Architecture of Standard-based, Interoperable and Extensible IoT Platform

Slavko Žitnik, Marko Janković, Klemen Petrovčič, Marko Bajec

***Abstract*—IoT Platform marketplace is gaining a lot of attention in many areas. The paper presents an interoperable and extensible IoT framework that follows oneM2M standard. The framework is validated by a reference implementation that includes automatic sensor recognition and pairing, NoSQL support, complex event processing and alarming and extensions for IoT devices communication over HTTP, WebSocket, CoAP, MQTT, Z-Wave, ZigBee and Bluetooth. The novelty includes the theoretical and practical solution to the proposed platform. Furthermore, we show the feasibility to build generally useful IoT platforms that are based on standards.**

***Index Terms*—IoT platform, Internet of things, OM2M, standardization, interoperability.**

I. Introduction

THE term Internet of Things (IoT) was first defined by Kevin Ashton in 1999. Since then, a variety and number of devices connected to the Internet has increased exponentially, especially in the recent years. We are witnessing an explosion of IoT deployments around the world, while IoT platforms are emerging as the main backbone of these deployments.

The IoT platforms are the key of development of IoT services that connect sensors or actuators, systems and people. The IoT Platform market represents a completely new segment that was non-existent a few years ago. Currently there are more than 300 IoT platforms available that offer different features. Normally they do not follow emerging IoT standards or are locked to IoT products of a specific vendor. It can be observed that the interoperability in this area is driven by the open-source community. We believe that the real value in the IoT World comes from data and connected services, which may globally happen only by following standards. In the history of the internetworking this has already happened - when network device vendors started to follow a common IEEE 802.3 standard and TCP/IP stack, networks quickly began to grow and connect.

Thus, in this paper we review IoT software platform landscape together with standards and protocols (Sec. II). We continue with the presentation of a general standard-based, interoperable and extensible framework, which is a theoretical design of an IoT platform. In Sec. IV we practically show feasibility of implementing and using such a platform, for which the implementation is available online. Lastly, in Sec. V we discuss the presented work and give some concluding remarks.

All authors are with the University of Ljubljana, Faculty for Computer and Information Science, SI-1000 Ljubljana (email: slavko.zitnik@fri.uni-lj.si).

The work has been supported by the Slovene Ministry of Education, Science and Sport of the Republic of Slovenia (EkoSMART program).

II. Related Work

A. Existing Platforms and Systems

The current IoT software platform landscape is enormous. All big vendors have already introduced their own platforms, for example Watson (IBM), HANA (SAP), Jasper (Cisco), AWS IoT (Amazon), Azure IoT (Microsoft), HomeKit (Apple), Brillo (Google), IoTivity (Intel) or AllJoyn (Qualcomm). Each of these platforms offers its own IoT ecosystem, which is hard to interconnect with others or integrate with additional unsupported devices. Therefore, in this paper we focus on non-commercial and open-source IoT platforms that support at least one of the key IoT protocols (see Sec. II-B). Still, there exist a few tens of such platforms and from them we shortlist only those that seem to be a part of an active community and are easy to install and extend. Also, the selected platform should include possibilities to interconnect to commercial platforms to further promote standards to the big vendors and remind them to start working together on at least a de-facto interconnection standard.

The platforms that are most suitable to our knowledge and defined constraints are the following: (1) Domoticz platform[1], which supports Z-Wave protocol, a number of RF-based devices, push notifications and definition of IFTTT rules. We can easily extend its functionality to connect to arbitrary sensors using custom scripts. (2) Platform Kaa[2] supports Bluetooth, ZigBee and Z-Wave protocol. It also comes with an SDK that allows for extending the platform to specific needs and with a tool for definition of a data schema for input data. Additionally it offers extensions for some popular analytics tools. (3) HomeAssistant[3] platform is mainly focused on pairing with commercial IoT products (such as Nest or Hue) with extensions for push notifications, rules definition or popular media software control. In addition to the use for non-technical users, it also provides an API for developers. (4) OpenHAB[4] leverages a large community with numerous extensions (i.e., enabling different protocols or integrations) that can be integrated into the platform. According to the previous platforms it is highly modular with OSGi. (5) OM2M [1] is also a highly modular IoT framework based on OSGi that

[1] https://domoticz.com/
[2] http://www.kaaproject.org/
[3] https://home-assistant.io/
[4] http://www.openhab.org/

by default supports CoAP and HTTP protocols only. Apart from all other platforms it is focused on standardization and implements oneM2M IoT standard [2]. The latter may be of the high importance for developing an IoT platform that is able to communicate with arbitrary devices and other IoT platforms using standardized messages.

A number of side-by-side IoT platform comparisons has already been conducted while focusing mainly on their functionalities separately. Consequently, many guidelines of how to build new IoT platforms have been proposed [3]. Therefore, it has already been identified that the key to a real IoT-centric world is the interoperability, which can be achieved by setting and obeying the standards [4], [5]. Analogously, for the development of computer networks, the standards have enabled the ease of communication among users from all over the World.

B. IoT Standards and Protocols

Currently, no universal IoT standard that industry would follow exists. Intel, GE and Qualcomm are playing a significant role in the IoT standards development. On the other hand, Industrial Internet Consortium attempts to accelerate best IoT practices instead of developing standards.

In the IoT standards space we can roughly consider four layers of standards. The first two are application and service layers that are important for developing IoT frameworks. The next layer is the network layer and finally access layer to support physical connectivity.

The main IoT standards initiatives are the following: (1) Thread, which is a wireless-centric standard that covers networking, power conservation, security and product compatibility. By default, each device would need to have an IPv6 address to ease the IoT networking issues. (2) AllJoyn is an open-source framework for connectivity and service layer operations in IoT. Its goal is "to create interoperable products that can discover, connect, and interact directly with other nearby devices, systems, and services regardless of transport layer, device type, platform, operating system, or brand." (3) IoTivity is Intel's alternative to Qualcomm's AllJoyn and another framework for device-to-device communications. (4) IEEE P2413 serves as an umbrella project for more than 300 IEEE standards that are applicable to IoT. Its goal is to build a reference architecture that "covers the definition of basic architectural building blocks and their ability to be integrated into multi-tiered systems". (5) oneM2M [6] is a standard whose main objective is to define a common service platform that can support various IoT application services. It defines IoT platform architectural blocks, standardized messages to provide interoperability and internetworking on multiple levels and includes Semantic Web schema definition for further automatic interconnection. It is very actively developed by multiple standard organization bodies TTA (Korea), ETSI (Europe), TIA, ATIS (USA), TTC, ARIB (Japan), CCSA (China) and India (TDSI). In the second release, new features for home domain enablement have been introduced and also internetworking functions for other technologies such as AllJoyn and LWM2M with semantic interoperability.

According to the current state in the IoT standardization we adopt the oneM2M standard as is already used in various companies and for deployment of large-scaled IoT projects, such as the Smart City in Busan. We first identify a number of features that an IoT platform must support to be applicable in arbitrary domain (i.e., health, traffic, smart city, smart home, smart industry) (Sec. III) and then implement those features according to the standard to show the usability in practice (Sec. IV).

III. Standard-based, Interoperable and Extensible IoT Framework

In this section we define the features for an interoperable and extensible IoT platform that follows a standard. From the standard point of view the important aspects are the IoT platform architecture and protocols (network and application) to enable seamless device-to-platform, inter-platform and intra-platform communications.

Fig. 1 shows a proposed IoT platform framework that extends the functionalities of the oneM2M standard-based platform - OM2M. The dotted line in the figure represents the existing implementation and the full line concepts represent the proposed extensions. The latest version of the OM2M platform includes device registration and discovery, device management, group management, security and notifications management. From the architecture point of view, OM2M also defines three types of entities - common service entity (CSE), application entity (AE) and network service entity. Each can be installed on multiple nodes and interconnected together into a platform, which is also defined by the standard. For simplicity of the visualization in the figure we show only an infrastructure node (IN, exactly one in the platform must exist) and a middle node (MN). The latter is used to connect sensors to the core of the platform. All of them could also be deployed on only one machine (e.g. smart home scenario) but generally sensors can be far away from the core (e.g. smart city).

We expect each sensor to be a small computer that can understand one of the network protocols and communications according to the oneM2M standard. Such logic of a sensor should be developed in an application entity that can connect to the platform. OM2M implements only HTTP standard to communicate with other sensors. It has already been proven that in larger environments and for sensors that generate time-critical data, it is useful to use WebSocket technology[7]. As it uses the same underlying HTTP protocol, the same type of messages can be used as are used for the RESTful API.

Next, the oneM2M standard defines the structure of conversations for MQTT and CoAP (first version of the standard), which also need to be supported. Apart from higher-level protocols, many IoT devices use protocols such as Z-Wave, ZigBee or Bluetooth (LE). As no oneM2M standard guidelines for them exists, we should look up to existing oneM2M standards and try to develop support for them following similar structure of conversations. Our focus is not a unified communication on physical level but a standardized definition of application-level messages for each physical protocol.

The standard already defines how to automatically attach a new sensor to a platform and that is why it is useful to

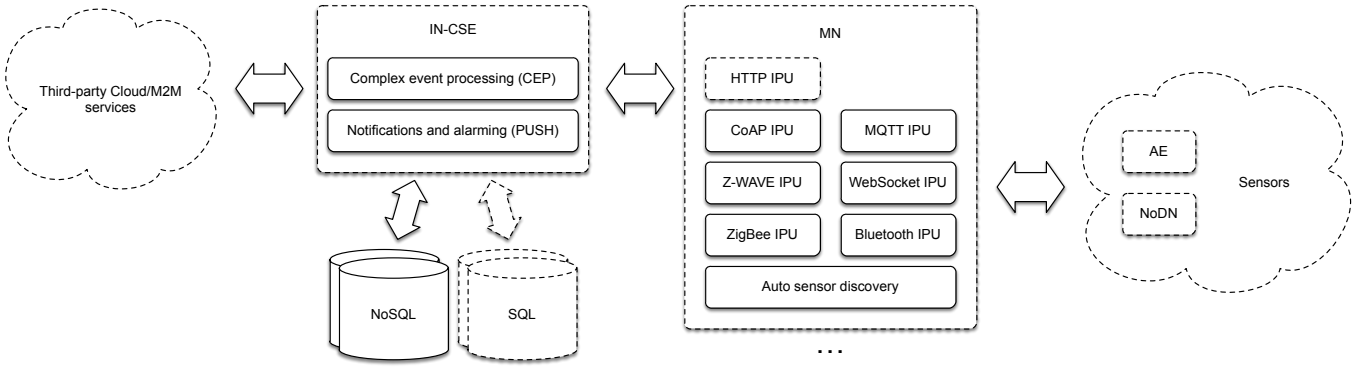

Fig. 1. Standard-based, interoperable and extensible IoT framework based on OM2M platform. The dotted line denotes the existing implementations while full line denotes our implementations.

implement automatic sensor discovery algorithm that would scan the area using all the supported protocols by the platform. When a new sensor is identified, it would be added to the platform using a standardized protocol.

A general IoT platform should support different types of data storage and database types. Moreover it should define an API to give a possibility to support arbitrary database, whether is it of relational or NoSQL type. The latter are appropriate for specific environments, such as with data intensive or highly time frequent sensors.

From the user perspective, a platform should support various notification or alarm mechanisms (e.g. PUSH, e-mail, SMS) and rules definition engine. Some smart home appliances support simple IFTTT rules, while the inclusion of more powerful complex event processing module would be more appropriate for a general use. Also, a platform should not be tightly coupled with a GUI but should implement services that would enable different UI implementations for arbitrary client devices.

The IoT platforms we have reviewed (some of them mentioned in Sec. II) define their own data and device representation. Therefore, a special mapping must be implemented for each interaction. The oneM2M already defines a standard for internetworking and already specifications to interconnect with OIC and AllJoyn platforms. In regards to enable compatibility on a semantic level [8], a base ontology has also been proposed. On the other hand, the Eclipse Vorto project[5] defines a repository of device descriptions that can be shared among platforms. We believe that a standardized protocol for data interchange among IoT platforms should exist together with definitions of device and data representations, which is already partly supported by oneM2M. Currently, a serious usage of the oneM2M ontology is missing, which may contribute to a definition of a generally usable public ontology.

IV. Reference Implementation

The oneM2M standard defines a multitiered, distributed architecture (see Sec. III) with the three main components (IN-CSE, MN, AE). Further we describe our implementation that is based on OM2M (see Fig. 2).

[5]http://www.eclipse.org/vorto/

We added support for multiple databases at a time, which can be of arbitrary (x-SQL) type. To show the applicability of the API, we concurrently use H2 and MongoDB database.

The IN-CSE is the main node, to which we added support for complex event processing (CEP) using EsperTech library. Each CEP rule generates values, that are stored as values for a new "virtual" sensor. The rules management is done using the OM2M administration console. It may seem that CEP implementation in the field of IoT is not necessary, but it can add important usable features when taking into account large deployments (i.e. not only smart home) that require specific rules, history data inclusion, speed, etc.

The OM2M supports oneM2M subscriptions and notifications using a special "monitor" aplication. We have extended this support to enable sending notifications and alarms over e-mail or PUSH notifications via the Google Cloud Messaging.

In our referential implementation we additionally implemented IPU (Interworking Proxy Unit) plugins for protocols CoAP, MQTT, Z-Wave and ZigBee. Analogous to the HTTP message headers, which represent various parts of the OM2M message, the CoAP specifications define a set of so called "custom options". The plugin and the client were implemented using the open source Californium library.

The main component of the MQTT protocol is the MQTT broker (server), which allows devices, communicating over this protocol, to publish (send) or subscribe (be notified) to messages in various, so called, topics. The MQTT client is implemented using the open source Paho library and for the broker, the Mosquitto library was used. Besides relaying messages between topics, the broker is responsible for client authentication and authorization, as described in the oneM2M.

We added support for two more protocols - Z-Wave and ZigBee. Both are used for building mesh networks and are nowadays primarily used in the home automation domain.

Initially, we overviewed the Z-wave libraries to determine the most suitable library, which we would use in our IPU plugin. The Zwave4j library appeared to be the most suitable. It is a wrapper of the most widely used open source Z-Wave library - OpenZwave. The main component of our Z-Wave IPU plugin is a singleton object, named Manager, which registers a set of callback methods, used for receiving command results (new node added, node removed, value of a specific command

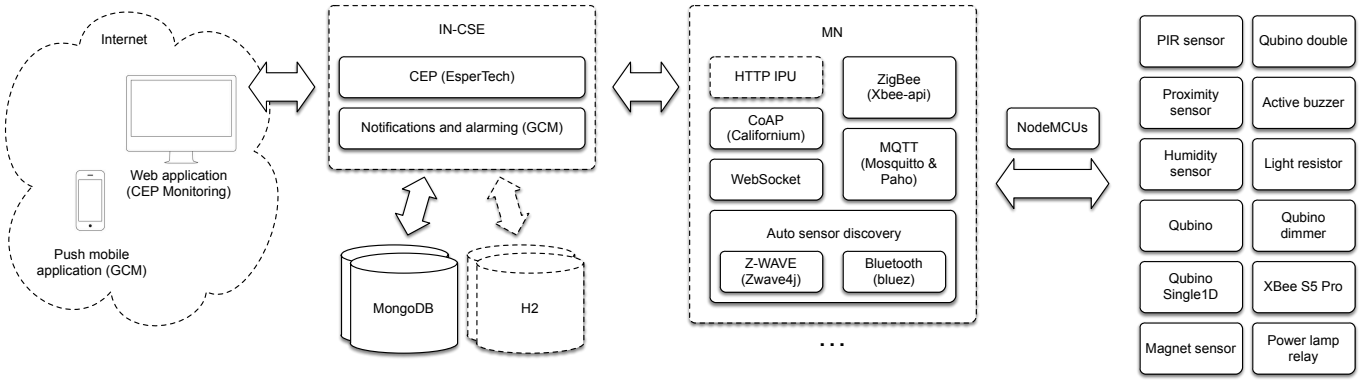

Fig. 2. Reference implementation of the proposed IoT framework. The dotted line denotes the existing implementations while full line denotes our implementations.

class changed). We use the AEON GEN5 USB dongle as a network controller for sending commands to the Zwave devices (various modules of the Qubino brand).

The ZigBee protocol is very similar to the Z-wave protocol. We implemented the IPU plugin using the Xbee-api library and tested on two Xbee S5 Pro modules. Similarly to the Z-wave plugin, this plugin also uses a singleton object and defines callback methods for receiving commands. Before using the modules in the platform, an initial configuration of the modules had to be performed, using the configuration tool X-CTU. With this tool, one can set various parameters of the modules using the AT (attention) commands. The parameters which have to be configured are: (1) the communication mode (should be set to API2, which is required by the library), (2) the address of the module with the role of the network coordinator and (3) basic parameters of the serial communication (baud rate, number of start and stop bits). For registering devices to the platform and sending data over this protocol we defined a set of messages in JSON format.

We implemented the Bluetooth (BT) protocol support using Bluecove library and using tools "hcitool" and "gattool" (for BT low-energy protocol). The IPU plugin first pairs with a visible BT device and then tries to connect to a service for data communication.

Currently we support sensor auto-discovery for Z-Wave and Bluetooth protocols. Therefore, we implemented a special IPU plugin that regularly scans the area for new devices. After a new device is found, the plugin tries to automatically connect to a device and then add it to the platform according to the oneM2M specifications.

All the OM2M extensions and reference implementations is being freely available for the research community[6].

V. Conclusion

Currently there exist no IoT platform that would follow an IoT standard, be freely available and generally applicable.

In the presentation of the reference implementation of the proposed standardized IoT framework we give the details of our extended OM2M platform that follows the oneM2M standard and is applicable to an arbitrary domain. To showcase the applicability we paired the platform with various sensors, which were connected at the same time using different protocols. We also showcase a web application that can be used for sensor data monitoring and all data sent from the platform is represented in a standardized format. Furthermore, we implemented a mobile application that can receive PUSH notifications according to manually defined rules.

To improve the platform, standardized connectors to interconnect with commercial platforms should be developed.

Our main contribution in this paper is a theoretically presented standardized framework, which we implement in practice and show the feasibility to build standardized IoT platforms for arbitrary needs. The reference implementation is publicly available and will be continuously updated.

[6]http://iot.data-lab.si/

References

[1] M. B. Alaya, Y. Banouar, T. Monteil, C. Chassot, and K. Drira, "OM2m: Extensible ETSI-compliant M2m Service Platform with Self-configuration Capability," *Procedia Computer Science*, vol. 32, pp. 1079–1086, 2014.

[2] J. Swetina, G. Lu, P. Jacobs, F. Ennesser, and J. Song, "Toward a standardized common m2m service layer platform: Introduction to onem2m," *IEEE Wireless Communications*, vol. 21, no. 3, pp. 20–26, 2014.

[3] M. Simeunović, A. Mihailovic, and M. Pejanović-Djurišić, "Setting up a multi-purpose internet of things system," in *Telecommunications Forum Telfor (TELFOR), 2015 23rd*. IEEE, 2015, pp. 273–276.

[4] J. Mineraud, O. Mazhelis, X. Su, and S. Tarkoma, "A gap analysis of internet-of-things platforms," *Computer Communications*, 2016.

[5] M. Zdravković, M. Trajanović, J. Sarraipa, R. Jardim-Gonçalves, M. Lezoche, A. Aubry, and H. Panetto, "Survey of internet-of-things platforms," in *6th International Conference on Information Society and Techology, ICIST 2016*, 2016.

[6] H. Park, H. Kim, H. Joo, and J. Song, "Recent advancements in the internet-of-things related standards: A onem2m perspective," *ICT Express*, 2016.

[7] M. Vujović, M. Savić, D. Stefanović, and I. Pap, "Usage of nginx and websocket in iot," in *Telecommunications Forum Telfor (TELFOR), 2015 23rd*. IEEE, 2015, pp. 289–292.

[8] M. B. Alaya, S. Medjiah, T. Monteil, and K. Drira, "Toward semantic interoperability in onem2m architecture," *IEEE Communications Magazine*, vol. 53, no. 12, pp. 35–41, 2015.