An Approach to the Unified Management of Heterogeneous IoT Environments

Ngombo Armando, *DEI, University of Coimbra, CISUC*, Jorge Sá Silva, *DEEC, University of Coimbra, CISUC,* and Fernando Boavida, *DEI, University of Coimbra, CISUC*

***Abstract*—While the traditional Internet of Things (IoT) relies on electronic sensors/actuators, today's IoT involves a variety of sensors, comprising not only physical, electronic-based devices, but also virtual and even human social sensors/actuators. In this context, how can we efficiently and effectively manage different types of IoT data sources? In this paper, we propose a solution to manage different types of sensors/actuators and analyze which management protocols are best in terms of performance. The proposal is based on open and broadly adopted technologies in IoT, with emphasis on the FIWARE middleware. We showed that the management of the heterogeneity of the sensing/actuating IoT devices is feasible, by presenting functionalities related to real use-cases. We took advantage of the implemented prototype to compare the performances of Lightweight Machine-to-Machine (LwM2M) and Ultralight device management services in FIWARE. In addition to demonstrating the viability of the proposed approach, the obtained results point to mixed advantages/disadvantages of one protocol over the other.**

***Index Terms*— IoT, Management, FIWARE, LwM2M, Ultralight**

I. INTRODUCTION

THE Internet of Things (IoT) is an extension of the Internet to encompass all sorts of smart and connected entities labelled "things". IoT applications cover activities for both monitoring and tracking purposes, including healthcare, critical infrastructure protection, and automated diagnostics.

In IoT, sensor nodes are entities in charge of collecting data from the surroundings and report them to a central unit for further processing. The concept of sensing has evolved considerably over the last decade, comprising a broad set of possibilities to collect data towards the development of smart applications and services. What is generically called the Internet of "things" is already the networked connection of physical things and beyond [1]. Nowadays, IoT involves a variety of sensors, including not only physical, electronic-based devices, but also virtual sensors (i.e., software agents that abstract one or more physical sensors), and even human sensors, such as human-originated data collected from Online Social Networks (OSN), which we call social sensors. Social sensors are excellent sources of data for, e.g., data-driven approaches and artificial-intelligence-based innovative services, along with traditional IoT-based services, leveraging emerging technologies such as edge/cloud computing and 5G networks.

Such heterogeneity of "devices" is inevitable and has an impact on the ITU reference model [2], that may require a redefinition of some other concepts already established in the device layer. Moreover, it is apparent that management solutions must deal with all sorts of sensors, in a way that is as transparent as possible. We, nevertheless, expect that different IoT management protocols will have different capabilities to deal with different types of sensors/actuators. In this context, our initial goal was to study the extent to which the existing IoT management protocols can be used for managing such a wide variety of entities in the IoT device layer, which deals with sensors and actuators. Subsequently, from this initial goal, an approach to the management of heterogeneous IoT environments was developed, and this is the central outcome presented in this paper.

The contributions of this paper can be summarized as follows. We propose and validate a solution for the unified management of heterogeneous sensing/actuating approaches in today's IoT. The proposed solution adopts relevant open standards for both data and device management. We show that the management of the referred three types of sensing/actuating devices - namely physical, virtual, and social - is feasible from a functional point of view. Furthermore, we analyze and compare the performance of two widely used management protocols. Specifically, we extensively assessed the latest stable versions of two relevant protocols for supporting management architectures, namely Lightweight Machine-to-Machine (LwM2M) 1.0 and Ultralight 2.0, available as services in a top reference IoT middleware named FIWARE [3].

The remainder of the paper is organized in the following way: Section II presents some base concepts and related work; in Section III, we present the proposed unified management approach, focusing on its underlying concepts and supporting technological ecosystem; Section IV is dedicated to the

Submitted for review at 26-June-2020. This paper was carried out within the scope of the 5G Project (POCI/01/0247/FEDER/024539), co-financed by COMPETE 2020, Portugal 2020 - Operational Program for Competitiveness and Internationalization (POCI I), European Union's ERDF (European Regional Development Fund) and the Portuguese Foundation for Science and Technology (FCT).

The authors are with the Centre of Informatics and Systems of the University of Coimbra (CISUC), 3000-214 Coimbra, Portugal (e-mails: narmando@dei.uc.pt, sasilva@deec.uc.pt, boavida@dei.uc.pt).

N. A is also with Escola Superior Politécnica do Uíge, Universidade Kimpa Vita, Nkondo Mbenza, Uíge, Angola.

presentation of the setups mounted for the management use-cases and the assessment of the management protocols in FIWARE. Both functional and performance results are presented and discussed in Section V. Lastly, in Section VI, we summarize the findings and provide guidelines for further work.

II. Related Work

This section starts with an overview of the IEEE 1451 standards family [4] [5], as these standards set a common language for sensor-based applications. Subsequently, we address the generic sensing loop concept. Finally, we tackle the problem of IoT device management by identifying existing solutions. The concepts and solutions addressed in this section will be used as the basis for extending IoT to software-based and human-based entities, and their respective management.

A. IEEE 1451 Standards Family

Transducers (sensors, actuators, filters) are the primary electronic interfaces to the real world. They are entities that receive a signal as input and generate a modified signal as output [6]. In this regard, smart transducers are analog or digital sensing/actuating units combined with both a processing unit and a communication interface. The family of smart transducer standards was created in 2007 under the designation of IEEE 1451, with the objective of facilitating device and data interoperability in the realm of IoT and Cyber-Physical Systems. Since then, IEEE 1451 standards have been largely adopted by the industry [4]. These standards define a set of common communication interfaces, network services, metadata concerning transducer connectivity to instruments, instrumentation systems, and control/field networks, in order to enable access, management, and control of networked transducers.

As shown in Fig. 1, the IEEE 1451 standards family defines two main blocks: i) the Transducer Interface Module (TIM), widely known in the literature as sensor node; ii) and the Network Capable Application Processor (NCAP), widely known in the literature as a network gateway. The TIM block comprises hard transducer end-units, metadata called Transducer Electronic Data Sheet (TEDS), signal processing units, and the necessary communication protocols to deal with the NCAP driver.

A TEDS contains manufacture-related information that allows both the self-identification and self-description of transducers to the system or network. For instance, in a TEDS, we may find transducers' manufacturer ID, serial number, measurement ranges, calibration data, and location information. A TEDS usually resides in embedded memory, typically EEPROM, within the transducer. The concept of virtual TEDS extends the benefits of the standardized TEDS to legacy sensors and applications where embedded memory is not available. That means a virtual TEDS can exist as a separate file downloadable from the Internet [7].

The NCAP block comprises communication modules to drive the Transducer Physical Interface and the essential functions required to control and manage transducers, communication protocols, and media-independent TEDS formats. It also comprises network services, for connectivity with user applications. The interface between the NCAP block and the TIM block is called Transducer Physical Interface, and it includes both serial and wireless links, such as SPI, I2C and RS-232/UART serial interfaces, defined in IEEE 1451.2, or wireless interfaces such as IEEE 802.15.1, IEEE 802.15.4, and 6LowPAN. As specified in IEEE 1451.5, 6LowPAN is intended to allow direct TIM access from the Internet.

B. Generic Sensing Loop

Fig. 2 illustrates a generic sensing loop in today's IoT [1]. In this sensing loop, sensing/actuating tasks may involve a variety of devices, comprising not only physical, electronic-based devices, but also virtual sensors (i.e., software agents that abstract one or more physical sensors), and even human sensors, such as human-originated data collected from OSNs.

While the sensors enable applications to monitor and provide an abstract representation of "things", the actuators enable the referred applications to interact with them. The heterogeneity in mechanisms for sensing/actuating leads to challenging questions in what concerns IoT solutions. For our study, we were interested in tackling the management of IoT devices beyond traditional electronic-based devices.

Let us take as an example the measurement of the overall ambience of a house. To this end, we will set three metrics for the measurement of the phenomenon under consideration namely, room temperature, perceived comfort and the mood

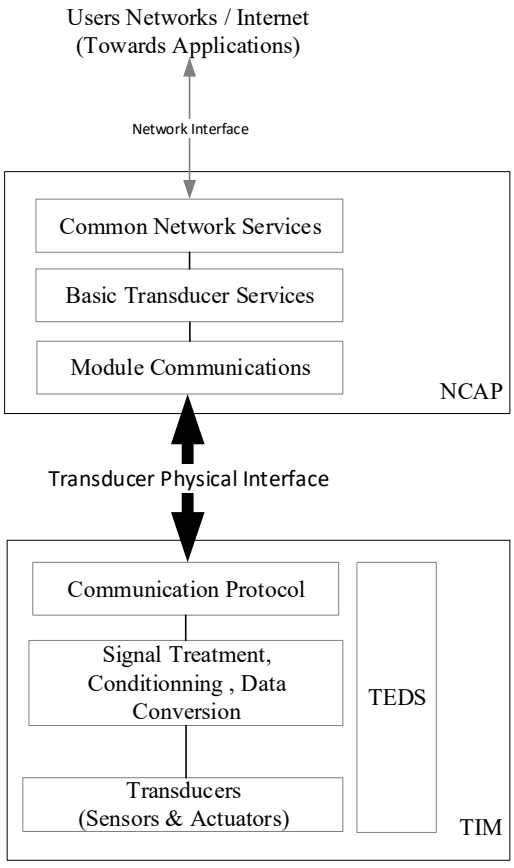

Fig. 1. Standardized Smart Transducer Scheme. Adapted from [5]

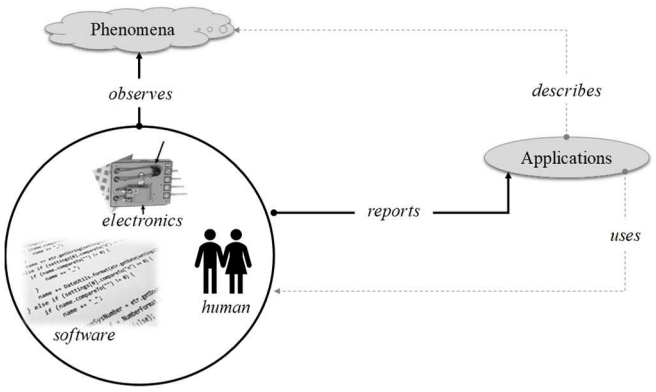

Fig. 2. Generic Sensing Loop in IoT.

expressed in the (electronic) messages sent by the house dwellers. To quantify each of these metrics, we will be using dedicated sensors to provide the inputs for the measurements of the temperature, comfort, and mood, respectively.

In a traditional approach, we would use an electronic-based sensor (EBS), e.g., a DHT11, connected to a network gateway to report the temperature values to the (smart) application. As for the collection of data that will provide information on perceived comfort, a computer program would be fed with measured values from a cloud of electronic devices, providing air temperature and relative humidity of the house. The comfort values would then be estimated upon a set of defined rules, according to both temperature and humidity [8]. The computer program that runs such a process is a software-based sensor (SBS) [9]. Lastly, we describe the approach that enables assessing the mood of the house dwellers. Here, from the application's perspective, humans play the role of sensing devices. Indeed, the text they post on OSNs can provide useful information to infer the overall ambience of the house. Using a suitable Application Programming Interface (API), the contents posted in OSNs and the associated metadata can be retrieved, and subsequently submitted to analytic techniques that produce scores for the target inference.

As we have seen in the provided example, the sensing activity may involve physical-electronic means as well as software-based and human-based sensors (HBS), to enable the representation of things that are part of the target scenario. We also highlight that the various types of sensors that make up the extended IoT environment are complementary, in the sense that they enrich the IoT data that will be used by applications to provide the agreed services.

C. Device Management in Today's IoT

Proper management of IoT systems is crucial to their operation, and this is even more so when dealing with the extended IoT because the heterogeneity of the involved sensors adds more complexity to the management platforms. Therefore, it is essential to have frameworks and architectures for enabling the abstraction of both technologies and protocols from different components of the IoT communication systems.

The authors in [10] addressed the complexity of jointly utilizing information on smart things, people, and the place where they socially interact. With a platform named Lilliput, the authors came up with a model for seamless integration of traditional sensing/actuating objects and OSNs, towards IoT social networks. They also provided a solution for the Access Control issue, enabling a trustful interaction between the "members" of IoT social networks. Finally, they tested the feasibility of their solution in a case study scenario named Sorcerer's Book, proving the feasibility of the proposed platform from three different perspectives, including the complexity alleviation for application developers who do not have to worry about IoT or OSN knowledge. The Lilliput architecture covers the monitoring, management, and security services involved between external components (covering both physical and online spaces) and end-user applications providing simple RESTful APIs for the developers. Although the study builds a real solution related to unified management of smart things and people, it does not focus on different IoT platforms. We believe that many capabilities described in the Lilliput Architecture can be directly implemented with the enablers from open and standard-based IoT platforms.

We have recently surveyed the literature on IoT management protocols and frameworks [11]. We showed that the leading standardization institutions work to ensure the integration, interoperation, and management of IoT systems, despite their growing complexity. Among them, the ITU-T defines common requirements and capabilities for IoT device management in its *Recommendation Y.4702*, oneM2M proposes a management architecture in its *Technical Specification OneM2M-TS-001-v3.11.0*, the Open Connectivity Foundation (OCF) defined a component for IoT management and control, named the *IoTivity*, and finally, the European Commission within the Seventh Framework Programme (FP7) fostered the development of *FIWARE middleware*. Among the available frameworks, we chose to adopt FIWARE because it offers the flexibility and scalability that we need for the different studies we exploit in our research group. Besides, it is a top reference IoT middleware [3].

The frameworks and architectures found in the literature offer open interfaces that support some management protocols, with a growing trend to use the protocol developed by the Open Mobile Alliance (OMA), named Lightweight Machine-to-Machine (LwM2M). LwM2M intends to provide a single, secure protocol for controlling and managing IoT devices and applications. This means that an IoT device implements and uses the same agent function for both the sensing/actuating purposes and management of the IoT device itself. There are numerous libraries, several products, and broad community support for LwM2M. The IoT management solutions from top leaders in the Information Technology market, such as Amazon Web Services, IBM Watson, and Google Cloud Platform, are developed for LwM2M. For the future, the new CoAP Management Interface (COMI), developed by the IETF, seems to be the most promising management protocol in IoT LwM2M. COMI will also include LwM2M as part of its IoT resources management solutions.

Management capabilities are a cross-layer component associated with the four layers of the ITU-T IoT reference model [12]. In each layer, IoT management covers fault,

configuration, accounting, performance and security management capabilities. In its *Recommendation Y.4702 (03/2016)*, the ITU-T defined two categories known as essential and specific management capabilities. Essential capabilities include device management, such as remote device activation and de-activation, diagnostics, firmware and software updating, device working status management, and traffic and congestion management. On the other hand, specific management capabilities are those tightly coupled with each application-specific requirements.

To the best of oud knowledge, many proposals addressing IoT management issues are focused on well-established physical sensing only [11]. Nevertheless, in many extended IoT applications, one may want to change the algorithm of a virtual sensor on-the-fly or prevent a social sensor from feeding the IoT system for a while. This gap in the literature motivated our proposal of a solution to manage different types of sensing/actuating devices and analyze which management protocol would be better suited in terms of performance. Our unified management approach will be described in the following section.

III. Unified Management Approach

This section presents the proposed unified IoT management approach. We start by providing a conceptual, high-level view and, then, proceed to detail its underlying technological components. Finally, we present the use-case scenario that we have developed to assess our proposal.

A. Conceptual view

Fig. 3 identifies the main building blocks of the proposed approach. The IoT middleware is the central component, enabling the abstraction of both technologies and protocols used in the various modules of the communication architecture. This component provides flexibility and scalability to the architecture. The combination of the flexibility of middleware with the robustness of well-established management standards is fundamental to a unified solution for both data and device management.

The Management Gateway deals with communication with the various sensing and actuation units for management purposes. As referred in Section II.A, during the description of the TIM, the Management Gateway is typically to reside in smartphones or stationary computing hosts, such as Raspberry Pi or Arduino-like devices. The IoT Managed Entities in Fig 3 comprise all sorts of heterogeneous elements with sensing or actuation capabilities, in the extended IoT environment, encompassing physical, virtual and social sensors/actuators.

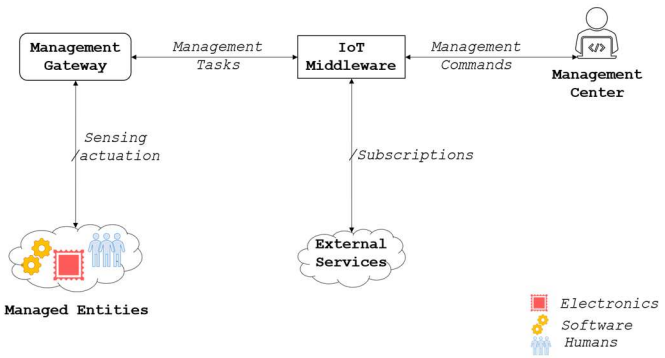

Fig. 3. Generic architecture for a unified management

The External Services component includes all the remote entities that can endow the unified management system with additional capabilities, such as natural language processing capabilities, machine learning, and other types of knowledge extraction functionality. Finally, the Management Center in Fig 3 is responsible for monitoring and control of the heterogeneous IoT environments.

In the proposed unified management approach, displayed in Fig. 3, FIWARE plays the role of the IoT middleware component. The documentation section available at http://ieee-dataport.org/2459 gives comprehensive insights on the FIWARE platform that we configured for the use-case scenario. Having designed the main conceptual blocks of the proposed approach, in the next subsection, we will present the use-case scenario that we have developed, named ISABELA (for IoT Student Advisor and Best Lifestyle Analyzer), to prototype and assess the unified management proposal.

B. The ISABELA Use-case

Initially, the aim of ISABELA was to explore the use of heterogeneous sensing approaches to infer the impact of the students' lifestyle on their academic performance [13]. ISABELA provides the students with recommendations for a set of good practices towards their academic achievement of better outcomes. Later, ISABELA also became a use-case scenario of the heterogeneous IoT management paradigm that we presented in the previous subsection.

Besides the backend, ISABELA has two main components. First, an Android mobile application that serves as an interface between the end-users (students) and the IoT resource pool, including a chatbot. Moreover, the application also serves to retrieve a set of EBS, SBS and HBS data, namely location, gyroscope, proximity, light, activity recognition, screen lock, phone calls and SMS statistics, alarm clock, and objects provided by Social Sensors. Second, we have developed from scratch an IoT set called ISABELA box to monitor phenomena in stationary delimited environments, namely temperature, humidity, noise and luminosity. All sensor data collected in ISABELA use-case are used towards the inference of the impact of the students' lifestyle on their academic performance.

Having designed the unified management solution, we set up a testbed in order to assess how it would behave under a variety of circumstances. We were specifically interested in triggering some essential management capabilities and in assessing the differences when the management commands were performed over LwM2M and Ultralight protocols.

IV. Configurations and Setups

This section provides information on the main configurations made to prototype and assess the unified management proposal, and to analyze which management protocol would be the best one to use in terms of performance.

A. Main Configuration and Code Adaptations

Although [14] is a recognized implementation of LwM2M, the software for communication with the FIWARE server had to be modified to make it work properly. This subsection presents the main modifications made to the configuration of both the clients and the server to overcome several incompatibility issues.

Mobile host LwM2M client

The configuration of the Android clients consisted firstly in creating a data model and the associated files for each object we wanted to manage. Then, we added new methods in the *MainActivity.java* file. Every time the LwM2M client starts, the pre-provision of N devices is performed in the agent (Create operation). The logout from the mobile application triggers the deregistration and the erasure of the managed object in the agent (Delete operation).

We used IPSO numbers both for the managed *ObjectId* and for the associated *ReusableResourcesId*, according to the OMA specifications in [15]. For the cases where we could not find the *ReusableResourcesId* for the attribute of an object, we overrode the one proposed by the OMA, selecting the most facsimile number possible from the not yet registered numbers, within the institution's reusable resources range (2048 - 26240) or from the private resources range (26241 – 32768). For instance, the *"Sample Frequency Value" 26040,* was inspired from IPSO 6040 *"Sample Frequency". "Sample Frequency"* is defined as "How often, in seconds, the inputs are read/sampled". We admit that in the physics domain, the units for "Period" would have been second^{-1}. However, we thought it was more convenient to be aligned to an already conventional entity instead of creating another IPSO object such as the "*Sample Period"* or "*Sampling Period"*. To summarize, in our study, the "lazy" attribute labelled *"Sample Frequency"* is to be interpreted as the one that drives the throughput of the device's activities, i.e., data acquisition rate and the actuation minimal intervals.

Stationary host LwM2M client

When the stationary host is switched on, a script launches our tailored NodeJS-based client from [16]. The *lwm2mclient.js* creates the managed objects, connects them to the agent and finally runs the python script that drives the Arduinos sensors in the ISABELA box, according to a *setInterval() scheduler ().* A function in *iotagent-lwm2m-client.js, childProcess.exec()*, fetches the *stdout* result from the Arduino sensor. E.g., the following code excerpt will enable change updates in the temperature sensor values provided by the Adafruit-based python script:

lwm2mClient.registry.setResource('/3303/0', 5700, stdout.trim(), handleObjectFunction).

Dedicated LwM2M NodeJS Server

Despite both the Android client and the dedicated LwM2M-iotagent being compliant with the same protocol, namely LwM2M version 1.0, the latter did not correctly decode the received values. The client sent the payload in TLV format, but the read values were in plain text. Concerning the numbers, for instance, we received values in ASCII instead of float or double, as sent by the client. Since the imported Leshan files in the Android clients are protected codes [14], we had to adapt the server from [17]. Thus, we inserted another layer of conversion in *lwm2m-node-lib/.../InformationReporting.js/ dataHandler: function (chunk) {}*, to overcome the received numerical values format (Float, Double). We also added reconnection patches in both mobile and stationary hosted LwM2M clients. Sometimes the former elements disconnect from the server and do not reconnect by themselves. Eventually, we created an image of the LwM2M server, with the following main patches:

- In commonLwm2m.js, a *trim()* function to delete spaces was added by the Android LwM2M client at the beginning of the URI objects;
- In informationRepport.js, we call a config.js file that accepts text/plain in *create.observe()* function;
- In deviceManagement.js, we also call a config.js file that accepts text/plain *in createrequest()* function.

The payload provisioned to the LwM2M agent is standardized according to what is known as a device model. In the next subsection, we will give more details on the configured LwM2M Device Models.

B. LwM2M Device Models

The device model in Table I is used to configure the Leshan *.xml* files embedded in the LwM2M mobile hosted client. Such files contain the resources, their IPSO number, some metadata, and their access type. As for the LwM2M stationary hosted client, the device model also structures the entities to be provisioned in the IoT agent. Typically, all (Active) attributes are registered with a Read-only in the .xml file, while the Lazy attributes are of Read-Write types.

Table 1(a) presents the generic device model in JSON format that we provided in the IoT agent. For the developed use-case, we did not make a difference between a managed device and its corresponding entity. Hence, we have the same value to both the *"device_id"* and *"entity_name"*. The (Active) attributes represent both the state and the activity of the managed devices. The first three elements of the *attributes* in Table 1(a) are based on the TEDS. As a consequence, the attribute *"Description"* contains information that describes the device.

We did not adopt the IPSO resource named *"Application Type"* (number 5750) [15], because it is a Read-Write attribute while, in our case, we wanted a Read-only attribute. The attribute *"DateTime Stamp"* adds contextualization to the information reported by the managed entities. Both "*Sample Frequency Value*" and *"Blocking Status"* attributes enable the tracing of the values reported in "*Sample Frequency Value*" and the command *"Set Block Status"* Lazy attributes, respectively. As for the Lazy attributes, we implemented one element to enable the change of the data acquisition rate of the sensors on-the-fly. This is labelled *"Sample Frequency"* (see Table 1(a)) corresponding to the IPSO *"*6040*"* defined by the OMA. The Commands attributes in our use-case comprise a single element to enable blocking and unblocking the activity of the managed

TABLE I
LWM2M DEVICE MODELS

(a) Generic

```
"devices": [ {
   "device_id": "IPSOObjectName_ISABELAUserId",
   "entity_name": "IPSOObjectName_ISABELAUserId",
   "entity_type": "Unifiedevice",

   "attributes": [
       Manufacturer, Model Number, Min & Max Range
       Values, Sensor Value, Sensor Units,
       Description, Sample Frequency Value,
       DateTime Stamp, Blocking Status
       ],

   "lazy": [Sample Frequency],

   "Commands": [Set Blocking Status],

   "internal_attributes": {
   "lwm2mResourceMapping": {
   "attributeNamex": {
       "objectType": IPSOObjectId,
       "objectInstance": 0,
       "objectResource": IPSOReusableResourceIdx
       },  …}}}}]
```

(b) Software-based Sensors

```
…
   "attributes": [
       Min & Max Range Values, Sensor Value ,
       Sensor Units, Description, Sample Frequency
       Value, Parameters, Sensor Algorithm,
       DataTime Stamp, Blocking Status
       ],

   "lazy": [Sample Frequency,
       Set Parameters, Set Algorithm],
…
```

(c) Human-based Sensors

```
…
    "attributes": [Language, Post Message,
                   Post Id, Description,
                   Sample Frequency Value,
                   DateTime Stamp],
…
```

(d) Actuators

```
…
    "attributes": [
          description, Parameters, Reply Text,
          DataTime Stamp, Blocking Status
          ],

    "lazy": [Set Parameters],

    "Commands": [Set Blocking Status],
…
```

devices. This provides the IoT system with the capability of preventing a sensor from sending its readings to the backend.

Based on the structure in Table 1(a), Table 1(b) is the generic device model for SBS. Thus, Table 1(b) only displays the elements that are particular to SBS, when compared to Table 1(a). In SBS, it is relevant to track and configure the inputs used to compute the virtual values. Besides, it is crucial to be able to configure the formula/algorithm used to compute the indirect values. For instance, in ISABELA, *Sociability* is an SBS that can be calculated in one of several possible ways, depending on the algorithm we adopt [13].

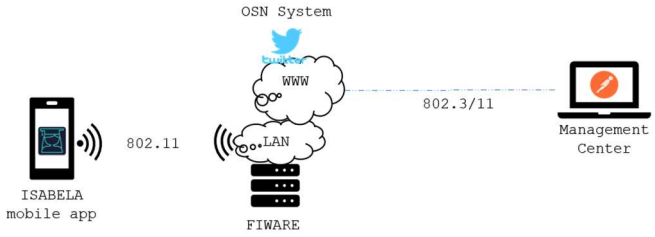

Fig. 4. Setup for management functionalites. *Adapted from* [19]

Also based on the structure from Table 1(a), Table 1(c) is tailored for HBS. The Active Attributes in Table 1(c) comprise a *"Post_Id"* to identify the latest retrieved tweet object in the Broker, before launching a new query to the OSN.

For instance, the Twitter SDK requires a *"Post_Id"* from where it will retrieve the objects in the *home_timeline*. If the *"Post_Id"* is not given, the SDK will retrieve the latest maximum number of objects, without any reference. Hence, there will be a risk to send duplicated objects to the Broker.

Finally, Table 1(d) shows the data model for a generic actuator, an addressable text, that we implemented in the use-case. The Lazy attribute labelled *"Set Parameters"* is fed by a string value from the Management Center. In ISABELA, this attribute is meant to be updated by a teacher who will send personalized recommendations to a student or a group.

The recommendations in ISABELA mobile application are received in the form of notifications and via a chatbot. On their turn, students can reply to the teacher, by witting a text directly in the chatbot textbox. This reply text feeds an attribute with the same name. Since the *"Reply Text"* is configured as an active attribute, it is observed by the agent, so that whenever its value changes, it updates in ORION. The *"Set Parameters"* attribute Table 1(d) is designed to contain values for the configuration of actuation tasks.

C. Testbed Setups

Fig. 4 is the layout set, in-lab, for testing the functionalities of unified management in the ISABELA use-case. It is an implementation of the Fig. 3, for a mobile scenario only, because in this scenario we could perform as many functionalities as possible for the unified management namely, the variation of the sensor's data acquisition rate, modification of addressable parameters for Software-based entities, and preventing/enabling a managed sensors/actuators to send the collected data to the backend.

An EBS is generally hard-programmed to be compliant with a specific management protocol and its IoT agent in the middleware. However, some IoT devices may support different communication protocols. In this context, it is essential to study the impact of using different protocols and analyze which aspects are to be considered when choosing the one to use. Specifically, we wanted to assess the communication

TABLE II
MANAGEMENT TECHNOLOGY STACKS

Standards and Protocols		Communication Layers	Management Layers
Management capabilities		--	Software App
IPSO Objects	*Simplified JSON*	--	Data Models
LwM2M 1.0	*Ultralight 2.0*	--	API and Services
CoAP	*HTTP*	Application	--
UDP	*TCP*	Transport	--
IPV4 or IPV6		Network	--
IEEE802.3 or IEEE802.11		Data Link & Phy	--

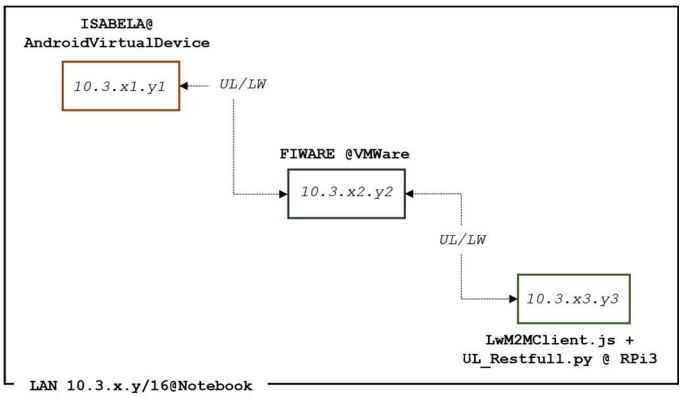

Fig. 5. Setup for the assessment of Ultralight (UL) versus LwM2M (LW)

performances of the LwM2M protocol, by comparing it to the transmission over Ultralight in FIWARE. We worked with Ultralight 2.0 and LwM2M 1.0 because they were the latest stable available versions implemented for both Android and Ubuntu OSes. Table II provides a general view of the various technologies and communication protocols on which LwM2M and Ultralight rely. It is important to mention that Ultralight is a version of a popular IoT standard developed by the Open Geospatial Consortium named Sensor Model Language (SensorML) [6], using an extremely simplified JSON codification for the payload transmitted over HTTP.

To analyze which management protocol would be the best one to use in terms of performances, we mounted the setup depicted in Fig. 5, and captured the outbound and inbound packets from IoT devices and the FIWARE server, respectively.

The whole testbed schematized in Fig. 5 ran inside a personal computer so that we could control the network conditions, including both the QoS and synchronization aspects.

The testbed was composed of three modules, namely a mobile phone, a stationary device, and a server. The mobile phone runs the ISABELA Android application, which implements the mobile client module. The Android application imports the LwM2M library from [14].

Since Ultralight is a simplified JSON codification for the payload transmitted over HTTP, its corresponding entity in the mobile application is developed with native libraries from Android SDK. The stationary device is a Raspberry Pi 3 which runs both Ultralight and LwM2M clients. The former client module also includes a python script that creates and sends a series of sensor values to the Ultralight agent in the server. The latter client of the testbed is an adaptation of the NodeJS-based client from *Telefónica I+D* [17]. Finally, the server module runs a cloud-based Virtual Machine where we installed a docker environment, a suite of containers that implement the Generic Enablers (GEs).

Because resource management systems should be able to dynamically adapt their operation in order to minimize latency and maximize throughput [18], we collected data pertaining to bandwidth occupancy, and delay. We added a third relevant indicator for assessing the communication performance, which is the packet loss.

The difference between the time at which a packet is captured in ORION and the packet timestamp at the source corresponded to the end-to-end delay. The network delay has been calculated as the difference between the time at which a packet arrived at the server interface and the timestamp at which the packet had left the client interface. The used bandwidth for each scenario includes all types of packets generated during the tests, such as *connection establishment packets*, *acks*, *retransmissions* and *posts*. Finally, the number of received packets in the Broker divided by the number of packets sent by the clients provided the packet delivery ratio, from which we could quickly determine the packet loss ratio.

We sent the traffic from clients to the server according to four scenarios, namely:

- Mobile Client to Server over Ultralight;
- Mobile Client to Server over LwM2M;
- Stationary Client to Server over LwM2M;
- Stationary Client to Server over Ultralight.

In each of these scenarios, each client sent 250 sensor readings to the server, using a given *"Sample Frequency"*. *"Sample Frequencies"* ranged from 1 second to 10 seconds. The datasets from testbed are available at http://ieee-dataport.org/2459. The root of the dataset comprises a .xlsx file, where we put the registered downtimes for the LwM2M clients in both mobile and stationary scenarios. We used Tshark (a Linux command-based, network packet analyzer) to capture the packets and generate such registries.

V. TEST RESULTS

In this section, we present and discuss the results concerning both the management functionalities and performances of the management services in FIWARE. We remind that for the latter, we were especially interested in analyzing the performance of LwM2M and Ultralight, in order to identify the circumstances under which each of these protocols can lead to advantages or disadvantages when supporting a solution for the unified management of heterogeneous environments.

A. Management Functionalities

Data Acquisition Rate

While we changed the *"Sample Frequency"* values, we registered Short-term History data in the FIWARE COMET GE. As a result, Fig. 6 depicts the effective variation of the data acquisition rate following the values of the *"Sample Frequency"* attribute. For the setup in Fig. 6, the sensors data embedded the epoch timestamp of when they were collected. Hence, we could calculate the delta timestamp between one sensor reading and its predecessor. The blue/white-dotted graph in Fig. 6 represents these delta values in seconds. We had also recorded the timestamps when a new *"Sample Frequency"* was set at ISABELA mobile application. As a result, we could fill the dataset represented by the dark-dotted graph in Fig. 6.

Addressable Text and Customizable Colour

We have implemented the addressable text entity according to the device model in Table I(d). The resulted management functionality, displayed in Fig. 7, shows how teachers can send a non-automatic and personalized recommendation to the students via the ISABELA platform.

In the left-up side of Fig. 7, we can see that the value for the *"Set_parameters"* is: "It works_yellow_advice". Such content is divided into three parts split by an underscore. The first part is the recommendation content, while the second part sets the colour of the background in the notification icon.

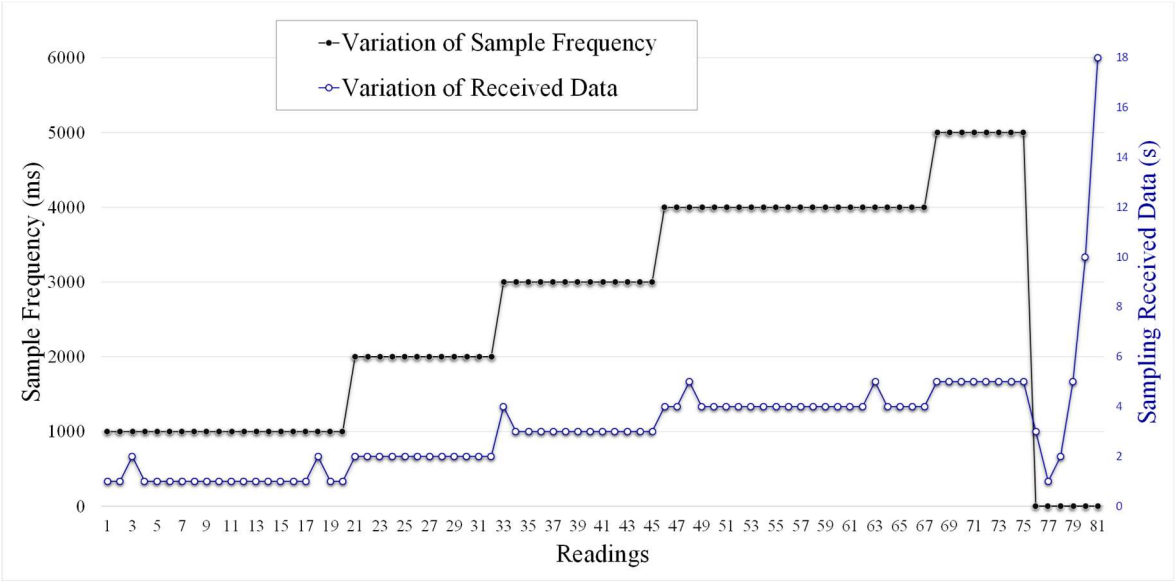

Fig. 6. Management of data acquisition rate

```
Fiware Orion Broker Query:
10.3.4.109:9001/v1/updateContext

Body:
{    "contextElements": [{
            "type": "mobile_xbx",
            "isPattern": "false",
            "id":
"addressabletext_XJ22P8qmbC01RnOrQK6K8Q",
            "attributes": [{"name": "set_parameters",
            "type": "string", "value": "It
works_yellow_advice"

}]}],"updateAction": "UPDATE"}
```

```
Fiware Orion Broker Query:
10.3.4.109:9001/v2/entities

Response:
 "datetime_stamp": {
            "type": "String",
            "value": "2019-11-13T19:28:28Z", "metadata":
{}},
…
 "parameters": {
            "type": "string",
            "value": "It works_yellow_advice", "metadata":
{}},
  "reply_text": {
            "type": "string",
            "value": "Ola","metadata": {}},
…
```

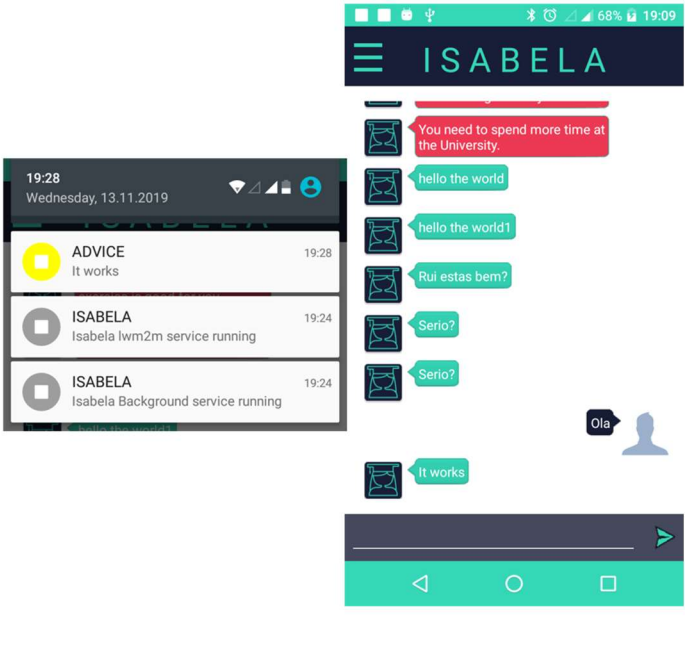

Fig. 7. Management of addressable text and displayed colours

The third part is tagged advice that is leveraged to differentiate it from the *"Blocking Status"* notification type. Finally, we have configured the Android notification slider so that when the users click on it, they are directed to the chatbot from where he can reply. The reply content feeds the *"reply_text"* attribute. Hence, its content updates the corresponding value in the Broker. The content of the *"reply_text"* in Fig. 7, i.e., "*Ola*", appears both in the chatbot and in the ORION context broker. Through the example in Fig. 7, we can see how the implementation of unified management can also serve as an instant messaging system.

Blacklisting

In [19], we showed that Human-based sensing was successfully integrated into the ISABELA system. Posts from the OSNs were collected and subsequently stored in the backend for further processing. By toggling the command *"Set Block Status"* described in IV.B, we change a Boolean variable in ISABELA app, that is configured to run or not the Android service that drives the managed sensors/actuator. This is how we can blacklist a "Social Sensor". Even if the work in [19] is focused on HBS, the blocking status works for any managed sensors/actuator.

B. LwM2M versus Ultralight

Packet Loss

Fig. 8 presents the packet loss values for stationary clients (Fig. 8a) and mobile clients (Fig. 8b) when using Ultralight or LwM2M. One crucial observation is that LwM2M leads to a more reduced packet delivery ratio (i.e., higher packet loss ratio) for both types of clients, namely stationary clients and mobile clients. In fact, in many cases, Ultralight led to 0% packet losses, while in the case of LwM2M losses as high as 32% were registered for mobile clients.

In the case of stationary clients, the results in Fig. 8(a) globally show low packet losses for LwM2M, starting at 0.8% and going up to 3.6%. We recall that the LwM2M client for the stationary scenario was developed by the teams in charge of the FIWARE server. At first sight, this low packet loss could make us conclude for relatively good client-server compatibility. Nevertheless, on closer look, we noticed that, contrary to what would be natural to expect, packet losses were higher when the period between samples was longer, i.e., for lower sampling rates. After careful analysis, such behaviour turned out to be related to client implementation problems that led to downtimes whenever the LwM2M client had to run for long periods of time. The solution for this was to re-launch the client whenever it crashed, at the expense of higher packet losses.

On the other hand, the mobile client LwM2M implementation had no such problems. Nevertheless, it was not problem-free, as it leads to heavy packet losses whenever the *"Sample Frequency"* is short, i.e., whenever the load is high, as can be seen in Fig. 8(b). We recall that the LwM2M implementation for the mobile scenario was not developed by the teams in charge of FIWARE, and this probably explains a lower performance when compared to the stationary scenario.

Bandwidth Occupancy

Fig. 9 presents the bandwidth occupancy values for stationary clients (Fig. 9a) and mobile clients (Fig. 9b) when using Ultralight or LwM2M. In both figures, we can see that bandwidth requirements mostly remain stable over the various *"Sample Frequencies"*, which was to be expected, as the number of total samples from the client to the server remains the same (i.e., 250 samples). The most notable fact in Fig. 9 is that the bandwidth required by Ultralight is much higher than the bandwidth required by LwM2M, regardless of the type of client. Moreover, the difference is higher in the case of mobile clients. The higher bandwidth occupancy of Ultralight is probably because this protocol runs over HTTP and TCP, as opposed to LwM2M, which runs over CoAP and UDP. Nevertheless, both the mobile and stationary client implementations are not optimized in terms of bandwidth.

Delay

Table III shows the differences in network delay and end-to-end delay when dispatching the packets via Ultralight and LwM2M in FIWARE. So, for instance, positive values mean that Ultralight has a higher delay than LwM2M.

Some conclusions can be drawn out of Table III. The first conclusion is that, in most cases, LwM2M leads to lower delays when compared to Ultralight. Nevertheless, there are cases in the stationary scenario for which Ultralight leads to better network delays. The second conclusion is that in the case of the stationary client's scenario, the differences in network delay and the end-to-end delay between Ultralight and LwM2M are tiny, in the order of few milliseconds. Finally, the third conclusion is that the processing load in mobile clients for the case of Ultralight is quite large; the difference to LwM2M being in the order of 700-800 milliseconds. Clearly, as in the case of bandwidth occupancy, the Ultralight implementation is not optimized in terms of delay.

VI. CONCLUSION

In this paper, we explored an approach to the unified management of IoT environments comprising not only physical/electronic IoT devices but also all types of sensing devices, such as virtual and/or software-based devices or human-based devices that build on data extracted from online social networks data. This mix of heterogeneous sensing approaches is increasingly being adopted on the Internet, and tools for the effective monitoring and control of such 'devices' are needed, as in any management system. Such tools may allow deciding on which 'sensors' to use or which data to extract/collect for a given IoT application, and which information to feedback into the users for their benefit.

The approach presented in this paper was prototyped using open, widely available protocols and platforms. We set up a testbed to assess the impact of two different management protocols on its performance. Due to their importance and widespread use in IoT management and the FIWARE middleware, we focused our attention on the comparison of LwM2M and Ultralight, for mobile and stationary scenarios.

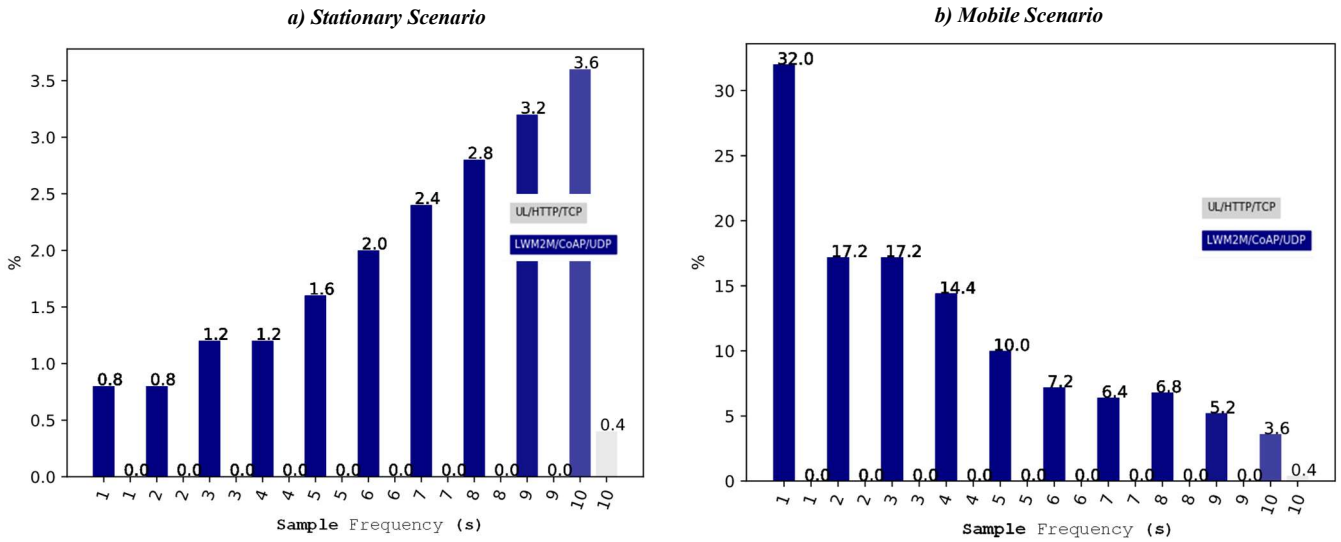

Fig. 8 Packet loss

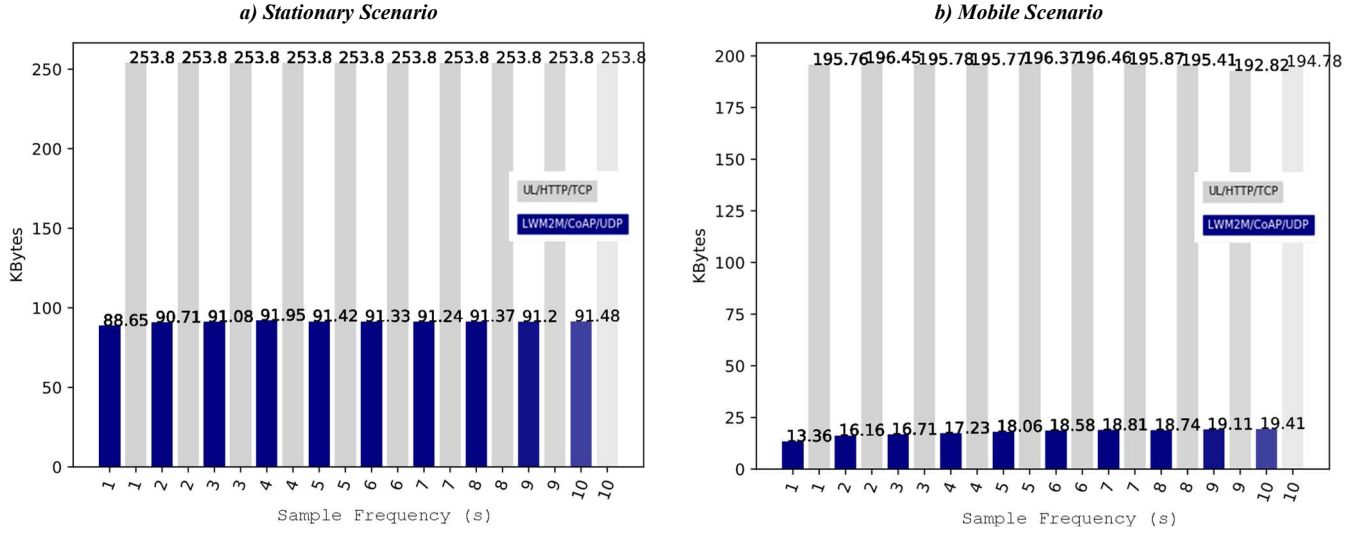

Fig. 9. Bandwidth occupancy

The evaluation results pointed to some limitations/problems of the existing implementations, namely in what concerns packet losses, bandwidth requirements, and delay. Ultralight packet losses are shallow for both stationary and mobile environments. Nevertheless, its performance in terms of required bandwidth is quite weak as compared to LwM2M.

As for the network delay, both protocols are quite similar, whereas Ultralight implementation for mobile clients requires high processing delay. As each management protocol has advantages and drawbacks, we can envisage solutions that dynamically select the best protocol depending on both the expected performance and the current conditions offered by the network.

For future work, we plan to deploy the whole testbed in a distributed scenario to study the variations of the various metrics further. Moreover, we will extend the functionality of the unified IoT management solution, especially addressing human-based sensing and human-in-the-loop concepts.

VII. REFERENCES

[1] N. Armando, A. Rodrigues, V. Pereira, J. S. Silva, and F. Boavida, "An Outlook on Physical and Virtual Sensors for a Socially Interactive Internet," *Sensors*, vol. 18, no. 8, p. 2578, Aug. 2018.

[2] ITU-T Study Group 20, *Common requirements and capabilities of device management in the Internet of things*. Geneva, Switzerland, 2016, p. 22.

[3] Pierre Audoin Consultants (PAC) GmbH, "FIWARE | IoT Platform Survey | IoT Survey," *PAC*, 2019. [Online]. Available: https://www.iot-survey.com/fiware. [Accessed: 13-Mar-2019].

[4] A. Kumar, V. Srivastava, M. K. Singh, and G. P. Hancke, "Current Status of the IEEE 1451 Standard-Based Sensor Applications," *IEEE Sens. J.*, vol. 15, no. 5, pp. 2505–2513, May 2015.

[5] K. Lee, *IEEE 1451 and Wireless Sensor Standard*. USA: https://www.nist.gov/sites/default/files/documents/el/isd/ieee/Information-on-1451_1588-V36.pdf, 2002, pp. 1–6.

TABLE III*
REGISTERED DELAYS

(A) ABSOLUTE VALUES IN STATIONARY SCENARIO

"Sample Frequency" (in seconds)	Network						End-to-End					
	Ultralight			LwM2M			Ultralight			LwM2M		
	Mean	**Median**	**Mode**	**Mean**	**Median**	**Mode**	**Mean**	**Median**	**Mode**	**Mean**	**Median**	**Mode**
1	**4.36**	4.34	4.05	**5.54**	5.51	5.48	47.28	44.69	17.07	37.4	34.44	14.16
2	4.86	4.8	4.53	6.28	6.23	6.23	42.67	42.26	15.79	33.91	32.79	14.47
3	4.09	4.04	4.21	5.85	5.84	5.86	43.14	41.81	41.07	32.83	31.53	14.1
4	4.33	4.27	4.06	5.48	5.45	5.47	43.41	41.01	15.32	34.51	32.49	13.88
5	3.91	3.82	3.73	4.55	4.54	4.56	44.67	43.18	45.3	35.03	33.35	15.58
6	3.19	3.08	2.9	0.52	0.33	0.38	43.37	41.8	27.63	33.8	31.08	9.42
7	2.99	2.9	2.5	2.04	2.03	1.98	44.27	41.51	15.22	30.55	29.16	11.37
8	1.89	1.8	1.78	1.88	1.87	1.9	41.02	39.9	16.48	32.55	31.59	9.33
9	1.58	1.45	1.47	0.85	0.77	0.75	44.16	42.29	12.69	34.54	28.8	9.68
10	1.02	0.97	0.97	0.45	0.39	0.36	54.54	47.13	12.12	31.18	31.08	10.71
(B) ABSOLUTE VALUES IN MOBILE SCENARIO												
1	8.69	8.57	8.35	4.91	4.28	4.22	1500.21	1491.15	1491.34	700.83	684.01	660.57
2	8.31	7.97	7.77	4.3	4.23	4.21	1502.46	1497.44	1454.09	713.38	683.78	660.75
3	8.24	7.99	7.82	4.21	3.94	3.84	1495.78	1490.44	1452.04	728.43	691.24	660.6
4	8.21	7.83	7.72	3.31	3.21	3.24	1493.7	1488.33	1454.82	758.82	684.45	661.48
5	8.8	7.9	7.89	3.42	3.24	3.11	1505.27	1491.07	1455.2	759.56	686.77	659.91
6	7.33	7.08	6.88	3.39	3.3	3.3	1517.67	1511.97	1452.03	720.91	683.93	661.78
7	6.89	5.31	5.32	3.24	2.98	3.02	1526.91	1486.42	1451.02	752.03	687.74	659.2
8	5.11	4.73	4.56	1.84	1.82	1.6	1504.73	1488.8	1449.1	761.23	681.85	660.61
9	4.19	4.11	4.08	1.49	1.45	1.37	1484.59	1479.65	1446.95	757.95	684.03	683.54
10	3.78	3.68	3.78	0.79	0.79	0.69	**1488.59**	1485.69	1447.18	**775.36**	682.95	659.29

(C) DELAY DIFFERENCES BETWEEN ULTRALIGHT AND LWM2M ABSOLUTE VALUES *(EXAMPLE: 4.36 -5.54 = -1.18,* 1488.59-775.36 = 713.23*)*

--	Stationary Scenario						Mobile Scenario					
	Network			End-to-End			Network			End-to-End		
	Mean	**Median**	**Mode**	**Mean**	**Median**	**Mode**	**Mean**	**Median**	**Mode**	**Mean**	**Median**	**Mode**
1	**-1.18**	-1.17	-1.43	9.88	10.25	2.91	3.78	4.29	4.13	799.38	807.14	830.77
2	-1.42	-1.43	-1.70	8.76	9.47	1.32	4.01	3.74	3.56	789.08	813.66	793.34
3	-1.76	-1.80	-1.65	10.31	10.28	26.97	4.03	4.05	3.98	767.35	799.20	791.44
4	-1.15	-1.18	-1.41	8.90	8.52	1.44	4.90	4.62	4.48	734.88	803.88	793.34
5	-0.64	-0.72	-0.83	9.64	9.83	29.72	5.38	4.66	4.78	745.71	804.30	795.29
6	2.67	2.75	2.52	9.57	10.72	18.21	3.94	3.78	3.58	796.76	828.04	790.25
7	0.95	0.87	0.52	13.72	12.35	3.85	3.65	2.33	2.30	774.88	798.68	791.82
8	0.01	-0.07	-0.12	8.47	8.31	7.15	3.27	2.91	2.96	743.50	806.95	788.49
9	0.73	0.68	0.72	9.62	13.49	3.01	2.70	2.66	2.71	726.64	795.62	763.41
10	0.57	0.58	0.61	23.36	16.05	1.41	2.99	2.89	3.09	**713.23**	802.74	787.89
(D) SUMMARISED VALUES												
Mean	-0.12	-0.15	-0.28	11.22	10.93	9.60	3.87	3.59	3.56	759.14	806.02	792.60
Median	-0.32	-0.40	-0.48	9.63	10.27	3.43	3.86	3.76	3.57	756.53	804.09	791.63
Mode	--	--	--	--	--	--	--	--	--	--	--	793.34

*Except for the *"Sample Frequency",* all values in Table III are in milliseconds

[6] D. W. Michael E. Botts, Alexandre Robin, Jim Greenwood, *OGC SensorML: Model and XML Encoding Standard*. http://www.opengeospatial.org/standards/sensorml, 2014, pp. 1–196.

[7] National Instruments, *An Overview of IEEE 1451.4 Transducer Electronic Data Sheets (TEDS)*. USA: https://standards.ieee.org/develop/regauth/tut/teds.pdf, 2006, pp. 1–19.

[8] S. Sharanya and S. John, "Comfort Sensor Using Fuzzy Logic and Arduino," in *Proceedings of 2nd International Conference on Intelligent Computing and Applications: ICICA 2015*, 2017, pp. 155–160.

[9] C. Sarkar, V. S. Rao, R. Venkatesha Prasad, S. N. Das, S. Misra, and A. Vasilakos, "VSF: An Energy-Efficient Sensing Framework Using Virtual Sensors," *IEEE Sens. J.*, vol. 16, no. 12, pp. 5046–5059, Jun. 2016.

[10] J. Byun, S. H. Kim, and D. Kim, "Lilliput: Ontology-based platform for IoT social networks: Towards socialized people, objects, and places," in *Proceedings - 2014 IEEE International Conference on Services Computing, SCC 2014*, 2014, pp. 139–146.

[11] S. Sinche *et al.*, "A Survey of IoT Management Protocols and Frameworks," *IEEE Commun. Surv. Tutorials*, vol. 22, no. 2, pp. 1168–1190, 2020.

[12] ITU-T Study Group 20, *Overview of the Internet of things*. Geneva, Switzerland: https://www.itu.int/rec/T-REC-Y.2060-201206-I, 2012, pp. 1–22.

[13] J. Fernandes *et al.*, "ISABELA – A Socially-Aware Human-in-the-Loop Advisor System," *Online Soc. Networks Media*, vol. 16, p. 100060, Mar. 2020.

[14] R. Haydarov, "LwM2M Demo Client Android," *Ericsson*, 2017. [Online]. Available: https://github.com/ApplicationPlatformForIoT/LwM2MDemoClientAndroid. [Accessed: 14-Mar-2019].

[15] Open Mobile Alliance, "LwM2M Registry and Resources." [Online]. Available: http://www.openmobilealliance.org/wp/OMNA/LwM2M/LwM2MRegistry.html. [Accessed: 15-Apr-2019].

[16] S. A. . Telefonica Investigación y Desarrollo, "IoT Agent Client." [Online]. Available: https://github.com/telefonicaid/lwm2m-node-lib/blob/master/bin/iotagent-lwm2m-client.js. [Accessed: 30-Jul-2019].

[17] Telefónica I+D, "lightweightm2m-iotagent." [Online]. Available: https://github.com/telefonicaid/lightweightm2m-iotagent.git. [Accessed: 30-Jul-2019].

[18] A. V. Dastjerdi and R. Buyya, "Fog Computing: Helping the Internet of Things Realize Its Potential," *Computer (Long. Beach. Calif).*, vol. 49, no. 8, pp. 112–116, Aug. 2016.

[19] N. Armando, J. M. Fernandes, A. Rodrigues, J. S. Silva, and F. Boavida, "Exploring Approaches to the Management of Physical, Virtual, and Social Sensors," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2020, pp. 954–959.

NGOMBO ARMANDO received the M.Eng. degree in information and communication technologies from IMT Lille-Douai, France, in 2010. He is currently a PhD candidate in information science and technology at the University of Coimbra, Department of Informatics Engineering (DEI), Coimbra, Portugal. He is a Member of IEEE. From 2010 to 2012, he was a telecoms consultant for Altran-T&M, Paris, France. Since 2013, he is a senior lecturer at Universidade Kimpa Vita, Escola Superior Politécnica do Uíge (ESPU), Uíge, Angola. At ESPU, he also heads the project management service in charge of both IT management and support, statistics and working good practices. Since 2018 he is a research fellow for 5G products at the Centre of Informatics Engineering of University of Coimbra (CISUC), Coimbra, Portugal. His research interests include IoT, Devices Management, 5G products and services. (https://www.cienciavitae.pt/en/5916-4A11-E165)

JORGE SÁ SILVA received his PhD in Informatics Engineering in 2001 from the University of Coimbra, where is Associate Professor with Habilitation at the Department of Electrical and Computer Engineering of Sciences and Technology of the University of Coimbra and a Senior Researcher of LCT, Portugal. His main research interests are IoT, Network Protocols, M2M, and WSNs. He has been serving as a reviewer and publishing in top conferences and journals in his expertise areas. His publications include 2 books, 5 book chapters and over 170 papers in refereed national and international conferences and magazines. He participated in European initiatives and projects such as FP5 E-NET, FP6 NoE E-NEXT, FP6 IP EuQoS, FP6 IP WEIRD and FP7 Ginseng (as Portuguese Leader). He actively participated in the organization of several international conferences and workshops, (e.g. he was the Workshop Chair of IFIP Networking2006, Publicity Chair of EWSN2009, General Co-Chair of EWSN2010, General Co-Chair of Mobiquitous2015, General Vice-Chair of WoWMoM2016) and he was also involved in program committees of national and international conferences. He is a senior a Member of IEEE, and he is a licensed Professional Engineer. (http://www.dei.uc.pt/sasilva)

FERNANDO BOAVIDA received his PhD in Informatics Engineering in 1990, and he currently is Full Professor at the DEI of the Faculty of Sciences and Technology of the University of Coimbra. His main research interests are people-centric Internet of Things, wireless sensor networks, mobility, and quality of service. He is author/co-author of more than 170 international publications (books, book chapters, refereed journals and conference proceedings) and 50 national publications. He was the chairman of the Program Committee of QofIS'2001, IDMS-PROMS'2002, NETWORKING 2006, WWIC 2007, FMN 2008, EWSN 2010, FMN 2012, IWQoS 2012, ACM SIGCOMM FhMN 2013, Mobiquitous 2015, and WoWMoM 2016 international conferences/ workshops. He is a senior Member of the IEEE and a licensed Professional Engineer. He is a member of the Editorial Advisory Board of the Computer Communications journal. (http://www.uc.pt/go/boavida)