# EC-IoT: An Easy Configuration Framework for Constrained IoT Devices

Enri Dalipi, Floris Van den Abeele, Isam Ishaq, Ingrid Moerman, Jeroen Hoebeke
*Department of Information Technology, Ghent University iMinds*
*Technologiepark Zwijnaarde 15, B-9052 Gent, Belgium*
*enri.dalipi@intec.ugent.be*

***Abstract*—Connected devices offer tremendous opportunities. However, their configuration and control remains a major challenge in order to reach widespread adoption by less technically skilled people. Over the past few years, a lot of attention has been given to improve the configuration process of constrained devices with limited resources, such as available memory and absence of a user interface. Still, a major deficiency is the lack of a streamlined, standardized configuration process. In this paper we propose EC-IoT, a novel configuration framework for constrained IoT devices. The proposed framework makes use of open standards, leveraging upon the Constrained Application Protocol (CoAP), an application protocol that enables HTTP-like RESTful interactions with constrained devices. To validate the proposed approach, we present a prototype implementation of the EC-IoT framework and assess its scalability.**

***Keywords*-Internet of Things; configuration; CoAP; usability**

## I. Introduction

The Internet of Things (IoT) has been continuously growing during the past years and it is estimated that the total number of connected things will double each 5 years for the coming decade [1], [2]. The high diversity of contexts and environments where these things operate in, goes in line with the proliferation of potential application areas.

The current trend in IoT is to move away from proprietary and closed ecosystems to open standards and cross-technology solutions. Moreover, there has also been an evolution of vertical domain applications into cross-domain applications, mainly referred to as polymeric applications. This evolution is seen as an enabler to improve and create new business models, thereby reducing costs and risks [3].

Considering this context, it is very unlikely that a single manufacturer will provide all nodes of a complete IoT system. Often, devices need to be integrated into existing systems. As such, the coexistence and interoperability of devices supporting different technologies is important. It is required that a device is sufficiently configurable and flexible to adapt and operate depending on a specific context and user need.

Such configuration of devices is required during different phases of their lifetime, including installation, commissioning within a network, user customization and device control. On the other hand, the constraints related to most of these devices in terms of energy supply, available memory and lack of user interface make these operations not straightforward for an installer or end user.

Generally, this problem is faced by interfacing the user to an application utility or a dedicated web server in order to perform basic commissioning and management. Given the lack of common configuration solutions, each vendor develops its own solution, causing a myriad of different applications and interfaces that a user needs to install and use in parallel for managing all his connected devices. This situation results in high integration and development costs for the manufacturer. At the same time, it hinders the usability, resulting in high operational costs for the installer, frustrations for the end user and slow down of the uptake of IoT solutions.

Until now, efforts that have been put in the definition of converged solutions were based on closed frameworks, and therefore, risk to remain isolated commercial solutions. Our main contribution is the design of an open standard based configuration framework for constrained IoT devices, called EC-IoT. A prototype implementation was built in order to demonstrate the flexibility and usability benefits that our solution offers to device manufacturers, installers as well as end users. Further, our evaluation demonstrates that the framework can effectively and efficiently configure heterogeneous constrained devices with low operational overhead.

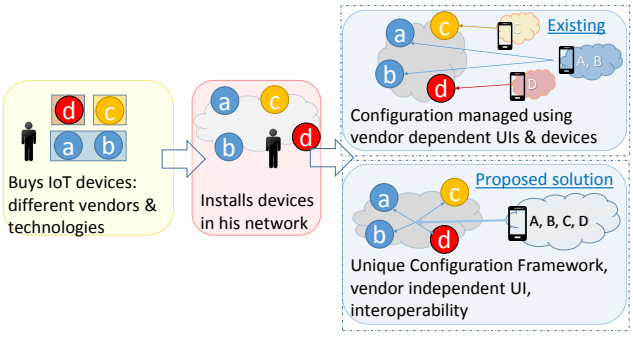


Figure 1. Proposed vs. existing configuration solution

The remainder of this paper is organized as follows. Section II discusses in more detail the problem statement, followed by the challenges and requirements that have driven the design of our framework. Section III elaborates on the

978-1-5090-4130-5/16/$31.00 

design, architecture and operation of the EC-IoT framework, whereas Section IV presents a prototype implementation, together with a description of the resulting configuration flow. The evaluation of our framework is discussed in Section V. Finally, existing IoT configuration solutions and related work are discussed in Section VI, before concluding this work with a summary and outlook in Section VII.

## II. Problem Statement and Requirements

When deploying a new IoT device or integrating it in a larger system, users need to be able to perform proper installation, configuration and customization. The user expects from his devices the same operational flexibility as the one he is used to when operating normal unconstrained devices. As a first step, possibly performed by a technician, initial commissioning has to take place, including the configuration of network, security and other context related parameters.

Next, further device configuration and control has to be done,including setting parameters related to the usage context, such as name of device, location, function etc.

Mostly, constrained devices possess no physical user interface that would allow any direct configuration of the device by an installer or a user. Moreover, the constraints in terms of resources such as CPU and memory, make it hard to store on the device generic configuration parameters that might be known already during the design. Considering the above, it is almost impossible for any device manufacturer to fully cover any future device configuration.

The existing trend is that these devices are designed to offer a limited set of capabilities, using a specific data standard and a rigid way of interacting and exchanging the data with the user. There is no unique open standard that would allow a common configuration and control interface for different types of constrained devices that need to be used in the same environment. As a result, when the user installs a new constrained device, he often needs to install a vendor specific application on a smartphone, tablet or PC in order to be able to configure it. In other cases, the user is required to manage the process by browsing to a web server, running on a remote cloud based solution. In both cases, the solution is vendor and device dependent. The configuration and control of devices using such parallel interfaces and tools tend to be confusing and impractical for an unexperienced user. Often, issues are experienced by the lack of connection or the discontinuity of the remote cloud based service. Another side effect of vertical ecosystems is the lack of interoperability and information exchange between different devices, which on the other side, need to coexist and cooperate in the same environment.

Often, these factors push the user towards vertical ecosystems, where all his IoT products are offered by the same manufacturer [4]. This trend contributes to the vendor lock-in and to high project integration costs. By hindering overall system integration flexibility, it also causes high operational costs, blocking the easy adaptation and penetration of innovative IoT solutions [5]. Contrary, the existence of flexible and standardized tools to configure such constrained IoT devices would result in more attractive and user friendly products, narrowing existing application gaps. Moreover, the exchange of valuable information between different devices during the configuration, operation and control would be of added value for the user and contribute to an augmented perception of efficiency and intelligence of the devices as a system. This last concept, together with the usability, goes in line with the Web-of-Things (WoT) paradigm.

The above discussion reveals the need of a streamlined configuration and management framework that builds on open standards and targets todays key problems regarding the deployment and usage of constrained devices. In order to properly design such a framework it is important to clearly understand the main configuration requirements and challenges. In terms of supported devices, the framework should be able to support resource limited devices without user interface that possibly operate in sleep mode. In order to be sufficiently versatile and performant, it should build on open solutions, have a small footprint and low communication overhead, and be scalable, cost effective and easy to integrate in either local or cloud operated solutions. Its impact on the devices (e.g. code changes) must be minimal. Finally, towards the user, the framework must offer a user friendly interface, operate on a variety of heterogeneous user devices and able to automatically adapt to context, user and device.

These requirements have driven the design of our EC-IoT framework, which is discussed in detail in the following section.

## III. Design, Architecture and Operation

### A. Design choices

The previous challenges and requirements were transformed into a set of design choices for our EC-IoT framework. In order to support the most resource constrained devices that can be connected to the Internet, we minimally support Class 1 devices, with 10 KiB RAM and 100 KiB ROM, as in [15].

Next, in order to be open, low overhead, scalable and easy-to-integrate, CoAP protocol [16] has been chosen as the application protocol for our framework. Using UDP as transport layer, CoAP benefits from a smaller UDP header overhead, compared to TCP headers. On the other side, the message exchange reliability is managed by CoAP itself. CoAP is designed to use minimal resources and therefore ideally suited for constrained IoT devices in low-power and lossy networks. It supports the well-known REST paradigm and can be easily mapped to HTTP methods. A variety of implementations are available, which can be easily reused for realizing the communication part of the user interface on a variety of devices.

Finally, it supports data models for semantic interoperability, such as IPSO [13] and OMA LWM2M [14], which are relevant to automatically derive the device type and the meaning of REST resources. This semantic recognition of the type of resources enables the visualization of device specific information on a user interface.

As such, it is clear that CoAP and its corresponding REST model are perfectly suited to meet our requirements with respect to device support, communication and the realization of the user interface.

### B. Architecture and operation

The high-level architecture of the proposed EC-IoT framework is shown in Figure 2. The configuration of sensors using heterogeneous wireless technologies is managed by the EC-IoT framework, which, in this example, runs on a gateway that is able to communicate with all constrained devices and with the user interface device using the CoAP protocol. There are two main components that run on the gateway: the Resource Directory (RD) [18] and the Configuration Directory (CD). RDs are mainly used to indirectly discover the presence and properties of CoAP servers, mainly in the cases when, due to the specific characteristics of the nodes in a LLN, direct discovery of the nodes is not possible or practical. As by our initial design challenges, we did not modify the standard behavior of the RD. Instead, we introduce the novel CD component. The architecture of the CD has similarities with the one of the RD, but the CD is used specifically for the configuration of the nodes.

In order to become informed about new devices joining the network, the CD observes a resource on the RD that contains the list of the nodes that are registered with the RD.

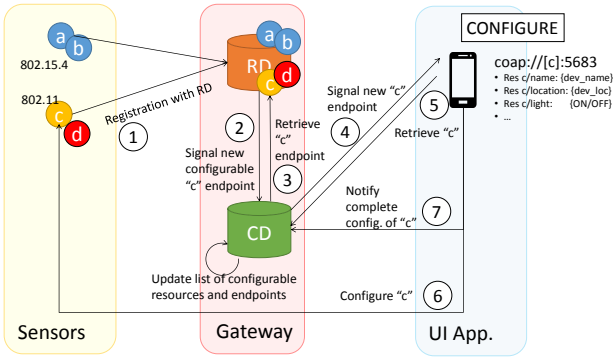


Figure 2. Resource configuration process

The main steps, starting from the moment that a new CoAP sensor is added to the network until its configuration is complete, are shown in Figure 2 and described below.

A vendor designs a CoAP-enabled IoT device (e.g. sensors) that hosts a set of CoAP resources modeled according to OMA LWM2M and IPSO. Every configurable or to be configured resource has been assigned a '*conf*' link attribute. Next, the unconfigured sensor is powered on and added to the Wi-Fi or WSN network. Immediately, the sensor triggers the registration with the RD by sending a CoAP POST request to '/rd' resource on the RD. The link of its resources in CoRE link format is included in the payload. As soon as the request is received, the RD registers the sensor and updates the /rd-lookup/ep resource containing information about all registered end points (STEP 1). As the CD continuously observes the /rd-lookup/ep resource on the RD, it is being informed about the presence of a new endpoint (STEP 2). The CD will now retrieve from the rd-lookup/res resource on the RD, the list of all resources that are present on the sensor and that have the '*conf*' attribute set. The CD now updates its corresponding cd-lookup/ep and cd-lookup/res resources. These resources contain all configurable endpoints and their resources. The configuration status of a configurable resource is also stored here (STEP 3). The client application on the user device observes the cd-lookup/ep resource on the CD and is being informed about the presence of newly to be configured devices (STEP 4). The client can then retrieve the list of configurable resources and their attributes and render a suitable interface (STEP 5). As soon as the user confirms and submits the configuration parameters for the available configurable resources, the client application configures the respective resources by sending CoAP POST requests (STEP 6). Upon completion of the configuration of the sensor, the user application updates the status of the resources and endpoints into configured and notifies the CD (STEP 7). The device configuration status is now marked as configured.

The above functional internal processes are transparent to the user. The steps that a user needs to follow when managing his devices are described in detail in Section IV.

## IV. Prototype of EC-IoT Framework

A prototype implementation of the proposed EC-IoT configuration framework has been built in order to evaluate the feasibility and performance of the proposed approach. The following main hardware components have been used:

1) Zolertia Z1 nodes. A headless device, such as Zolertia Z1, was chosen as constrained sensor node, as this module falls into the Class 1 device categorization.
2) nodeMCU ESP8266 nodes. A less constrained device, such as the NodeMCU ESP8266 module, was chosen to demonstrate the support of heterogeneous wireless technologies. Both, Z1 and nodeMCU modules, can operate as battery powered devices and different sleep mode mechanisms can be enabled.
3) Raspberry Pi. The Raspberry Pi runs a CoAP gateway implementation, called CoAP++, realized in Click Router [17]. On top of it, an implementation of a CoAP Resource Directory (RD) [12] and Configuration Directory (CD) has been built. The gateway is

able to communicate to all constrained devices and the user interface device.

4) Wi-Fi IEEE 802.11n Access Point, used to connect both the user interface device and the Wi-Fi sensor nodes.
5) User interface device. A consumer Android tablet has been selected as the user interface device. For this purpose, an Android CoAP client and user interface application have been developed.

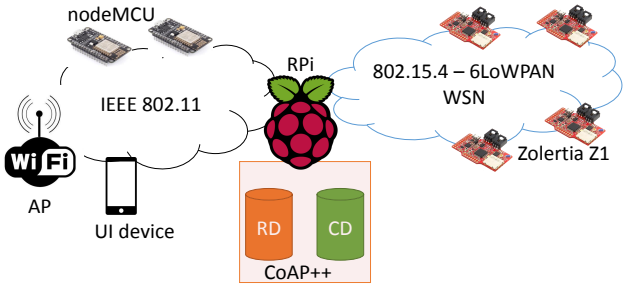


Figure 3. Devices and network setup

Figure 3 shows the respective components and their interconnection. All these components have been integrated in a flightcase housing. The resulting setup models a home automation use case that incorporates a variety of sensors distributed on the model surface. The respective sensors were connected to LEDs, and were mains powered (in case of Z1 devices), or battery powered (in case of nodeMCUs). The setup of the flightcase, including the tablet user interface and a standalone nodeMCU device to demo the EC-IoT framework are shown in Figure 4.

Figure 4. Setup and configuration of a Wi-Fi nodeMCU LED using EC-IoT

After the installation and the commissioning of the devices in their respective networks, the user configuration flow proceeds as below. First, the user launches the device-independent configuration UI to check for any configured and unconfigured devices in his network. These two categories, independently of each device technology, are displayed on the UI. At the same time, the UI is able to visualize specific icons, depending on each sensors device type link attribute. Link attributes, being specified in the '.well-known/core' resource of each device, are retrieved during the discovery of the devices.

When a new unconfigured sensor is installed, an unconfigured sensor icon pops up on the UI, as shown in Figure 4. The user can now click on the sensor icon, after which a UI screen is rendered that shows all resources that can be configured. The user now fills in the configuration parameters, as required by the UI and performs the configuration. The whole process is fully device independent, with the UI being able to handle heterogeneous devices.

In case of successful configuration, the sensor icon would signal this by changing color and by getting listed under the configured devices category. The LED sensor would signal this to the user by blinking one time.

Using the same UI and in the same uniform way, the user can always check and modify the parameters of the already configured devices. He can configure device description related resources (i.e. location, name, owner, etc.) or can check the status of measurement resources (i.e. living room temperature in the case of a temperature sensor) using standard CoAP interactions. He is also able to modify, and therefore control, the status of a LED sensor or an actuator installed in his network.

## V. Evaluation

The realized prototype enables the evaluation of the usability and flexibility of the proposed solution. From a functional point of view, our solution makes the integration of heterogeneous constrained devices straightforward, thereby requiring only a single, vendor independent UI. For an IoT device to be able to interface with and get configured by our framework, it is sufficient that i) it has a built-in CoAP server able to trigger the registration with the RD, and ii) it marks configurable resources with a conf link attribute. Considering our implementation and the existing CoAP-based solutions, both these minimal requirements can be easily satisfied by any considered Class 1 constrained device. An additional requirement would be the installation and network commissioning of the device in order to be able to communicate with the EC-IoT framework. However, this is a prerequisite to the configuration process, and therefore is out of the scope of this research work.

From a performance point of view, a low communication overhead and good scalability are key requirements for our solution. To this end, we have evaluated and modeled the overall message exchange that takes place during the configuration of a configurable endpoint (EP "c") device, assuming that there is only one RD and one CD entity. Figure 5 shows the resulting CoAP message exchange, with the steps in the first column referring to the steps previously described in Figure 2. In this figure, we consider the most simple case when the amount of data required for representing the

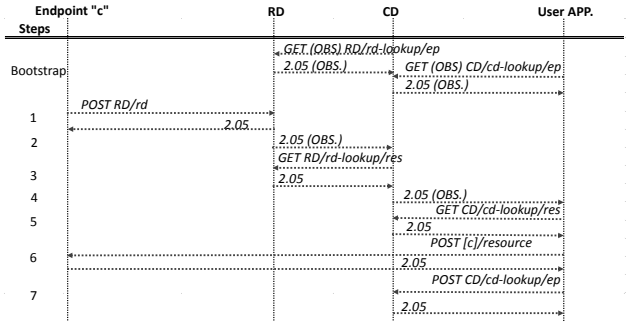


Figure 5. Total number of CoAP message exchanged in each step for configuring $S$ devices

device resources in CoRE link format is contained within the maximum supported block size, i.e. block transfer is not being used. With $R$ resources on a single device, $b_R$ bytes needed to represent a single resource and its attributes in CoRE link format and B being the maximum supported block size in bytes, this means that $\lceil R * (b_R/B) \rceil$ is equal to 1. Considering that the CD and RD may not necessarily reside on the same device, we have also counted the message exchange between these two components. Finally, we also assume that the configuration of a single resource requires the transfer of $b_C$ bytes (i.e. no block transfer if $\lceil b_C/B \rceil$ is equal to 1), with the device having $C$ configurable resources in total $(C \leq R)$.

Using the above notations, we can further generalize the idea, by considering the configuration of more devices ($S$ in total), hosting resources that are not necessarily representable within one maximum block size and that may require configuration input exceeding the maximum block size. We notate with $B^{'}$ the maximum supported block size in bytes in the unconstrained network. Under these assumptions, the total number of messages being exchanged is shown in Figure 6. From Figure 6, we observe that there is

| Steps | Messages | Entities |
|---|---|---|
| Bootstrap | 2 | RD <---> CD |
| | 2 | CD <---> APP |
| 1 | S*[2*⌈R*br/B⌉] | EP <---> RD |
| 2 | 2*S | RD <---> CD |
| 3 | S*[2*⌈C*br/B'⌉] | RD <---> CD |
| 4 | 2*S | CD <---> APP |
| 5 | S*[2*⌈C*br/B'⌉] | APP <---> CD |
| 6 | S*[2*C*⌈bc/B⌉] | APP <---> EP |
| 7 | 2*S | APP <---> CD |

Figure 6. Message exchange in each step for configuring *S* devices

a constant number of messages exchanged during bootstrap. The number of messages exchanged during steps 2, 4 and 7 is purely proportional on the number of configurable devices (S). The messages exchanged in steps 1, 3, 5 and 6, also depend on the overall number of resources, the number of configurable resources and the required amount of data to represent and configure the resources ($b_R$ and $b_C$). From these steps, we observe that the messages in steps 3 and 5 are exchanged internally in the framework or in the less constrained wireless network, between the user interface device and the gateway. Assuming a large maximum supported block size in the unconstrained network, then $\left\lceil C * b_R/B^{'} \right\rceil \approx 1$ and:

$$M_3 \approx 2 * S$$

$$M_5 \approx 2 * S$$

On the other side, the messages exchanged in steps 1 and 6 are exchanged with the constrained device itself, possibly over a constrained network and therefore have the most impact in terms of energy consumption and latency. We observe that the number of exchanged messages in step 6 is proportional to the number of configurable resources and the worst case scenario results when all the resources are configurable (C=R):

$$M_1 = 2 * S * \lceil R * b_R/B \rceil \leq 2 * S * R * \lceil b_R/B \rceil$$

$$M_6 = 2 * S * C * \lceil b_C/B \rceil \leq 2 * S * R * \lceil b_C/B \rceil$$

The messages in step 1 are linked to the RD concept itself and inherent to every solution that incorporates RD-based discovery. The messages in step 6 are introduced by our solution, but other solutions performing resource-based configuration will exhibit similar scalability. Therefore, we may conclude that the proposed solution respects the scalability requirement.

## VI. Related Work

There has been continuous effort to define converged solutions that are able to deal with the configuration and management of heterogeneous IoT devices. These solutions are proposed both by academic and industrial organizations, and target mainly the standardization of interfaces, protocols and IoT data models [6].

In [7]–[10], solutions for the configuration of IoT heterogeneous devices are described. However, these solutions do not make use of open standards and require additional hardware. As a specific protocol, software and hardware are required, these solutions fall out of the scope of our work.

A sensor discovery and configuration framework is described in [11]. This work makes use of semantics and focuses mainly on the commissioning of the sensors and the connection to a cloud IoT middleware. However, there is less focus on the usability and user configuration interface.

In [12], an open standards-based self-configuration solution is described. It facilitates the deployment, discovery

and resource access for CoAP servers. However, it is not focused on the user configuration of heterogeneous devices. Our work leverages on these results, but goes beyond them with respect to user interfacing, the CD concept and the use of lightweight semantics for UI generation.

## VII. Conclusions and Future Work

In this paper, we have presented our EC-IoT framework, a configuration framework for IoT devices that is fully based on open standards. The proposed solution offers a streamlined configuration and management solution that can help installers and non technical users when dealing with the roll-out of heterogeneous IoT devices. It is beneficial also to IoT device manufacturers, when designing and integrating complex systems that include heterogeneous and vendor dependent technologies and standards.

This is a crucial step in enabling the proliferation of IoT applications, the interoperability and the usability of constrained devices.

In order to evaluate the proposed solution, we have built a prototype system that demonstrates the feasibility of the concept. The experimental results show a successful configuration of constrained heterogeneous devices, improved usability by making use of a single user interface, and a contained message exchange overhead.

In the future we plan to improve the framework towards context-aware self-configuration of constrained devices.

## Acknowledgment

The research from DEWI project (www.dewi-project.eu) leading to these results has received funding from the ARTEMIS Joint Undertaking under grant agreement n 621353 and from the agency for Flanders Innovation & Entrepreneurship (VLAIO). The research from the ITEA2 FUSE-IT project (13023) leading to these results has received funding from the agency for Flanders Innovation & Entrepreneurship (VLAIO).

## References

[1] M Meekers, S Devitt, L Wu, Morgan Stanley internet trends 04/12/2010, http://www.morganstanley.com/institutional/techresearch/pdfs/Internet_Trends_041210.pdf

[2] The Internet of Things: A survey of topics and trends, Andrew Whitmore, Anurag Agarwal and Li Da Xu, Journal: Information Systems Frontiers, 2015, Volume 17, Number 2, Page 261

[3] A. Botta, W. De Donato, V. Persico, and A. Pescap, Integration of Cloud computing and Internet of Things: A survey, Futur. Gener. Comput. Syst., vol. 56, pp. 684700, 2016.

[4] S. Leminen, M. Westerlund, M. Rajahonka, and R. Siuruainen, Towards IOT ecosystems and business models, in Lecture Notes in Computer Science, 2012, vol. 7469, pp. 1526.

[5] I. Lee and K. Lee, The Internet of Things (IoT): Applications, investments, and challenges for enterprises, Bus. Horiz., vol. 58, no. 4, pp. 431440, 2015.

[6] Internet of Things: Applications and Challenges in Technology and Standardization, Wireless Personal Communications, May 2011, Volume 58, Issue 1, pp 49-69 Debasis Bandyopadhyay, Jaydip Sen

[7] S. Guoqiang, C. Yanming, Z. Chao, and Z. Yanxu, Design and implementation of a smart IoT gateway, Proc. - 2013 IEEE Int. Conf. Green Comput. Commun. IEEE Internet Things IEEE Cyber, Phys. Soc. Comput. GreenCom-iThings-CPSCom 2013, pp. 720723, 2013.

[8] I. Chatzigiannakis, H. Hasemann, M. Karnstedt, O. Kleine, A. Krller, M. Leggieri, D. Pfisterer, K. Rmer, and C. Truong, True self-configuration for the IoT, Proc. 2012 Int. Conf. Internet Things, IOT 2012, pp. 915, 2012.

[9] T. P. Works, The Thingsquare Blog (/ blog /) How the Thingsquare Platform Works What the User Sees, 2014.

[10] Q. Zhu, R. Wang, Q. Chen, Y. Liu, and W. Qin, IOT Gateway: BridgingWireless Sensor Networks into Internet of Things, 2010 IEEE/IFIP Int. Conf. Embed. Ubiquitous Comput., pp. 347352, 2010.

[11] C. Perera, P. P. Jayaraman, A. Zaslavsky, D. Georgakopoulos, and P. Christen, Sensor discovery and configuration framework for the Internet of Things paradigm, 2014 IEEE World Forum Internet Things, pp. 9499, 2014.

[12] I. Ishaq, J. Hoebeke, J. Rossey, E. De Poorter, I. Moerman, and P. Demeester, Enabling the web of things: Facilitating deployment, discovery and resource access to IoT objects using embedded web services, Int. J. Web Grid Serv., vol. 10, pp. 218243, 2014.

[13] J. Jimenez, M. Koster, and H. Tschofenig, IPSO Smart Objects, 2016.

[14] Data models for the IoT, http://openmobilealliance.org/data-models-for-the-internet-of-things/

[15] RFC7228, https://tools.ietf.org/html/rfc7228

[16] Constrained Application Protocol, https://tools.ietf.org/html/draft-ietf-lwig-coap-03

[17] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, The click modular router, ACM Trans. Comput. Syst., vol. 18, no. 212, pp. 263297, 2000.

[18] CoAP Resources Directory, https://tools.ietf.org/html/draft-ietf-core-resource-directory-07