Smart M2M Gateway based Architecture for M2M Device and Endpoint Management

Soumya Kanti Datta, Christian Bonnet
Mobile Communication Department
EURECOM, Biot, France
Emails: {dattas, bonnet}@eurecom.fr

Abstract— Internet of Things (IoT) envisions connecting and managing billions of devices and endpoints. This paper describes a smart M2M gateway based architecture to manage the huge volume of M2M devices and endpoints. The architecture is compliant with both ETSI and oneM2M standards recommendations. The resources and the attributes of the M2M devices and endpoints are described using CoRE Link Format. The measurements of the sensors and commands for actuators are carried using Sensor Markup Language (SenML). We have also introduced an extension to CoRE Link for the SenML unit resource description aimed to provide skeuomorphic experience to the end users. The gateway is composed of RESTful web services. The internal structure of the gateway and APIs to manage the M2M devices, endpoints, their discovery and interaction with the mobile clients are described in detail. The prototype implementation of the proposed system is discussed along with the lightweight development of FI-WARE Generic Enablers relevant to the work. Finally the paper summarizes the state-of-the-art, analyses the contribution of the work and concludes with future directions.

Keywords- CoRE Link Format; FI-WARE; Generic Enablers; IoT; M2M device and endpoint; M2M gateway; Sensor Markup Language.

I. INTRODUCTION

Internet of Things (IoT) is a novel paradigm that is shaping the evolution of internet [1, 3]. It envisions billions of physical objects or things like sensors, actuators, RFID tags being connected to the internet. In order to deploy an IoT system, these things must be uniquely addressable and accessed by end users through network. But often these things do not have the capabilities to interact with the internet on their own and are assisted by a M2M gateway [2]. Although powerful and intelligent M2M devices can directly connect to the internet and provide services to the end users, the gateway is indispensable for connecting legacy things and for several other scenarios like smart home. To deploy IoT based system, billions of such M2M devices and endpoints or things will have to be connected together. The management of such volume of devices is a cumbersome task, especially when it has to be tackled without any human intervention.

In this paper, the notion of M2M device and endpoint management using a smart M2M gateway is proposed. The gateway also settles the heterogeneity between M2M devices and internet and bridges the traditional internet with endpoint networks. We have proposed an architecture which emphasizes on gateway based M2M device management. The architecture follows the specifications of both ETSI and oneM2M standards. The internal structure of such a gateway along with its APIs and interactions with M2M devices, endpoints and the mobile clients are the focus of the paper. We have used the CoRE Link Format and IPSO Alliance Framework to configure the M2M device and endpoint resources and attributes [5, 23, 25]. When an M2M device with sensors and/or actuators are connected to the gateway, the device loads its configuration file onto the gateway. The gateway APIs examine the file and stores the device and endpoint configures in a local database. The architecture takes advantage of Sensor Markup Language (SenML) to exchange metadata for both sensors and actuators. Although SenML draft defines metadata only for sensors, extensions to the draft have been developed to address actuators using the same software implementation [18, 26]. We have also extended the capabilities of CoRE Link to add additional resource types for SenML units. These resource types provide additional information on the unit and measurements and drive the user interface of the mobile applications from the M2M gateway. This is explained in Section IV. The internal APIs of the gateway also relate to several FI-WARE Generic Enablers which are also described in Section V [15].

The main contributions of the work are: (i) CoRE Link Format based lightweight resource and attribute description of the M2M device and endpoints, (ii) internal structure of the gateway, its web services, APIs and interactions, (iii) ETSI and oneM2M compliant IoT architecture and (iv) lightweight implementation of relevant FI-WARE Generic Enablers. One possible deployment scenario of the mentioned architecture is in smart homes where one gateway can manage all the connected devices. But the same can be extended to e-Health domain, intelligent transportation system (ITS) and more verticals of IoT ecosystem.

The rest of the paper is explained as below. Section II describes the proposed gateway based architecture. The components of the architecture are referred to that of ETSI and oneM2M. Section III focuses on internal structure and the APIs of the gateway for configuration management of M2M devices and endpoints and the implementation. Interaction of the gateway APIs with the mobile clients are

presented in Section IV. The prototype implementation of the entire architecture includes the development of several Generic Enablers (GE) proposed by FI-WARE project. A brief analysis of the GEs is given in Section V. Section VI examines the performance issues while Section VII provides a study of relevant state-of-the-art.

II. PROPOSED ARCHITECTURE

The proposed architecture is depicted in Figure 1. The gateway acts as the backbone of the architecture and is a collection of RESTful web services divided into two interfaces, north and south. The service capabilities layer (SCL) interfaces at the north interface of the M2M gateway while the M2M devices and endpoints are connected to the south interface of the gateway. The architecture is compliant with the ETSI standards [4]. The M2M gateway together with the M2M devices and endpoints constitute the Device Domain. The SCL and mobile clients running an application (which can also be termed as network applications) constitute the network domain. With respect to oneM2M standardization efforts – (i) the SCL corresponds to the Discovery Domain, (ii) core and access network constitute the Interaction Domain and (iii) M2M gateway, devices and endpoints belong to Resource Domain [24].

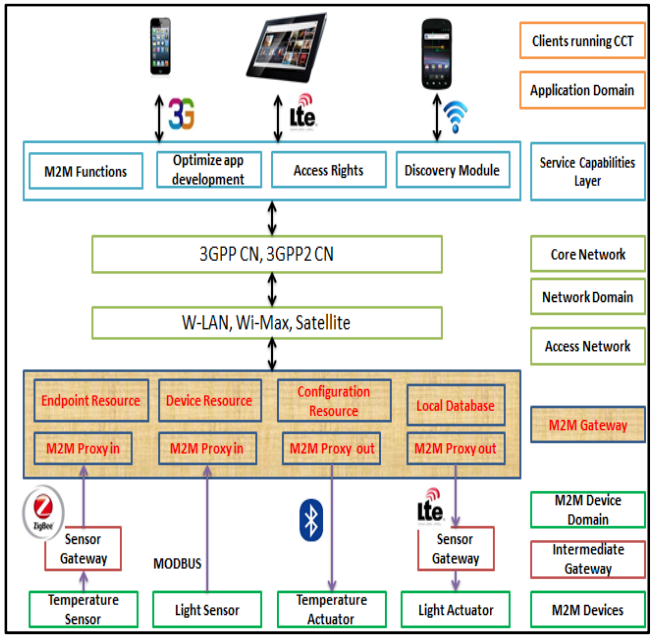

Figure 1. Proposed architecture to manage M2M devices and endpoints.

A. M2M Devices and Endpoints

M2M devices contain the actuals sensors and actuators which are the endpoints or things. A smart device can associate name, id, unit, software version to the measurement value creating a SenML compliant metadata. The interface of the smart device allows the metadata to be read using a GET request or can push the metadata to the gateway. For a non-smart or legacy device, the same is done using another gateway called intermediate gateway (IG). It is configured with the necessary information to create the metadata. In case of a legacy actuator, IG converts the command sent from mobile devices to machine executable format using a predefined protocol.

The work concentrates on providing lightweight description of these devices and endpoints in CoRE Link Format. When a device containing a sensor or actuator is attached to a gateway, the configuration file is pushed to the gateway. The internal APIs of the gateway read the descriptions and store them in appropriate local database. Such configurations also play an important role during dynamic device discovery.

B. M2M Gateway

The mobile clients access the devices and endpoints via the gateway which acts as the backbone of the architecture [17]. The necessary services are developed using REST paradigm. The north interface of the gateway (i) exposes the APIs for managing the things, (ii) implements an API to provide push notification containing sensor measurements, and (iii) assists in dynamic device discovery. The south interfaces employs proxy-in and proxy-out. The sensors are connected to proxy-in web service which facilitates real time interaction between the mobile clients and sensors. This web service can send a GET request to read the sensor metadata or receive the same when the sensor pushes it. The proxy-out links the clients with actuators. This web service receives the URI of the actuator and the command as SenML metadata. It is then converted to machine readable format and sent to the actuator corresponding to the URI.

C. Service Capabilities Layer

This layer exposes the M2M functions through a set of open interfaces that are shared by different applications. It also exploits the core network functionalities to provide reliable service to the clients. SCL provides several advantages – (i) it decouples the mobile applications from the actual sensors and actuators, (ii) provides optimizing in mobile application development, (iii) to provide access rights to the sensors and actuators and (iv) implements the APIs necessary for dynamic discovery of devices and topics [24].

D. Mobile clients

The mobile clients consist of smartphones and tablets running Android, iOS and other mobile operating systems. The clients are equipped with a mobile application called "Connect and Control Things" (CCT) which receive the sensor measurements and can issue commands to control actuators [18].

III. DEVICE CONFIGURATION MANAGEMENT USING THE GATEWAY

The management of M2M devices and endpoints is efficiently done using the gateway. The sensors are connected to the proxy-in while the actuators are connected to the proxy-out of the gateway. We propose to define a function set dedicated based on the CoRE Link specification

as listed in Table 1. All the elements are implemented using separate APIs and are exposed to the SCL and mobile clients through the north interface. This section explains each API and how it interacts with the devices and endpoints through south interface. The following APIs together constitute the Gateway SCL as per the ETSI standards [4].

TABLE I. LIST OF PROPOSED FUNCTION SET

Function Set	Root Path	Resource Type
Device	/d	wg.dev
Endpoint	/e	wg.endpoint
Unit	/unit	wg.senml.unit
Configuration	/cf	wg.config

A. Configuration Resource Description API

An initial configuration of the device and its endpoint(s) must be created. This can be done using a XML or JSON file containing the static description. When the device is first linked to the gateway, this API reads the configuration file using GET request or the file can be pushed to it. Following table portrays the configuration resource description.

TABLE II. CONFIGURATION RESOURCE DESCRIPTION

Type	Path	RT	IF
Device	/cf/d	wg.dev	p
Endpoint	/cf/e	wg.endpoint	p
Unit	/cf/unit	wg.unit	p

The configuration file can be created and/or updated from the mobile clients by making a connection to this API. Access to the resources of devices and endpoints is restricted to authorized clients as determined by the access rights. The resource types “device” and “endpoint” are further explained in following sub-sections.

B. Device Resource Description API

Table III provides the device resource description. The list is specific to the M2M devices connected to the gateway. The “location” is taken from IPSO Alliance Framework which currently supports only “ipso” as namespace in the resource type (RT). Rest of the attributes has namespace “wg” which stands for the wireless M2M gateway. The API reads the device resource description from the configuration file and stores that in a local storage.

TABLE III. DEVICE RESOURCE DESCRIPTION

Type	Path	RT	IF
Location	/d/loc	ipso.loc.gps / ipso.loc.xy / ipso.loc.sem	p
Id	/d/id	wg.dev.id	rp
Name	/d/n	wg.dev.name	p
Model	/d/mdl	wg.dev.model	p
Endpoint	/d/end	wg.dev.endpoint	p
destination	/d/dst	wg.dev.destination	p
proxy-out	/d/po	wg.dev.proxy-out	rp
proxy-in	/d/pi	wg.dev.proxy-in	rp

The attribute types are described below. It is to be noted that, each device is configured with location, id, name, endpoint, model and destination. The proxy-out and proxy-in are assigned by the gateway depending on the type of endpoint attached to the device. The introduction of these two proxies as resource types is our unique addition to CoRE Link specification.

- **Location:** It signifies the type of device’s location which can be described using GPS co-ordinates, X, Y value (X and Y are in meters with respect to an anchor position) or a semantic (Building A / Room 313).
- **Id:** It is the unique identification of the device. It can either be configured in each device or the gateway can assign it. The entire above description is stored in a table (for devices) in a database in the gateway. This id serves as the primary key to for each entry of the table.
- **Name:** It contains the name of the device.
- **Model:** It gives the model name number of the hardware. This is optional.
- **Endpoint:** It provides information on the type(s) of endpoint (sensor, actuator, RFID tag, transducer or logger) attached to the device. This is optional.
- **Destination:** This attribute denotes the URI of the device. For a sensor, the GET request to read metadata is sent to this URI. Similarly, a POST request containing the command for an actuator is sent to this destination.
- **Proxy-out:** This is the web service URI at south interface to which a device with actuator is connected.
- **Proxy-in:** This is the URI to which a device with sensor is connected.

In case of a device that contains both senor and actuator, the device must be connected to both the proxies.

C. Endpoint Resource Description API

This is in charge of retrieving and storing the description of endpoints connected to a device. Table III lists the mentioned description.

TABLE IV. ENDPOINT RESOURCE DESCRIPTION

Type	Path	RT	IF
id	/e/id	wg.endpoint.id	rp
name	/e/n	wg.endpoint.name	p
device	/e/d	wg.endpoint.device	p
senml	/e/senml	wg.endpoint.senml	rp

The attribute types are described below and they are preconfigured into each endpoint.

- **id:** It is the unique identification of the endpoint and can also be called the resource identifier.
- **name:** It is the human readable name of the endpoint.
- **device:** It denotes the URI of the device managing the endpoint.
- **senml:** It carried the SenML metadata related to the endpoint. The metadata includes the information about the type of the endpoint (i.e. sensor, actuator, RFID tag, transducer etc.)

D. Unit Resource Description API

A unit resource is a semantic associated with a SenML unit and is listed in Table IV. Two most important types are graphical representation and allowed range of values for operation. It is necessary for a client to know the operating range of endpoints, particularly for an actuator in order to control it. The graphical information is to be associated with the measurement value and allowed range of values and serves as a feedback for the users. The main aim of using this is to drive the user experience from the gateway itself and it is one of the main contributions of the work.

TABLE V. ENDPOINT RESOURCE DESCRIPTION

Type	Path	RT	IF
Unit	/unit/{senml.unit}	wg.senml.unit	rp
ui-graphical	/unit/{senml.unit}/gr	wg.senml.gr	rp
allowed-range	/unit/{senml.unit}/ar	wg.senml.ar	rp
allowed-list	/unit/{senml.unit}/al	wg.senml.al	rp

- **unit:** It refers to any SenML compliant unit.
- **ui-graphical:** This is a graphical representation (image) associated with the unit.
- **allowed-range:** The range of operation for the endpoints. It is to be noted that this denotes a continuous range of values. The maximum and minimum values are written side-by-side, separated by a semicolon. For example, the allowed operating range of a temperature sensor can be “10; 50” in Celsius.
- **allowed-list:** It is used when the operating range is not continuous. Consider a light controlling actuator having two states only (on and off) and does not permit dimming the light. Instead of using the allowed-range, allowed list is used and the permitted values are written in the same fashion as previously stated.

If the initial configuration file of the device does not contain information about these attributes, then they can be created by accessing the Configuration Resource Description API from the authorized mobile clients. It is to be noted that, the introduction of unit resource description is solely aimed at providing skeuomorphic representation of sensor and actuator measurements, allowed range of values, unit to the end users.

E. Push Notification API

This API is necessary to update the mobile clients about sensor measurement updates even when the application CCT is running in background. To receive the notification, CCT must subscribe to this API. It is also used to notify user when a device and/or endpoint is dynamically added to or removed from the gateway [17].

F. Actuator Control API

When a client issues a command to control an actuator (e.g. switch off a light, reduce the room temperature using AC), the command is received by this API. At first it extracts the URI of the proxy-out to which the actuator is connected. Then the rest of the SenML coded metadata is examined to find out the URI of the actuator, whether it is smart or legacy. In case of smart actuator the metadata is forwarded by the proxy-out. But for the legacy actuator, the command is translated into a machine readable format and then communicated to the actuator.

G. Semantic API

There is another API which is reasoning on the sensor and actuator metadata to enrich overall M2M data using semantic web technologies across several domains (e-Health, transportation, smart home). The API follows Machine-to-Machine Measurement (M3) Ontology proposed by the authors of [27, 28]. The sensor metadata received at the proxy-in is fed to this API which annotates the metadata following RDF, RDFS and OWL. The API then further reasons to classify the M2M devices, their metadata and the domain of operation. The main advantage of the API is that it can combine cross domain ontologies to propose new information to the end users. They have to subscribe to receive the semantic information, same as the push notification.

The main API for dynamic device discovery is implemented in the SCL. When the client initiates the discovery phase, the SCL queries the gateway based on predefined rules and access rights to learn about the attached devices and endpoints. Thus there is no additional API for this. Similarly the access control and authentication of the clients to the gateway are implemented with the help of SCL.

IV. INTERACTION OF THE GATEWAY APIS WITH MOBILE CLIENTS

This section provides insight on the interaction of the gateway APIs with the client via the SCL of the architecture. An example scenario as depicted in Figure 2 which is used to

explain the implementations. Two M2M devices are connected to the gateway at the south interface and the mobile client is connected to the north interface through the service capabilities layer. The APIs are developed in Java and the interaction with the M2M devices and clients can be done using XML/JSON.

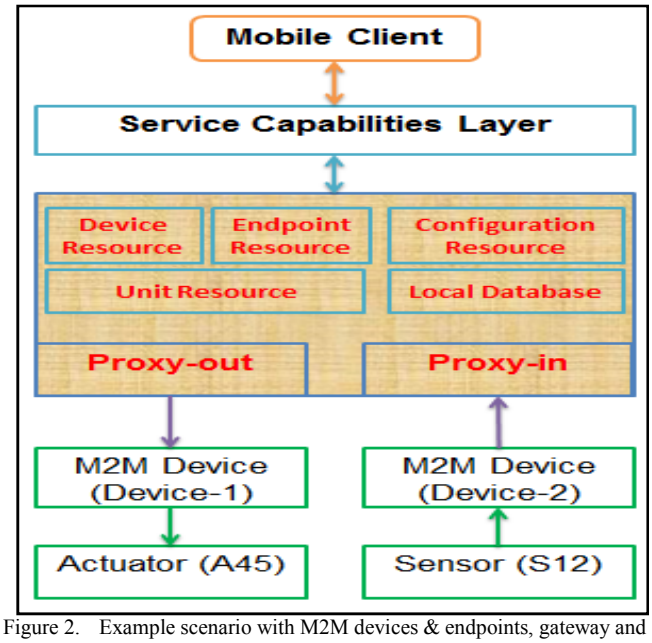

Figure 2. Example scenario with M2M devices & endpoints, gateway and mobile client.

The initial configuration files are pushed to the gateway and are examined by the configuration resource API. Then the device, endpoint and unit resource descriptions are extracted from those files by the corresponding APIs and stored in the local database. After this, when the mobile client issues a GET request to receive the details of the devices connected to the gateway, the Device Resource Description API responds with the full list of devices and their descriptions. With respect to Figure 2, an example of such reply is given below.

```
<d>
<r path="/d/dev1">
<a path="loc" value="Room 313" rt="ipso.loc.sem", if="p"/>
<a path="id" value="AxZi21d" rt="wg.dev.id", if="rp"/>
<a path="n" value ="Device-1" rt="wg.dev.name", if="p"/>
<a path="mdl" value="Model-1" rt="wg.dev.model", if="p"/>
<a path="po" value="po1-webservice.mydomain.com", rt="wg.dev.proxy-out", if="rp"/>
<a path="dst" value="device-1.m2mdevices.com", rt="wg.dev.proxy-out", if="rp"/>
</r>
<r path="/d/dev2">
<a path="loc" value="Room 311" rt="ipso.loc.sem", if="p"/>
<a path="id" value="Sz7wb8" rt="wg.dev.id", if="rp"/>
<a path="n" value ="Device-2" rt="wg.dev.name", if="p"/>
<a path="mdl" value="Model-2" rt="wg.dev.model", if="p"/>
<a path="pi" value="pi2-webservice.mydomain.com", rt="wg.dev.proxy-in", if="rp"/>
<a path="dst" value="device-2.m2mdevices.com", rt="wg.dev.proxy-out", if="rp"/>
</r>
</d>
```

From the field proxy-out and proxy-in, it is understood that if the device has an actuator or a sensor. Now that the client received the information about the devices, it can choose a device and request the gateway to send the details of the connected endpoint. For example, if the user chooses device 2 of Figure 2 then the request and reply are as follows.

```
GET:/e?d="dev2"
Returns:
<e>
<r path="/e/S12">
<a path="id" value="Sn2DnCE1" rt="wg.endpoint.id", if="rp"/>
<a path="n" value="Sensor12" rt="wg.endpoint.name", if="p"/>
<a path="d" value="/d/dev2" rt="wg.endpoint.device", if="p"/>
<senml bn=urn:dev:mac:6399877>
    <e n="temp" t="0" v="17.5" u="Cel" xif="s"/>
</senml>
</r>
</e>
```

As seen from the response, it contains the description of a sensor along with the latest SenML metadata containing the measurement. It is to be noted that the field "xif" is an interface definition introduced by us so as to differentiate between different types of endpoints like sensors (s), actuators (a), RFID tag (r) and transducers (t). The notion is introduced so that the same SenML software implementation can be used not only for sensors but also for other endpoints [17, 18]. The SenML unit in this case is Celsius. Thus the unit resource description sent to the mobile client is as follows.

```
<unit>
<r path="/unit/Cel">
<a path="unit" value="Cel" rt="wg.senml.unit", if="rp"/>
<a path="gr" value="slider.png" rt="wl.senml.gr", if="rp"/>
<a path="allowed-range" value ="10;30" rt="wl.senml.ar", if="rp"/>
</unit>
```

From the unit resource description, it is clear that the range of operation of the sensor is 10 to 30 degree Celsius. The latest sensor measurement 17.5 should be represented using a slider (semantic of the graphical representation) and the image is located at "slider.png". The maximum and minimum value of the slider will portray 10 and 30 while the pointer of the slider will be at 17.5 to indicate the current temperature. This is done to drive the user interface and user experience from the M2M gateway. Thus the gateway not only manages the devices and the connected endpoints, but also contributes to the user experience in the mobile clients.

Configuring the devices and endpoints is possible from the mobile clients. After they are linked to the gateway, the clients can access the configuration file of a device using GET request as shown below. The configuration resource API exposes only the attributes which can be created or updated by users like the location.

```
GET /cf?d="dev1"
Response:
<cf>
<r path="/cf/d" rt="wl.dev>
<a path="loc" value="" rt="ipso.loc" if="p"/>
</r>
</cf>
```

In the above example, the location attribute of the dev1 is empty and can be created by the user. The client pushes the value (Building-A which is semantic location) to the configuration file as seen below. The POST is issued for the Configuration API which internally updates the location attribute for dev1 at the Device Resource API.

```
POST /cf/d
<r path="/cf/d" rt="wl.dev>
<a path="loc" value="Building-A" rt="ipso.loc.sem"/>
<a path="n" value="dev1" rt="wg.name"/>
</r>
Response: 204, No Content
```

Apart from the location, name and model can also be created and/or updated using the same procedure from the client.

V. Implementation of Lightweight FI-WARE Generic Enablers

FI-WARE is an initiative that provides an open cloud-based infrastructure to create and deliver cost-effective applications and services for future internet (FI) [15]. The initiative aims to build Generic Enablers (GE) for IoT service enablement. This will in turn allow the physical things to be available, searchable, accessible and usable by high level applications and end users for different purposes. Each GE is a building block of FI-WARE and consists of a set of functionalities, APIs and interoperable interfaces compliant with open specifications. The prototype development of the different components of the architecture, especially the M2M gateway APIs relates to development of several GEs. In this section, we give an overview of several such GEs which are relevant to our work.

The GEs have been broadly classified into two domains (e.g. IoT Gateway Domain and IoT Backend Domain) from IoT architecture point-of-view [16].

A. IoT Gateway GE

Our lightweight implementation of this GE runs within the Gateway SCL and provides inter-networking, protocol conversion & network traffic optimization for IoT backend. This GE combines all the internal APIs of the gateway. In addition to that the GE extends its capabilities to offer gateway based M2M device discovery and assists legacy sensors and actuators to be a part of the IoT ecosystem.

B. IoT Backend GE

The backend is composed of several applications and services and can be hosted in web servers or cloud. This GE typically addresses the IoT domain specific applications. In our architecture, we have developed this GE which interacts with the Android application to provide services (e.g. update sensor measurement, push notification, controlling actuators) to the end-users.

Considering the functional part of the IoT architecture, following GEs are important to mention.

C. Data/Context Management GEs

These GEs are part of both Device SCL and Gateway SCL. The main tasks include the following and more details of the prototype implementation can be found in [17].

- Generate sensor measurement and combine it with additional data (e.g. unit, type, id, name, version and timestamp) to create sensor metadata.
- Various context informations (e.g. location of M2M devices) are also made available.
- The Gateway SCL is capable of generating new information (e.g. semantic notation of the measurements) from the received metadata using the M3 ontology [27, 28].
- The gateway also acts as an aggregation point for various sensor data. Such data can be analyzed using Big Data algorithms to provide further value added services.

D. Applications/Services Ecosystem and Delivery Framework GEs

We are currently in the process of designing and building an ecosystem of mobile applications for different verticals of Internet of Things like home automation, e-health, intelligent transport system, smart metering and more. In order to accelerate the application development process, reduce the time-to-market and release on multiple mobile OS platform, we suggest the use of cross platform mobile application development tools. A detailed survey, comparison and evaluation based on memory, CPU and power consumption are provided in [19]. The proposed ecosystem will foster accelerated innovation in developing novel use cases and relevant mobile applications. Special attention will be paid to support different industrial business models.

E. Interface to Networks and Devices (I2ND) Architecture GEs

FI-WARE defines I2ND as a provider of an enabler space where GEs can run an open and standardized network infrastructure [20]. The architecture includes four generic enablers and two of them have been so far addressed as mentioned below:

- **Connected Device Interface (CDI)**: This GE equips the mobile clients (terminals, tablets, smartphones) with real time and remote access to M2M devices and endpoints. In this case, the GE has been implemented as the mobile application CCT [18].
- **Service Capability, Connectivity and Control (S3C)**: This GE is running in the SCL layer and the

functionalities include – (i) a self-adaptive framework for battery and context aware mobile application development [21], (ii) a framework to optimize the mobile application development using cross platform tools, (iii) an API for dynamic device discovery and (iv) an ecosystem of mobile application to serve different IoT domains which is under development.

VI. Performance Evaluation

The file containing the M2M device and endpoint(s) is typically less than 1KB in size. The configuration being written in JSON format in the file accounts for such small footprint. Thus With an available internal memory of the M2M gateway of 10MB, it can store and manage about 10,000 configuration files. Thus it is possible to utilize such concept in smart home and other industrial applications of IoT. Even if the number of connected objects increases, one gateway can handle thousands of such objects. Thus the proposed architecture is highly scalable.

VII. State-of-the-Art

Researches in Internet of Things domains and Machine-to-Machine Communication have identified the necessities and requirements in M2M gateways [12, 13, 14]. Authors Huang and Hsieh have presented an implementation of a programmable and low-cost IoT gateway in an embedded system which has been used in monitoring systems in IoT test bed [6]. The authors have presented the hardware platform (microcontroller, SPI, battery module and different sensors), system stack and system protocol in detail. Guoqiang, Yanming, Chao and Yanxu have described a smart IoT gateway that aims to bridge the gap between traditional network and sensor network [7]. The gateway architecture provides modules with different communication protocols and therefore can be attached to different networks. Software development is made easy through unified external interfaces. The paper also discusses a protocol to translate different sensor data into a uniform format but it does not use SenML. In order to avoid repetitive manual configurations on the gateway due to increased volume of devices, a server-assisted provisioning method is proposed in [8]. This method aims to avoid extra load at the gateway. The evaluation of the prototype system promises considerable reduction in operational steps and time required to configure the gateway. In [9], the authors present an IoT gateway traffic model in CobraNet based digital broadcast system (CDBS). The system architecture (including signaling information, management information, audio flow), system design, message function are mentioned in detail. Methods for data aggregation for M2M gateways are described in [10]. Costantino, Buonaccorsi, Cicconetti and Mambrini have studied the suitability of using LTE to connect the M2M gateways to the internet [11]. The system makes use of CoAP as the transmission protocol instead of HTTP as the former is specifically designed to be used on constrained resource scenarios like M2M devices and gateways. The paper also presents simulation results of a typical IoT scenario using ns3.

But the above research directions do not consider the internal structure of M2M gateway, its APIs and interaction with things and mobile clients. The CoRE Link based description of M2M devices and endpoints for managing them are not covered. This is one of the main contributions of this paper. Such implementations are lightweight to suit the constrained nature of the M2M gateway. The paper also provides insight about several relevant FI-WARE GEs.

VIII. Conclusion and Future Direction

To summarize, the paper provides an insight into managing M2M devices and endpoints using a smart M2M gateway. The resources and attributes are expressed using CoRE Link Format. An extension to that is proposed to enhance the operability of SenML units. The main idea behind that is to drive the user interface of the mobile applications from the gateway. The internal structure of the gateway is mentioned along with the implementation of the APIs necessary for the management. The APIs store configuration of the M2M devices and endpoints and also assist the mobile clients and SCL for dynamic device discovery. The architecture in Figure 1 follows the recommendations of both ETSI and oneM2M. We have further briefed the development of relevant FI-WARE Generic Enablers for the overall service enablement.

Comparison with current literature is done to identify that most of the works have not focused on efficient management of the billions of things. Therefore our work advances the related state-of-the-art by providing a methodology to manage those things using a smart M2M gateway. The same APIs can be implemented in the SCL also in case the gateway is not employed in IoT system. The overall system also takes advantage of open source implementation of SenML extensions to control actuators.

As a future direction, we are working on multi-protocol gateway architecture to provide seamless integration approach for heterogeneous devices using different protocols for communication [22]. Further research is being carried out on semantic-based machine learning algorithms.

Acknowledgment

The authors express their acknowledgment to the consortium of the French research project WL-Box-4G Pole SCS.

References

[1] Handong Zhang; Lin Zhu, "Internet of Things: Key technology, architecture and challenging problems," *Computer Science and Automation Engineering (CSAE), 2011 IEEE International Conference on* , vol.4, no., pp.507,512, 10-12 June 2011.

[2] Qian Zhu; Ruicong Wang; Qi Chen; Yan Liu; Weijun Qin, "IOT Gateway: BridgingWireless Sensor Networks into Internet of Things," *Embedded and Ubiquitous Computing (EUC), 2010 IEEE/IFIP 8th International Conference on* , vol., no., pp.347,352, 11-13 Dec. 2010.

[3] Glitho, R.H., "Application architectures for machine to machine communications: Research agenda vs. state-of-the art," Broadband and Biomedical Communications (IB2Com), 2011 6th International Conference on, pp.1-5, 21-24 Nov. 2011.

[4] ETSI Technical Specification on Machine-to-Machine Communications; Functional Architecture, ETSI TS 102 690, V2.1.1 (2013-10).

[5] Constrained RESTful Environments (CoRE) Link Format, IETF RFC 6690.

[6] Ji-De Huang; Han-Chuan Hsieh, "Design of Gateway for Monitoring System in IoT Networks," Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing, pp.1876-1880, 20-23 Aug. 2013.

[7] Shang Guoqiang; Chen Yanming; Zuo Chao; Zhu Yanxu, "Design and Implementation of a Smart IoT Gateway," Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing, pp.720-723, 20-23 Aug. 2013.

[8] Hattori, Masaharu; Yagi, Hikaru; Yoshihara, Kiyohito, "A server-assisted provisioning method for machine-to-machine gateway," Network Operations and Management Symposium (APNOMS), 2013 15th Asia-Pacific, pp.1-3, 25-27 Sept. 2013.

[9] Xianyang Jiang; Deshi Li; Shaobo Nie; Jing Luo; Zhonghai Lu, "An Enhanced IOT Gateway in a Broadcast System," Ubiquitous Intelligence & Computing and 9th International Conference on Autonomic & Trusted Computing (UIC/ATC), 2012 9th International Conference on, pp.746-751, 4-7 Sept. 2012.

[10] Matamoros, J.; Anton-Haro, C., "Data aggregation schemes for Machine-to-Machine gateways: Interplay with MAC protocols," Future Network & Mobile Summit (FutureNetw), 2012, pp.1-8, 4-6 July 2012.

[11] Costantino, L.; Buonaccorsi, N.; Cicconetti, C.; Mambrini, R., "Performance analysis of an LTE gateway for the IoT," IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012, pp.1-6, 25-28 June 2012.

[12] Riedel, T.; Fantana, N.; Genaid, A.; Yordanov, D.; Schmidtke, H.R.; Beigl, M., "Using web service gateways and code generation for sustainable IoT system development," Internet of Things (IOT), 2010, pp.1-8, Nov. 29 2010-Dec. 1 2010.

[13] Qian Zhu; Ruicong Wang; Qi Chen; Yan Liu; Weijun Qin, "IOT Gateway: BridgingWireless Sensor Networks into Internet of Things," Embedded and Ubiquitous Computing (EUC), 2010 IEEE/IFIP 8th International Conference on, pp.347-352, 11-13 Dec. 2010.

[14] Singh, S.; Kuei-Li Huang, "A robust M2M Gateway for effective integration of capillary and 3GPP networks," Advanced Networks and Telecommunication Systems (ANTS), 2011 IEEE 5th International Conference, pp.1-3, 18-21 Dec. 2011.

[15] FI-WARE Project, http://www.fi-ware.org/

[16] http://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/Internet_of_Things_(IoT)_Services_Enablement_Architecture

[17] S. K. Datta, C. Bonnet and N. Nikaein, "An IoT Gateway Centric Architecture to Provide Novel M2M Services", IEEE World Forum on Internet of Things (WF-IoT), pp. 514-519, 6-8 March 2014.

[18] S. K. Datta, C. Bonnet and N. Nikaein, "CCT: Connect and Control Things", 9th International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 21-24 April 2014.

[19] Dalmasso, I.; Datta, S.K.; Bonnet, C.; Nikaein, N., "Survey, comparison and evaluation of cross platform mobile application development tools," *Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International*, pp.323-328, 1-5 July 2013.

[20] https://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/Interface_to_Networks_and_Devices_(I2ND)_Architecture

[21] S. K. Datta, C. Bonnet and N. Nikaein, "Self-Adaptive Battery and Context Aware Mobile Application Development", 10th IEEE International Wireless Communication and Mobile Computing Conference (IWCMC), 4-8 August 2014. [in press].

[22] Jung, M.; Weidinger, J.; Reinisch, C.; Kastner, W.; Crettaz, C.; Olivieri, A.; Bocchi, Y., "A Transparent IPv6 Multi-protocol Gateway to Integrate Building Automation Systems in the Internet of Things," *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*, pp.225-233, 20-23 Nov. 2012.

[23] Work-in-Progress Draft on CoRE Interfaces, http://tools.ietf.org/html/draft-shelby-core-interfaces-05.

[24] http://www.onem2m.org/library/oneM2M-TR-0002-Architecture_Analysis_Part_1-V0_2_0.doc

[25] IPSO Alliance Framework, http://www.ipso-alliance.org/wp-content/media/draft-ipso-app-framework-04.pdf.

[26] IETF SenML Draft, http://tools.ietf.org/html/draft-jennings-senml-10.

[27] A. Gyrard, C. Bonnet and K. Boudaoud, "A machine-to-machine architecture to merge semantic sensor measurements, " in 22nd International World Wide Web Conference (WWW 2013), May 2013, Rio de Janeiro, Brazil.

[28] A. Gyrard, C. Bonnet and K. Boudaoud, "Enrich Machine-to-Machine Data with Semantic Web Technologies for Cross-Domain Applications," in IEEE World Forum on Internet of Things (WF-IoT), pp. 559-564, 6-8 March 2014.