Computer Networks 171 (2020) 107101

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

A literature review of IoT energy platforms aimed at end users

Miguel M. Martín-Lopo*, Jaime Boal, Álvaro Sánchez-Miralles

Instituto de Investigación Tecnológica, Escuela Técnica Superior de Ingeniería ICAI, Universidad Pontificia Comillas, Madrid, Spain

ARTICLE INFO

Article history:
Received 7 October 2019
Revised 27 December 2019
Accepted 7 January 2020
Available online 11 January 2020

Keywords:
Internet of Things
Energy platforms
Energy middlewares
Communication protocols
Energy services
Physical layer
Server layer
Application layer
Security

ABSTRACT

The rising interest in connecting everything to the Internet has not gone unnoticed in the energy sector. New actors that aim to remotely monitor and control home devices such as heating, ventilation and air conditioning (HVAC), light bulbs, or distributed energy resources (e.g., batteries, PV panels…) have come into play. However, transitioning from isolated often not interoperable home automation systems to open, yet secure, solutions that integrate external sources of information and cloud computing to make a more efficient use of energy, is not trivial. It requires designing and implementing hierarchical architectures and standard solutions to facilitate interoperability, one of the challenges of cross-domain smart-city applications as no standard solution has been established yet. Even though most solutions share a set of building blocks that have fostered the appearance of Internet of Things (IoT) middlewares to accelerate development, most existing energy platforms are still tailor-made for specific applications. This paper targets three audiences. First, for those interested in using or selecting an energy platform, the study carries out a comparative analysis of some of the most popular alternatives. Second, for those that are considering building new energy platforms, this paper analyzes the necessary hierarchical blocks, and the main design options and strategies. Finally, for those interested in comparing platforms, a new set of IoT levels that evaluate the adoption of IoT technologies is proposed.

1. Introduction

The electricity sector is currently undergoing significant changes due to several reasons like the expansion and development of renewable generation technologies or the new regulations approved by governments attempting to fight climate change and reduce pollution. This means that the outlook for the following decades predicts low-carbon energy generation (many European countries like France or the UK have already announced that they will shut down all coal plants by 2025) and the emergence of Distributed Generation (DG). The operation of distribution networks is already evolving into a model where energy efficiency services, Distributed Energy Resources (DER), and energy storage devices will be ubiquitous. Together with the fast improvements in information and communication technologies (ICT) and the rising of the Internet of Things (IoT), this is fostering the appearance of new business models and new energy systems that range from microgrids (MG) to remote load aggregation and electric vehicles' smart charging. These solutions typically imply deploying physical sensor and actuator networks monitored and controlled from a central server, located either in the same local network or in the cloud, across devices, vehicles, and buildings.

At a large scale, IoT systems will generate an overwhelming amount of information, from measurements to user preferences, which needs to be processed jointly. However, there are more requirements and challenges that energy platforms have to address. Patti and Acquaviva analyze some of these requirements, highlighting the need for interoperability across smart city platforms [1]. Besides collecting and processing huge amounts of data, energy platforms must also be able to issue commands based on control algorithms in real-time. For this reason, another challenge that energy platforms must bear in mind is scalability, since the traditional way of dealing with growing systems by increasing computational power might not be suitable for handling billions of IoT devices. Furthermore, to develop cross-domain applications at a smart-city level, platforms must exploit interoperability by adopting standard solutions and technologies, which is an issue due to the low maturity of most of them. Finally, energy platforms also face security challenges. Every node and every communication frame must be protected against physical and cyber-attacks. The consequences of a compromised system range from the disclosure of sensitive data (e.g., users' routines and preferences could be inferred from their energy consumption habits) to loss of integrity

* Corresponding author.
E-mail addresses: miguel.martin@iit.comillas.edu (M.M. Martín-Lopo), jaime.boal@iit.comillas.edu (J. Boal), alvaro.sanchez@iit.comillas.edu (Á. Sánchez-Miralles).

https://doi.org/10.1016/j.comnet.2020.107101
1389-1286/

(i.e., unauthorized modification of data), which would allow the attacker to take over the devices and put safety at risk.

Every single one of these challenges must be considered in the first stage of the development, design. This paper analyzes energy platforms aimed at end-users from different perspectives and provides a classification of their characteristics. The features analyzed cover early design decisions (e.g., platforms' architectures), technical decisions (e.g., communication protocols and networks), and functional features related to the services provided by the system (e.g., compatibilities with other platforms). This paper is complementary to other published reviews such as [2–6]. The study presented in [2] was one of the first surveys found in the literature addressing the concept of IoT and it may still offer a useful first-contact guide. It analyses different definitions of IoT, common elements found in IoT's architectures and different environments in which IoT could potentially be applied. [3] focuses more on functional aspects, analyzing several platforms in order to help the reader make a choice, pointing out open source alternatives and the developer tools they provide. Ngu et al. [4] puts forward a deep study of IoT middlewares by classifying existing architectures into three different groups (service-based, cloud-based, and actor-based) and studying the different issues and challenges IoT middlewares need to tackle, such as service discovery or security and privacy. Bedi et al. [6] present a review of IoT applied to Electric Power and Energy Systems (EPES). The study covers the impacts of IoT under different points of view, such as economic, environmental and societal. Then, it analyses the integration of IoT with different parts of the energy system (e.g., generation and transmission, among others), and the current limitations and challenges IoT for EPES must tackle, from connectivity to big data. This extensive review serves as a good first contact with the design of IoT ecosystems; however, it does not go into much depth since it simply analyses the main concerns and trends of IoT for EPES.

As aforementioned, this review is complementary to previous studies as it is conceived to assist comparisons between IoT platforms aimed at end-users of the energy sector. IoT platforms without an immediate impact on the energy sector have been excluded. For example, systems such as LIFX [7], Philips Hue [8], or Savant Systems [9] whose only objective is to increase users' comfort without taking into account neither energy consumption nor efficiency. Additionally, platforms aimed at electric vehicles such as Loop [10] have also been left out as they are considered to be a niche by themselves. In architectural and technological levels, they rely on similar elements than the platforms included, mainly differing in terms of services and interoperability.

The main contributions of this study are the following. First, the introduction of a new methodology to easily classify and compare the degree of adoption of IoT technologies. Secondly, a comprehensive review of energy platforms that updates and extends earlier works by analyzing energy platforms from the perspective of every hierarchical level of their architecture. Finally, the identification of common characteristics found in existing energy platforms, which should contribute to lay the foundations for new and more advanced, interoperable, and secure systems.

The rest of the paper is structured as follows. First, Section II presents the blocks that usually underpin the architecture of IoT platforms. Sections III, IV, and V discuss, respectively, different topics regarding platforms' physical, server, and application layers. Section VI provides insights about several considerations and security issues that IoT energy platforms should take into account. Section VII studies different specific services that are remarkably important in energy platforms and interoperability. Section VIII presents the results derived from the analysis of the state-of-the-art. Section IX highlights lessons learned from the study and provides design guidelines for IoT platforms. Finally, Section X summarizes the main conclusions drawn from the study.

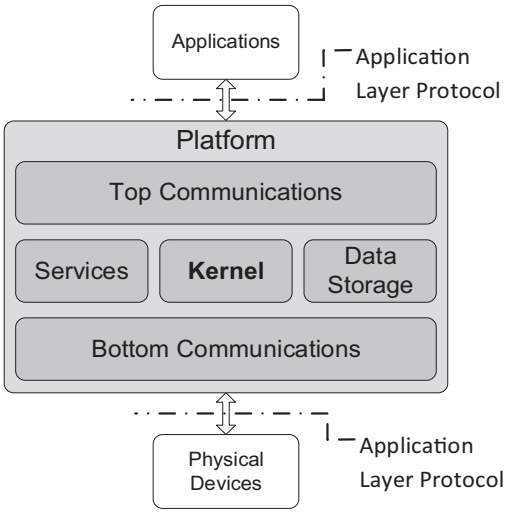

Fig. 1. Building blocks of an energy platform cloud architecture.

2. Cloud-based architectures

The emergence of DG and the ongoing improvements on ICT and IoT are fostering the development of new business models in which end-users request more personalized services and features. One of these relatively new business models are energy platforms. Their objective is to increase users' comfort by means of home automation systems while reducing energy bills by improving energy efficiency or through Demand Side Management, or specifically, Demand Response programs [11].

Fig. 1 shows a common architecture for this type of platforms, where a set of devices (sensors and actuators) are deployed across users' households. As stated in [12], IoT architectures are based on a three layer system: physical, network, and interfaces. The physical layer is in charge of two main tasks. The first one is to provide information about the environment, without which, platforms would not be able to run any algorithm or provide almost any service. The second is to execute user commands and apply the settings obtained as an output of the algorithms that optimize energy usage and improve comfort levels. The physical layer communicates with the core of the system, usually a server located in the cloud, which is composed of five main block types.

The kernel supervises every other block and is responsible for the overall state of the system. Besides, it also manages and coordinates the execution of the available services, providing them with the required information from the data storage. The kernel must handle different types of algorithms with distinct execution triggers. To achieve all this, the design process of the whole system is essential.

There are two clearly demarcated entry points to the platform: one for devices and another for applications. Despite the fact there is no big difference between them in terms of software programming, apart from the specific protocol –the one used by the physical layer is likely to be lighter due to hardware constraints– and the data transmitted, they have been labeled differently for the sake of clarity. According to the Open System Interconnection (OSI) model, the physical layer appears in the lowest level, which is why it is referred to as "*Bottom*", as opposed to the one that handles applications that is tagged as "*Top*".

The application layer fulfills two main tasks. First, applications are the tools that make remote control and interaction with the system possible. Users can access their personal account anywhere and monitor the devices in the physical layer. In addition, they are essential for the expansion of the system since the same data

Table 1
Classification of the studied systems according to bottom characteristics.

Platform	Device Types	Bottom Architecture	Internet Connectivity	Connectivity Exploitation	Home Automation	Thermal	Electric	DES / Storage
Afero [16]	Sensors and actuators	Mixed	1, 2	1	✓	✗	✗	✗
Al Faruque and Vatanparvar [17]	Sensors and actuators	Local Server	1	0	✓	✓	✗	✓
Ali-Ali et al. [18]	Sensors and actuators	Mixed	1, 2	1	✓	✓	✓	✗
Ayla [19]	Sensors and actuators	Mixed	1, 2	0	✓	✗	✗	✗
Bosch Smart Home [20]	Sensors and actuators	Local server	1	0	✓	✓	✗	✗
Altair SmartCore [21]	Sensors	–	–	–	–	–	–	–
Cisco [22]	Sensors and actuators	Local server or Gateway	1	1	–	–	–	–
DeviceHive [23]	Sensors and actuators	–	–	–	–	–	–	–
DIMMER [1]	Sensors	–	–	–	✓	✓	✓	✗
Ecobee [24]	Sensors and actuators	Mixed	2	0	✗	✓	✗	✗
E-IoT [25]	Sensors and actuators	Local Server	0	0	✓	✗	✗	✗
FLEXMETER [1]	Sensors	–	–	–	✓	✓	✓	✗
GridPoint [26]	Sensors and actuators	Local server	1	0	✓	✓	✗	✗
HEMS [27]	Sensors and actuators	Gateway	1	1	✗	✗	✓	✗
Honeywell [28]	Sensors and actuators	Local server	1	0	✓	✓	✓	✗
iChipNet [29]	Sensors and actuators	Mixed	1, 2	0	✓	✗	✗	✗
Insteon [30]	Sensors and actuators	Mixed	1, 2	1	✓	✓	✗	✗
Javed et al. [31]	Sensors and actuators	Local server	1	1	✗	✓	✗	✗
Kaa [32]	Sensors and actuators	–	–	–	–	–	–	–
LG SmartThinq [33]	Actuators	Direct connection	2	0	✓	✓	✗	✗
LoBEMS [34]	Sensors and actuators	Local server	1	0	✓	✓	✓	✗
Lopez et al. [35]	Sensors	Gateway	1	1	✗	✗	✓	✓
MyDevices [36]	Sensors and actuators	–	–	–	–	–	–	–
Nest [37]	Sensors and actuators	Direct connection	2	1	✓	✓	✗	✗
Netatmo [38]	Sensors and actuators	Mixed	2	1	✓	✓	✗	✗
Nexia [39]	Sensors and actuators	Local server	1	0	✓	✓	✓	✓
Particle [40]	Actuators	Direct connection	2	1	✓	✗	✗	✗
Predix [41]	Sensors and actuators	Mixed	1, 2	1	–	–	–	–
Qarnot [15]	Sensors and actuators	Direct connection	2	1	✗	✓	✗	✗
Scheneider Wiser [42]	Sensors and actuators	Local server	1	0	✓	✓	✓	✗
SensorCloud [43]	Sensors	–	–	–	–	–	–	–
Shinde et al. [44]	Sensors	Local server	1	0	✓	✗	✗	✗
Siemens Desigo [45]	Sensors and actuators	Local server	1	0	✓	✓	✗	✗
Smappee [13]	Sensors and actuators	Local server	1	1	✓	✗	✓	✓
SmartThings [68]	Sensors and actuators	Gateway	1	0	✓	✓	✗	✗
ThingPlus [46]	Sensors and actuators	–	–	–	–	–	–	–
ThingWorx [47]	Sensors and actuators	Direct connection	2	0	✓	✗	✗	✗
Ubidots [48]	Sensors and actuators	–	–	–	–	–	–	–
Watson IoT [49]	Sensors and actuators	–	–	–	–	–	–	–
Wattio [50]	Sensors and actuators	Mixed	1	1	✓	✓	✓	✗
WattsOn [51]	Sensors	Local server	1	0	✗	✗	✓	✗
Wibeee [52]	Sensors	Mixed	1, 2	1	✗	✗	✓	✗
Wink [14]	Sensors and actuators	Local server	1	0	✓	✓	✓	✗
Yaghmaee and Hejazi [53]	Sensors	Local server	2	0	✓	✗	✓	✗

–: Does not apply.

(gathered in the physical layer) can serve different purposes depending on the goal and how it is processed.

Apart from the architecture depicted in Fig. 1, which corresponds to end-to-end systems that can operate on their own, there exist a wide variety of middlewares and software platforms. Their objective is to accelerate the development of custom platforms by providing the infrastructure required to connect devices with front-end applications. Middleware architectures do not differ much, but they usually do not include the blocks colored in white. Furthermore, as middlewares are designed to fit a wide range of contexts, the services embedded are typically general purpose (e.g., data analysis and event processing). Software platforms do not include the physical layer but offer a complete software solution – including processing capabilities and applications– to their users.

3. Physical layer

Energy platforms may be the missing link that finally makes it possible for users and energy markets to start interacting effectively, thus allowing companies to offer customized services to people with a higher degree of understanding of how the energy system operates. Table 1 classifies 44 platforms that provide at least one energy-related service from the list below, either because they allow to manually and remotely control loads, or because they monitor and control energy generation and consumption:

- *Home automation services:* These platforms allow to check or operate devices such as lightbulbs, blinds or presence sensors remotely. Blinds can be used, for instance, to improve insulation when required, whereas presence sensors can help determine if there is someone in a room and turn off unnecessary loads (e.g., lighting) automatically.
- *Thermal services:* Platforms that manage thermal demand by controlling devices such as radiators, heat pumps, furnaces, and air conditioning systems.
- *Electric services:* Platforms that monitor power consumption, generally through sensors installed in the distribution board or between the plug and the socket of major loads. Some simply display electricity consumption information, while more sophisticated systems also provide usage recommendations towards the reduction of the bill.
- *DES and storage services:* As aforementioned, the energy system is evolving towards distributed generation. For this reason, some platforms already integrate batteries or photovoltaic panels. Smartly managed, these DER devices can help flatten demand and reduce energy purchase prices if the optimization system is fed with sensor measurements, user behavior models, and external information sources.

The previous services are neither exclusive nor interdependent. For instance, several of the studied platforms offer services that

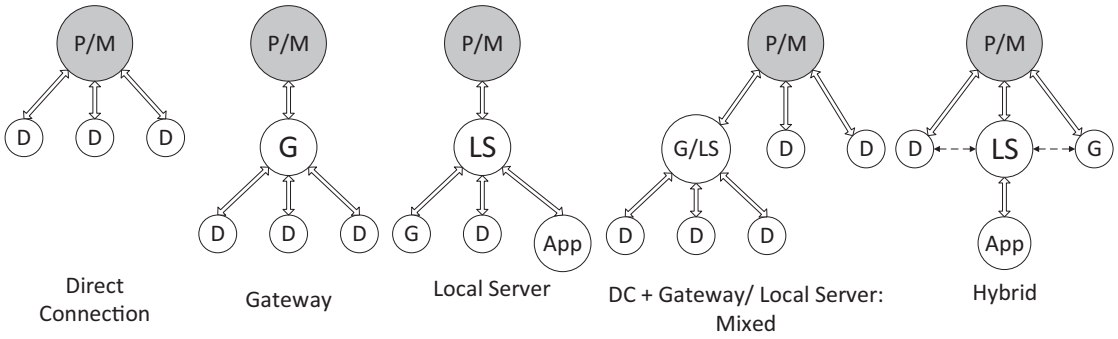

Fig. 2. Bottom topologies identified to connect physical devices (D) to a platform (P) or middleware (M). G refers to gateway and LS to local server. In some cases, the smartphone or desktop applications (App) can work within the local network (i.e., without Internet connection).

belong to multiple groups (e.g., Smappee [13] or Wink [14]). Platforms should exploit synergies between devices and integrate different services to achieve truly efficient energy management systems.

3.1. Bottom architecture

In order to be able to provide these services, the first step is to deploy physical devices (i.e., sensors and actuators) that connect to the platform using different topologies and protocols. Five topologies have been identified (Fig. 2) after reviewing the systems presented in Table 1:

- *Direct Connection (DC)*: Each device has an open socket and connects directly through the Internet to the system's server. For the purpose of this study, the router has been deemed "transparent" since it does not modify message contents. This topology minimizes the number of intermediate devices, hence reducing latency. In addition, as every node opens and holds a different connection with the server, this structure provides more robustness, since failure of an individual node does not affect other devices. In exchange, the server is forced to handle more concurrent connections, especially if the sockets are kept permanently open, with the corresponding increase in computational burden. This topology is obviously not compatible with meshed networks as devices are configured only as clients. An example could be that of Qarnot [15], where every individual lightbulb can connect to the cloud server to retrieve pending commands.
- *Gateway (G)*: In this topology, devices and server use different types of networks or different protocols, so an additional element (a gateway) is required to perform translations. This structure reduces hardware requirements at the expense of increased latency. Devices can exchange data among themselves and with the gateway using lighter and even meshed protocols (e.g., ZigBee). This approach increases coverage and autonomy of battery-powered devices, while the gateway takes care of IP-based communications. The server benefits in terms of complexity and scalability.
  If all devices are connected through gateways, the number of simultaneous sockets is drastically reduced and, in principle, it only needs to be able to parse one protocol. Supporting new devices with different protocols affects the gateway-side only. Examples of platforms that use this structure are Cisco [22] and SmartThings [68].
- *Local Server (LS)*: The main difference with the previous configuration is that besides translating protocols, the gateway (now referred to as local server) also parses the contents of some if not all frames to provide additional services that are typically application dependent. For example, it could quickly trigger an action in response to an event, which would otherwise take a few seconds if the order had to come from the cloud server, or allow users to interact with devices from within the local network. This ability represents a huge improvement in terms of resiliency as the system remains partially functional in case of an Internet outage. In addition, it also reduces external traffic as the local server may compress and aggregate data, or even assume part of the computational load. However, as more tasks are delegated to the local server, keeping both servers in sync to avoid issuing conflicting orders becomes more challenging.
  The distinction between gateway and local server is not usually found in the literature. For instance, [54–56], talk about intelligent gateways with processing and aggregating capabilities. Furthermore, [57] presents a middleware architecture deployed in both device and gateway sides.
- *Mixed (M)*: This topology is one of the most widely used. Some devices connect directly to the server (Direct Connection). Others go through a gateway or local server to enable using low-consumption communication technologies, which is almost mandatory for battery-powered nodes. Some examples include Insteon [30], Netatmo [38], or Predix [41].
- *Hybrid (H):* A final configuration that has not been found during the review process but that could be useful to improve resiliency, is somewhere in between the DC and LS topologies. Rather than coexisting, if there is an Internet outage in a DC setup, one of the devices starts acting as local server until Internet access is restored.

As expected, middlewares only establish how the physical network should connect to the platform, but not how to operate, leaving design decisions to developers. In fact, they usually provide solutions for several structures supporting different communication protocols and APIs [32,49,58].

3.2. IoT levels

The IoT is fostering a new way of operating energy systems towards one where the number of end-nodes will increase substantially. These nodes have the ability of generating new data and applying commands uninterruptedly. However, the concept of Internet of Things is still unclear. Are the topologies studied in section III.A equivalent in an IoT scale? To address this question, two scales have been defined in order to classify the degree of integration of IoT in different architectures and platforms.

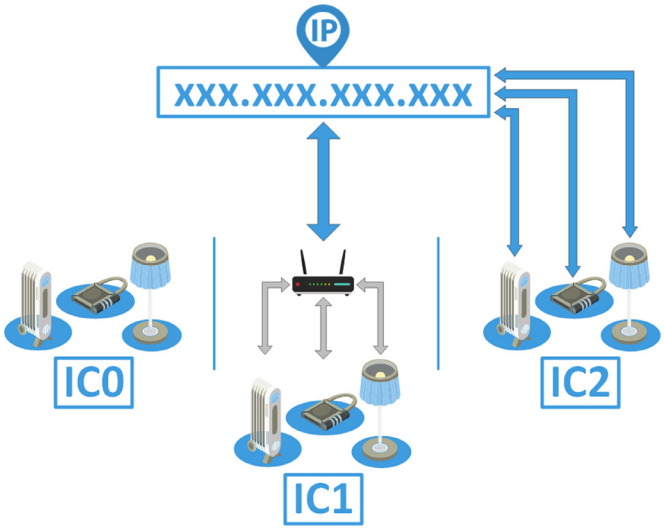

Fig. 3. Internet connectivity tiers. IC0: Systems that do not have access to the Internet or do not handle "Things". IC1: Platforms that handle Internet communications, but only some devices handle IP-based communications, providing gateway services for the rest of them. IC2: Platforms that handle Internet communications, and every device handles IP-based communications.

3.2.1. Internet connectivity (IC)

This classification measures the communication topology between devices and cloud. As shown in Fig. 3, it distinguishes the following three levels.

- *Tier IC0*: Energy systems prior to IoT. In other words, platforms that do not have access to the Internet (just operating in local networks) or that have Internet connection but do not handle "Things" (i.e., common objects). An example of this would be a platform communicating only with computers.
- *Tier IC1*: Platforms that handle Internet communications between a server and devices, only some of which support IP-based communications. In other words, there are more devices than unique IP addresses in the network. The bottom architectures used in this tier are Local Server and Gateway.
- *Tier IC2*: Platforms that handle Internet communications between devices and a server. Every device supports IP-based communications and, therefore, is identified by a unique address across the platform's network. This tier is not exclusive to DC architectures. For instance, there are protocols such as ZigBee IP [59] in which, even though everyone supports IP communications, due to the meshed nature of the network only one node acts as the exit point for the rest.

There is a large number of platforms in the market that combine the topologies studied in section III.A to the point that determining whether devices are acting as gateways or not is not straightforward. Take the following four examples. Philips Hue [8] clearly belongs to tier 1 since individual light bulbs are not addressable with a unique IP, but rather communicate via ZigBee with a bridge that connects to the server. Netatmo's [38] outdoor weather stations communicate using radio frequency with their indoor counterparts to be able to send their measurements to the cloud. The reason behind this architecture is that outdoor stations might not have enough Wi-Fi coverage to communicate by themselves. Are indoor stations gateways in this case? Should they be considered equivalent to Philips Hue bridges? Third, Netatmo thermostats are composed of two devices paired using radio frequency: the relays that control boilers and furnaces, and an interface that allows users to interact with the system. This way, users can place the interface wherever suits them best and only the relays support Internet communications. Are relays acting as gateway for the interface? Would it be different if the two devices were connected through a wire instead of wirelessly? Finally, Wibeee Box Mono [52] handles up to three consumption sensors connected through a wire. Is this just a single device measuring different circuits or three devices communicating with a gateway? Once again, would it be different if they were connected wirelessly?

Algorithm 1 Algorithm for determining a platform's internet connectivity tier.

Output: The IC tier of a platform.
Define: ic as the IC level.
ic_t as the IC for each iteration of the loop.
MD a set of devices which can hold an open connection through the internet.
if MD is empty **then**
  $ic = 0$;
else
  foreach *device* in MD **do**
    if *device* has slaves **and** slaves are added modularly **and** *device* cannot fulfill its main role if there are no available slaves **then**
      $ic_t = 1$;
    else
      $ic_t = 2$;
    if $ic_t < ic$ **then**
      $ic = ic_t$;
  end
return ic;

The methodology presented in Algorithm 1 has been developed in order to objectively determine the level of Internet connectivity of these unclear cases. All four previous examples have in common that there exist slave devices that may or may not be present (i.e., they can be modularly added or removed). If after removing all slave devices, the node they used to connect to loses its main purpose, then it is a gateway and the system should be classified as tier 1. In Philips Hue, the bridge makes no sense if there are no lights bulbs to control; consequently, it is a tier 1 system. Netatmo's outdoor weather stations are indeed added modularly to an available indoor station. However, the indoor station's main role is not fulfilled thanks to any outdoor station, it will still measure every household variable as it is designed to do, so it belongs to the second tier. The third example is simpler. As the interface is not added modularly to the relays, there cannot be neither several interfaces for one relay nor several relays with a single interface, so it belongs to the second tier. Finally, Wibeee's consumption sensors belong to the first tier, as they are added modularly to Wibeee Box Mono which will be deemed useless without them.

3.2.2. Connectivity exploitation (CE)

The previous scale measured the infrastructure of the platform. This one, presented in Fig. 4, aims to study the functionality of connected systems and how they benefit from the web. It is worth highlighting that this scale does not consider remote control and monitoring as a benefit, as this is immediately achieved after connecting the system to the Internet.

- *Tier CE0*: Platforms that do not benefit from being connected to the Internet beyond remote access.
- *Tier CE1*: Platforms that exploit the advantages of being connected to their own benefit. For instance, systems might run their algorithms in the cloud –as there is more computational power available–, increasing their intelligence or reducing their time complexity. Additionally, expanding the system to the web gives it access to an immense amount of data that could be used to increase the number of inputs to the system, or even avoid some computations (on either nodes or the cloud). For example, use specialized weather

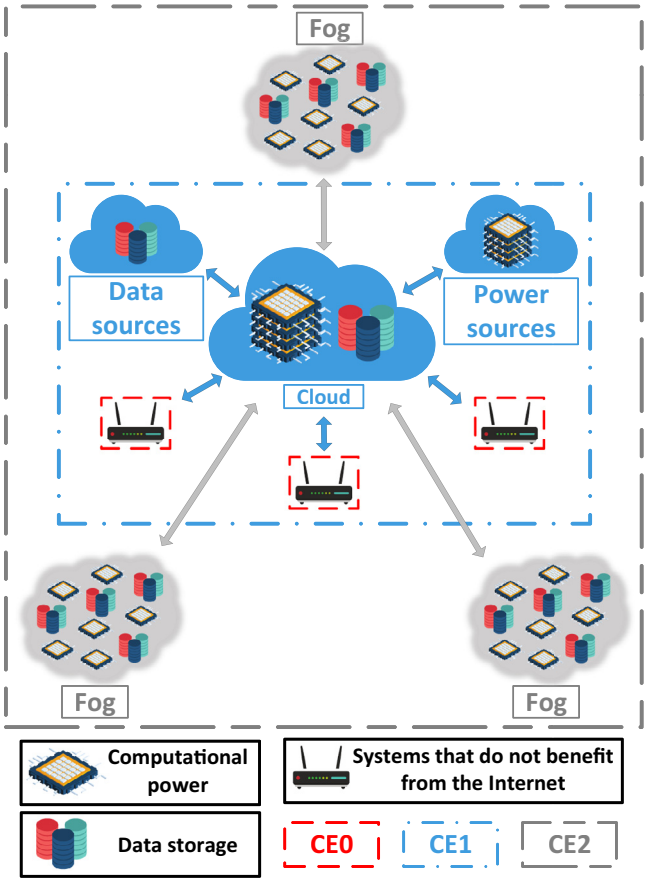

Fig. 4. Connectivity exploitation tiers. CE0: Platforms that do not benefit from being connected to the Internet beyond remote access. CE1: Platforms that benefit from being connected because they have access to additional computational power and/or new data sources. CE2: Platforms that address the limitations of previous levels in terms of latency, bandwidth efficiency, load balancing, resiliency, and security (e.g., by distributing load and information). The icon enclosed within CE0 bounds represents a full sensor/actuator network together with its local server (if the latter exists).

prediction services or retrieve data in real-time, such as energy prices.

- *Tier CE2*: This second tier addresses the limitations of previous levels in terms of latency, bandwidth efficiency, load balancing, resiliency, and security. Although the current trend is to provide users with remote control and upload as much information as possible to the cloud, this approach presents several challenges. For example, locating the centralized server in the cloud translates into higher latency and network traffic. Additionally, the fact that intelligence is centralized brings issues related to the resilience of the system, as it cannot withstand Internet blackouts or cloud server crashes.

Several of these aspects are already being tackled in the literature and there is not a unique way of dealing with them. For instance, combining cloud with the so-called fog computing could be one option. Munir et al. [60] present an architecture for a system where cloud and fog are integrated and exploit their synergies. In this study, fog computing is set apart from cloud computing through characteristics such as the proximity to end devices (the fog is located at the edge of the network) and the number and computing power of the nodes that conform them (the fog is more distributed, but the nodes are less powerful). Additionally, although fog computing and edge computing are usually treated as synonyms in the literature, in [60] edge computing is considered to be slightly more centralized. Puliafito et al. [61] state that the combination of cloud and fog results in lower latency, a reduction of bandwidth consumption, and more privacy and context awareness. Additionally, Puliafito et al. [61] analyzes several case studies where not only the fog, but also quick algorithms could become essential in order to keep the active fog node topologically near a mobile IoT node. Kelaidonis et al. [62] states that technologies such as 5G mobile will contribute to the integration of cloud computing in IoT systems. This study presents an architecture underpinned by four main components: semantic abstraction and virtualization of things, cognitive management and composition of virtual things, semantic storage system and cognitive management of services.

Fog computing is a relatively new paradigm in constant evolution. There is a lot of ongoing research attempting to provide insights into its definition, characteristics, advantages and challenges (e.g., security, scalability, or hybridization of cloud computing and IoT).

Bellavista et al. [63] present an extensive study where they compare a large number of fog solutions for IoT and provide some design guidelines for IoT applications based on fog. The authors propose a unified conceptual architecture for fog computing and a taxonomy to compare different solutions. Both cover different architectural dimensions, such as security, communications, and data management. Additionally, they analyze how scalability, interoperability or real-time responsiveness (essential to real time energy solutions) can be tackled using fog-based solutions. Chiang and Zhang [64] state that latency, bandwidth, or resiliency over intermittent connectivity for constrained devices, cannot be properly addressed with the current architectures. They suggest that the combination of fog and cloud computing could ensure uninterrupted control, computation, storage and communication along the IoT system. Jalali et al. [65] present a survey on IoT energy consumption that highlights the benefits of fog over cloud computing when developing energy efficient IoT architectures, as communicating with a remote cloud server will necessarily consume more energy than a closer fog node. Furthermore, as the computing load is distributed, fog servers will have more idle time than cloud servers due to device concurrency ratios. Ren et al. [66] present an architecture for a fog IoT ecosystem influenced by transparent computing that pleads for flexible distribution of computation and storage capabilities. The authors also address the key challenges of implementing the proposed architecture.

There are other architectures in the literature, such as the one presented by Al Faruque and Vatanparvar [17], that distribute computation responsibilities (e.g., in home energy management control panels) albeit neither flexibly nor in real-time. We adhere to the definitions found in [60,67] and will not consider them fog computing, but rather a set of Local Servers communicating with each other.

Table 1 shows the current tier of the platforms studied and reveals several clusters that classify the current state of the art. First, platforms such as GridPoint [26], Honeywell [28], or Siemens Desigo [45], which used to operate locally and that are now beginning to integrate IoT. They currently rely on an Energy Management System to monitor and control the operation of the network, being this the only device able to communicate with the cloud. Therefore, they belong to IC1 and CE0. As they start moving some of the algorithms to the cloud, since it scales better and has more computational power, they will upgrade to a CE1 level. However, to

increase their IC level they will have to change their topology. Platforms like SmartThings [68], Nexia [39], or Bosch Smart Home [20], which have an LS architecture to give more resiliency to the system but do not provide services that require complex algorithms or optimizations, are also classified as IC1 and CE0. Third comes the group conformed by IC1 systems, which chose a Gateway topology over LS, with cloud-based intelligence (CE1), represented by Cisco [22], or Wattio [50]. Fourth, there are systems evolving towards higher IoT levels with mixed topologies, where some devices leave the local network through a gateway or a local server and others connect directly with the cloud. This is the case for platforms such as Afero [16], Ayla [19], or Predix [41]. Finally, Table 1 also shows that there are a great number of platforms that already operate in tiers IC2 and CE1. Most of these platforms respond to one of the following cases: (a) systems that rely on just a single device (e.g., a thermostat or a consumption sensor) connected to a power source (so they do not need to optimize their own energy consumption aggressively); (b) industry-oriented systems; (c) systems aimed at autonomous vehicles (not included in the study). Examples of these platforms are Smapee [13], Wibeee [52], Qarnot [15], or Hum [69]. Platforms with several devices usually avoid Direct Connection topologies. Refer to Section VIII For a detailed analysis of the characteristics of the solutions analyzed.

3.3. Communications

Energy platforms usually rely on two different types of communication blocks: one dealing with back-end devices and software (referred to as *Bottom*) and another for front-end applications and other services (*Top*). As front-end applications and services are expected to run on systems with enough computational power, the most extended application protocol of the World Wide Web, HTTP, is usually employed. However, the decision is not so clear in the case of bottom blocks. There is not a single correct choice, but a range of protocols that serve different purposes. Solapure and Kenchannavar [3] analyze several IoT architectures and, based on a three layered approach (application, network, and perception), classify protocols into four different categories: application, service discovery, infrastructure, and other influential protocols. Furthermore, the selection of a particular protocol is determined by other factors, such as the three level device classification carried out in [70]:

- *Class 0* devices are very constrained in terms of memory and processing capabilities (usually sensor-like nodes). These devices are typically not able to transmit information over the Internet in a secure way, so they need the assistance of a gateway or a local server that can encrypt data before leaving the local network.
- *Class 1* devices are less restricted than Class 0 in terms of computational power. They are able to handle communication protocols specifically designed for constrained nodes, such as MQTT (Message Queue Telemetry Transport) [71] or CoAP (Constrained Application Protocol) [72], but not a full protocol stack as HTTP with SSL/TLS cipher suites.
- *Class 2* devices support most of the protocol stacks used by servers and can still benefit from lightweight protocols when running on batteries.

Table 2 puts forward the main characteristics of several communication protocols suited for IoT applications. Currently, MQTT and CoAP are drawing attention because they were specifically designed to meet IoT requirements (i.e., they have a lightweight stack). MQTT exploits one-to-many communications following a publish/subscribe (PS) paradigm. Clients are connected through a message broker that keeps track of topic and subscription lifecycles. As publishers and subscribers are decoupled, a single data object pushed by a source can reach many destinations without exhausting the node. Although CoAP also has PS capabilities [3], MQTT is specifically designed to keep the client side as simple as possible. The complexities are left to the broker, which should be the most powerful node in the network. Furthermore, MQTT suits applications whose communications are event-based, allowing for great scalability. Antonic et al. [73] present a taxonomy to compare IoT PS solutions and use it to confront MQTT and the CUPUS middleware. After studying several characteristics, such as latency, mobility, and performance, they conclude that CUPUS is more appropriate for mobile environments with context changes, whereas MQTT outperforms CUPUS in heterogeneous environments and sensor networks.

As defined in [72], CoAP is a “specialized web transfer protocol for use with constrained nodes and constrained networks in the Internet of Things. The protocol is designed for machine-to-machine applications such as smart energy and building automation”. CoAP was conceived to meet RESTful design patterns to ease the migration from HTTP-based systems. However, the client-server structure is not as clear as in MQTT, since both endpoints act as clients and servers. Palattella et al. [74] present a complete protocol stack that addresses the most important requirements of IoT platforms, from hardware to network and application layers, and propose 6LoWPAN and CoAP for the latter. After a deep analysis of CoAP and its features, such as the frame structure, methods, and caching and proxying capabilities, the conclusion is that CoAP is suitable as is for low-power applications.

When designing IoT platforms, it is important to determine which communication protocols are the most appropriate for the applications that will be developed. For the analyzed solutions, Tables 2 and 3 present a set of features that need to be taken into account when selecting communication protocols, such as the Quality of Service (QoS), or the supported security. Furthermore, systems are not limited to one protocol and could benefit from using several of them and taking advantage of their synergies. For instance, Bellavista and Zanni [75] proposes and evaluates the performance of an architecture that uses both MQTT and CoAP, claiming that for large-scale IoT scenarios MQTT is not enough and CoAP is the perfect complement to solve its limitations (i.e., direct communications with low reliability requirements).

4. Server layer

Section III analyzed the physical layer of energy platforms, which corresponds to the lowest blocks of the architecture presented in Fig. 1. This section aims to study key aspects of the processes running in the cloud, which includes not only software architecture but also its paradigm and the services it offers. Service Oriented Architecture (SOA) platforms are composed of a set of elements or blocks, each of which is responsible for performing different tasks. The selection and definition of these blocks, and the design of the interactions among them is known as software architecture. The study carried out in [76] analyzes the roles of software architecture and the characteristics of Service Oriented Architecture (SOA). SOA is a design pattern that enhances modularity by dividing services in self-contained blocks with several interfaces to interact with them. This way, services can be used to build applications without too many dependencies. One of the main benefits of SOA is the inherent improvement in the system’s scalability, due to its modularity and interoperability. This modularity allows recycling several blocks with different purposes. For instance, in the case of middlewares like Hydra [77] and Lysis [78], the blocks that conform the service-based architectures for the physical and application sides follow the same structure, with minor adaptations to meet the requirements of each side.

Table 2
Classification of different communication protocols used in the IoT domain.

	Transport layer	Paradigm	Scope	Discovery	Security	QoS	Minimum class
MQTT	TCP	Pub/sub	D2C C2C		TLS	√	1
CoAP	UDP	Req/res	D2D D2C	√	DTLS	√	1
AMQP	TCP	Pub/sub Req/res	D2D D2C C2C		TLS	√	2
DDS	TCP/UDP	Pub/sub Req/res	D2D D2C C2C	√	TLS DTLS DDSS	√	1
MQTT-SN	TCP/UDP	Pub/sub	D2C C2C		TLS	√	1
XMPP	TCP	Req/res Pub/sub*	D2C C2C	√	TLS	√*	2
HTTP	TCP	Req/res	D2C C2C		TLS	√	2
LLAP	TCP/UDP	Req/res	D2D D2C		–		1
LWM2M	UDP	Req/res	D2D D2C	√	DTLS	√	1
SSI	TCP/UDP	Req/res	D2D D2C		–		1
VSCP	TCP/UDP	Depends on transport layer	D2D D2C	√	–	–	1

* Available with extensionsD2C = Device to cloud; C2C = Cloud to cloud; D2D = Device to Device

Table 3
Classification of the studied systems according to the bottom communications used.

Platform	Technology between devices	Security between devices	Protocol between devices	Technology devices-cloud	Security devices-cloud	Protocol devices-cloud (Data format)	Local network
Afero [16]	Bluetooth	ECDH-256, AES-GCM	Custom	Wi-Fi, Cellular network	SSL	Custom	√
Al Faruque and Vatanparvar [17]	ZigBee	ND	Custom	Ethernet	ND	Custom	√
Ali-Ali et al. [18]	ND	ND	ND	Wi-Fi	SSL	MQTT	✗
Ayla [19]	Multiple (ZigBee, Z-Wave, Bluetooth)	ND	Private	Wi-Fi	SSL	Private	√
Bosch Smart Home [20]	Radio	Private	Private	Ethernet	Private	Private	✗
Altair SmartCore [21]	ND	ND	ND	ND	SSL	HTTP, MQTT (JSON, XML)	✗
Cisco [22]	Private	Private	Private	Private	Private	Private	Private
DeviceHive [23]	–	–	–	ND	SSL	HTTP, MQTT, WebSockets (JSON, XML)	√
DIMMER [1]	ND	ND	ND	ND	TLS	MQTT	–
Ecobee [24]	Radio	Private	Private	Wi-Fi	TLS	Private	Private
E-IoT [25]	AC mains, Wi-Fi, Bluetooth	ND	Private	Ethernet	TLS	Private	√
FLEXMETER [1]	ND	ND	ND	ND	TLS	MQTT	–
GridPoint [26]	Private	Private	Private	Private	Private	Private	Private
HEMS [27]	Wiring	–	Custom	Wi-Fi, Ethernet	SSL	HTTP	√
Honeywell [28]	Multiple	Private	Multiple (BACnet)	Private	Private	Private	Private
iChipNet [29]	Wi-Fi	Private	Private	Wi-Fi, Ethernet	SSL	Private	√
Insteon [30]	Radio	No	Insteon protocol	Ethernet	TLS	Insteon protocol	√
Javed et al. [31]	Radio	ND	Private	Private	Private	Private	Private
Kaa [32]	ND	ND	ND	ND	SSL	HTTP (JSON)	√
LG SmartThinq [33]	–	–	–	Wi-Fi	Private	Private	✗
LoBEMS [34]	Radio, Wi-Fi	ND	Private	Wi-Fi	SSL	HTTP	√
Lopez et al. [35]	Radio	Private	Private	Wi-Fi	SSL	MQTT, SWE, Modbus	√
MyDevices [36]	ND	ND	ND	ND	SSL	MQTT	✗
Nest [37]	–	–	–	Wi-Fi	SSL	Nest Weave	✗
Netatmo [38]	Radio	Private	Private	Wi-Fi	TLS	Private	✗
Nexia [39]	Z-Wave	AES128	Private	Ethernet	TLS	Private	✗
Particle [40]	–	–	–	Wi-Fi, Cellular network	TLS	Spark	√
Predix [41]	ND	ND	ND	ND	SSL	WebSockets	✗
Qarnot [15]	–	–	–	Wi-Fi, Ethernet	Private	Private	✗
Scheneider Wiser [42]	ZigBee	Private	Private	Ethernet	Private	Private	✗
SensorCloud [43]	ND	ND	ND	ND	SSL	HTTP (XDR)	✗
Shinde et al. [44]	ZigBee	No	Custom	Wi-Fi, Ethernet	SSL	HTTP (JSON)	√
Siemens Desigo [45]	Multiple	Private	Multiple (KNX, MBus)	Private	Private	Private	√
Smappee [13]	Wiring, Radio	Private	Private	Wi-Fi, Ethernet	Private	Private	√
SmartThings [68]	ZigBee, Z-Wave	No	Multiple	Ethernet	TLS	Private	√
ThingPlus [46]	Multiple	ND	Multiple (MQTT)	ND	SSL	HTTP, MQTT	✗
ThingWorx [47]	–	–	–	Wi-Fi	SSL	HTTP	✗
Ubidots [48]	ND	ND	ND	ND	SSL	HTTP, MQTT (JSON)	✗
Watson IoT [49]	ND	ND	ND	ND	SSL	HTTP, WebSockets, MQTT (JSON, XML, Plain text, Binary)	✗
Wattio [50]	ZigBee	Private	Private	Wi-Fi, Ethernet	TLS	Private (VPN)	✗
WattsOn [51]	Wiring	–	Voltage levels	Wi-Fi, Ethernet	TLS	Private	√
Wibeee [52]	Wiring	–	Private	Wi-Fi	AES128	Private	✗
Wink [14]	Multiple (Bluetooth, ZigBee, Z-Wave, Kidde, Lutron, Clear Connect)	Private	Multiple	Wi-Fi, Ethernet	SSL	Private	✗
Yaghmaee and Hejazi [53]	Wi-Fi	No	CoAP	Wi-Fi, Ethernet	ND	HTTP (JSON) ND = Not Defined	√

–: Does not apply.

Table 4
Classification of the studied systems according to server layer characteristics.

Platform	Type	IaaS	PaaS	SaaS	SOA	Open Source	Focus	Play Store popularity
Afero [16]	Platform	✓	✓			✗	Commercial	*
Al Faruque and Vatanparvar [17]	Platform			✓	✓	✓	Research	
Ali-Ali et al. [18]	Platform			✓	✓	✗	Research	
Ayla [19]	Platform		✓			✗	Industry	*
Bosch Smart Home [20]	Platform			✓		✗	Commercial	**
Altair SmartCore [21]	Middleware		✓		✓	✗	Commercial	
Cisco [22]	Platform		✓	✓	✓	✗	Industry	
DeviceHive [23]	Middleware	✓	✓		Microservices	✓	Research	
DIMMER [1]	Platform		✓		Microservices	✗	Research	
Ecobee [24]	Platform		✓	✓		✗	Commercial	***
E-IoT [25]	Platform			✓		✗	Research	
FLEXMETER [1]	Platform		✓		Microservices	✗	Research	
GridPoint [26]	Platform			✓		✗	Commercial	*
HEMS [27]	Platform			✓		✗	Research	
Honeywell [28]	Platform			✓		✗	Commercial	
iChipNet [29]	Platform		✓	✓		✗	Commercial	
Insteon [30]	Platform		✓	✓		✗	Commercial	***
Javed et al. [31]	Platform			✓		✗	Research	
Kaa [32]	Middleware	✓			✓	✓	Research	
LG SmartThinq [33]	Platform			✓		✗	Commercial	****
LoBEMS [34]	Platform			✓		✓	Research	
Lopez et al. [35]	Platform			✓		✓	Research	
MyDevices [36]	Middleware		✓		✗	✗	Commercial	***
Nest [37]	Platform		✓	✓		✗	Commercial	****
Netatmo [38]	Platform		✓	✓		✗	Commercial	***
Nexia [39]	Platform			✓		✗	Commercial	***
Particle [40]	Platform	✓	✓			✗	Commercial	**
Predix [41]	Platform		✓		Microservices	✓	Industry	
Qarnot [15]	Platform	✓	✓	✓		✗	Commercial	*
Scheneider Wiser [42]	Platform			✓		✗	Commercial	**
SensorCloud [43]	Platform		✓			✗	Commercial	
Shinde et al. [44]	Platform			✓		✗	Research	
Siemens Desigo [45]	Platform			✓		✗	Commercial	
Smappee [13]	Platform			✓		✗	Commercial	**
SmartThings [68]	Platform		✓	✓		✗	Commercial	*****
ThingPlus [46]	Middleware		✓			✗	Commercial	
ThingWorx [47]	Middleware		✓	✓		✗	Industry	
Ubidots [48]	Platform		✓			✗	Commercial	
Watson IoT [49]	Middleware		✓			✗	Commercial	
Wattio [50]	Platform		✓	✓		✗	Commercial	**
WattsOn [51]	Platform			✓		✗	Research	
Wibeee [52]	Platform		✓	✓		✗	Commercial	**
Wink [14]	Platform		✓	✓		✗	Commercial	***
Yaghmaee and Hejazi [53]	Platform			✓		✗	Research	

As stated in [79], there are two basic scaling models. Vertical scaling boosts resources and computational power in existing nodes, while horizontal scaling increments the number of nodes in the system. Even though both strategies allow for increased workloads, vertical scaling is constrained by physical and hardware limitations, whereas horizontal scaling could theoretically be extended to no matter how many nodes. This study also presents microservices as the solution for scaling issues in the IoT domain. In [80], SOA and microservice architectures are analyzed and compared. The microservices approach is a type of SOA where each service is defined as an individual application that is able to operate on its own. Usually, these applications are divided into three basic layers: logic, interface, and data storage. Since IoT relies on many distinct technologies, opting for a monolithic approach is not very reasonable, especially when microservices allow reusing components already present in the ecosystem [79]. On the contrary, Lan et al. [81] proposes an Event-Driven Service Oriented Architecture (EDSOA) that introduces event mechanisms in SOA. Services are not only executed when a certain event is triggered, but also to allow them to generate and receive events from other custom services through an event bus structure. Table 4 details the platforms that use SOA as their design philosophy.

Additionally, Table 4 highlights several complementary characteristics. First, the platforms' purpose: whether they were developed for or during research projects or if they are market oriented. Second, if platforms have been released as open source projects, thus offering users the opportunity to modify or extend them. Finally, their popularity according to the number of downloads in Google's Play Store App market [82]. The purpose of this characteristic is to provide an index that allows measuring the performance of the platform in terms of users' penetration.

4.1. Cloud paradigm

Cloud-based servers were originally designed to provide a particular set of services determined by the targeted end-users. This section aims to establish a connection between the design and architecture of the systems and the nature of the services provided. For instance, [83,84] use the term Energy as a Service (EaaS) to define cloud infrastructures that focus on the energy sector. This could be considered as a common characteristic among every platform analyzed. The study carried out in [85] stipulates three levels of services provided by general cloud computing that we have identified with energy platforms as follows:

- *Software as a Service (SaaS):* Clients use applications running on cloud infrastructures and are freed from the responsibility of managing and maintaining the platform. This is the only design pattern suited for non-developers, clients that use the system to fit their needs without requiring advanced IT skills. Some example platforms are Smapee [6], Wibeee [31], or SmartThinq [38].
- *Platform as a Service (PaaS):* In this case, the client is provided with an operating system, applications, and other development environments. This can be found in a large number of SaaS systems (e.g., LIFX [7], Nest [37], Philips Hue [8], or Wattio [50]) where users can use the existing platform (both software and hardware devices) to develop their own applications and add new functionalities at different levels. Those middlewares that are not open source and offer their services from the cloud also fall in this category. As Table 4 reveals, SaaS and PaaS services are usually offered together as they are not incompatible and can benefit from synergies. In other words, platforms that can be used as is but that are accessible to developers that want to customize user experience or upload new applications for third parties.
- *Infrastructure as a Service (IaaS):* It is the most basic model where the client is provided with the resources needed to deploy and run arbitrary software, such as operating systems and applications. IaaS energy platforms can be divided into two general scenarios. First, platforms such as Afero [16] or Particle [40] offer users the necessary tools and/or hardware to program devices to build their own platforms. Second, open source middlewares (usually designed with research purposes) that either require configuration through coding or allow users to perform programmatic customizations (e.g., GSN [58], HomeStar [86], or Kaa [32]).

5. Application layer

The application layer could be considered as a counterpart of the physical layer. Instead of interacting with physical devices, it communicates with human beings and other systems. Represented by the top blocks of Fig. 1, this layer allows applications to access data uploaded by the physical layer. These applications are usually employed by end-users to supervise their device network through Graphical User Interfaces (GUIs). Ideally, GUIs should be intuitive and have a gentle learning curve, but this is not trivial to achieve since it depends on the design to a large extent.

The traditional way of acquiring data is through a request-response communication within a client/server structure. Different types of clients (e.g., GUIs or smartphone and web applications) connect to the cloud's server following a specific request type or protocol, the server process incoming queries, and answers with the desired data. Another way of handling data acquisition is the publish/subscribe paradigm. As stated in [73], it has some undeniable advantages thanks to its many-to-many capabilities. In this structure, the server deals with subscription lifecycles and usually acts as a message broker. As aforementioned in Section III.C, publishers and subscribers are decoupled, and a single data object pushed by a source can reach many destinations without exhausting it (a crucial issue among constrained devices). To be able to receive push notifications, subscribers establish a permanent connection with the server, increasing performance and reducing communication latency, at the expense of a slightly increased workload on the node side. Bellavista and Zanni [75] propose a communication structure that combines both strategies, reasoning that, despite the increased latency, for highly constrained or battery powered devices that receive commands (e.g., actuators), a request-response procedure is more suitable.

For instance, imagine a device running on batteries that can be turned on and off via a smartphone application. In a publisher-subscriber setup, the smartphone application would push the command to the server, which in turn would pass it to every device listening to that topic. In a request-response configuration, constrained devices can subscribe to different topics but without keeping the connection with the server open. They would rather connect periodically to fetch any pending notifications allowing them to enter sleep mode and save energy between connections. Table 5 classifies the methods supported by the studied systems.

6. Security

In light of recent global cyber-attacks, security has become part of the backbone of every system connected to the Internet. However, with the proliferation of IoT platforms new challenges need to be addressed, since this kind of platforms can be attacked in different ways and across different levels. Most of these levels are collected in [87], which classifies twenty security considerations applied to cloud-enabled IoT systems in eight different categories.

The first one is related to cloud access. Every message must be encrypted (e.g., using TLS or DTLS) to prevent any sensitive data from being exposed during transmission. Additionally, the server must monitor every connection to the system through user accounts and permissions. It must validate not only who is sending the message but also if the source has the privileges required to retrieve or post data. For example, this can be accomplished using OAuth. OAuth is widespread across the Internet and authenticates users through unique tokens. Table 5 reveals that approximately 25% of the platforms analyzed resort to this mechanism. Nonetheless, some platforms (e.g., HYDRA [88]), have developed custom authentication flows and methods.

Another group deals with data management and integrity in the cloud. IoT platforms must identify and protect sensitive data from unauthorized access or modification. For instance, energy-related platforms handle data that could disclose consumption patterns that can indicate when dwellers are away. One of the considerations presented in [87] states that this kind of data should be encrypted (independently of the encryption used to transmit the whole message securely) just in case it is intercepted. However, this could become excessively complex as the server, devices, and applications would have to manage encryption keys that need to be revoked and renewed periodically. As aforementioned, servers must control the access to the system. This does not only affect front-end applications, but also "things", the weakest nodes in terms of computational power and, consequently, one of the preferred targets for cyber-criminals. Therefore, the cloud must also identify and control any device that connects to it. For example, if device privileges are not handled properly, someone could pass off as a thermostat and take advantage of a backdoor in order to steal sensitive data that hardware agents should not have access to.

Opposite to the previous study, more focused on the cloud side, Mohamad Noor et al. [12] present a survey of current solutions applied to IoT security. It provides insights into several security issues regarding communications between the perception layer and the rest. The authors study several security mechanisms found in the literature: authentication, encryption, trust management, and secure routing, among others. Analyzing security vulnerabilities found among the studied systems, the conclusion is that several of them stem from devices' computational and memory constraints. These limitations prevents them from encompassing full implementations of effective authentication and encryptions mechanisms. Furthermore, manufacturers usually apply hard-coded credentials or passwords that lead to authentication failures, as those credentials are easier to compromise. The survey highlights that research on IoT security is focusing on improving

Table 5
Classification of the studied simulators according to top characteristics.

Platform	Request Response	Publish Subscribe	Available GUIs	Application layer communication protocol	OAuth2
Afero [16]	✓		Smartphone, Web	HTTP(JSON)	✓
Al Faruque and Vatanparvar [17]	✓		Web	HTTP	✗
Ali-Ali et al. [18]	✓		Smartphone	HTTP	Private
Ayla [19]	✓		Smartphone, Web	Private	✓
Bosch Smart Home [20]	Private	Private	Smartphone	Private	Private
Altair SmartCore [21]	✓	✓	Web	HTTP, MQTT (JSON, XML)	API key
Cisco [22]	✓		Web	Private	Private
DeviceHive [23]	✓	✓	Web	HTTP, MQTT (JSON, XML)	✗
DIMMER [1]	✓	✓	Smartphone, Web	HTTP	Private
Ecobee [24]	✓		Smartphone, Web	HTTP (JSON)	✓
E-IoT [25]	✓	✓		HTTP	✗
FLEXMETER [1]	✓	✓	Smartphone, Web	HTTP	Private
GridPoint [26]	✓		Smartphone, Web	Private	Private
HEMS [27]	✓		Web	HTTP	Private
Honeywell [28]	✓		Web	Private	Private
iChipNet [29]	✓		Smartphone	Private	Private
Insteon [30]	✓		Smartphone	HTTP	Private
Javed et al. [31]	Private	Private		Private	Private
Kaa [32]	✓		Web	HTTP (JSON)	✓
LG SmartThinq [33]	Private	Private	Smartphone, Web	Private	✓
LoBEMS [34]	✓			HTTP	✗
Lopez et al. [35]	✓	✓	Web	HTTP	Private
MyDevices [36]		✓	Smartphone, Web	MQTT	✗
Nest [37]	✓	✓	Smartphone, Web, Device	HTTP (JSON)	✓
Netatmo [38]	✓		Smartphone, Web	HTTP (JSON)	✓
Nexia [39]	Private	Private	Smartphone, Web, Device	Private	✓
Particle [40]	✓	✓	Smartphone, Web, IDE	HTTP (JSON)	✓
Predix [41]	✓		Web, IDE	HTTP(JSON)	✓
Qarnot [15]	Private	Private	Smartphone	Private	Private
Scheneider Wiser [42]	Private	Private	Smartphone, Web	Private	Private
SensorCloud [43]	✓		Web	HTTP (XDR)	✓
Shinde et al. [44]	✓		Web	HTTP (JSON)	
Siemens Desigo [45]	✓		Web	Private	Private
Smappee [13]	Private	Private	Smartphone, Web	Private	Private
SmartThings [68]	✓	✓	Smartphone, Web, IDE	HTTP (JSON/SOAP)	✓
ThingPlus [46]	✓	✓	Web	HTTP	✓
ThingWorx [47]	✓		Web	HTTP	Private
Ubidots [48]	✓	✓	Web	HTTP, MQTT (JSON)	✓
Watson IoT [49]	✓	✓	Web	HTTP, MQTT (JSON, XML, Plain text, Binary)	Custom
Wattio [50]	✓		Smartphone, Device, Web	HTTP (JSON)	✓
WattsOn [51]	✓		Smartphone	HTTP (JSON)	✓
Wibeee [52]	✓		Smartphone, Web	HTTP (JSON)	Custom
Wink [14]	✓	✓	Smartphone	HTTP (JSON)	✓
Yaghmaee and Hejazi [53]	✓		Web	HTTP (JSON)	API key

lightweight mechanisms for constrained devices. However, it also reveals that most of the current mechanisms are applied to network and application layers rather than to perception.

As Singh et al. [87] is focused on the cloud side of the system, it does not tackle every vulnerable point of IoT platforms. Gateway, or Local Server architectures (Section III.A) that, besides communicating with the cloud, also handle local communications with physical devices, are not covered. These communications are far less dangerous as they do not leave the local network, but they need to be protected as well. Otherwise, someone could take over automated devices of neighbors, or public building nearby. As shown in Table 3, it is not strange to find platforms that do not cypher these local network messages due to the increased complexity (e.g., Wink [14], or Kontakt [89]). This observation has also been highlighted in other surveys and studies [4,90,91]. In addition, Mohamad Noor and Hassan [12] states that, ideally, devices should incorporate dedicated security hardware, such as cryptographic code processors or security chips (e.g., Trusted Platform Modules (TPM) [92]), which can be used to guarantee data integrity regardless of the platform the device will connect to.

After analyzing several IoT middlewares available in the literature, Ngu et al. [4] conclude that existing systems can be classified into three different groups: Platforms that do not address security, platforms that address some security issues, and platforms that offer theoretical security models but do not show any real implementations. Besides, Khorshed et al. [90], analyze security in a three-layered approach: things, cloud, and big data, and state that as far as things are concerned, security is almost nonexistent among current platforms, which is rather worrying. This work also presents a machine learning algorithm trained to identify different kinds of cyber-attacks, such as Man in the Middle or DDoS. Furthermore, Sain et al. [91] studies not only current IoT-related projects (ranking their security levels), but also different wireless communication technologies, such as Bluetooth or ZigBee. The previous articles coincide in the possible solutions that could be applied to solve all these security issues. For instance, Ngu et al. [4] suggests implementing lightweight device authentication and end-to-end security, while [91] remarks once again the importance of controlling access to the system and keeping sensitive data protected. Notwithstanding, security will always be a challenge as it is impossible to develop a system completely free of vulnerabilities.

7. Services and interoperability

The previous sections have discussed the blocks presented in Fig. 1 from a technical perspective. This section deals with the ser-

vices offered to end-users. As mentioned in Section II, these platforms may be a manner of increasing the understanding of users in the energy market. However, this is not an easy task and could be assisted by the development and implementation of different intelligent, but intuitive, algorithms running on the cloud and/or on devices. An example could be a thermal control that not only operates radiators and heat pumps, but also make the blinds lift to heat the house using sunlight instead of electric power. Increasing the number of controls operating on a same device could result in synchronization issues and conflicting commands. As happens with energy markets, there should be different kinds of management algorithms in energy platforms. For instance, to plan the operation of every device in advance (e.g., on a daily basis), predict the expected behavior of users, or react smartly to unexpected events. Table 6 distinguishes three main categories of algorithms: control, planning, and learning.

One of the most widespread approaches to improve the quality and number of these intelligent algorithms is to let users shape the platform to some extent. Many of the studied platforms have open Application Programming Interfaces (APIs), Software Development Kits (SDKs) and developer portals, to allow third parties to create custom applications and services. Some platforms, such as Altair SmartCore (formerly Carriots) [21] or onesait (formerly Sofia2) [93] restrict the availability of these custom applications to developers, while others such as SmartThings [94] or ThingSpeak [95], have deployed a custom application market where users can share their extensions. This is a winning strategy both for users and platforms as they put users' creativity to work in benefit of the whole community.

Another key element is interoperability. Soursos et al. [96] highlight some key aspects that need to be addressed to achieve full cross-domain interoperability, such as the need for collaboration by sharing resources or the creation of IoT alliances. An example of this would be recent alliance between Apple, Google and Amazon to implement a new IP-based standard to increase compatibility between smart home devices [97]. The system is evolving towards a scenario with fully interconnected platforms. However, current platforms are implementing their own custom solutions to tackle interoperability. Systems such as Fiware [98] or Legato [99] attempt to offer complete solutions to different applications with the main objective of achieving standardization in the IoT domain. For example, within Fiware's solutions catalogue there are sections for message brokers, data handlers, or even security enablers. [100] and [101] discuss the approach pursued by the H2020 project symbIoTe. Zarko et al. [100] introduce the concept of IoT platform federations and their role within symbIoTe's architecture. The authors define four compliance levels for IoT platforms operating in symbIoTe's ecosystem. In the first level, platforms open up only its interworking interface to third parties to offer its resources. Platform federations are enabled in the second level, as platforms index content of other platforms and make it available for their applications and services natively. In the third level, platforms integrate components to simplify the integration and reconfiguration of devices within local spaces, preventing vendor lock-in. The fourth level enables device roaming and the interaction of smart devices with visited smart spaces. Smart spaces are detailed in [101]. One extrapolation that can be deduced from the previous studies is that the number of proposed solutions to the interoperability problem is increasing. This might result paradoxical as the only way of reaching full interoperability is by platforms adopting the same solution. In other words, one of the solutions must prevail above the others. This is just a natural effect of the youth of IoT technologies and platforms; they are evolving quickly and, with each step, solutions are deprecated while new ones appear. Gambi et al. [102] review different projects aimed at living environments (e.g., AllJoyn or DomoInstant) and how they have approached interoperability. They conclude that an outstanding solution has not been reached yet. Zitnik et al. [103] present an architecture that is compliant with oneM2M's standards. IoT standards are appearing in four different layers: application, service, network, and access. The first two, application and service, are essential for developing new frameworks that allow the interconnection of platforms at a semantic level. The networking layer allows communications at a more technical level (e.g., device to platform, or intraplatform). The access layer deals with users and permissions management. The study concludes, once again, that no outstanding solution has been reached yet. However, alike communication protocols, it is difficult that one solution stands above the rest, as each is conceived with its own purpose, and has its own advantages and disadvantages. This will probably lead to a scenario where a set of standard solutions will coexist allowing developers to select the one that suits their application best. For instance, Fiware [98] and oneM2M [104] provide architectural blocks, and standard APIs and data formats for the network and service layers aimed at IoT platforms. In the case of the network layer, AllJoyn [105] and IoTivity [106] have appeared as solutions for inter-device communications. The main objective of the IEEE P2413 project [107] is to define "an architectural framework for the Internet of Things (IoT), including descriptions of various IoT domains, definitions of IoT domain abstractions, and identification of commonalities between different IoT domains".

To achieve interoperability in the energy sector, the system must rely on open devices and services. Why deploy several sensor networks managed by distinct platforms across the energy system? What if one of them could offer already processed data to the rest or even make its devices available directly? To accomplish this, a standard device-naming system is necessary to find devices operating in a network. Heo et al. [108] approach discoverability by proposing a scanning procedure based on OliotONS (Object Name Service). Koo and Kim [109] take a deeper approach by reviewing several identification systems and introducing the concept of device ID translator —named Device DNS (Device Name System). It uses oneM2M's naming structure as its base to relate each naming policy to it. This way, manufacturers are involved in the process as they can introduce their naming design into the Device DNS database. Once discovery is performed, the next step is to communicate with the found devices.

The traditional way of integrating devices of different manufacturers is to parse their protocols so the rest of the system understands it. Some general purpose platforms, like Savant Systems [9] or Insteon [30], rely on this approach to allow integrating other devices into one single control interface. A simpler way is to offer data through APIs so that any platform has access to it, decoupling physical and application layers. Although this is a basic yet powerful solution for several smart-city information providers (e.g., air quality or parking), the intrinsic requirements of the energy sector increase the difficulty of developing real-time applications.

For example, lets imagine two platforms: platform A —from now on referred to as PA—, managing a sensor/actuator network, and platform B —PB— that intends to access PA's data to solve a contingency that might cause an outage. To succeed, PB must act within seconds. In this time-lapse, the process represented in Fig. 5 must take place. First, the sensor network performs a measurement and sends it to PA's cloud where it is processed and stored in a database. In this step, the bottom architecture of PA impacts parameters such as communication latency and efficiency. Second, once a set of measurements is stored, PA might process its raw data and aggregate it into another storage, the one accessed by APIs. Third, PB connects to PA's server to request the data. After that, PA's server will process PB's request, access the dataset

Table 6
Classification of the studied systems according to the services provided.

Platform	Learning	Planning	Control	Connect custom software	Connect Custom Hardware	Open API	REST	SDKs	Compatibility with Third Parties	Extra Services
Afero [16]				Cloud	√	√	√	√	–	OTA
Al Faruque and Vatanparvar [17]			√		√	✗	–	✗	Hardware	–
Ali-Ali et al. [18]									–	Bill Savings
Ayla [19]				Cloud	√	√	√	√	–	Data analytics, OTA
Bosch Smart Home [20]			√			✗	–	✗	–	Bill savings
Altair SmartCore [21]				Cloud	√	√	√	√	Hardware	–
Cisco [22]				Cloud	√	✗	–	✗	–	Data analytics
DeviceHive [23]				Cloud	√	√	√	√	Hardware, Alljoyn	Data analytics, Machine learning
DIMMER [1]				Cloud	√	√*	✗	✗	–	Geographic correlation simulation
Ecobee [24]		√	√	Cloud		√	√	✗	Homekit, Amazon Alexa, Samsung SmartThings, IFTTT	Bill savings
E-IoT [25]						✗	–	✗	–	AC/DC wiring communication
FLEXMETER [1]				Cloud	√	√*	√	✗	–	Demand response
GridPoint [26]			√			✗	–	✗	Hardware	Data analytics
HEMS [27]	√					✗	–	✗	–	Machine learning
Honeywell [28]			√			✗	–	✗	Hardware	Data analytics
iChipNet [29]				Cloud		√	Private	✗	–	–
Insteon [30]		√	√	LAN, Cloud		√*	√	✗	Home automation systems	–
Javed et al. [31]	√		√			✗	–	✗	–	Machine learning
Kaa [32]				Cloud	√	√	√	√	Hardware	Data analytics
LG SmartThinq [33]			√			Under development	–	✗	–	–
LoBEMS [34]	√		√			✗	–	✗	–	Bill savings
Lopez et al. [35]						√	√	✗	–	–
MyDevices [36]		√		Cloud	√	√	√	✗	Hardware	–
Nest [37]	√	√	√	Cloud	√	√	√	√	Home automation systems	Bill savings, Security algorithms
Netatmo [38]		√	√	Cloud		√	√	√	Homekit, Amazon Alexa	Bill savings, Weather Forecasts
Nexia [39]			√			✗	–	✗	Home automation systems	–
Particle [40]				Cloud	√	√	√	√	–	OTA
Predix [41]	√			Cloud	√	√	√	✗	–	Data analytics, Machine learning
Qarnot [15]			√	Cloud		✗	–	✗	–	Bill savings, Cloud computing heating, Intrusion detection
Scheneider Wiser [42]			√			✗	–	✗	–	–
SensorCloud [43]				Cloud	√	√	√	√	–	Data analytics
Shinde et al. [44]				LAN		✗	–	✗	–	–
Siemens Desigo [45]			√			✗	–	✗	Hardware	Data analytics
Smappee [13]	√					✗	–	✗	IFTTT, Stringtify, Comfort Plugs	Bill savings, Appliances electric identification

(*continued on next page*)

Table 6 (*continued*)

Platform	Learning	Planning	Control	Connect custom software	Connect Custom Hardware	Open API	REST	SDKs	Compatibility with Third Parties	Extra Services
SmartThings [68]		√	√	LAN, Cloud		√	√	√	Home automation systems	Internal SmartApps
ThingPlus [46]				Cloud	√	√	√	√	Hardware	Data analytics
ThingWorx [47]				Cloud	√	√*	√	√	–	Data analytics
Ubidots [48]				Cloud	√	√	√	√	Hardware	Data analytics
Watson IoT [49]				Cloud	√	√	√	√	Hardware	Data analytics, Blockchain, Weather Services
Wattio [50]		√	√	Cloud		√	√	✗	IFTTT	Bill savings, Presence Simulator
WattsOn [51]						✗	–	✗	–	–
Wibeee [52]	√			Cloud		√	√	✗	–	Bill savings, Appliances electric identification
Wink [14]			√	Cloud		√	√	✗	Home automation systems	–
Yaghmaee and Hejazi [53]				LAN		√	√	✗	–	–

* Account required; **Only LAN–: Does not apply

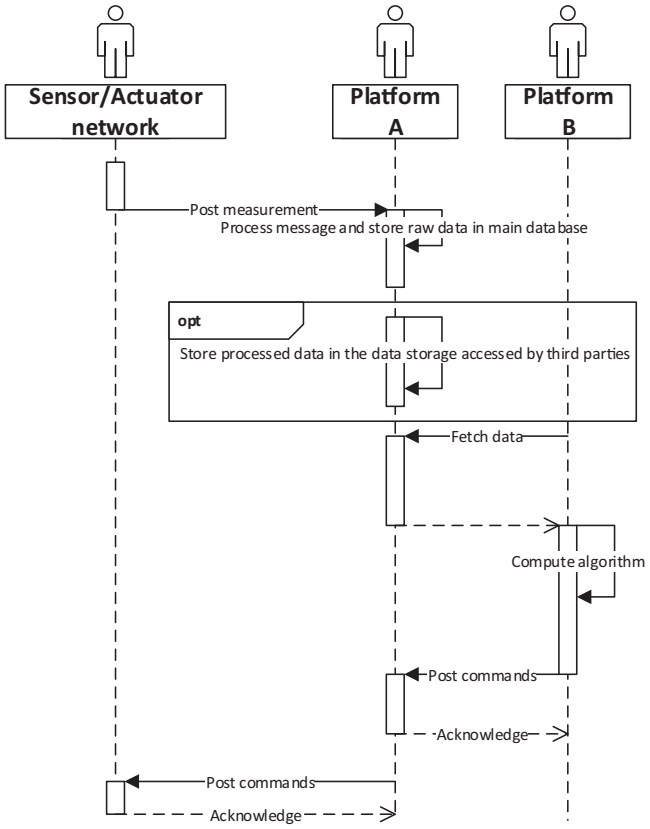

Fig. 5. Sequence diagram of the message exchange between two platforms: Platform A, owner of energy data needed by Platform B who intends to solve a contingency.

through a cache or a Database Management System (DBMS) and answer back. Then, PB receives the data and triggers a computation to calculate the action that must be carried out. Once the algorithm has finished, PB connects another time to PA to send the obtained results. Finally, PA forwards the received commands to the actuators involved.

This whole process must take place within seconds to ensure grid stability and quality of service. For this reason, efficient communications are key to platforms that strive for real-time interaction with the energy sector. Most APIs offer their data in standard formats, such as JSON or XML, that are light enough for most applications due to their undeniable advantages, such as readability and the use of key-value pairs and hierarchical trees. However, some energy-oriented applications might require lighter protocols or even binary formats.

Another trend among IoT energy platforms are those targeting makers. Do-It-Yourself (DIY) systems provide toolkits designed to monitor, test, and operate "things" to allow developers to build their own custom hardware and/or software. Obviously, the vast majority of these systems are middlewares, since their objective is to ease the development of custom platforms by offering some of the most common blocks. Additionally, there exist platforms, such as Afero [16] or Ayla [19], whose business model is to assist current companies in their integration of IoT. For instance, traditional companies that operate in the energy sector and want to embrace this technology and move the core of their businesses to the Cloud. Platforms that permit testing and operating hardware have also a great potential in the educational and training sector, as users can run simulations and emulate how their devices would behavior in a real setup.

8. Results

After reviewing the main characteristics of 44 platforms and middlewares, this section carries out a statistical analysis with a view to determining which architectures, technologies, and services cannot be missed out when developing an energy platform.

8.1. Physical layer

Energy platforms are attempting to take a holistic approach to energy management that combines all four services in an attempt to take advantage of their synergies. Home automation services are widespread, being present in the three most common configurations (H-T, H-T-E and H). Conversely, DES and storage services are hardly considered, probably because storage devices must be controlled properly to make them profitable and avoid battery aging [110]. This typically translates into implementing complex and robust algorithms and optimizations to forecast the weather and day-ahead energy consumption. By contrast, home automation services usually cover devices that do not need neither maintenance nor management (e.g., light bulbs or automated blinds). Several combinations are not even considered by any platform. Some of them would not make much sense, such as DS or T-DS. As it is hard to optimize DES and storage without electric services to monitor consumption, DS and E services should usually appear together. However, there are other combinations that could have a greater presence, such as H-E, T-E or T-E-DS. Eight platforms exploit two types of energy services, and nearly all of them handle H-T services, a combination with few synergies. Consumption information could assist the operation of other devices, such as radiators or heat pumps. Apparently, there is an invisible wall between thermal and electric devices.

Regarding Internet connectivity (one of the IoT levels presented in section III.B), the most extended tier is the first one. It seems like the first and second tiers are levelling as more systems are adopting architectures where every device is identified by a unique IP address. However, in the second tier there are several platforms, such as Wibeee [52] that are usually underpinned by just a single device per household. Deploying another device to act as gateway for the first one would increase costs unnecessarily. In the case of CE levels, the exploitation of connected resources is higher among energy platforms than in other IoT environments where most systems still do not take advantage from the network and use cloud services to provide mainly data storage, remote access, and user management. This translates into a higher presence of CE1 platforms. Several systems are currently in a transitioning state evolving towards higher tiers. For example, Insteon [30] and Wattio [50] have coexisting topologies with some devices connecting through a LS or a gateway and some accessing the cloud directly. It is interesting to point out that there are currently no tier 2 platforms.

The transition state of the literature can be observed in the fact that there are more platforms using Mixed than pure Gateway topologies. As happened with tier 2 of connectivity exploitation, no systems are using a Hybrid topology. Even though this architecture could result in an improvement of the ecosystem's resiliency thanks to the distributed intelligence, it is also true that it is harder to deploy. There is no clear dominating protocol for inter-device communications, whereas for Internet communications, platforms usually rely on more standard solutions such as HTTP. As analyzed in section VII, this dispersion is an issue in terms of interoperability, as none of the available protocols is settling as a one-and-only solution for energy platforms. Furthermore, most of the studied solutions use private protocols to connect their devices.

8.2. Security

Platforms are aware of the importance of protecting sensitive information that is shared over the Internet. Among the studied ecosystems, fortunately none of them relies on unencrypted communications between devices and cloud However, the scenario changes drastically for inter-device messages. The number of platforms that do not specify how they protect their local communications increases as nearly 80% of the systems are not completely open about their security settings or have known issues at some level —usually the physical layer. This might be explained by the usage of meshed networks and computational power constraints.

8.3. Services

Section VII discussed how systems could benefit from the usage of three main types of algorithms (i.e., control, planning and learning algorithms) to optimize device behavior. However, most systems have discarded implementing complex algorithms for the time being. Of those systems which have, approximately 65% of the solutions consider just a single type. In other words, platforms sacrifice customization for simplicity, synergies are not exploited and, most importantly, the lack of computational intelligence might translate into less robustness under specific circumstances.

The most implemented types are learning and control algorithms. Learning algorithms intend to customize user experience, adapting to daily routines and habits, for example. As this type of algorithm is the most visible to the user, it is not surprising that it is the most common option among current solutions. Control algorithms monitor one or more dimensions of the household and prevent them from going out of bounds. The most implemented type of control algorithm observed are thermal controls (e.g., a thermostat in charge of the temperature of a room). Planning algorithms might be the most complex of the three types as they require predicting user preferences, weather conditions, and energy prices, among other variables.

As mentioned in Section VII, the number of platforms that are opening their infrastructure is increasing. This way, developers can expand the system uninterruptedly. Only 40% of the studied platforms support custom hardware connection whereas 60% support the connection of custom software. An explanation could be the fact that developing an application to monitor a household is easier than designing, building, and testing physical devices.

9. Guidelines

After analyzing the current trends among IoT platforms in terms of architectural layers and services, this section aims to assist future developments by explicitly stating lessons learned and providing design guidelines for IoT energy platforms. Finding and exploiting synergies is probably the most important takeaway. In order to increase energy efficiency or reduce energy bills, every source of data is useful. For example, monitoring the evolution of energy intraday prices and the consumption of the whole household might assist the management of thermal devices.

9.1. Bottom architecture

There is not an ideal solution that suits every possible scenario and context. If devices are highly constrained in terms of computational power (e.g., class 0), or their consumption needs to be highly optimized, a Gateway (G) becomes mandatory. Direct Connection (DC) architectures may assist in keeping manufacturing and deployment costs low as they reduce the number of indispensable devices.

The location and connectivity of the devices should also be taken into account. Devices installed in environments with low Internet coverage (e.g., a basement), or that may operate disconnected from the Internet for extended periods of time, may benefit from a Local Server (LS) architecture, as the intelligence is always reachable through the local network.

Another important factor is resiliency and how critical each node is for the rest of the system. Let's compare a weather monitoring station that stops measuring the outdoor temperature with a thermostat that controls the temperature of a hot water cylinder. One is a data source, the other interacts directly with users, which always has a higher priority. Resiliency can be improved with distributed intelligence (e.g., by using a Hybrid (H) architecture), or by combining fog and cloud computing.

In IoT platforms, the decisions made in a layer affect the rest of them. For example, the centralized server will need less computational power if the selected bottom architecture is a LS. However, this architecture also translates into synchronism issues between local and centralized servers. Several of the studied platforms have opted to avoid these problems by moving the intelligence to local servers. However, these systems can be difficult to debug as everything happens within the local network.

9.2. Communications

Each protocol suits different applications and purposes. For instance, MQTT could be a good fit for systems with a large number of devices, or applications related to smart-cities where subscriptions would allow new devices to easily acquire information from already deployed hardware. CoAP may suit platforms that handle constrained devices in a RESTful structure as a substitute for HTTP, or for inter-device communications, which would be difficult to achieve using MQTT or HTTP. As aforementioned, platforms do not have to bind to a single solution and might consider implementing different protocols in distinct situations.

9.3. Server layer

This is not a design decision but a consequence of the niche or business model that the platform pursues. SaaS platforms are closed solutions for clients who seek ease of management and reduced maintenance complexity. PaaS offer specialized clients a platform they can somewhat customize. This intermediate strategy might be a way of increasing customer loyalty, as their self-made ecosystems are underpinned by the platform. Finally, IaaS platforms are probably the most open and flexible alternative, yet the most difficult for developers to work with.

9.4. Security

Security is always essential when designing any type of system, but it becomes even more important in IoT platforms due to the many potentially vulnerable points, as mentioned in Section VI. There are three main things that cannot be forgotten when designing the security of an IoT platform. First, the encryption of every communication message: inter-device, device-to-server (this one is affected by constrained devices) or server-to-server. Second, user and device management. On the one hand, user credentials need to be secured and users' access must be restricted according to their permissions; on the other hand, devices' connections need to be mutually authenticated to prevent cyber-attacks on either end of the socket. Finally, it is important to protecting user data from being compromised, not only during transmission, but also when it is processed and stored in the cloud.

9.5. Interoperability

Even in early stages, designers should be aware of current design guidelines for interoperability. Closed platforms might not need to worry about integrations with other systems, but it is still desirable to follow the guidelines to enable the use of available solutions (open source or not) that might reduce development periods and runtime errors. For example, offer RESTful APIs or standard protocols instead of creating and relying in custom ones. IoT platforms' interoperability affects every layer and it is difficult to be open in all of them. For instance, it is not the same making a device suitable to be handled by different platforms than to make platforms handle devices from different manufacturers.

10. Conclusion

This paper targets three audiences. First, for those interested in comparing energy platforms, a new methodology that aims to classify and analyze the degree of adoption of IoT technologies is presented. This methodology presents two different classifications: Internet connectivity and connectivity exploitation. The former analyzes the communications' topology and the latter studies the benefits obtained by the systems as a result of being connected to the Internet. Second, for those interested in selecting an energy platform, the study carries out a classification of several energy platforms, available in the literature and in the market, highlighting their strengths and weaknesses. Finally, for those interested in building new platforms, the study presents a comprehensive review of energy platforms that updates and extends earlier works by analyzing systems from the perspective of every hierarchical level of their architecture, evaluating every block necessary to develop a full operative energy platform and the different options available for applications.

This review reveals that systems are in the process of transitioning towards more efficient versions where the integration of IoT is higher. The standard solution is designed to provide home automation services using a mixed architecture. It relies on meshed networks to communicate (tier IC1) and are starting to use the Internet for more than just providing remote access to devices and information (tier IE1). The transition should end up in a scenario where the predominant architectures are Direct Connection and Hybrid, combined with spare Mixed or meshed solutions when necessary due to physical constraints or design requirements. Additionally, most energy platforms should rely on external data sources and computational power to increment the efficiency of management algorithms to operate the system in real-time. Some of them should also distribute the computational load across all available devices to increase the efficiency of the communication network.

However, there are still several challenges that energy platforms are facing. First, there are crucial security issues that should be solved in inter-device communication networks, as most platforms do not encrypt them. To solve this, it is necessary to prioritize end-to-end security and protect sensitive data during its whole lifecycle [4,91]. Second, interoperability between energy platforms is essential for the development of new smart-city application and services. Nonetheless, platforms are approaching this issue individually, which translates in a group of divergent solutions and protocols, such as different naming interfaces or different APIs for each energy platform [102]. Third, most systems have opted to prioritize simplicity keeping the architecture as straightforward as possible in terms of functionality, instead of implementing complex algorithms that require convoluted developments and maintenance processes. For instance, the reduction of users' energy bills and the optimization of DERs' lifetime, such as batteries or photovoltaic panels.

The presented review allows the abstraction of common features and characteristics and could represent the first step in standardizing the development of new faster and simpler energy solutions. As the study has revealed, there are similar energy solutions operating on the market that only differ on specific details and that have been designed and developed simultaneously. This could be avoided by abstracting the studied architectures to obtain standard architectures with sets of configurations for specific constraints. This abstraction could also stem the development of ecosystems designed to narrow the gap between energy simulators and real-time applications.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Miguel M. Martín-Lopo: Investigation, Methodology, Visualization, Writing - original draft. **Jaime Boal:** Supervision, Visualization, Writing - review & editing. **Álvaro Sánchez-Miralles:** Supervision, Funding acquisition.

References

[1] E. Patti, A. Acquaviva, IoT platform for Smart Cities: requirements and implementation case studies, in: Proceedings of the IEEE 2nd International Forum Research Technology Society and Industry Leveraging a better tomorrow, 2016, pp. 1–6.
[2] L. Atzori, A. Iera, G. Morabito, The Internet of Things: A survey, Comput. Netw. 54 (15) (2010) 2787–2805.
[3] S.S. Solapure, H. Kenchannavar, Internet of Things: A survey related to various recent architectures and platforms available, in: Proceedings of the International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2016, pp. 2296–2301.
[4] A.H.H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, M.Z. Sheng, IoT middleware: a survey on issues and enabling technologies, IEEE Internet Things J. X (X) (2016) 1–1.
[5] P. P. Ray, "A survey on Internet of Things architectures," J. King Saud Univ. Comput. Inf. Sci., vol. 30, no. 3. King Saud bin Abdulaziz University, pp. 291–319, 01-Jul-2018.
[6] G. Bedi, G.K. Venayagamoorthy, R. Singh, R.R. Brooks, K.C. Wang, Review of Internet of Things (IoT) in electric power and energy systems, IEEE Internet Things J. 5 (2) (2018) 847–870.
[7] L. Labs, "LIFX." 2019. Available: http://www.lifx.com/. [Accessed: 16-Dec-2019].
[8] "P. Hue." 2019. Available: http://www2.meethue.com. [Accessed: 20-Dec-2019].
[9] "Savant Systems." 2019. Available: https://www.savant.com/. [Accessed: 20-Dec-2019].
[10] Litmus Automation, "Loop." 2019. Available: http://www.litmusautomation.com/. [Accessed: 27-Dec-2019].
[11] B. Dupont, R. Belmans, Residential Demand Response Based on Dynamic Electricity Pricing: Theory and Practice, PhD Thesis, KU Leuven, 2015.
[12] M. Binti Mohamad Noor, W.H. Hassan, Current research on Internet of Things (IoT) security: a survey, Comput. Netw. 148 (2019) 283–294.
[13] "Smappee." 2019. Available: http://www.smappee.com. [Accessed: 19-Dec-2019].
[14] Wink Labs Inc, "Wink." 2019. Available: https://www.wink.com/. [Accessed: 20-Dec-2019].
[15] "Qarnot." 2019. Available: https://www.qarnot.com/. [Accessed: 19-Dec-2019].
[16] Afero Inc, "Afero." 2019. Available: https://www.afero.io/. [Accessed: 15-Dec-2019].
[17] M.A. Al Faruque, K. Vatanparvar, Energy management-as-a-service over fog computing platform, IEEE Internet Things J. 3 (2) (2016) 161–169.
[18] A.R. Al-Ali, I.A. Zualkernan, M. Rashid, R. Gupta, M. Alikarar, A smart home energy management system using IoT and big data analytics approach, IEEE Trans. Consum. Electron. 63 (4) (2017) 426–434.
[19] Ayla Networks Inc, "Ayla." 2019. Available: https://www.aylanetworks.com/. [Accessed: 15-Dec-2019].
[20] "Bosch Smart Home." 2019. Available: https://www.bosch-smarthome.com/. [Accessed: 15-Dec-2019].
[21] Altair, "Altair SmartCore." 2019. Available: https://www.altairsmartworks.com/. [Accessed: 15-Dec-2019].
[22] Cisco, "Cisco." 2019. Available: https://www.cisco.com/c/en/us/solutions/internet-of-things/overview.html. [Accessed: 15-Dec-2019].

[23] DataArt Solutions, "DeviceHive." 2019. Available: https://devicehive.com/. [Accessed: 15-Dec-2019].
[24] "Ecobee." 2019. Available: https://www.ecobee.com/. [Accessed: 15-Dec-2019].
[25] D. Di Zenobio, K. Steenhaut, S. Thielemans, M. Celidonio, An IoT platform integrated into an energy efficient DC lighting grid, in: Proceeding of the Wireless Telecommunications Symposium, 2017, pp. 1–6.
[26] GridPoint, "GridPoint." 2019. Available: https://www.gridpoint.com/. [Accessed: 16-Dec-2019].
[27] Z. Kokolanski, T. Shuminoski, C. Gavrovski, Architectures and challenges for the household energy management systems, in: Proceeding of the *IEEE 27th Intermnatioal Science Conference Electronics Proceeding*, 2018, pp. 1–4.
[28] Honeywell International Inc, "Honeywell." 2019. Available: https://buildingsolutions.honeywell.com/en-US/solutions/hvacbuildingmanagement/Pages/default.aspx. [Accessed: 16-Dec-2019].
[29] Connect One, "iChipNet." 2019. Available: http://www.connectone.com/. [Accessed: 16-Dec-2019].
[30] "Insteon." 2019. Available: http://www.insteon.com/. [Accessed: 16-Dec-2019].
[31] A. Javed, H. Larijani, A. Wixted, Improving energy consumption of a commercial building with IoT and machine learning, IT Prof. 20 (5) (2018) 30–38.
[32] KaaIoT Technologies LLC, "Kaa." 2019. Available: https://www.kaaproject.org/. [Accessed: 16-Dec-2019].
[33] LG, "SmartThinq." 2019. Available: http://www.lg.com/us/discover/smartthinq/thinq. [Accessed: 16-Dec-2019].
[34] B. Mataloto, J.C. Ferreira, N. Cruz, Lobems—IoT for building and energy management systems, Electron 8 (7) (2019).
[35] E. Lopez, J. Monteiro, P. Carrasco, J. Saenz, N. Pinto, and G. Blazquez, "Development, implementation and evaluation of a wireless sensor network and a web-based platform for the monitoring and management of a microgrid with renewable energy sources," in 2019 International Conference on Smart Energy Systems and Technologies (SEST), 2019, pp. 1–6.
[36] MyDevices, "MyDevices." 2019. Available: https://mydevices.com/. [Accessed: 16-Dec-2019].
[37] "Nest." 2019. Available: https://store.google.com/us/category/connected_home?hl=en-US&GoogleNest&utm_source=nest_redirect&utm_medium=google_oo&utm_campaign=GS102776&utm_term=control. [Accessed: 27-Dec-2019].
[38] "Netatmo." 2019. Available: https://www.netatmo.com/. [Accessed: 19-Dec-2019].
[39] "Nexia Home." 2019. Available: http://www.nexiahome.com/. [Accessed: 19-Dec-2019].
[40] "Particle." 2019. Available: https://www.particle.io/. [Accessed: 19-Dec-2019].
[41] General Electric, "Predix." 2019. Available: https://www.ge.com/digital/predix. [Accessed: 19-Dec-2019].
[42] "Schneider Wiser." 2019. Available: http://www.schneider-electric.com/en/product-range/61209-wiser-home-energy-management-system?N=3322220873. [Accessed: 19-Dec-2019].
[43] Lord MicroStrain, "SensorCloud." 2019. Available: http://www.sensorcloud.com/. [Accessed: 19-Dec-2019].
[44] V.R. Shinde, P.P. Tasgaonkar, R.D. Garg, Environment monitoring system through Internet of Things(IOT), in: Proceedings of the International Conference Information, Communication Engineering, Technol. ICICET 2018, 2018, pp. 1–4.
[45] Siemens Inc., "Siemens Desigo." 2019. Available: https://new.siemens.com/global/en/products/buildings/automation/desigo.html. [Accessed: 19-Dec-2019].
[46] Daliworks Inc, "ThingPlus." 2019. Available: https://thingplus.net/en/. [Accessed: 20-Dec-2019].
[47] PTC, "ThingWorx." 2019. Available: https://www.thingworx.com/. [Accessed: 20-Dec-2019].
[48] "Ubidots." 2019. Available: https://ubidots.com. [Accessed: 20-Dec-2019].
[49] IBM, "Watson IoT." 2019. Available: https://www.ibm.com/internet-of-things/. [Accessed: 20-Dec-2019].
[50] "Wattio." 2019. Available: https://wattio.com/. [Accessed: 20-Dec-2019].
[51] C. Fulk, G. Hobar, K. Olsen, S. El-Tawab, F. Rahman, P. Ghazizadeh, Cloud-based low-cost energy monitoring system through Internet of Things, in: Proceedings of the IEEE Internation Conference Pervasive Computer Communication Work PerCom Work, 2019, pp. 322–327.
[52] "Wibeee." 2019. Available: https://wibeee.com/. [Accessed: 24-Dec-2019].
[53] M.H. Yaghmaee, H. Hejazi, Design and implementation of an Internet of Things based smart energy metering, in: Proceedings of the 6th IEEE International Conference Smart Energy Grid Engineering SEGE, 2018, pp. 191–194.
[54] A. Gyrard, S.K. Datta, C. Bonnet, K. Boudaoud, A semantic engine for Internet of Things: cloud, mobile devices and gateways, in: Proceedings of the 9th International Conference Innovation Mobile Internet Services in Ubiquitous Computing, 2015, pp. 336–341.
[55] M. Liyanage, C. Chang, S.N. Srirama, Energy-efficient mobile data acquisition using opportunistic Internet of Things gateway services, in: Proceedings of the IEEE International Conference Internet Things IEEE Green Computing and Communications IEEE Cyber, Phys. Soc. Comput, IEEE Smart Data, 2016, pp. 217–222.
[56] J. Granados, A.M. Rahmani, P. Nikander, P. Liljeberg, H. Tenhunen, Towards energy-efficient HealthCare: an Internet-of-Things architecture using intelligent gateways, in: *Proceedings of the 4th International Conference on Wireless Mobile Communication and Healthcare - "Transforming Healthcare Through Innovations in Mobile and Wireless Technologies*, 2015, pp. 279–282.
[57] Y. Wen, Z. Li, X. Peng, H. Zhao, A middleware architecture for sensor networks applied to industry solutions of internet of things, in: Proc. 2011 2nd International Conference Digital Manufacturing Automation, ICDMA 2011, 2011, pp. 50–54.
[58] EPFL, "GSN." 2019. Available: https://github.com/LSIR/gsn. [Accessed: 20-Dec-2019].
[59] ZigBee Alliance, "ZigBee IP." 2019. Available: https://zigbeealliance.org/. [Accessed: 27-Dec-2019].
[60] B. A. Munir, P. Kansakar, and S. U. Khan, "IFCIoT: integrated fog cloud IoT,", 2017.
[61] C. Puliafito, E. Mingozzi, G. Anastasi, Fog computing for the Internet of Mobile Things: issues and challenges, in: Proceedings of the IEEE International Conference on Smart Computing (SMARTCOMP, 2017, pp. 1–6.
[62] D. Kelaidonis, A. Rouskas, V. Stavroulaki, and P. Demestichas, "A federated edge cloud-IoT architecture," in *Proceedings of the European Conference on Networks and Communications (EuCNC)*, pp. 230–234.
[63] P. Bellavista, J. Berrocal, A. Corradi, S.K. Das, L. Foschini, A. Zanni, A survey on fog computing for the Internet of Things, Pervasive Mob. Comput. 52 (2019) 71–99.
[64] M. Chiang, T. Zhang, Fog and IoT: an overview of research opportunities, IEEE Internet Things J. 3 (6) (2016) 854–864.
[65] F. Jalali, S. Khodadustan, C. Gray, K. Hinton, F. Suits, Greening IoT with fog: a survey, in: Proceedings of the IEEE International Conference on Edge Computing, EDGE 2017, 2017, pp. 25–31.
[66] J. Ren, H. Guo, C. Xu, Y. Zhang, Serving at the Edge: a scalable IoT architecture based on transparent computing, IEEE Netw. 31 (5) (2017) 96–105.
[67] S. Yi, Z. Hao, Z. Qin, Q. Li, Fog computing: Platform and applications, in: Proceedings of the 3rd Workshop Hot Topics Web Systems and Technologies HotWeb, 2016, pp. 73–78.
[68] Samsung, "SmartThings." 2019. Available: https://www.smartthings.com/. [Accessed: 20-Dec-2019].
[69] Verizon, "Hum." 2019. Available: https://www.hum.com/. [Accessed: 27-Dec-2019].
[70] C. Bormann, M. Ersue, A. Keranen, Terminology for constrained-node networks abstract, J. Chem. Inf. Model. 53 (9) (2013) 1689–1699.
[71] "MQTT." 2019. Available: http://mqtt.org/. [Accessed: 27-Dec-2019].
[72] "CoAP." 2019. Available: http://coap.technology/. [Accessed: 27-Dec-2019].
[73] A. Antonic, M. Marjanovic, P. Skocir, I.P. Zarko, Comparison of the CUPUS middleware and MQTT protocol for smart city services, in: Proceedings of the 13th International Conference Telecommunication, ConTEL 2015, 2015.
[74] M.R. Palattella, et al., Standardized protocol stack for the internet of (important) things, IEEE Commun. Surv. Tutor 15 (3) (2013) 1389–1406.
[75] P. Bellavista and A. Zanni, "Towards Better Scalability for IoT-Cloud Interactions via Combined Exploitation of MQTT and CoAP," in 2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI), 2016, pp. 1–6.
[76] M.H. Valipour, B. Amirzafari, K.N. Maleki, N. Daneshpour, A brief survey of software architecture concepts and service oriented architecture, in: Proceedings of the 2nd IEEE International Conference Computer Science and Information Technology, 2009, pp. 34–38.
[77] M. Eisenhauer, P. Rosengren, P. Antolin, A development platform for integrating wireless devices and sensors into Ambient Intelligence systems, in: Proceedings of the 6th IEEE Annual Communications Society Conference on Sensor, Mesh and Ad Hoc Communications, 00, Networks Work. SECON Work. 2009, 2009, pp. 1–3.
[78] A. Meloni, L. Atzori, A cloud-based and RESTful Internet of Things platform to foster Smart Grid technologies integration and re-usability, in: Proceedings of the IEEE International Conference on Communications, Work. ICC 2016, 2016, pp. 387–392.
[79] T. Vresk, I. Cavrak, Architecture of an interoperable IoT platform based on microservices, in: Proceedings of the 39th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2016 - Proc., 2016, pp. 1196–1201.
[80] Z. Xiao, I. Wijegunaratne, X. Qiang, Reflections on SOA and Microservices, in: Proceedings of the 4th International Conference on Enterprise Systems Adv. Enterp. Syst., 2017, pp. 60–67.
[81] L. Lan, F. Li, B. Wang, L. Zhang, R. Shi, An event-driven service-oriented architecture for the internet of things, in: Proceedings of the Asia-Pacific Services Computing Conference, APSCC 2014, 2015, pp. 68–73.
[82] Google Inc, "Play Store." 2019. Available: https://play.google.com/store. [Accessed: 23-Dec-2019].
[83] M. Altamimi, R. Palit, K. Naik, A. Nayak, Energy-as-a-Service (EaaS): on the efficacy of multimedia cloud computing to save smartphone energy, in: Proceedings of the 2012 IEEE 5th International Conference on Cloud Computing, CLOUD 2012, 2012, pp. 764–771.
[84] M. Altamimi, K. Naik, A practical task offloading decision engine for mobile devices to use energy-as-a-service (EaaS), in: Proceedings of the IEEE World Congress on Services, 2014, pp. 452–453.
[85] B. Mohammed, M. Kiran, Analysis of Cloud Test Beds Using OpenSource Solutions, in: Proceedings of the International Conference Future Internet Things Cloud, International Conference Open Big Data, OBD 2015, 2015, pp. 195–203.
[86] "HomeStar." 2019. Available: https://github.com/iotdb/iotdb.github.io/tree/master/docs. [Accessed: 27-Dec-2019].
[87] J. Singh, T. Pasquier, J. Bacon, H. Ko, D. Eyers, Twenty security considerations for cloud-supported Internet of Things, IEEE Internet Things J. 3 (3) (2016) 269–284.

[88] H. Akram, M. Hoffmann, Requirements analysis for identity management in ambient environments: the HYDRA approach, in: Proceedings of the CEUR Workshop, 371, 2008, pp. 17–29.
[89] Kontakt, "Kontakt." 2019. Available: https://kontakt.io/. [Accessed: 27-Dec-2019].
[90] M.T. Khorshed, N.A. Sharma, K. Kumar, M. Prasad, A.B.M.S. Ali, Y. Xiang, Integrating Internet-of-Things with the power of cloud computing and the intelligence of big data analytics-a three layered approach, in: Proceedings of the 2nd Asia-Pacific World Congress on Computer Science and Engineering, APWC CSE 2015, 2016.
[91] M. Sain, Y. J. Kang, and H. J. Lee, "Survey on Security in Internet of things: state of the art and challenges," in 2017 19th International Conference on Advanced Communication Technology (ICACT), 2017, pp. 699–704.
[92] T. Morris, Trusted platform module, in: Encyclopedia of Cryptography and Security, Springer, 2011, pp. 1332–1335.
[93] Indra, "Onesait." 2019. Available: https://www.minsait.com/es/productos/platform/. [Accessed: 27-Dec-2019].
[94] "SmartThings." 2019. Available: https://www.smartthings.com/. [Accessed: 20-Dec-2019].
[95] MathWorks, "ThingSpeak." 2019. Available: https://thingspeak.com/. [Accessed: 27-Dec-2019].
[96] S. Soursos, I.P. Zarko, P. Zwickl, I. Gojmerac, G. Bianchi, G. Carrozzo, Towards the cross-domain interoperability of IoT platforms, in: Proceedings of the European Conference on Networks and Communications, 2016, pp. 398–402.
[97] Apple Inc, "Amazon, Apple, Google, Zigbee Alliance and board members form working group to develop open standard for smart home devices." 2019. Available: https://www.apple.com/newsroom/2019/12/amazon-apple-google-and-the-zigbee-alliance-to-develop-connectivity-standard/. [Accessed: 24-Dec-2019].
[98] "Fiware." 2019. Available: https://www.fiware.org/. [Accessed: 27-Dec-2019].
[99] "Legato." 2019. Available: http://legato.io/. [Accessed: 27-Dec-2019].
[100] I.P. Zarko, et al., Towards an IoT framework for semantic and organizational interoperability, in: *Proceedings of the* Global Internet of Things Summit (GIoTS), 2017, pp. 1–6.
[101] G. Carrozzo, et al., Interoperation of IoT platforms in confined smart spaces: the SymbIoTe smart space architecture, in: *Proceedings of the* Fifth International Conference on Internet of Things: Systems, Management and Security, 2018, pp. 38–45.
[102] E. Gambi, L. Montanini, L. Raffaeli, S. Spinsante, L. Lambrinos, Interoperability in IoT infrastructures for enhanced living environments, in: Proceedings of the IEEE International Black Sea Conference on Communications, Networking, BlackSeaCom 2016, 2017.
[103] S. Zitnik, M. Jankovic, K. Petrovcic, M. Bajec, Architecture of standard-based, interoperable and extensible IoT platform, in: Proceedings of the 24th Telecommun. Forum, TELFOR 2016, 2017, pp. 0–3.
[104] "oneM2M." 2019. Available: http://www.onem2m.org/. [Accessed: 27-Dec-2019].
[105] Open Connectivity Foundation, "AllJoyn." 2019. Available: https://openconnectivity.org/developer/reference-implementation/alljoyn. [Accessed: 27-Dec-2019].
[106] The Linux Foundation, "IoTivity." 2019. Available: https://iotivity.org/. [Accessed: 27-Dec-2019].
[107] IEEE, "IEEE P2413." 2019. Available: https://standards.ieee.org/project/2413.html. [Accessed: 27-Dec-2019].
[108] S. Heo, S. Woo, J. Im, D. Kim, IoT-MAP: IoT mashup application platform for the flexible IoT ecosystem, in: Proceedings of the 5th International Conference Internet Things, IoT 2015, 2015, pp. 163–170.
[109] J. Koo and Y. Kim, "Heterogeneous IoT Platforms," pp. 26–29, 2017.
[110] I.J. Fernández, C.F. Calvillo, A. Sánchez-Miralles, J. Boal, Capacity fade and aging models for electric batteries and optimal charging strategy for electric vehicles, Energy 60 (Oct. 2013) 35–43.

Miguel Martín Lopo obtained an Industrial Engineering degree (major in Electronics) in 2014, a master's degree in Engineering Sciences in 2016 and is currently a Ph.D. candidate in Engineering Systems Modelling, all of them from Comillas Pontifical University. In December 2014, he joined the Institute for Research in Technology (IIT), where he is part of the Smart Industry and Cities group. Currently, his work focuses mainly on improving energy efficiency at dwelling and building levels through the wireless monitoring and smart control of loads and the integration of distributed energy resources. Notwithstanding, his areas of interest include IoT devices, wireless communications, artificial intelligence, smart cities, and Industry 4.0.

Jaime Boal obtained an Industrial Engineering degree (major in Electronics) in 2010 and holds a Ph.D. in Engineering Systems Modeling since 2014, both from Comillas Pontifical University. In September 2014, he joined the ICAI School of Engineering as Lecturer and the Institute for Research in Technology (IIT), where he is part of the Smart Industry and Cities group. Currently, his work focuses mainly on improving energy efficiency at dwelling and building levels through the wireless monitoring and smart control of loads and the integration of distributed energy resources. Notwithstanding, his areas of interest include IoT devices, wireless communications, artificial intelligence, mobile robotics, and computer vision.

Alvaro Sánchez Miralles holds a Doctor's degree in Engineering Sciences from Comillas Pontifical University. He is an experienced researcher at the IIT from Comillas University, where he has been working for over 20 years. He is currently the coordinator of the Intelligent Systems research group at IIT. The goal of his research activity is to improve the value of companies, institutions and society applying artificial intelligence, optimizing systems and processes at all the layers: physical (electronics, sensors, power converters), communication (FIWARE, CIM, SEP), intelligence and business models. The main fields of interest are smart grids and smart cities: integration of renewable generation and storage in distribution networks (smart storage, microgrids), electric vehicles (V2G and VPP), planning power grids (reference network models, system planning with central and distributed energy resources evolution and tariff design) and energy management in buildings and districts (aggregation of prosumers with market price maker and taker participation, microgrids, building and home energy management systems, electric and thermal synergies optimization). Furthermore, he has a lot of experience in security systems and autonomous land and aerial vehicles: computer vision, simultaneous location and mapping, indoor localization of people and active and robust security systems. He is an Associate Professor at the Electronics and Automation Engineering Department.