Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2014, Article ID 158252, 18 pages
http://dx.doi.org/10.1155/2014/158252

Research Article

Protocol and Architecture to Bring Things into Internet of Things

Ángel Asensio, Álvaro Marco, Rubén Blasco, and Roberto Casas

Aragon Institute of Research, University Zaragoza, 50018 Zaragoza, Spain

Correspondence should be addressed to Álvaro Marco; amarco@unizar.es

Received 1 December 2013; Revised 16 February 2014; Accepted 16 February 2014; Published 13 April 2014

Academic Editor: Zuqing Zhu

Copyright © 2014 Ángel Asensio et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The Internet of Things (IoT) concept proposes that everyday objects are globally accessible from the Internet and integrate into new services having a remarkable impact on our society. Opposite to Internet world, *things* usually belong to resource-challenged environments where energy, data throughput, and computing resources are scarce. Building upon existing standards in the field such as IEEE1451 and ZigBee and rooted in context semantics, this paper proposes CTP (Communication Things Protocol) as a protocol specification to allow interoperability among *things* with different communication standards as well as simplicity and functionality to build IoT systems. Also, this paper proposes the use of the IoT gateway as a fundamental component in IoT architectures to provide seamless connectivity and interoperability among *things* and connect two different worlds to build the IoT: the *Things world* and the *Internet world*. Both CTP and IoT gateway constitute a middleware content-centric architecture presented as the mechanism to achieve a balance between the intrinsic limitations of *things* in the physical world and what is required from them in the virtual world. Said middleware content-centric architecture is implemented within the frame of two European projects targeting smart environments and proving said CTP's objectives in real scenarios.

1. Introduction

Since the last decade, we are assisting in a progressive jump from a nonubiquitous Internet, where humans access Internet using a computer at their work or at home, to the current ubiquitous Internet where we access the Internet using smartphones, tabs, or TVs, anytime, anywhere. In the same way, now comes the time of the Internet of Things (IoT) when not only humans but also *things,* any object surrounding us, are present in Internet [1].

Things must be considered in the broadest sense of the word as real or virtual entities that exist and evolve in a context and time and have univocal identifiers. On the other hand, the term Internet applied to them conveys the idea that all these *things* are heavily communicated and interrelated among them. Commonly IoT can be approached from different perspectives: *Internet* for communications, cloud, and services; *things* for physical elements, sensor networks, and user interfaces; and *semantic* that considers ontology of *things* in Internet [2].

Ideally, *things* on the IoT will have full interconnectivity and computation resources, being natural to consider connecting these *things* to the web using current paradigms. Different works investigate the types of *things* (sometimes called smart objects), their nature, and relationship with the IoT; according to the different perspectives, a *thing* has awareness, representation, interaction, and so forth [3]. The classification of *things* into standard groups helps to show that they are the physical part of IoT with constraints and needs that must be taken into account. So, coming down to implementation, while IoT concept talks about ubiquitous, invisible, and context aware *things*, technology poses hurdles such as energy supply, price and size of devices, seamless connectivity, or interoperability [4]. Additionally, as, currently, Internet has a wired backbone, "being there" forces all the *things* to be IP compatible and use access points or gateways to bridge global fiber optic or cabled infrastructure.

Wireless communication protocols are mandatory if *things* need to be mobile and ubiquitous. Depending on the specific application, and leaving aside proprietary protocols,

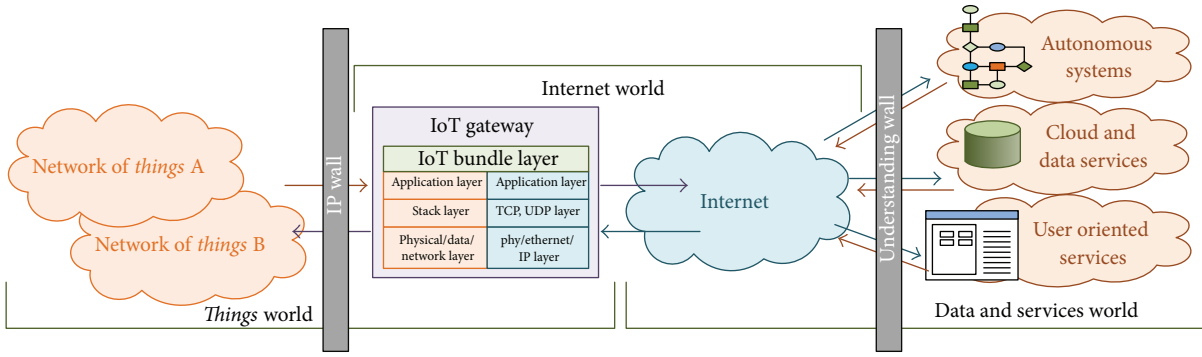


Figure 1: IoT architecture.

different standards are commonly used to provide connectivity: ZigBee, RFID, Bluetooth, 6lowPAN, WIFI, 3G, and so forth. Many works evidence the importance of selecting the most appropriated technology in each situation comparing them in terms of network topology, coverage, data throughput, or energy consumption. In any case, the more data you need to exchange, the further you need to communicate, the more time you need to be online, then more energy you need, and consequently the quicker you will run out of batteries.

The arrival of IoT will lead to appearing multitude of new services, improving the quality of life of people, and offering new business opportunities. Ultimately, the IoT is expected to bring a revolution in the concept of society, similar to how Internet changed the concept of communications and information [5]. In the last few years, initiatives and project related with IoT have been growing in number and scope throughout the entire world. The IoT concept is becoming tremendously popular and already appearing systems that claim to offer IoT solutions to the final user without high technical requirements [6]. Current developments are in early stages being most of the services based on monitoring or sensing variables to extract information and then analyze and represent them [7].

It is considered that the development of IoT will play an important role in the near future; thus, public and private investments have been made in R&D, demonstration, and deployment activities [8, 9]. To date, most of IoT solutions are small subnets of interconnected objects. It is not possible to talk about IoT until all objects are interconnected and to improve this interconnectivity, an architecture that ensures interoperability between systems is mandatory. Thus, both public and private initiatives are currently focusing on standardization of the IoT [10–14].

In summary, the evolution of IoT implies overcoming real *things*' limitations enabling them to communicate with the Internet using a common language. In this paper, we propose CTP (Communication *Things* Protocol) as an ontology-based solution to enable understanding among *things* and the use of an IoT gateway to take *things* to the Internet world.

2. Internet of Things

2.1. Architecture. Most IoT implementations follow an architecture that contains different worlds each of them with their own characteristics; Figure 1 shows an example [15].

The *things* world relates to michroelectromechanical systems, smart sensors, simple human-machine interfaces (HMIs), and so forth, which associate in networks (Networks of *Things*, NoT) to ubiquitously interact with other *things*, the environment and/or people. NoT are usually resource-challenged ecosystems, typically with medium-high time access delays, high error rates, low data throughput, and limited online time where energy consumption must be optimized to the maximum. In a simplified model of a *thing*, basic blocks of communication, computation, and interaction (sensors, actuators, and HMI) are distinguished.

The Internet world usually is constructed around computers, centralized software infrastructures, or in the cloud [16]. Applications and services use *things* to provide context awareness, artificial intelligence, affective computing, and so forth [16]. Depending on their consideration, tablets and smartphones can be included in both worlds. Nevertheless, due to their power (computing, communications, battery lifetime, user interfaces, etc.) we find it more appropriate to consider them closer to the world of computers and Internet than to the world of *things*. Currently, Internet acts as the base infrastructure for the exchange of information. However, access to it has several constraints such as the need for unique identifier, the change of communication technology, and the adoption of the Internet Protocols. Nowadays, this "IP wall" is usually jumped through an IoT gateway.

Internet is the global interconnection method where multitude of services and applications use extracted information of IoT to provide services to end users. On each of them, needs are a virtual representation of the *things* of the IoT in order to enable interaction. Once "IP wall" is saved and thanks to the connectivity provided by the Internet, services can access the information, but, for the information to be useful, it must be understood and this poses what we call the "understanding wall."

2.2. Interoperability. IoT interoperability implies capacity to both exchange data (crossing the IP wall) and understands the information embedded in data (crossing the understanding wall). Communication standards ensure stack layer's communication between *things* in the same network sharing the same protocol, that is, network management and maintenance, security, data exchange, and so forth. Nevertheless, if application layer is not defined, *things* will not understand among them unless previously agreed between developers, that is, information understanding. This is the case of 6lowPAN, RFID, WiFi, or cellular; for example, if two manufacturers want to develop 6lowPAN temperature sensors and thermostats are able to interoperate among them, there is no common definition to adopt; they will need to agree on the protocol exchange application-related information, temperature data format, procedure to virtually bind devices, and so forth.

Automation and control networks such as Lonworks, BACnet, Konnex, or CANOpen define how devices are represented in their networks; objects and variables are the most common approaches. Bluetooth and ZigBee go one step further in interoperability defining profiles and device objects within the application layer. Bluetooth defines profiles (hands-free, health device, human-interface device, etc.) corresponding to vertical applications. For example, we could build a monitoring infrastructure with Bluetooth microphones streaming audio according to hands-free profile; no matter their manufacturer, any certified host compliant with the profile will play the audio gateway role without any additional programming [17].

ZigBee not only defines vertical profiles (home automation, energy metering, healthcare, etc.), but also defines horizontal functional domains to specify how devices must exchange application data attending their functionality [18]. For example, every ZigBee compliant temperature sensor must implement "measurement & sensing functional domain" and any other device in the network (e.g., a thermostat) would be able to get temperature information as defined in the specification. Additionally, ZigBee allows instantiation of intelligence in the network by defining how to coordinate devices to produce scenes and create virtual bindings among devices [19], for example, to program a switch to turn on two lights and open a motorized door.

Sometimes two specifications coordinate to solve interoperability, such as ZigBee that specifies how to interoperate with BACnet. Devices using any wireless communication standard (e.g., ZigBee) cannot directly interoperate with devices over other standards (e.g., WiFi) unless there is a protocol aggregator and translator connecting both worlds, the IoT gateway.

Although IP connectivity is not necessarily required to ensure inter-*thing* communication, many standards include IP as part of their specification to ease the process to connect *things* to the Internet. Nevertheless, this is not enough to ensure interoperability at required IoT application level. As *things* are resource-challenged communication nodes, efficient data transmission mechanisms are needed at IP level; CoAP [20], XMPP [21], RESTful HTTP [22], and MQTT are relevant specifications at this level.

Once efficient connectivity between devices is granted, standards such as EEML (Extended Environments Markup Language), SensorWeb (including SensorML and TransducerML), or SenseWeb provide interpretation to bytes exchanged integrating sensors and actuators with the virtual world. These schemes just express queries and data modeling, lacking of semantics, and ontologies that are necessary for complex information processing and support to service composition and adaptation at higher levels of abstraction.

IEEE1451 standard is a network-independent specification for smart transducers (sensors or actuators) that provides a common language regardless of the protocol used. The standard defines different application profiles: environmental (climate monitoring, greenhouse gases, and other chemical sensors), smart meter (monitor water, gas, or electricity consumption), health care (monitor the body using external or implantable sensors), and smart home automation and industrial (pipe's monitoring, comfort, and surveillance sensors) [23]. The standard is based on the Transducer Electronic Data Sheet (TEDS) to describe a set of communication interfaces for connecting transducers to microprocessors, instrumentation systems, and control/field networks. According to IEEE1451.0 specification, TEDS provides information about transducer's identification, operation, calibration, manufacturer, and so forth.

Lying on IEEE1451.5 specification for radio-specific protocols [24], it is possible to implement wireless sensor networks using IEEE1451 over communication standard protocols such as Wifi [25], Bluetooth, ZigBee [26], or 6LowPan [27]. Similarly, IEEE1451.2 defines cabled SPI and UART, IEEE1451.6 over CANOpen, and IEEE1451.7 over RFID. This interoperability is enabled by the Network Capable Application Processor (NCAP) that aggregates the different Transducer Interface Module's communication standards (TIM) over the same common language. Through NCAP and following IEEE1451 it is possible to implement web services that make *things* discoverable, accessible, and controllable via the Internet 22. In the same line, ZigBee defines the ZigBeeGateway Public Application Profile that specifies how devices must be connected to the Internet and to service providers.

Initiatives, such as Sensei project [7], COSE [28], DogOnt [29], and other [30] different ontological approaches to model *things* (sensors, actuators, simple human-machine interfaces, appliances, etc.) in smart environments, associate contained devices through semantic relationships [31] and seamlessly integrate *things* with web services.

3. Common Things Protocol (CTP)

3.1. Rationale. Networks of *things* (NoT), defined as smart infrastructures of embedded devices with local intelligence and access to the "information ether," are the base of the IoT. As a part of the Internet, one of the basic needs is the interaction mechanism between agents which implies that, between them, there must be connectivity and understanding in order to provide interoperability. To achieve this goal, different IoT protocols are being proposed to provide efficient,

seamless, and robust connectivity (CoAP, XMPP, RESTful HTTP, MQTT, etc.) but not providing semantics. From an integral perspective, IEEE1451 appears to be a good suited option for the IoT as it tackles specification from sensor to network interface providing independence from the communication protocol, enabling self-identification of devices, long-term self-documentation, plug and play capacity to ease field installation, upgrade, and maintenance [32]. On the other hand, the standard has a limited penetration in IoT applications out of the electronic instrumentation field. Reasons for that can be the little compatible hardware available (no commercial wireless sensor networks consider it) because IoT developers mainly come from the computer science field (usually considering semantic approaches), due to complexity of the standard [33], or the excessive detail in electronic aspects that are not commonly used by the application (e.g., transducer channel reads delay time or incoming propagation delay through the data transport logic). Communication standards such as ZigBee or 6LoWPAN are much more used in IoT applications but involve hardware restrictions and lack of interoperability among them.

The Common Things Protocol (CTP) aims to provide a specification that allows interoperability among communication standards and coexistence with IoT protocols, but prioritizing simplicity, efficiency, and functionality to build final IoT systems.

CTP takes into consideration existing specifications in the standards and the needs from the final applications that are to be supported by the *things* in the field. It integrates the strategies, concepts and terms of some of the alternatives, for example, IEEE1451 TED's concept to provide device's information, sensor and actuator working modes, and so forth, or ZigBee concepts of clusters and endpoints. CTP definition is approached from an ontological perspective considering *things*, not just as sensors [34] but as electronic devices that perform some sort of function in the IoT application being in contact with user and context and usually fulfilling the following paradigms:

(i) *context interaction:* embedded sensors (to sense the user/context), actuators (to modify the environment), and/or simple human interfaces (to interact with people);

(ii) *computin:* to have computing capabilities and memory that allow them to implement from the simplest logic to complicated services or data processing algorithms;

(iii) *communication:* to have at least one, usually wireless, communication media commonly following a standard and adapted to communication requirements (range, power consumption, and data throughput). It is required to interoperate among them and integrate in the Internet. Thus, unless they are able to directly connect to Internet, a network gateway is required to allow interoperability among different networks of *things* and to provide Internet access;

(iv) being an *electronic thing*: as anything in this world, regardless it is electronic or not, each *thing* is unique and "lives" in a specific space and time. Additionally as any electronics, it needs energy to operate.

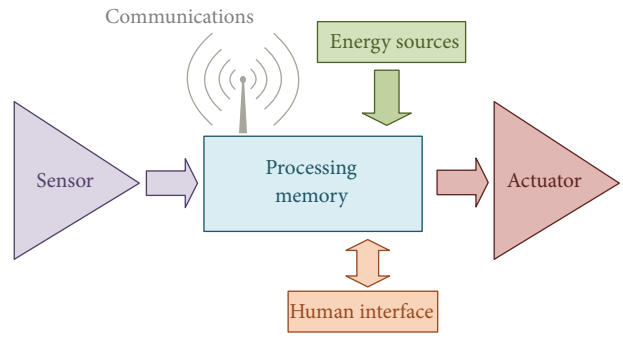


Figure 2: *Thing* block diagram.

Figure 2 represents said paradigms in the block diagram of a *thing* able to integrate in the IoT.

Thus, derived from their capacity to interact with the context, CTP considers that *things* can have three different functionalities:

(i) sensors that gather and process information from the real world in environmental and person-centric contexts [35]. Information from the environmental context is useful to create an objective snapshot of what is happening in every moment, while person-centric helps to understand and evaluate the objective data in order to extract conclusions, to define guidelines, or to identify patterns adapted to each user;

(ii) actuators that provide the ability to act over the environment. IoT applications may need to act over the environment when the user is not able to control (e.g., because he/she has a disability) or he/she is not aware of specific situations that require actuation (e.g., forget about the heating when going to sleep) or simply he/she is not present in the environment;

(iii) pervasive human-machine interfaces including traditional (switches and dimmers) and new interaction paradigms, providing the user with relevant information or notifying him/her about events in new and natural ways (e.g., a color device turns red when house's energy consumption is high).

Regardless its functionality, each *thing* will always "live" in a certain location and time and will have a processor, communication transceiver, and power source. Similar to MetaTEDS in IEEE1451 and ZigBeeDevice Object (ZDO) in ZigBee, this constitutes the basic set of attributes and functionalities of any device, which is modeled by the endpoint BASE in CTP. Depending on the specific functionalities it integrates, it will also implement multiple application endpoints that should be of any of the said categories: SENSOR, ACTUATOR, or HMI.

Thus, an endpoint can be defined as each of the subdevices that have a complete functionality and together with others build the *thing*; in total we just define the said 4 endpoints to model any*thing*. Additionally, a cluster is defined

as a set of commands, events, and responses (some mandatory to implement in order to guarantee interoperability) which together define a communication interface between two endpoints. Clusters constitute the implementation of the ontological representation of *thing*'s nature; for example, endpoint BASE includes location, time, and power clusters. Endpoint and cluster concepts are shared with ZigBee and are similar to transducer channels in IEEE1451.

Commands are usually action requests to an endpoint (of the same or different *thing*) that should send back a response informing about the action result, for example, ask for a sensor value and get it back. The ontology defines attributes which are implemented in the protocol as GET/SET requests and responses. Events are asynchronously generated messages sent to previously subscribed endpoints; for example, presence detected by a sensor is sent to light actuator. When defining a communication interface, two strategies can be adopted: (i) using a small, but well defined, number of messages, where the versatility is achieved by parameters (ii) or defining a greater number of messages allowing more specific control with lower parametric content [36]. An example of this duality is to define a single configuration command with many parameters, or several commands to adjust each parameter independently. The selection of one or another philosophy conditions ease of use, generalization capacity, adaptability to different scenarios and future maintenance, and backwards compatibility. CTP chooses to define a reduced and simple communication interfaces defining the meaning type and range of the parameters when needed. For example, many clusters have in common a read-only information attribute; in the case of sensors it provides information of ranges, accuracy, format, units, and so forth of the measure it provides, while for actuators the same attribute indicates how the actuator should be controlled. Similarly, some clusters include a read-and-write configuration attribute, whose specific parametrization depends on the device.

3.2. Endpoints and Clusters. Figure 3 shows the four endpoint types (BASE, SENSOR, ACTUATOR, and HMI) with their associated clusters and the main attributes, commands, responses, and events.

3.2.1. Endpoint BASE. All devices, regardless their nature, must have endpoint BASE containing the generic and common characteristics whatever their functions are. Related with this endpoint we define the following clusters.

Cluster DEVICE. It manages *device identification*, *description* of their functionalities (as in IEEE1451 TEDS's globally unique identifier and transducer channels), and minimum *self-operation* functionalities (as BIOS). Its implementation is mandatory in all CTP devices, as it is the basis for ensuring interaction with other system elements. Devices may have the ability to operate in different ways depending on the context; the *mode* parameter is used to switch between them and to facilitate the use of the *thing*, by providing to a nonadvanced user a set of predefined modes of operation that do not need any previous settings to set mode and operate. Also, this cluster has several capabilities oriented to work with low power devices, for example, and similar with TEDS, a *timeout* parameter that indicates the amount of time after an action for which the lack of reply following the receipt of a command may be interpreted as a failed operation.

Cluster LOCATION. Each device will always have a location that can be essential information for many services. It can be known or not, it can change or remain, and device can self-calculate it or be written externally, whichever the case, this cluster allows getting and setting *device's position*, programming its timed update, or reporting it in response to specific events.

Cluster POWER. As location, every device will have a *power source* (battery, mains, energy harvesting, etc.); its type, consumption profile, and energy remaining are the main information deal within this cluster.

Cluster TIME. Again as location and power, every device "live" in a specific instant of time, and whether relative or absolute this cluster allows the management of temporal information such as time synchronization and scheduling of timed actions.

Cluster PROXY. Similar to TEDS, this cluster acts as an aggregator of endpoints (or transducer channels) and datalogger manager (if memory is available) simplifying access to data. It reduces communication burden and thus increases battery efficiency.

Often, implementation of networks of smart sensors and actuators goes through a central device that receives data from sensors, processes the information, and then orders actions to the actuators. This has a number of problems: (i) density of data traffic, (ii) increased energy consumption, (iii) masking the mesh concept, since there is a central node acting as sink and source of information, and (iv) reducing robustness to failure. To face that, it is possible to define a basic distributed intelligence by subordinating the behavior of some devices to the events generated by other devices.

Cluster BEHAVIOR. Similar to ZigBee this allows the creation of groups of devices or endpoints and the establishment of logical relationships (bindings) among them. We expand its native definition defining the concept of a trigger event for a *binding* which allows that any endpoint generating events (e.g., timing, threshold event, etc.) could trigger any endpoint(s) in one or more devices with their corresponding parameters (e.g., changing the mode to low power of a device). Also, as in TEDS when defining embedded actuators, these bindings can be internal to a device and serve, for example, to trigger actions; button event triggers a light actuator.

This cluster also allows delegation of system's intelligence in the devices as it provides methods to program autonomous operation based on the specification of operational rules. It is based on logical rules that verify compliance with certain conditions of different endpoints of the device. Each situation

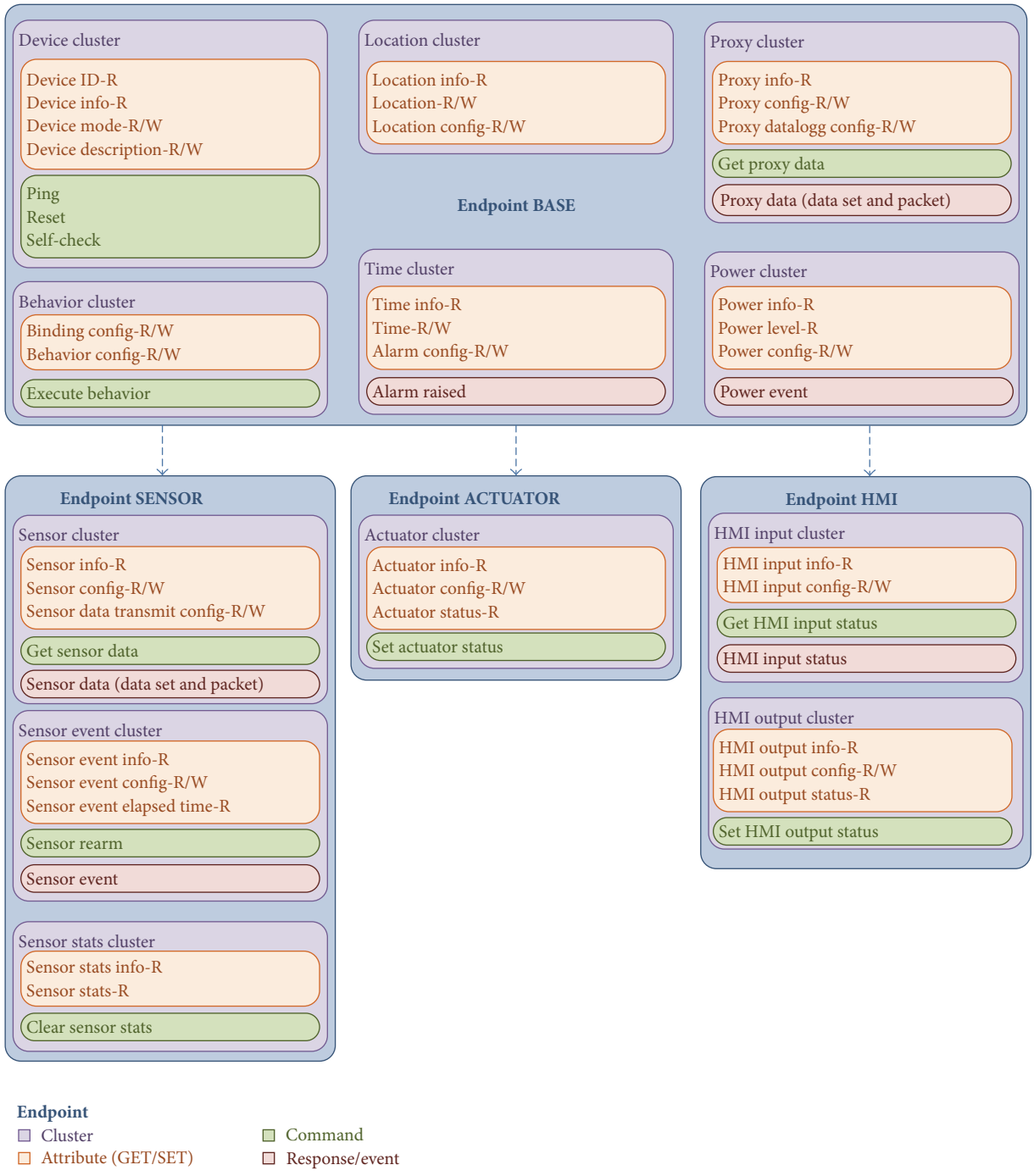


Figure 3: Summary of endpoints, clusters, and main interfaces in CTP.

that activates a behavior is called activation scenario. Notice that as a result of a rule of behavior a binding could be triggered.

3.2.2. Endpoint SENSOR. Sensors are elements that can extract information from the context. CTP allows management of basic features and supports more abstract and complex capabilities.

Cluster SENSOR. As transducer channels TEDS, it manages the basic actions to request measure, defines sensor settings (e.g., sampling period), and defines the mode for transmitting the data set or data packets aggregating several measurements (e.g., on command, timed, or buffer full). It also provides essential attributes of the measurement such as units, accuracy, data ranges, warm-up time, vectors, and data packets. Every sensor type must implement this cluster.

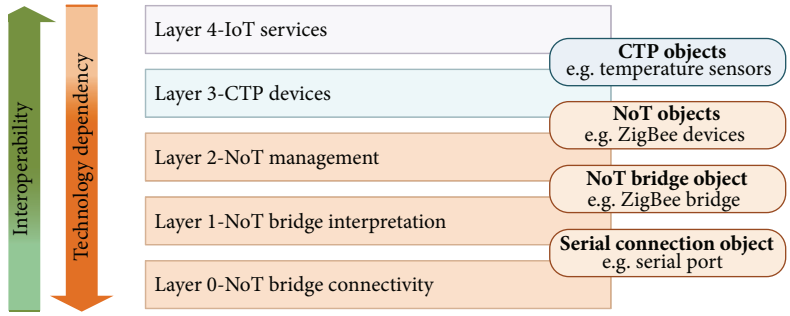


Figure 4: IoT gateway middleware architecture.

Cluster SENSOR_EVENT. Again as transducer channels TEDS it configures event triggers associated with a sensor (e.g., upper and lower thresholds, bit patterns, etc.) and provides the events.

Cluster SENSOR_STATS. This cluster performs local preprocessing and analysis of the captured data. This is of special interest when we are interested in obtaining a summary of the observation (e.g., maximum values, average values, etc.). As energy consumption between communication and computation in a wireless sensor node has a high ratio ("sending one bit" versus "computing one instruction") [37], its implementation is especially interesting in low power applications.

3.2.3. Endpoint ACTUATOR. While sensors allow obtaining contextual data, actuators are the elements that provide means to act over the environment. In relation with the IoT, an actuator can be considered as a device that transmits information or energy to another power mechanism or system (motors, electromagnets, thermocouples, heaters, coolers, etc.) that finally alters the environment. Thus, a *thing* with actuation capabilities is usually a device with control outputs, which is connected to an electromechanical system that opens doors, windows, shutters, control lighting, heating, and so forth.

Cluster ACTUATOR. It manages basic actions, controls the states of the actuator, and defines its configuration parameters. These states can be as simple as on/off or define complex operation patterns such as open the door for two minutes and then close and lock it.

3.2.4. Endpoint HMI. HMI (Human Machine Interface) devices are aimed to interact with a human user. In fact, they could be considered as sensors and actuators to interface people (in the same way that sensor and actuator connect with the environment). But given its importance and different use from the application perspective, it is convenient to assign its own type of endpoint. CTP considers the following clusters.

Cluster HMI_INPUT. It manages configuration and operation of simple input devices interface, such as buttons, switches, or dimmers. It also considers groups of elements such as arrays of keys to model a keypad.

Cluster HMI_OUTPUT. It manages basic operation of simple output devices such as LEDs and buzzers. It also considers groups of elements such as arrays of LEDS to model a LED strip.

4. IoT-Gateway Architecture

According to the proposed architecture, the different *things*, maybe on several NoTs, must interconnect to the Internet, that is, jumping to the IP world. In most cases, this involves changing not only the communications protocol but also communication technology, which imposes the need of a gateway interfacing *things* and Internet worlds.

In the proposed architecture, the gateway is not a single bridge between protocols; it is also an intelligence aggregator and distributor of services. The IoT gateway must support management (discovering, recruiting, and connecting devices) in the NoTs and provide interoperability between them. Such duty requires that the IoT gateway have enough power computation to handle requests and responses from both domains and a flexible architecture to ensure interoperability. To accomplish such task, the layered middleware architecture shown in Figure 4 is proposed.

Lower layer of the middleware is in charge of providing base connectivity with the NoT through the device acting as the network bridge, typically a USB dongle, a Bluetooth device, or a TCP socket which results in a sort of serial connection object allowing sending and receiving bytes as data streams.

Second layer adds meaning to those bytes, implementing the specific communication protocol of the bridge and allowing effective communication with the NoT. This results in an object representing the bridge, which encapsulates the communication protocol with appropriate methods and fields.

Middle layer is in charge of managing the NoT through the usage of the bridge representation service, being capable of discovering network devices (*things*), recruiting them, and handling network unavailability. At this layer, objects representing network devices are available, although they are already described in terms of the underlying technology.

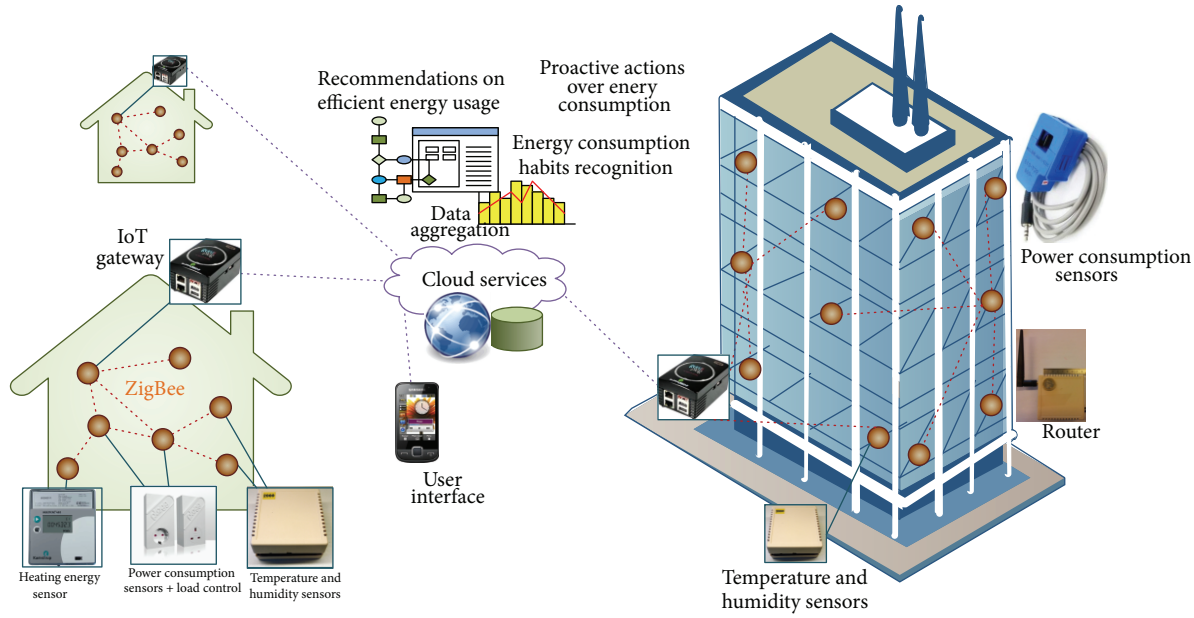


Figure 5: Test scenarios.

In the next layer, objects related to NoT devices are described as CTP objects, regardless their network technology, and full interoperability among *things* is achieved. CTP objects are described as a main device representing the endpoint BASE plus a collection of *channels* related to remainder endpoints.

Upper layer is devoted to gateway IoT services, which use CTP objects to link them to remote services (such as sending data to a remote database or allowing accessing a *thing* from a remote client) or build local services to perform offline operations.

5. Experimentation and Results

CTP has been successfully applied in several projects, but the largest implementations have been done in two European projects: "Easy Line Plus" [38] in the domain of Ambient Assisted Living and "Renaissance" [39] in the domain of Energy Efficient Buildings. The Easy Line Plus project sets the frame to define the CTP protocol, but it was within the Renaissance project where all the aspects described above have been formally deployed and tested. Therefore, we will describe below the Renasissance project setup.

5.1. Test Scenario. "Renaissance" main objective is energy saving, through the implementation of bioclimatic buildings, urban planning and rehabilitation, incorporating renewable energy, and reduction of energy consumption in households by improving habits of energy usage by users. Through remote data analysis, the system derives user patterns relating their energy consumption and comfort variables and then issues recommendations in order to increase their awareness and enhances energy savings. This requires an accurate and long-term monitoring of energy consumption and environmental conditions in order to generate enough data to extract relevant information.

To meet the needs of this project, the IoT paradigm is the ideal solution. The simplified structure of the system is a number of *things* that ubiquitously extract context information (temperature, humidity, heating energy, and power consumption). Through the proposed architecture, this information is handled (data aggregation, data base management, and cloud computing) and analyzed (habits recognition) to provide a range of services (data visualization, user profiling, assessment of the best practices, and user information through multimodal user interfaces).

Technically the system developed followed the architecture in Figure 1; *things* use ZigBee standard plus CTP; IoT gateway is a Linux embedded PC running the already presented middleware, and services run in different hosts (PC, mobile phones) using data stored in the cloud. As evaluation has been done in a real and operative deployment, the system previously ensured a number of quality requirements such as stability, ease of deployment and maintenance, and low intrusiveness in the context. Similarly, a number of limitations related to the *things* have been solved, namely, reducing power consumption to ensure months of operation with the same batteries, cost-effectiveness, and reduced size.

Two different types of installations have been done as follows (Figure 5):

(i) 80 m^2 dwellings with one ZigBee network formed by 3 ambient (temperature + humidity) sensors, 1 heating energy sensor, 1 monophasic power consumption sensor, and an IoT gateway;

(ii) 20.000 m^2 building with one ZigBee network formed by 70 ambient sensors, 20 triphasic power consumption sensors, 17 routers, and an IoT gateway.

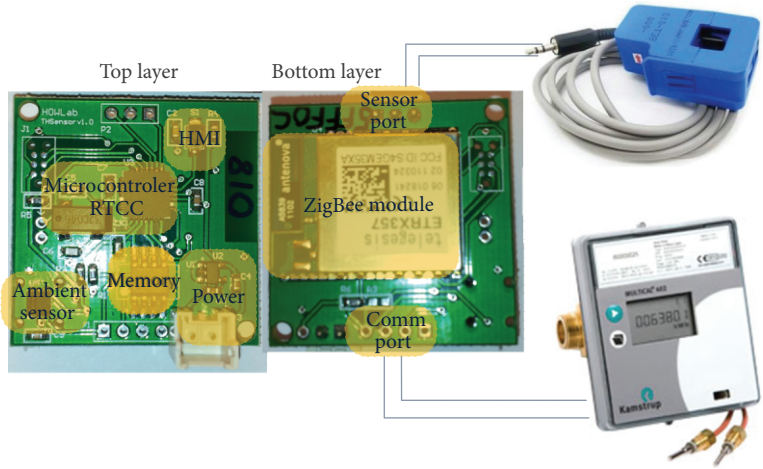


Figure 6: Zigbee *thing* hardware implementation.

In both cases, although the environment is quite different, the technological development is similar, allowing assessing and evaluating CTP in different scenarios.

The system has been installed and was running for 9 months in 43 dwellings simultaneously (i.e., 43 systems) in Zaragoza, proven to be stable, no maintenance was needed, and the expected functionality was provided. The building installation has been running for 6 months (already running) at the University of Zaragoza [40].

5.2. ZigBee Network of Things. Besides forming a ZigBee network, the sensors developed allow sensing the physical environment, connecting to analog or digital simple external sensors (e.g., presence detector, magnetic contact, etc.), and connecting to external commercial smart meters (e.g., commercial electricity or heating water consumption).

The aforementioned 20.000 m^2 building with long corridors, anti-fire doors, and so forth is a challenging scenario for wireless communication network with a hundred of nodes. A multihop mesh topology with an overlap of coverage areas and redundant communication paths was deployed. The ZigBee network backbone is formed by 17 routers, 1 network coordinator, and a data sink connected to the IoT gateway. The infrastructure is devoted to keep routing tables to date and interconnect 90 CTP *things* (70 ambient sensors and 20 power consumption sensors) that are ZigBee Sleepy End Devices. Any sensor enters low power mode between measurements, being disconnected from the network along this time, and its *father* (a router) holds its messages. Periodically, sensors poll their parents to check for incoming messages. This time between polls introduces latency in the communication but reduces energy consumption of the *thing*; to avoid possible loss of messages it is set to 4 seconds. Also, time between measurements, time between sending data, timestamping, and so forth can be configured in order to balance power consumption and application requirements. In this application, environmental nodes measure and immediately send data every 10 minutes, while power consumption sensors measure and store data every minute and then send it every 10 minutes. As it is not necessary that measures are simultaneous, the update process of the system is randomized to reduce the probability of several *things* trying to send messages at the same instant, which would increase medium access time and, therefore, energy consumption.

5.2.1. Hardware. Hardware implementation (Figure 6) is designed to be versatile. Device implementation is based in a dual hardware architecture where a low power microcontroller (PIC18F26J11) runs the main application and controls the network coprocessor that implements ZigBee Pro stack (Ember 357). After deep analysis and experimentation, we found architecture more efficient in terms of power consumption than using a system on chip solution (embedding a radio module plus a programmable microcontroller) as it allows splitting tasks between two specialized microcontrollers, one for sensing and processing tasks and the second for communication [41]. The device additionally has an EEPROM memory, a real time clock, expansion ports, a button, a LED, and a temperature and humidity digital I2C sensor (Sensirion SHT21). Along with these onboard capacities, the hardware features two expansion ports: first for connection of external analog/digital sensors and second for communications to external systems. Thanks to them, different *things* are built: a noninvasive AC current sensor (a SCT-013-000 0-100A split core current transformer) enables power consumption sensor and a communication port connected to a commercial device (Kamstrup) for measuring heating water consumption.

5.2.2. Firmware. Accordingly, the basic configuration of the device (only onboard capabilities) presents 5 endpoints: base (base type), temperature (sensor type), humidity (sensor type), button (HMI type), and LED (HMI type). Note that although SHT21 sensor is a single electronic component that measures temperature and humidity, there are two different endpoints one for each magnitude.

Nevertheless, similar to IEEE1451, access to both endpoints is possible using the proxy cluster within the base endpoint. According to the devices connected to the ports described above, news endpoints (power and heating water consumption, both sensor types) are added when necessary. The CTP protocol is therefore suited to the different characteristics of the hardware, while performing an abstraction of it.

Clusters implemented per endpoint are as follows:

(i) BASE: device, power (for battery level monitoring), time (to provide timestamps), proxy, and behavior (to program clima control when event happens),

(ii) SENSOR: sensor (to provide single measurements), sensor event (to get events when upper and lower thresholds are surpassed), and sensor stats (to provide maximum and minimum levels),

(iii) HMI: HMI input (to handle button events) and HMI output (to handle LED operation).

The use of CTP provides a structured and simple programming strategy, which results in modular code with high reusability. Figure 7 shows a simplified view of the proposed structure for a common firmware of a *thing*: configuration of the microcontroller, peripherals, ZigBee communications, and CTP and system behavior.

5.3. IoT Gateway. An embedded computer running Linux, where the middleware described before was implemented, acts as the IoT gateway. The middleware architecture falls within the SOA (Service Oriented Architecture) paradigm, and we use OSGi [42] (Open Services Gateway initiative) as development framework. OSGi defines a framework where pieces of code are organized into bundles that can be managed separately. OSGi bundles are agents which might be dedicated to specialized tasks, such as handling a serial port, providing a command line interface, collecting, aggregating and analyzing data, and so forth. These bundles communicate and interact with each other by means of services which are published within the framework, and each bundle can acquire and utilize them. The main strength of OSGi is that the framework manages these bundles dynamically, allowing them to be upgraded without terminating the full application, as well as enabling the availability of the services to other bundles depending on the situation. That allows us provinding new features and capabilities to the IoT Gateway by adding new services, which may use the services already existing in the framework, but keeping current features unaltered.

Using OSGi as development framework also eases development of the middleware layers, as we may focus on one single layer when developing, comprising one of more bundles, while the whole application layers will be arranged on execution time. Middle-layer interfaces are also defined to specify interoperability among adjacent layers and eliminate direct dependencies among bundles implementing each layer. According to Figure 8, the different layers have the following objectives.

(i) The network bridge is a USB dongle with a ZigBee module that can be controlled through an AT command interface on a serial port. On the lower layer, the Serial Communication middle interface defines a SerialConnection service, which basically allows writing bytes and notifies about incoming bytes, and a SerialDriver service, which is in charge of creating and discovering available SerialConnection services. Main bundle implementing this layer operates the UART interface, but other stream-based interfaces, such as a Bluetooth connection, may adopt the same middle-layer interface, allowing communication with Bluetooth devices transparently to the upper layers. In our particular case, we have implemented an application service that wraps a SerialConnection into a TCP socket, and at the remote client, another bundle creates a SerialConnection services that allows communication with the SerialConnection services on the server side like a local one. This would allow any service on the Internet to remotely handle the serial port traffic, which is very useful for debugging or auditing deployed IoT systems.

(ii) On the NoT Bridge Interpretation layer, a ZigbeeGateway bundle looks for newly created SerialConnection services, checks if they correspond to the Zigbee USB dongle, and creates ZigbeeGateway services implementing the AT command protocol in such a case. Again, the ZigbeeGateway service is defined in the ZigbeeGateway middle interface, which isolates layers and allows using different versions of the USB dongle.

(iii) Above that, there is a ZigbeeDriver, which uses the ZigbeeGateway to accomplish network management tasks, such as network creation or device enumeration, and a Driver Manager which uses the ZigbeeDriver to perform automatic maintenance tasks such as message route maintenance. It would be possible to aggregate these services into a single one but that will introduce complexity and hinder interoperability, whereas using separated services increases reliability and robustness as units of code are smaller and easier to test. This allows also deploying gateway facilities to accomplish transversal tasks, such as network monitoring, debug, maintenance, and commissioning, which may use limited parts of the Driver Layer. In the case of having other networks, the scheme is similar. The ZigbeeDriver will create ZigbeeNode services representing nodes in the network which allows sending and receiving messages to and from the physical device and provides Zigbee related properties such as its MAC address or its type.

(iv) On the CTP Devices layer, a CTP Driver service will use the ZigbeeNode services to create CTP Device services representing the real devices available on the NoT, following the CTP definition. This layer isolates devices' technology and registers them attending to their nature in order to provide interoperability; a temperature sensor always provides temperature the

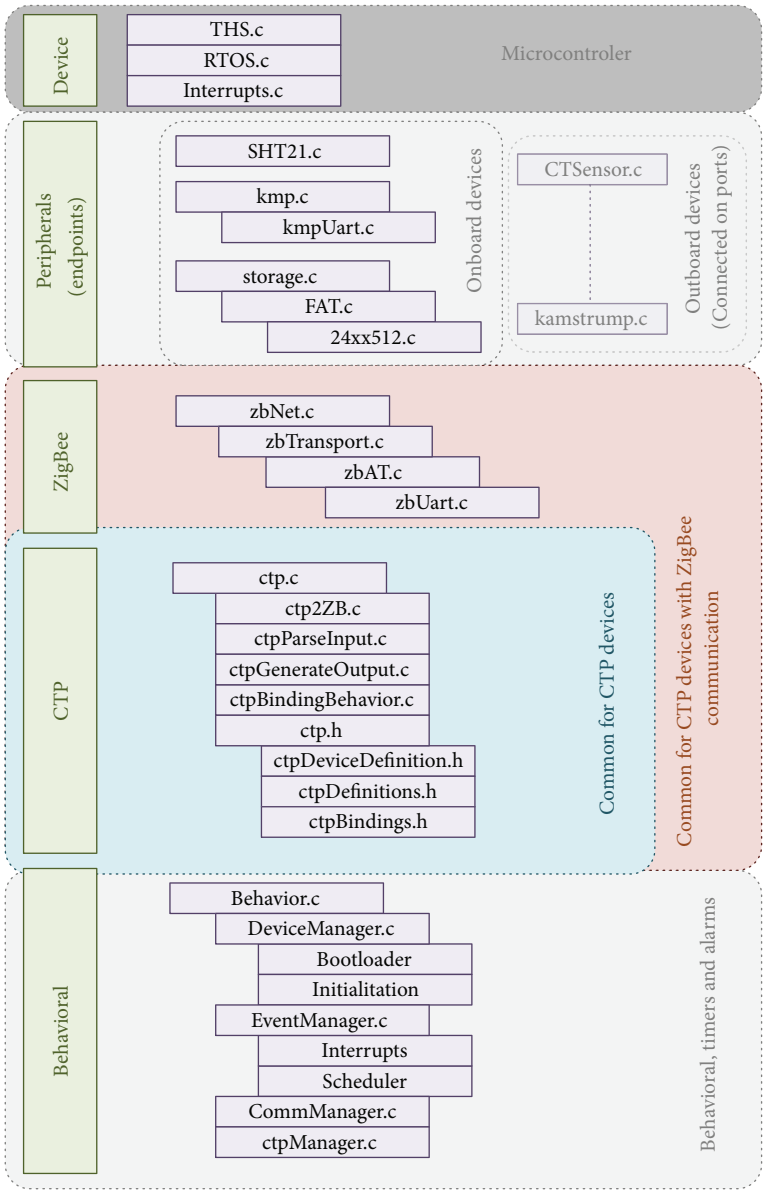


Figure 7: ZigBee *thing* firmware implementation.

same way regardless its technology. There are also different gateway facilities at this level (web services and TCP sockets) providing Internet access and virtual representation in order to allow remote applications to use NoT infrastructure.

(v) At the IoT services layer, a Data Collector service tracks for CTP Device services of type Sensor, subscribes itself to receive a notification when a new sensor value is received, and sends a data report to a remote database, where it can be accessed from elsewhere.

At this point, local network devices, which were not able to reach the IoT by themselves, are now represented by OSGi services which are smart enough to operate on the IoT and among them. Internet applications accessing those *things* require address resolution services which allow reaching them through an URL [2]. This is overcome by delegating *things*' addresses resolution to the IoT gateway which forwards incoming requests to the physical *thing*. Thus, accessing *things* from the Internet is granted by knowing the IoT gateway address. It is also feasible that *things* themselves access the IoT services directly; for example, we feed data from sensors to IoT services on the cloud specialized on data managing like Cosm [43] or Nimbits [44].

On top of the middleware described and thanks to the OSGi modularity, we also have developed communications terminal to sniff ZigBee communication (mainly intended as

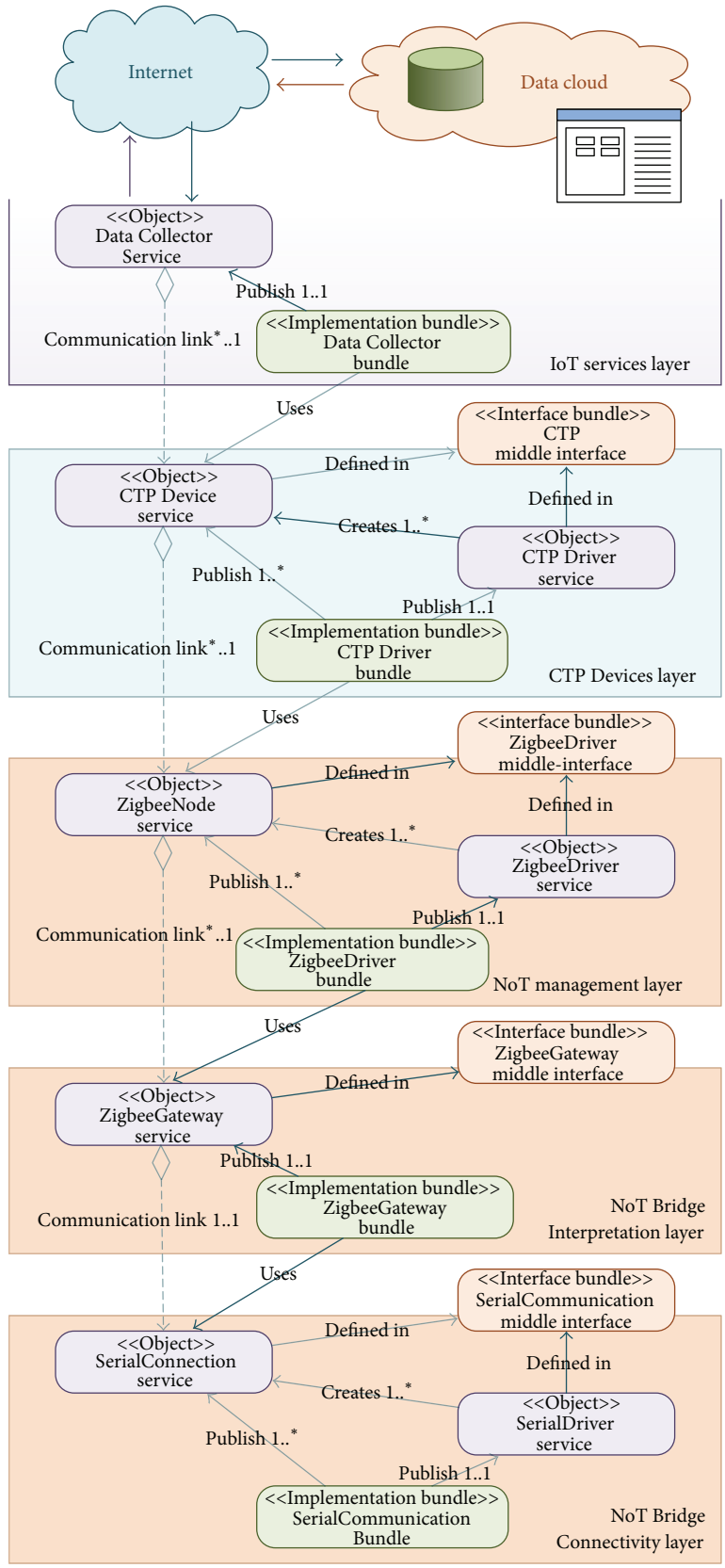


Figure 8: IoT gateway middleware architecture.

a development tool for hardware/firmware/software developers) and a Network Manager that allows full management and commissioning of deployed ZigBee networks.

5.4. System Performance. The system deployed is built by a backbone of 19 nodes (1 coordinator, 17 routers, and 1 data sink) that exchange network management messages. Also 90 sleepy sensors poll their parents every 10 seconds and report the data sink every 10 minutes. Due to network topology, some sensor messages must be relayed by routers up to five times to get to the data sink which passes them to the IoT gateway that maintains network's virtual image, checks data inconsistencies, registers loss of expected messages, and uploads data to exploitation database. These processes allow network commissioning, node-problem targeting, and data integrity validation.

Evaluating the quality of service (QoS) of an IoT application is not easy as it is a large and complex system based on heterogeneous technology and high application dependency. Currently, there are not well-defined standard metrics that can be used as a common agreed and widespread comparison of IoT systems; derived from this application specificity, researchers usually define their own metrics [45–47]. As IoT is usually made up by different subsystems, it is possible to analyze the layers separately; for example, there are many metrics to evaluate the effectiveness of data transfer in networks [48]; however, the success of an IoT should be assessed as a whole not just one particular aspect of the architecture [49].

The system proposed uses 90 sensors to deterministically measure ambient and power consumption information and make it available in the IoT. Three different areas are assessed: relative and absolute energetic cost, data efficiency, and message latency.

Data efficiency considers the amount of data generated in the ZigBee network and the data correctly stored in the cloud database. Similar indicators are used to assess the quality of a communication network, in which case it is common to consider lost messages and total messages. However, for the global evaluation of the IoT is preferable to consider both ends of the system, thus we use the messages generated by the physical *things* and the amount of data available in the logical scope. Note that if there would be nondeterministic events (e.g., alarms, presence detection, etc.) the data generated in the IoT will not be known a priori and also difficult to measure. We define the data efficiency of the IoT, $\mathrm{DE}_{\mathrm{IoT}}$, as

$$\mathrm{DE}_{\mathrm{IoT}} = \frac{\mathrm{DA}_{\mathrm{IoT}}}{\mathrm{DG}_{\mathrm{NoT}}} = \frac{\mathrm{DA}_{\mathrm{IoT}}}{\sum_{i=1}^{n} \left(\mathrm{MS}_i \times \mathrm{ND}_i\right)}, \tag{1}$$

where $\mathrm{DA}_{\mathrm{IoT}}$ is the data available on the IoT, $\mathrm{DG}_{\mathrm{NoT}}$ is the data generated by the NoT, n is the number of *things*, MS_i is the number of messages sent by *thing* I, and ND_i is the amount of data sent in each message (we assume this is constant for all the messages sent by a *thing*). This value can be measured accumulated or disaggregated in different periods to assess the variation of stability over the time. In our case, we check the number of new registers in database and considering that the total number of inputs should be 12960 per day (defined by the number of *things* and their scheduling), we can calculate the ratio. In our case, the accumulated data efficiency is 98.3%. It has been found that in general, errors are due to failures on the electrical supply or Ethernet network not directly attributable to the system.

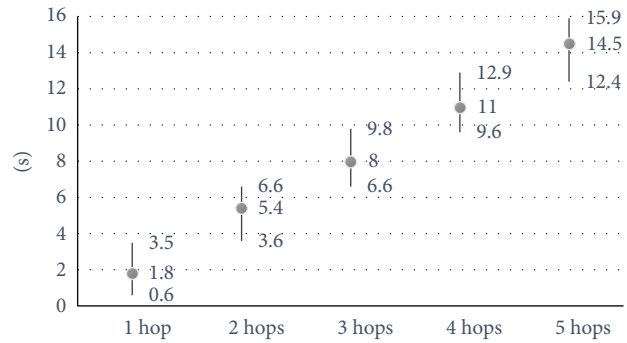


Figure 9: Latency time intervals.

Message latency quantifies the availability of a *thing* in order to exchange information with it. Again, we consider this availability from the IoT layer, that is, how much time takes to force a sensor to refresh, and effectively update the data in the cloud. There are two main factors that influence this indicator: how many hops away from the gateway is the sensor and how much time does the sensor spend in sleep mode (i.e., disconnected from the ZigBee network). Thus, we define the message latency per hop, MLH_j, as:

$$\mathrm{MLH}_j = \frac{\sum_{i=1}^{n_j} \mathrm{RT}_{ij}}{n_j}, \tag{2}$$

where RT_{ij} is the response time for *thing* i, which is j hops away from the gateway, and n_j is the number of *things* being j hops away from the gateway. Figure 9 shows the latency intervals (in seconds) within a 95% confidence interval according to the distance in hops between sensor and gateway. If we need to provide a global metric of the system latency we would say it will be between 0,6 s and 15,9 s.

Energy consumption of a device is a common indicator of its efficiency. In the particular case of IoT, we must take into consideration the consumption of the *things* andalso communications infrastructure (routers) and logical infrastructure (gateway). Given the data volume managed, energy consumption of routers and gateway is independent from the amount of data sent by *things*, and the absolute energetic cost of the system over time $\mathrm{AEC}(t)$ can be defined as follows:

$$\mathrm{AEC}\,(t) = \left[P_{\mathrm{Gtw}} + n_R \times P_R + \sum_{i=1}^{n} P_i\right] \times t, \tag{3}$$

where P_{Gtw} is the power consumption of the gateway, P_R is the power consumption of a router, n_R is the number of routers, P_i is the power consumption of *thing* i, and n is the number of *things*. As we show in [41] to calculate a *thing's* power consumption, a good characterization of its duty cycle is required. We have defined the figure of merit of power

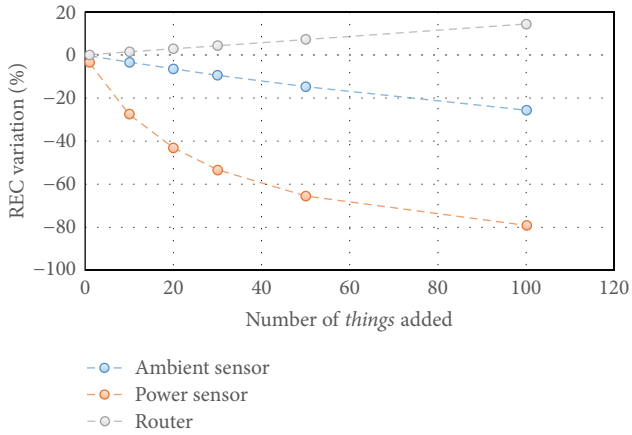


Figure 10: Relative energetic cost variation with respect to the proposed scenario when including new devices.

consumption for each type of *thing* and the routers and we consider constant the power consumption of the gateway. For the proposed IoT, the absolute energetic cost in one day is 19,76 MJ (5,49 kW·h). It is important to remark that only 0,013% of the energy is due to *things*, battery-powered devices. Given that in the proposed scenario, the number of data and time are directly related, we can also define the relative energetic cost REC as the energy required per byte of processed data during a day:

$$\mathrm{REC} = \left.\frac{\mathrm{AEC}(t)}{\mathrm{DA}_{\mathrm{IoT}}}\right|_{1\,\mathrm{day}}. \quad (4)$$

For the present IoT application we have a relative energetic cost of 121,97 J per byte sent. This includes 15 mJ corresponding to the energy drained from batteries. This parameter is related to the scalability of the network; if we enlarge the communications infrastructure (more routers to have broader coverage) and increase the number of *things* (to have more measurement points), relative energetic cost varies as in Figure 10.

Including new sensors supposes decreasing the REC as they provide additional data with reduced energy consumption. Specifically, power sensor consumption is related to a better REC than ambient sensor as it sends more data (44 bytes versus 4 bytes) with a similar energy consumption. As expected, including routers improves network backbone and/or range but worsens the metric, as they do not increase $\mathrm{DA}_{\mathrm{IoT}}$.

6. Discussion

Main conclusion of the described field trial is that architecture and the protocol proposed are feasible in large and long-term IoT deployments. Also it demonstrates that implementation of CTP over ZigBee in order to create low power *things* (running with batteries for a year) that efficiently integrates in an IoT system is possible. As we developed the whole system, from the hardware design of *things* to the service implementation, our concerns have been from the limited resources of hardware and firmware to the versatility and generality required by applications.

In order to analyze under which circumstances CTP is a convenient option, Table 1 compares it with main IoT protocols mentioned in Section 2. Three different levels are identified: (1) standard communication protocols (6lowPAN, RFID, WiFi, Cellular, ZigBee, and Bluetooth), (2) protocols abstracting from communication media and being mounted over the protocol's payload (IEEE1451, CTP), and (3) protocols assuming IP connectivity on any*thing* (CoAP, RESTfull-HTTP, XMPP, MQTT, SensorWeb, EEML, and SenseWeb). Comparison items are as follow.

(i) Application layer interoperability among *things* indicates whether *things* can effectively exchange information among them, for example, if a light controller (protocol A) can be operated by a light sensor (protocol B). Communication standards (1) are limited on this because they are not intended to define how to interoperate with other standards. On the other side, protocols abstracting from communication standard (2, 3) are precisely designed to enable this feature. Also IP-based protocols (3) are more restrictive as they need that *things* are IP enabled.

(ii) *Things*' representation models indicate if the protocol provides a virtual representation of *things*; for example, a temperature sensor has some characteristics (range, accuracy, etc.) and provides some methods (get temperature measurement, configure it, etc.). The protocols that just consider data exchange (e.g., 6LowPAN, WiFi, or MQTT) do not provide this definition hindering interoperability. ZigBee and Bluetooth define some *things*, those considered in their applications. IEEE1451, CTP, and SensorWeb define how any*thing* needs to be specified, for example, the datasheet format.

(iii) *Things*' interaction model: interaction means on step further than representation as it implies providing relationships among *things*, for example, how a light controller can be operated by a light sensor.

(iv) Suitability for low power and lossy networks: this relies very much on the possibility to run on wireless sensor network standards, mainly ZigBee and 6LowPAN.

(v) Simplicity and exhaustiveness are important features that determine adoption of protocols. For example, IEE1451 is a very exhaustive protocol defining everything in a sensor, but precisely this makes its use complicated by an app developer that just wants temperature value of a sensor.

According to Table 1, most similar protocols are CTP, IEEE1451, and ZigBee Cluster Library. In order to make a more detailed comparison among them, we define a scenario based on a ZigBee temperature sensor where these three specifications are encapsulated in the payload of the ZigBee application layer (Figure 11).

Table 1: IoT protocols comparison.

		Application layer interoperability among things	*Things'* representation models	*Things'* interaction model	LLN (Low Power and Lossy Networks) suitability	Simplicity	Exhaustiveness and level of detail
1	6lowPAN, RFID, WiFi, and Celular	Limited to standard (stack layer)	No	No	6lowPAN: excellent Others: fair	Not applicable (App. layer not defined)	Not applicable (App. layer not defined)
	ZigBee Cluster Library, Bluetooth Profiles	Limited to standard (App. layer)	Yes, but limited to specific applications	Yes, but limited to specific applications	ZigBee: excellent Bluetooth: fair	Good	Excellent
2	IEEE1451	Full (delegated to NoT protocol)	Yes, but limited to sensor and actuators	Yes, but limited to sensor and actuators	Excellent	Good	Excellent
	CTP	Full (delegated to NoT protocol)	Yes	Yes	Excellent	Excellent	Good
3	CoAP, RESTfullHTTP, XMPP, MQTT	Full (delegated to IP protocol)	No	No	CoAP: good (UDP) Others: fair (TCP)	Not applicable (App. layer not defined)	Not applicable (App. layer not defined)
	SensorWeb, EEML, SenseWeb	Full (delegated to IP protocol)	Yes	No	Fair (TCP)	EEML: Excellent Others: Fair	EEML: Good Others: Excellent

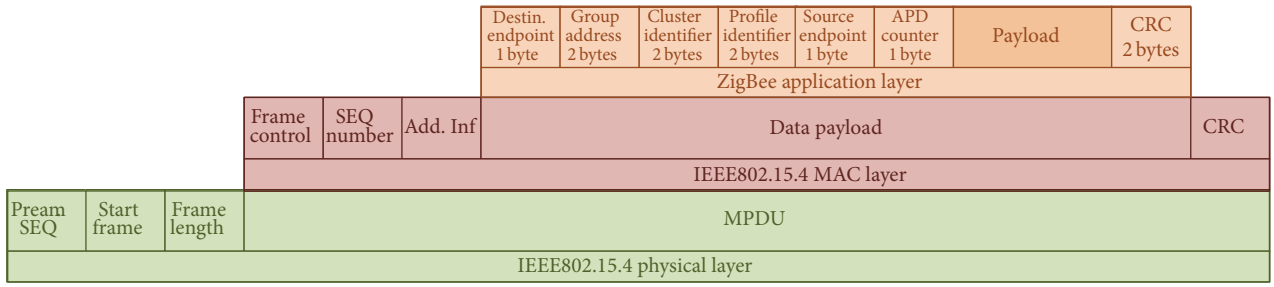


Figure 11: ZigBee protocol encapsulation.

Communication is established through requests and corresponding responses; Table 2 contains the frame descriptions for the three cases. Each frame includes different fields defined in the specifications.

In order to use the temperature sensor several requests need to be answered by the sensor: (1) retrieving general information about the device, (2) retrieving channel/endpoint mandatory information, and (3) retrieving temperature measurement. If using ZCL, sensor temperature must be 2 bytes in length, in degrees Celsius, and specified in the range from −273.15°C to 327.67°C with aresolution of 0.01°C. Thus, step 2 is not needed as information about the measure is already defined. This is positive in terms of easy use but limitative in some cases; if using CTP or IEEE 1451, variables such as accuracy, range, units, and so forth are open to the hardware developer and must be specified in the channel/endpoint mandatory information. Table 3 indicates the number of bytes exchanged for each action and the number of variables encoded. It should be noted that just mandatory fields are considered in the table.

Regarding protocol efficiency, it is obvious that ZCL is the most efficient, but this has important implications: as ZCL focuses on specific application domains (home and building automation, health, etc.) if the *thing* is not specified in the standard or the representation of the information differs from it, its implementation is not possible and thus it will not be interoperable. For example, there is no way to use inertial sensors according to ZCL.

On the other hand, IEEE 1451 is the heaviest protocol because it leaves open a lot of variables. This flexibility turns to be its main drawback to be used in IoT applications because it makes it too complicated for developers not versed into electronics. Additionally, as it is more oriented to electronics than to application, it does not consider human-machine interfaces in its specification, just sensors and actuators.

Table 2: Frame description of IEEE 1451, CTP, and ZCL.

		Header	Payload
IEEE 1451	Request (6 + *n*) bytes	Destination transducer channel: 2 bytes Command Class: 1 byte Command Function: 1 byte Offset Length: 2 bytes	Data: *n* bytes
	Response (3 + *n*) bytes	Success flag: 1 byte Offset Length: 2 bytes	Data: *n* bytes
CTP	CTP Frame (5 + *n*) bytes	Endpoint destination: 1 byte Length: 2 bytes Cluster identification: 1 byte Command identifier: 1 byte	Data: *n* bytes
ZigBee Cluster Library	ZCL Frame (5 + *n*) bytes	Frame Control: 1 byte Manufacturer code: 2 bytes Transaction sequence number: 1 byte Command identifier: 1 byte	Id attribute: 2 bytes Value attribute: *n* bytes ⋮ Id attribute *m*: 2 bytes Value attribute *m*: *n* bytes

Table 3: Number of payload bytes (differentiating request + response) and variables associated.

	IEEE 1451		CTP		ZigBee Cluster Library		Observations
	Bytes	Variables	Bytes	Variables	Bytes	Variables	
Retrieve general information about the device	50 (7 + 43) 42 (7 + 35)	21	45 (5 + 31)	5	14 (7 + 7)	1	In case of ZCL, information obtained is partial, same amount/type of variables must be done from a lower level of the protocol; at ZigBee Device Object.
Retrieve channel/endpoint mandatory information	111 (7 + 104) 33 (7 + 26)	18	34 (5 + 29)	7	—	—	ZCL strictly defines the nature of the variables in each cluster, so no need to declaration.
Retrieve temperature measurement	12 (7 + 5)	1	12 (5 + 7)	1	14 (7 + 7)	1	

As a summary, CTP is a balanced protocol especially suited for the IoT allowing definition of any device and just specifying those variables that are needed at application level. As it can be efficiently encapsulated in any communication protocol, it is strongly focused on the interoperability of *things* while considering their technical limitations (energy, processing, and communication capabilities). CTP is based on an ontological representation and interaction model; thus, besides interoperability, it allows local intelligence and interaction among *things*. Also as it describes the characteristics and properties of *things*, higher-level protocols can use it as a source of information.

7. Conclusions

IoT is perceived as a huge generator of services and applications, but it requires that two important issues to be solved: *things*' connectivity (communication of the physical *things* with the Internet) and interoperability at all levels (understanding of the information exchanged). This paper emphasizes the need to maintain an integrative and broad point of view when considering architectures and protocols for IoT. Considering IoT development areas and their associated technologies, an architecture and protocol with a comprehensive view considering current technological capabilities at all levels (services, gateway communications, firmware, and hardware) are proposed.

In order to tackle connectivity challenge, the appropriateness of using gateway-based solutions to connect Networks of *Things* with the Internet has been discussed. This is considered an optimal way to solve the problem of the heterogeneous networks, jumping from the real world to the IP world and ensuring the unambiguous designation of each of the *things* in the Internet, performing a tunneling process (e.g., the need of a unique identification). Regarding the functionality of the IoT gateway, the use of layered and modular middleware architecture to constitute a versatile, interoperable, scalable, and easy to maintain system has been proposed.

Common *Things* Protocol (CTP) has been presented as a solution to provide interoperability among *things*. Main guidelines considered in its design are an ontological representation and interaction model of *things* andimplementation feasibility in standard communication protocols. CTP reflects the need for equilibrium between the networks of *things*, data traffic, and virtualized world of *things*, which

is not common in the current proposals. Main principles of CTP are self-description of the nature and capabilities of the *thing*, organization capabilities through the endpoint and cluster concepts, simplifying the use of the *thing* in standard situations through mechanisms such as modes and scenarios, allowing distributed intelligence by using bindings and behaviors, and being simple and compact to ease its implementation fitting on current standard communication protocols.

The IoT architecture (ZigBee network of *things*, IoT gateway and related Internet services) and CTP proposal have been implemented in two European research projects related to smart environments for Ambient Assisted Living and energy efficiency. The paper describes specific hardware and software implementations and deployment in large scale environments, with real end users, during periods of several months. We also measure energetic cost, data efficiency, and message latency metrics demonstrating proposal's feasibility and suitability in order to meet current IoT requirements.

This paper does not intend to define a standard but an initial proposal to adopt a minimum agreement (currently not considered in the state of the art) with a global and inclusive vision that facilitates the development of the IoT. To this end, it should be mentioned that most of the developments shown have been carried out under open source license, in particular CTP is accessible and well documented [50].

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] I. T. Union, "ITU Internet Reports 2005: The Internet of *Things*," 2005.

[2] L. Atzori, A. Iera, and G. Morabito, "The Internet of *Things*: a survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[3] G. Kortuem, F. Kawsar, V. Sundramoorthy, and D. Fitton, "Smart objects as building blocks for the internet of *things*," *IEEE Internet Computing*, vol. 14, no. 1, pp. 44–51, 2010.

[4] R. V. Kulkarni, A. Förster, and G. K. Venayagamoorthy, "Computational intelligence in wireless sensor networks: a survey," *IEEE Communications Surveys and Tutorials*, vol. 13, no. 1, pp. 68–96, 2011.

[5] ISTAG, "Orientations for EU ICT R&D & Innovation beyond 2013-10 keys recomendation," July 2011, http://cordis.europa.eu/fp7/ict/istag/documents/istag_key_recommendations_beyond_2013_full.pdf.

[6] M. Swan, "Sensor Mania! The Internet of *Things*, Wearable Computing, Objective Metrics, and the Quantified Self 2. 0," *Journal of Sensor and Actuator Networks*, vol. 1, no. 3, pp. 217–253, 2012.

[7] "sensei (Real World Dimension of the NEtwork of the Future)," http://www.ict-sensei.org/index.php.

[8] "IoT- A, (Internet of *Things* Architecture)," http://www.iot-a.eu/public.

[9] "Internet of *Things* Strategic Research Roadmap," European Commission-Information Society and Media DG-BU31 01/18 B-1049, Brussels, Belgium, 2009.

[10] D. Uckelmann, M. Harrison, and M. Florian, "An architectural approach towards the future internet of *things*," in *Architecting the Internet of Things*, pp. 1–24, Springer, 2011.

[11] D. Miorandi, S. Sicari, F. de Pellegrini, and I. Chlamtac, "Internet of *things*: vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, 2012.

[12] MQTT, http://mqtt.org/.

[13] "ipSo Alliance," http://www.ipso-alliance.org/.

[14] T. Usländer, A. Berre, C. Granell et al., "The future internet enablement of the environment information space," in *Proceedings of the International Symposium on Environmental Software Systems (ISESS '13)*, Neusiedl am See, Austria, 2013.

[15] C. Gómez, J. Paradells, and J. Caballero, *Sensors Everywhere. Wireless Network Technologies and Solutions*, Fundación Vodafone España, 2010.

[16] A. Kapadia, S. Myers, X. Wang, and G. Fox, "Secure cloud computing with brokered trusted sensor networks," in *Proceedings of the International Symposium on Collaborative Technologies and Systems (CTS '10)*, pp. 581–592, May 2010.

[17] A. Ibarz, G. Bauer, R. Casas, A. Marco, and P. Lukowicz, "Design and evaluation of a sound based water flow measurement system," *Lecture Notes in Computer Science*, vol. 5279, pp. 41–54, 2008.

[18] ZigBee Specification, "ZigBee Cluster Library Specification," (053474r17), 2008.

[19] Y.-F. Lee, H.-S. Liu, M.-S. Wei, and C.-H. Peng, "A flexible binding mechanism for ZigBee sensors," in *Proceedings of the 5th International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP '09)*, pp. 273–278, December 2009.

[20] C. Bormann, A. P. Castellani, and Z. Shelby, "CoAP: an application protocol for billions of tiny internet nodes," *IEEE Internet Computing*, vol. 16, no. 2, pp. 62–67, 2012.

[21] XMPP Standards Foundation, "XMPP Standard," http://xmpp.org/.

[22] "Constrained RESTful Environments (core)," https://datatracker.ietf.org/wg/core/charter/.

[23] "How Can IEEE, 1451 Be Applied," June 2007, http://ieee1451.nist.gov/.

[24] I. I. a. M. Society, *IEEE Standard for a Smart Transducer Interface for Sensors and Actuators*, 2007.

[25] D. Sweetser, V. Sweetser, and J. Nemeth-Johannes, "A modular approach to IEEE-1451.5 wireless sensor development," in *Proceedings of the IEEE Sensors Applications Symposium*, pp. 82–87, February 2006.

[26] J. Higuera, J. Polo, and M. Gasulla, "A ZigBee Wireless sensor network compliant with the IEEE 1451 standard," in *Proceedings of the IEEE Sensors Applications Symposium (SAS '09)*, pp. 309–313, February 2009.

[27] J. E. Higuera and J. Polo, "IEEE 1451 standard in 6LoWPAN sensor networks using a compact physical-layer transducer electronic datasheet," *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 8, pp. 2751–2758, 2011.

[28] Z. Wemlinger and L. Holder, "The COSE ontology: bringing the semantic web to smart environments," in *Toward Useful Services for Elderly and People with Disabilities*, vol. 6719, pp. 205–209, Springer, 2011.

[29] D. Bonino, E. Castellina, and F. Corno, "DOG: an ontology-powered OSGi domotic gateway," in *Proceedings of the 20th IEEE International Conference on Tools with Artificial Intelligence (ICTAI '08)*, pp. 157–160, November 2008.

[30] W. Wang, S. De, G. Cassar, and K. Moessner, "Knowledge representation in the internet of things: semantic modelling and its applications," *Automatika*, vol. 54, no. 4, pp. 388–400, 2013.

[31] B. Christophe, "Semantic profiles to model the 'web of *things*'," in *Proceedings of the 7th International Conference on Semantics, Knowledge, and Grids (SKG '11)*, pp. 51–58, October 2011.

[32] E. Y. Song and K. Lee, "Understanding IEEE 1451—networked smart transducer interface standard—what is a smart transducer?" *IEEE Instrumentation and Measurement Magazine*, vol. 11, no. 2, pp. 11–17, 2008.

[33] R. Johnson and S. P. Woods, "Proposed enhancements to the IEEE, 1451. 2 standard for smart transducers," September 2009, http://archives.sensorsmag.com/articles/0901/74/main.shtml.

[34] M. Compton, P. Barnaghi, L. Bermudez et al., "The SSN ontology of the W3C semantic sensor network incubator group," *Web Semantics*, vol. 17, pp. 25–32, 2012.

[35] L. Feng, P. M. G. Apers, and W. Jonker, "Towards context-aware data management for ambient intelligence," in *Database and Expert Systems Applications*, vol. 3180 of *Lecture Notes in Computer Science*, pp. 422–431, Springer, 2004.

[36] S. Petersen and S. Carlsen, "WirelessHART versus ISA100.11a: the format war hits the factory floor," *IEEE Industrial Electronics Magazine*, vol. 5, no. 4, pp. 23–34, 2011.

[37] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[38] European 6th Framework Programme for Research and Technological Development, "Easy Line Plus," http://www.easylineplus.com/.

[39] C. P. E. Commision, "Renaissance," http://www.renaissance-project.eu/?lang=en.

[40] U. o. Zaragoza, "Liga Energetica edificio I+D+i," http://proyectosostenibilidad.unizar.es/institutos/index.php/monitorizacion.

[41] Á. Asensio, R. Blasco, Á. Marco, and R. Casas, "Hardware architecture design for WSN runtime extension," *International Journal of Distributed Sensor Networks*, vol. 2013, Article ID 136745, 11 pages, 2013.

[42] O. A. website, "OSGi Alliance website," http://www.osgi.org/Main/HomePage.

[43] http://www.cosm.com.

[44] http://www.nimbits.com.

[45] M. W. Ryu, J. Kim, S. S. Lee, and M. H. Song, "Survey on internet of *things*: toward case study," *Smart Computing Review*, vol. 2, no. 3, 2012.

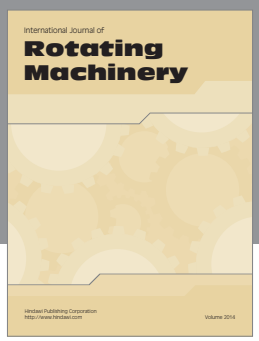
[46] M. Paschou, E. Sakkopoulos, E. Sourla, and A. Tsakalidis, "Health Internet of *Things*: metrics and methods for efficient data transfer," *Simulation Modelling Practice and Theory*, vol. 34, pp. 186–199, 2013.

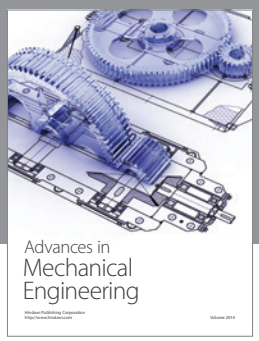
[47] A. P. Castellani, N. Bui, P. Casari, M. Rossi, Z. Shelby, and M. Zorzi, "Architecture and protocols for the internet of *things*: a case study," in *Proceedings of the 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM '10)*, pp. 678–683, April 2010.

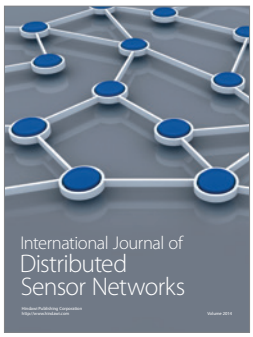
[48] S. Forsström, P. Österberg, and T. Kanter, "Evaluating ubiquitous sensor information sharing on the internet of *things*," in *Proceedings of the IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, 2012.

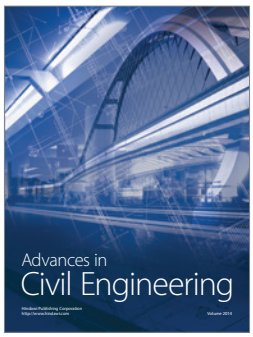
[49] J. Gubbia, R. Buyyab, S. Marusic, and M. Palaniswami, "Internet of *Things* (IoT): a vision, architectural elements, and future directions," *Future Generation Computer Systems*, no. 29, pp. 1645–1660, 2013.

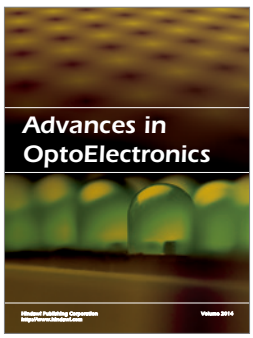
[50] HOWLab, 2013, http://openlab.unizar.es/index.php/conocimiento/ctp.

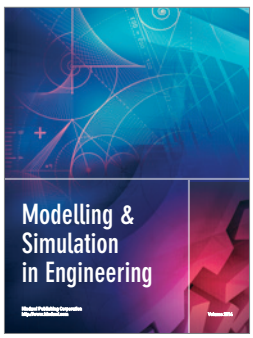


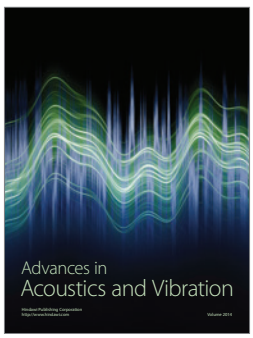


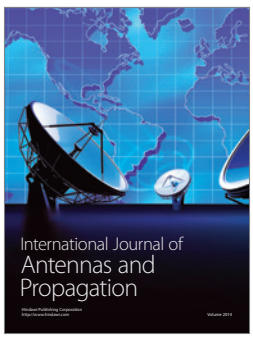


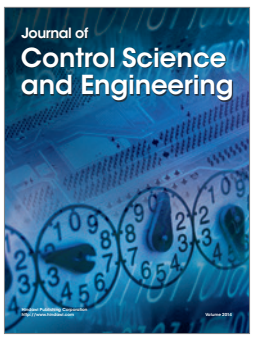


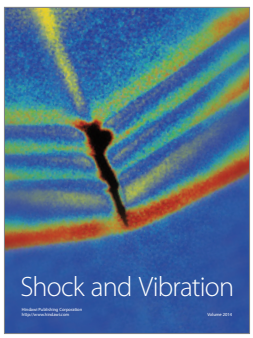












Hindawi

Submit your manuscripts at
http://www.hindawi.com