware control functions for AEs and others CSEs such as data and device management, M2M subscriptions and location services. The CSE is deployed in M2M nodes and in each server node [9]. An application entity (AE) that contains the application logic of IoT or M2M solutions such as an application for transport and logistic, sleep apnea monitoring and agriculture. Fig. 1 shows how these entities are deployed within the nodes. In addition, oneM2M defines a network service entity (NSE) which involves basic network services such as transport and connectivity to be used by the CSEs. The connection and exchange information between these entities is done through reference points: Mca, Mcc, and Mcn. The Mca reference point exposes the services included in the CSE to AE running on the devices. The Mcc reference point allows a CSE to use the services included in another CSEs.

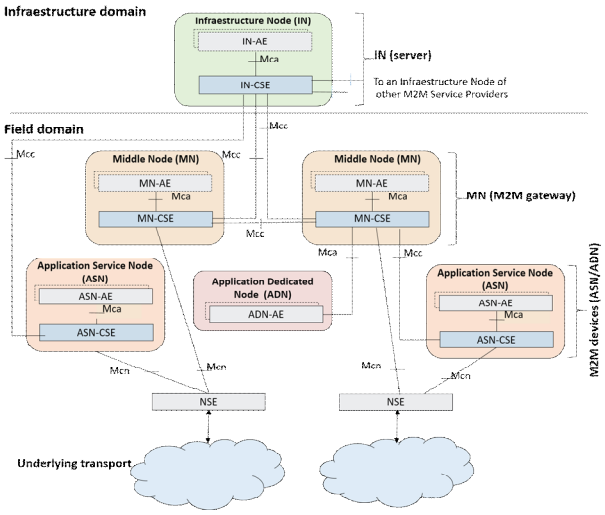

Fig. 1. oneM2M High-Level Functional Architecture

The Mcn reference point allows the CSE to use the supported services by the NSE [2].

The latest version of the oneM2M Release 2 was published in August 2016. This version is intended to provide new functionalities and capabilities for expanding the IoT ecosystem. The updated standard includes enhanced security, semantic interoperability, features for home and industrial domain enablement, and interworking with devices of industry-driven IoT connectivity such as Open Mobile Alliance LWM2M, Open Connectivity Foundation (OCF) and AllSeen Alliance. These updates and their status are disclosed in [9]. In particular, the updated internetworking specifications incorporate improvements based on the early implementation experience in order to achieve high-level interoperability by supporting the interconnection of oneM2M with non-oneM2M complaint devices based on IPE.

B. Interworking proxy entity (IPE)

IPE is a specialized AE defined to enable the interworking between a non-oneM2M compliant node (NoDN) and the CSE of oneM2M. In particular, the IPE is capable of interfacing with various NoDNs and reallocating the NoDN data models to oneM2M resources and vice versa through the oneM2M-specified interfaces.

An example of the operation of IPE is illustrated in Fig. 2 (a), a NoDN (e.g., 6LowPAN-based motion MEMS sensor) is connected to an MN, which, in turn, is connected with an IN, which consists of a CSE to which an AE is enrolled. In order to provide interworking, the IPE needs to convert 6LowPAN-based protocols from non-oneM2M device side to the common protocol like HTTP on the MN side. Furthermore, the IPE needs to map the data model used by the 6LowPAN device into oneM2M resources and then set up the respective resources in the IN-CSE using the MN-CSE services. The AE consisting of a mobile application for fall detection of elderly people, and registered in the IN-CSE can access to the acceleration data gathered from the NoDN.

Fig. 2 (b), depicts some scenarios that can be supported by the result of interworking provided by IPE. Several mixed deployments could be enabled by the combination of these scenarios.

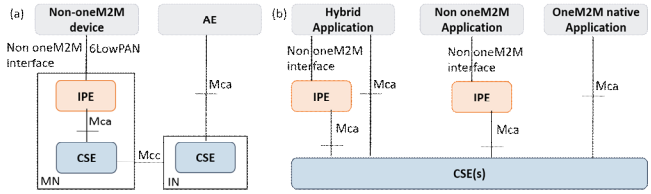

Fig. 2. (a) Example of the IPE operation. (b) Some possible scenarios supported by IPE.

C. *OneM2M Interworking specification via (IPE)*

The OneM2M defines three approaches that can be used to interworking NoDNs with oneM2M systems.

1) Mapping all the NoDN data model to the oneM2M data model, based on containers. In this case, the IPE includes all the interworking protocol logic. Depending on the complexity of the NoDN data model, it may imply that the IPE builds a complex set of resource instances (from the oneM2M core resources) in the CSE. These resources are oneM2M representations of the legacy data model. They allow to CSEs and AEs access to NoDN entities.

2) Using containers for the transparent transport of encoded NoDN data and commands through the Mca interface. Both data and commands are packaged in oneM2M containers. In this case, the CSE or AE need to know the specific protocol coding rules to the NoDN in order to be able to decode the contents of the containers.

3) Using reassignment mechanisms.

In this research, the interworking architecture via IPE is designed and implemented using the first approach, which has been chosen because it offers a unique solution to allow communications between different protocols. In addition, the data model of each NoDN (i.e., the non-oneM2M heterogeneous device) is which determines the representation of resource instances (the names, data types, and structure of the containers) in the M2M system. As a result, this approach allows the interworking of protocols, the exchange of syntactic information, the use and sharing of data between different solutions and deployments.

III. Interworking Architecture

An architecture for achieving the technical and syntactic interoperability of heterogeneous devices in the IoT was proposed in our preliminary work [8]. In this work, we extend this architecture for enabling the integration of these devices with oneM2M ecosystem through of the implementation of an oneM2M-based IPE. The proposed interworking architecture integrates and consolidates several blocks as shown in Fig. 3

The protocol translation block handles the reception and sending of messages to or from IoT devices. To do this, it coordinates communication tasks through different adapters and resolves the problem of incompatibility of different protocols by the encapsulation of the data sent by the source protocol in a format compatible with the destination communication protocol. This module enables the technical interoperability.

The data transformation block is focused on the data types and data schemas. Given that heterogeneity is also present in the different data formats supported by the different IoT devices. According to the type of data collected, this module is in charge of transforming this data to a common data standard defined in the architecture through a syntactic mapper, in order to enable IoT devices to recover the complete information contained in the message. The data-flow in this module is enabled by an interface engine. This module enables the syntactic interoperability.

The integration block is represented by the IPE and it facilitates the common understanding of the collected data, manage access, and extract knowledge from different IoT devices by describing the resource instances in the oneM2M system. This module enables the integration of NoDNs with oneM2M system. It includes eight modules that can interact with each other as shown in Fig. 3.

- A message broker that enables communication streams between IPE components using a publish/subscribe mechanism. Each component can adopt the role of publisher, subscriber, or even both in order to fulfill the needed functionality. The red and blue dashed lines represent these communication streams.
- An activator that is in charge of activating and deactivating the IPE by the implementation of the start () and stop () method.
- A controller that performs two tasks: On the one hand, it starts and stops the IPE's internal components and handles the creation of oneM2M resources in the CSE at the start of the IPE. On the other hand, it executes the received request (e.g., retrieve the state of a device, change their state, etc.) from the oneM2M interface on the resource instances of the devices.
- An event handler able to real-time act by sending commands to the several NoDNs (e.g. actuators).
- An M2M resources mapper is the main component of the integration block that enables the reassignment of the received data model from the data transformation block in an oneM2M resources format through oneM2M interfaces.
- A monitor that retrieves the data of each device exposed to the oM2M system and push such data into the CSE using the data model provided by the M2M resources mapper.
- A router that defines a unique path to handles all request addressed to IPE in a simple resource controller and send the request to the corresponding method of the controller module. This module implements the interworking service interface.
- A request sender that is designed to create oneM2M requests to send to the CSE and provide the response of these requests.

The IPE is registered with the MN-CSE with the aim of hosting the interworking service, which enables the

synchronization the registered IoT devices with the resource instances that these represent in the oneM2M system.

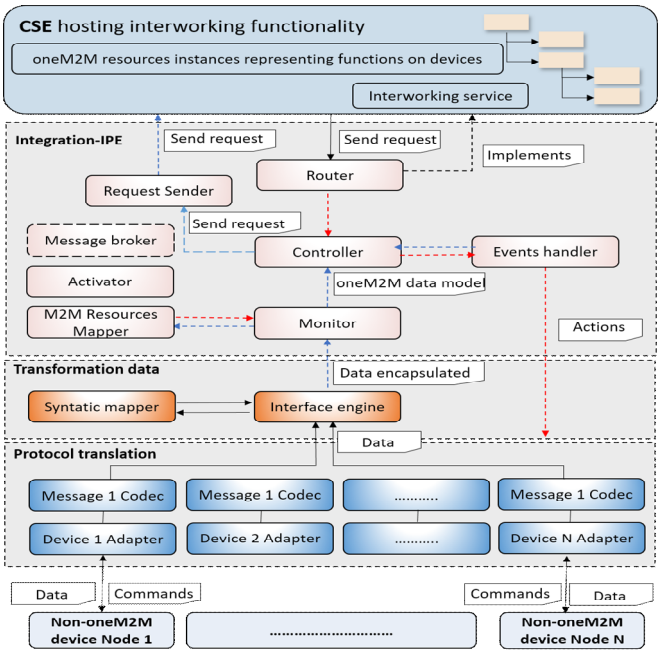

Fig. 3. Interworking architecture via IPE

IV. Testbed Implementation

In this work, a testbed is implemented in order to validate and exploit the oneM2M-based interworking architecture. In particular, we first describe a real scenario and, then, detail the implementation of this architecture and setup on a low-cost resource-constrained device. In addition, we describe various services created as a result of the interoperation, data sharing, cooperation among IoT devices (see Table I) and their integration with oneM2M system.

A. Use case

In the interest of furthering cooperation of IoT platforms through the seamless interoperability of heterogeneous IoT devices and enable the creation of new smart services, a testbed based on Inter-LogP use, which derives from an ongoing project, was carried out.

INTER-LogP use case [10] illustrates the need to achieve seamless interoperability of different heterogeneous IoT platforms, oriented to port transport and logistics at different levels or layers: device, networking, middleware, application and services. In this work, we focus the use case to achieve the interoperability at the device level.

The scope of the INTER-LogP includes several scenarios, in this work, we focus on the scenario of *access control, traffic, and operational assistance*. In this scenario, the interoperability can be directed at resolving several issues. A major issue in port container terminals is the high-level of traffic and congestion at the entrance of terminal gates in peak times. This is caused by the absence of coordination among terminal operators and road hauliers and aggravated by the continuous and fast increase of containers at the port. For instance, the freight forwarder informs the road haulier about the estimated date to pick up or deliver the goods in the container but the container terminal typically is not aware of the date and time the truck arrives at its gate. This lack of information prevents an optimal planning of the port terminal operations, and favors a massive arrival of trucks at the end of closing times, instead of having a more regular flow during the operational time. Consequently, on peak hours there are long queues in the port terminal gate and inside the container yard, thus affecting the quality, safety, and timing of the container handling operations. This inefficiency problem causes long waiting times to hauliers in the terminal and it is translated into a lower performance of terminal operations, loss of time and economic resources and more pollution. Traffic congestion can ultimately lead to considerable delays or even cancellations of transportation orders, becoming an important performance problem in the road transportation, the container terminal, and the port. Two global IoT platforms are involved to facilitate processes in the use case studied: Terminal IoT Platform (TIP) associated to NOATUM container terminal and Port IoT Platform (PIP) associated to the port authority.

Table I summarizes the different technologies and data shared with these platforms in the context of the IoT devices interoperability.

TABLE I. Devices and data shared through the interworking architecture in the use case

Devices	*Data*	*Communication technology*	*IoT Platforms*
GPS NEO-6 sensor	Truck GPS location	Bluetooth	TIP, PIP
Crowtail-weight sensor	Container status (loaded or unloaded)	ZigBee	PIP
MC-38 wired door/windows sensor magnetic switch	Opening or closing of the truck's doors and windows	Wi-Fi-MQTT	TIP
Light alarm actuator IKS01A2	Truck control	6LowPAN-CoAP	TIP

B. Implementation of the interworking architecture

We have implemented the interworking architecture employed only open standards, avoiding technological dependence on proprietary solutions and favoring the customization and development of new functionalities. All the architecture blocks are programmed in Python language, and the communication channels are defined through function calls. Python was chosen because of its inherent advantages for the development. It is an open source GPL-compatible distribution and facilitates the integration of application developed in different programming languages. In addition, it can be run on any machine and has a wide support in any operative system.

The architecture interworking along with the CSE are running in the same execution environment on a smart IoT gateway acting as MN and deployed on a Raspberry Pi 3.

Fig. 4 shows the IoT system implemented for the use case, which affords an overview of how our architecture is integrated within an oneM2M environment. The system consists of a gateway, an oneM2M IoT Server Platform and the IoT platforms of the stakeholders involved in this use case.

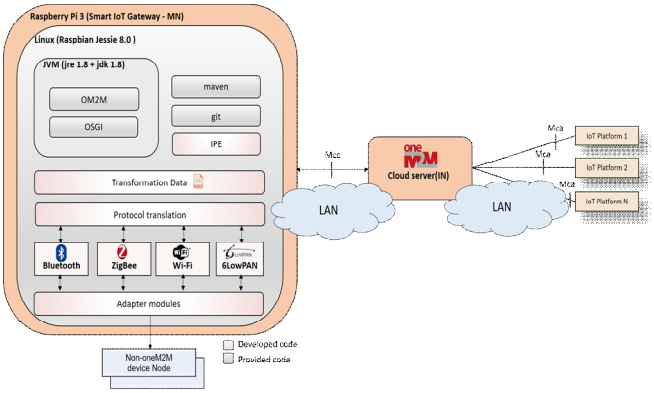

Fig. 4. Integration of proposed interworking architecture with the oneM2M ecosystem.

As oneM2M IoT Server Platform, we used the IN-CSE software provided by the Eclipse OM2M project, which is running on a private cloud server so that only platforms IoT concerned can to access the sensors data.

For the operation of the system, several oneM2M resource entities have been created from devices data. Once registered MN-CSE with IN-CSE, an <AE> resource type called "Smart truck" is created in the smart IoT gateway (MN-CSE) for mapping the truck to the oneM2M resource, through the IPE. The IoT platforms (i.e., TIP and PIP) are subscribing oneM2M IoT Server Platform (IN-CSE) so that any updates in the sensor readings will be notified to these platforms. Then, once the IoT devices are connected to the gateway (MN), the protocol translation block executes the discovery of devices and established the communication with them by a specific adapter according to the communication technology used. Each message received from devices is transformed to common data format defined in the system (i.e., JSON) through the transformation data block, in order to get data format suitable for IPE. Then, IPE determines the devices to be exposed to the oneM2M system and create a <container> resource type for each device under AE, according to the type sensor. In addition, the IPE also creations a <*container*> resource type called "Control" use for representing control data for the truck in the oneM2M system. Subsequently, the IPE translates the device's data encapsulated in a JSON object to oneM2M resource model and send to gateway (MN-CSE) via the creation of a <*contentInstance*> resource type, which has attributes that represent to the device's properties (e.g., value, unit, type, data type, technology). After this, the gateway (MN-CSE) responds with the content of instance created. Figs. 5 and 6 show an example of such reply and a view of the GUI of the oneM2M server (which depicts the oneM2M resource tree generated), respectively.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<om2m:cin xmlns:om2m="http://www.onem2m.org/xml/protocols">
<cnf>message</cnf>
<con>
      &lt;obj&gt;

        &lt;str name=&quot;appId&quot; val=&quot;Smart_Truck&quot;/&gt;

        &lt;str name=&quot;name&quot; val=&quot;GPS-sensor&quot;/&gt;

        &lt;str name=&quot;technology&quot; val=&quot;Bluetooth

        &lt;int name=&quot;value&quot; val=&quot;lat:39,6645_lng:-0,2268&quot;/&gt;

        &lt;int name=&quot;unit&quot; val=&quot;geographical coordinates&quot;/&gt;

        &lt;int name=&quot;data type&quot; val=&quot;float&quot;/&gt;

      &lt;/obj&gt;
</con>
```

Fig. 5. Reply to the creation of a ContentInstance.

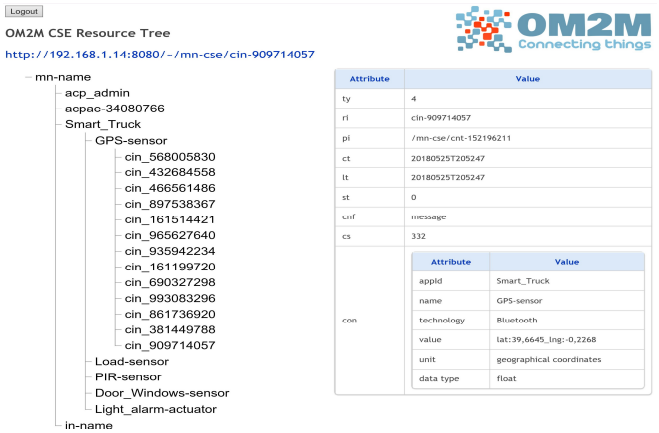

Fig. 6. OneM2M resource structure created for the use case.

The device's data are stored in a database integrated with the oneM2M platform, which implements persistence services. Finally, the oneM2M IoT Server Platform (IN-CSE) propagates these data to IoT platforms using the oneM2M subscription/notification services.

In order to establish the bidirectional communication, when a state change in IoT devices (i.e., actuator) is required, a <*contentInstance*> resource instance of <container> "Control" is created through the IPE. Then IPE maps the data to the NoDN data model supported by the IoT device (e.g., light alarm actuator) on which the action is executed.

C. *Services offered by interworking architecture*

Several services could be developed and delivered to the top the interworking architecture in order to solve existing issues in the *access control, traffic and operational assistance* scenario, including the following.

1) *Improved the access control to port facilities service*

Currently, the trucks for accessing the port must have a valid transport order and an appointment time. Through the use of proposed interworking architecture, the gateway automatically propagates the truck's GPS location and the number plate of the truck to oneM2M IoT Server Platform. Thus, based on this information, the PIP is able to early check the validity of the transport order whereas the TIP can do a cross-checks to validate the appointment. Once the truck arrives in the port, it is identified by means of an LPR (number plate reader) and the port gate is activated automatically by the PIP, thanks to the information provided

by the oneM2M IoT Server Platform. In the same way, when the truck arrives in the container terminal, the TIP allows the truck access to terminal facilities to deliver and/or collect containers.

2) Guidance service

Frequently, some trucks do not immediately find the right route to the container collection and/or delivery points. The haulier may resort to the GPS guidance service on his cell phone as a mobile application offered by the port authorities by sharing the truck's GPS location through our IPE. The mobile application guides the truck haulier directly to the container or pick-up point. The transition from the terminal area to the port would be transparent for the user and would not require switching over to a different application. This service would be active while the truck is inside the port or terminal areas.

3) Improved inspection of empty containers service

In addition, the information sharing by the gateway through IPE supports a more effective and efficient inspection of empty containers. Trucks exit the port through the empty lane in case they carry an empty container. With current procedures, the containers are randomly selected for inspection. As an improvement for this process, the load sensor installed on the container can provide container status information through our IPE. Therefore, all platforms that register this information in their systems can receive this information from oneM2M IoT Server Platform. So once a truck arrives at the port gate in the empty lane, the TIP or PIP can inform to the border police whether the container should be checked or not. In the case that it is not necessary, the exit will be automatic.

4) Improved the Dynamic Lighting service

Currently, the Dynamic Lighting System (DLS) [11] of TIP in container port terminal is capable to significantly reduce the luminary energy consumption an intelligent and efficient way at night time. Though, in the use case scenario, DLS requires to be aware of the GPS position of every vehicle in the terminal area be able to apply the low-consumption lighting mode. If not, the terminal area will be fully illuminated for security reasons. Thus, it is only active if there are no trucks operating in the container terminal, which is a rare situation. The DLS could know this information by sharing the truck's GPS location through our IPE. As a result, the DLS can operate at its full potential, being the low-consumption mode active the whole night time, achieving energy savings up to 78% (based on the initial results provided by the port terminal) compared to the results previous to the sharing of trucks' GPS position.

These services verify that the implementation of our proposed interworking architecture to the *access control, traffic and operational assistance* scenario can achieved several benefits including the minimization of queues and waiting times at the entrance of the terminal, the increase of control and security in the port area, the regulation of the traffic flows inside the terminal and the optimization of port operations.

V. Conclusions and Future Work

The lack of interoperability between IoT devices is a significant barrier to the development and deployment of horizontal IoT solutions able to interoperate with each other. This paper discusses the interoperability of IoT devices and their integration with oneM2M system using international standards specifications. We have proposed and deployed an interworking architecture through an IPE. This architecture enables the interworking of different protocols and communication technologies, the exchange of syntactic information, a continuum use and sharing of data among heterogeneous devices. Moreover, it can interoperate with external parties (e.g., IoT platforms, services, and applications) interested in using the information coming from IoT devices through the oneM2M platform as well as send control messages to IoT devices in a ubiquitous way. A use case derived from an ongoing project validates the feasibility of the architecture proposed that facilitates the creation of smart services in a real IoT scenario focused on transport and logistic.

Acknowledgements

This work has received funding from the European Union's "Horizon 2020" research and innovation programme as part of the "Interoperability of Heterogeneous IoT Platforms" (INTER-IoT) under grant agreement no 687283; ACTIVAGE project under grant agreement 732679; the Escuela Politécnica Nacional, Ecuador; and SENESCYT, Ecuador.

References

[1] ETSI, "Interoperability Best Practices," 2013.

[2] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Commun. Surv. Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[3] Mckinsey Global Institute, "the Internet of Things : Mapping the Value Beyond the Hype," 2015.

[4] J. Swetina, G. Lu, P. Jacobs, F. Ennesser, and J. Song, "Toward a standardized common M2M service layer platform: Introduction to oneM2M," *IEEE Wirel. Commun.*, vol. 21, no. 3, pp. 20–26, 2014.

[5] J. Yun, R. C. Teja, N. Chen, N. M. Sung, and J. Kim, "Interworking of oneM2M-based IoT systems and legacy systems for consumer products," in *2016 International Conference on Information and Communication Technology Convergence (ICTC)*, 2016, pp. 423–428.

[6] C. W. Wu, F. J. Lin, C. H. Wang, and N. Chang, "OneM2M-based IoT protocol integration," in *2017 IEEE Conference on Standards for Communications and Networking, CSCN 2017*, 2017, pp. 252–257.

[7] J. Kim, J. Yun, S. C. Choi, D. N. Seed, G. Lu, M. Bauer, A. Al-Hezmi, K. Campowsky, and J. Song, "Standard-based IoT platforms interworking: implementation, experiences, and lessons learned," *IEEE Commun. Mag.*, vol. 54, no. 7, pp. 48–54, 2016.

[8] D. Yacchirema, C. Palau, and M. Esteve, "Smart IoT Gateway For Heterogeneous Devices Interoperability," *IEEE Lat. Am. Trans. VOL. 14, NO. 8, AUG. 2016*, vol. 14, no. 8, pp. 3900–3906, 2016.

[9] OneM2M, "TS-0001 Functional Architecture 2.19.0," 2018.

[10] Inter-IoT, "Interoperability of Heterogeneous IoT Platforms," 2018.

[11] Valenciaport, "Dynamic real-time lighting system for port terminals," 2015.