A novel cloud-based IoT architecture for Smart Building automation

David Sembroiz[1], Sergio Ricciardi[1]andDavide Careglio[1]

[1]Technical University of Catalonia (UPC) - BarcelonaTech, Department of Computer Architecture, Barcelona, Spain

November 3, 2016

1 Introduction to Internet of Things

Internet of Things or IoT is meant to be the future of the current Internet. It is commonly defined as a network of physical and virtual objects, devices or things that are capable of collecting surrounding data and exchanging it between them or through the Internet. To enable these data collection, devices are embedded with sensors, software and electronics and the exchange capability is achieved by connecting them to local area networks or to the Internet.

The origins of the Internet of Things are diffuse. Even though the word was first coined in 1999 by Kevin Ashton, co-founder and executive director of the Auto-ID Center at MIT, for companies such as CISCO, the IoT was born in 2009, when more devices than people were connected to the Internet. At that time, the number of connected devices were 10 billion, but the expectations are generous. It is thought that by 2020, more than 50 billion devices will be connected to the Internet.

As it can be extracted from the numbers, during the last few years, the Internet of Things has seen an unexpected increase in popularity, mainly thanks to the following technology improvements:

- **Smaller, more durable and powerful sensors.** New manufactured sensors are seeing their size substantially reduced, allowing their placement in small spaces and also in delicate and dangerous scenarios.
- **Increased efficiency.** One of the key aspects of the Internet of Things paradigm is the wireless interconnection between devices. Thus, these devices must be equipped with autonomous power supplies that limit their lifespan. To cope with this problem, manufacturers are aiming for efficient processors and software engineers are specifically designing software and communication technologies for IoT in which lower energy consumption

https://doi.org/10.1016/B978-0-12-811373-8.00010-0

is the main requirement. To achieve this, sensors usually work in low-power-usage mode. Devices remain in sleep mode until a new sample message needs to be generated. Then, it wakes up, creates the message and transmits it by powering up the RF power amplifier. When the message is transmitted, both the RF power amplifier and the device are turned down until the next cycle.

- **Lower production cost.** The improvements in industry and the easiness in which mass production is currently achieved allow companies to lower the price of each component.

The combination of these improvements created new market opportunities that companies have foreseen. Since the Internet of Things is in a very young state, the lack of coordination and the rapidness with which new gadgets are created are hampering the standardization of the future Internet.

The possibility to attach small hardware to any kind of electronic or mechanic device enables the possibility to monitor everything. That is why the Internet of Things is also defined as the interaction with *anything, anytime, anywhere.*

Smart Buildings are gaining popularity and many companies are putting their efforts in this field since the number of potential clients is very high. Elements such as smart fridge, smart thermostat, smart lightning are appearing to ease daily life and to increase peoples comfort.

In the case of the health-care systems, companies aim for taking advantage of smart and small devices to monitor the condition of people in order to detect anomalies and immediately inform familiars or even hospitals.

2 Main enabling technologies and protocols

Since the beginning of the current Internet, many groups have been created for helping in the standardization of protocols and technologies. Internet Engineering Task Force (IETF), World Wide Web Consortium (W3C), Institute of Electrical and Electronics Engineers (IEEE) are some of the most important ones.

In the initial steps of IoT, protocols such as RFID and NFC were the standard the facto mainly due to its low production cost. However, its transmission limitations in terms of range coverage and inability to communicate through the Internet hampers their usage in new IoT scenarios such as smart buildings or cities. In the industry sector, they are still widely used for packet tracking and object identification.

During the last years, groups have put their efforts in creating standards for protocols directly related to the Internet of Things. Even though it is possible to use the old communication protocols such as Bluetooth or Wi-Fi, their characteristics do not fit with IoT device requirements. Many IoT devices rely on the necessity of having external power supplies such as batteries to work, requiring reduced power consumption and cost, while maintaining similar communication

range with respect to their analogous current Internet protocols. To cite some, Bluetooth Low Energy, Wi-Fi HaLow or LoRaWAN are IoT focused protocols with the stated requirements. Even though the core of such protocols is very similar, depending on the kind of application being developed, one may fit best than another. Moreover, the usage of multiple protocols inside the same system is not forbidden. For instance, a general system combining many information sources, each of them using the specific sensor devices, can use the protocol that fits best by taking into account devices connectivity and location.

IoT enabling protocols can be divided into two major groups, named Infrastructure protocols and Application protocols. Infrastructure protocols refer to the ones that actuate inside the underlying infrastructure and create the communication between system layers. For instance, the connection between the perception layer and the network layer, or the one between the network layer and the cloud layer. Application protocols are responsible for interconnecting the infrastructure with the application.

In the following sections, the most relevant protocols of both groups are explained more in depth. To summarize, a table comparing the relevant features is also shown.

2.1 Wireless Infrastructure Protocols

2.1.1 Bluetooth Low Energy (BLE)

Bluetooth Low Energy or Bluetooth Smart, as it has been branded, is an enhancement of the Bluetooth technology in which connectivity and power usage are smarter than its predecessor. However, devices with Bluetooth Smart technology attached are not compatible with previous versions. To cope with this problem, Bluetooth Special Interest Group completed the Bluetooth Core Specification version 4.0 to include compatibility between versions. Current devices include this new core protocol making them able to communicate with any Bluetooth device.

The shifting in the connection paradigm performed by the Internet of Things has forced new protocols to include new behavioral modes. Bluetooth Smart includes ultra-low peak, average and idle modes. Once the pairing between two devices is performed, Bluetooth Smart focuses on sending small bits of data when needed and putting the connection in a low power consumption mode in order to drastically reduce energy usage.

According to the Bluetooth SIG specification, this protocol has been specifically designed for smart home, health, sports and fitness sectors. These sectors can take advantage of the following Bluetooth Smart features [1]:

- Low power requirements, allowing the devices to operate for months or even years.
- Small size and low cost.
- Compatibility with a large base of mobile phones, tablets and computers, allowing the interoperability between such devices.

As for its technical details, Bluetooth Smart operates in the same spectrum range than its predecessor, the 2.4 GHz-2.4835 ISM band. However, the set of channels used vary significantly. Instead of the classic 79 1-MHz channels, Bluetooth Smart uses 40 2-MHz channels. Regarding bit rate and maximum transmission power, they are limited to 1 Mbit per second and 10 milliwatts respectively. Its range coverage is ten times that of the classic Bluetooth (10 meters versus 100 meters approximately). Latency wise, it is 16 times shorter (100ms versus 6ms) [2].

	Bluetooth Classic	**Bluetooth Smart**
Spectrum Range	2.4-2.4835 GHz	2.4-2.4835 GHz
Channel Bandwidth	1 MHz	2 MHz
Number of channels	79	40
Max. Bit Rate	3 Mbps	1 Mbps
Max. Transmission Power	100 mW	10 mW
Avg. Range	10 m	100 m
Avg. Latency	100 ms	6 ms

Table 1: Comparison between Bluetooth Classic and BLE [2].

2.1.2 ZigBee

ZigBee is a standard based on the IEEE 802.15.4 specification specially targeted for long battery life devices in wireless mesh networks. This protocol has been evolving since its appearance in 1999, and its last specification is called ZigBee PRO, from 2007. Even though it shares features with Bluetooth, ZigBee is intended to be simpler and less expensive.

Regarding operation bands, ZigBee uses the same as Bluetooth [3], the 2.4 GHz band. In some locations, this band varies. For instance, China uses the 784 MHz band, Europe uses the 868 MHz band, whilst USA and Australia uses the 915 MHz one.

Its simplicity also limits some important aspects such as transmission rate and communication range. Unlike Bluetooth, data transmission is limited to a maximum of 250 Kbit per second, which may be enough depending on the scenario under use. Communication range varies between 10 and 20 meters for indoor transmissions, depending on power output and environmental characteristics.

2.1.3 6LoWPAN

The IPv6 over Low power Wireless Personal Area Networks or 6LoWPAN was created by a concluded working group in the Internet area of the IETF to fulfill the necessity to allow any kind of devices, even the smallest ones with limited power usage and processing capabilities, to participate in the Internet of Things.

6LoWPAN is a combination of IEEE 802.15.4 and IP in a simple well understood way. The key features of this protocol are the encapsulation definition and header compression that allow the compatibility between local area networks and wide area networks with IEEE 802.15.4 based networks.

Since 6LoWPAN pertains to the network layer of the OSI model, it does not have a specific transmission specification. Instead, the underlying link layer protocol is responsible for providing them. As mentioned before, this protocol has been designed to work on top of IEEE 802.15.4 based networks which provides the transmission characteristics already explained in section 2.1.1.

2.1.4 Wi-Fi HaLow

Wi-Fi HaLow is a very new technology presented in January 2016 in the Computer Electronic Show (CES) by the Wi-Fi Alliance [4]. This new Wi-Fi specification is directly suited to meet the unique requirements of IoT environments such as Smart Homes, Smart Cities and Industrial markets. It extends Wi-Fi, and specifically its 802.11ah specification, to operate into the 900 MHz band, enabling the low power connectivity necessary for applications including sensors and wearables which hardly rely on battery lifetime. Its range has been extended to almost twice that of current Wi-Fi, and will not only be capable of transmitting further, but also providing a more robust connection in harsh environments thanks to its ability to penetrate walls or other barriers more easily.

Devices with HaLow support are expected to also support current 2,4 and 5 GHz Wi-Fi bands, allowing interoperability between current devices and newly ones. They also support IP-based connectivity to natively connect to the Internet. Another important point worth mentioning is the ability to connect thousands of devices to a single access point to create dense device deployments. As for its transmission power, HaLow is expected to work between 150 Kbps and 18 Mbps depending on the requirements of the application. To support such transmission rates, different channel setups are required: 150 Kbps only requires a 1 MHz channel but the maximum transmission rate requires a 4 MHz-wide channel.

This new technology is the answer to Bluetooth, and Wi-Fi alliance expects to begin certifying HaLow products in 2018.

2.1.5 LoRaWAN

LoRaWAN [5] is a Low Power Wide Area Network created by the LoRa Alliance as a solution for wireless battery operated devices. It specifically target the IoT main requirements such as secure communication, mobility and localization services. In a typical LoRaWAN network, devices and gateways compose a star of stars topology in which only the gateways are connected to the Internet, whereas devices use single-hop wireless communication to transmit their data to a single or multiple gateways simultaneously. The transmission between devices

and gateways is bi-directional, but it also enables the possibility for multi-cast messaging for Over The Air software upgrade.

LoRaWAN supports a wide range of frequency channels and data rates. Moreover, transmission with different specifications to the same gateway do not interfere with each other. Every transmission is encapsulated in a separate *virtual* channel which increases the capacity of the gateway significantly. Data rates range between 0,25 Kbps and 50 Kbps.

LoRaWan defines three classes for end point devices to address the different needs reflected in the wide range of possible applications:

- Bi-directional end devices (Class A): asynchronous transmissions in which every uplink message is followed by 2 short downlink windows that the gateway can take advantage of to send messages to the end devices. After these windows have finished, the end devices is set to idle until the next uplink transmission. This class operates in the lowest power and is suitable for applications that only need end device to gateway communication.
- Bi-directional end device with scheduled receive slots (Class B): In addition to Class A random receive windows, end devices are told by means of a time synchronized Beacon from the gateway which time slot they must listen for any possible downlink communication.
- Bi-directional end devices with maximal receive slots (Class C): end devices are continuously listening for downlink messages and this window is only closed when transmitting to the gateway. This class is usually targeted for AC-powered devices because of its high power consumption.

	BLE	**ZigBee**	**6LoWPAN**
Spectrum Range	2.4-2.4835 GHz	2.4-2.4835 GHz	868/915/2400 MHz
Bit Rate	1 Mbps	20 - 250 Kbps	250 Kbps
Peak Consumption	$<$ 15 mA	30 - 40 mA	$<$ 15 mA
Range	10 - 100 m	10 - 100 m	10 - 200 m
	Wi-Fi HaLow	**LoRaWAN (EU)**	
Spectrum Range	900 MHz	868 MHz	
Bit Rate	150 Kbps - 18 Mbps	0,25 - 50 Kbps	
Peak Consumption	$\sim$ 50 mA	$\sim$ 38 mA	
Range	1 km	2 - 22 km	

Table 2: Comparison between the main enabling infrastructure IoT protocols.

Table 2 acts as a summary and comparison between the main infrastructure protocols presented. To clarify, communication range shows the distance to which these technologies can transmit depending if the transmitter and the receiver are in Line Of Sight (LOS) or not. Regarding spectrum usage, it can vary depending on the location since the legislation varies in every continent.

2.2 Application Layer Protocols

2.2.1 Hypertext Transfer Protocol (HTTP)

The Hypertext Transfer Protocol (HTTP) is an application layer protocol designed for distributed, collaborative, hypermedia information systems. This protocol is the foundation of data communication for the World Wide Web.

HTTP was initiated in 1989 at the European Organization for Nuclear Research (CERN). However, the development of standards was coordinated by the IETF and the Word Wide Web Consortium (W3C), culminating in the publication of a group of RFCs in 1997, with the definition of the HTTP/1.1 version in [6] firstly, and updated in [7]. During many years this has been the standard *de facto*. In 2015, the successor HTTP/2 was standardized [8].

HTTP works as a request-response protocol in the client-server computing model. The majority of the time, it uses TCP as a transport protocol for reliability. However, it can be adopted to use unreliable protocols such as UDP.

Even though its usage has been extended to the IoT world, it was not specifically designed for this purpose. If compared to other IoT-oriented protocols, HTTP may not be the best choice due to its protocol overheads and communication requirements. However, it has served as a strong base for newly developed protocols such as CoAP.

2.2.2 Constrained Application Protocol (CoAP)

The Constrained Application Protocol (CoAP) is defined as a *specialized web transfer protocol for use with constrained nodes and constrained networks in the Internet of Things* [9]. As it can be extracted from the definition, this protocol is specifically tailored for the IoT and M2M applications. The major standardization of this protocol has been carried out by the IETF Constrained RESTful environments (CoRe) Working Group and the core is specified in [10]. This application layer protocol can be seen as an enhancement of HTTP for low power devices. It is based on the successful REST model, in which resources are available under a URL and clients can access those resources using the GET, PUT, POST and DELETE methods. Additionally, CoAP also supports publish-subscribe thanks to the usage of an extended GET method.

Even though it shares similarities with HTTP, CoAP is specifically designed to run over UDP only. As UDP is inherently not reliable, CoAP defines two types of messages, namely *confirmable messages* and *non-confirmable messages* to define its own reliability mechanism. The former requires an acknowledgment similar to the ACK used in TCP communications whilst the latter does not require any kind of acknowledgment.

2.2.3 Message Queue Telemetry Transport (MQTT)

Message Queue Telemetry Transport (MQTT) is a client-server publish-subscribe messaging transport protocol standardized under the ISO/IEC PRF 20922 [11].

It is lightweight, simple and very easy to implement. Its lightness characteristic make it ideal for environments in which communication capabilities are limited such as M2M or IoT scenarios. Unlike CoAP, MQTT has been designed to run over TCP/IP or other network protocols that provide ordered, lossless and bi-directional communication. In this regard, MQTT is similar to HTTP. However, the former has been designed to have less protocol overhead.

The reliability of messages in MQTT is taken care by three Quality of Service (QoS) levels:

- At most once: messages are delivered in a best effort manner and messages loss can occur. This QoS is tailored for scenarios in which the loss of a message is not relevant.
- At least once: messages are assured to arrive, but duplicates can occur.
- Exactly once: messages arrive exactly one time, without duplicates. This QoS is reserved for systems that must operate reliably all the time, such as banking systems.

Thanks to the publish/subscribe model, it also allows for one-to-many message distribution with application decoupling.

	HTTP	**CoAP**	**MQTT**
Main Transport Protocol	TCP	UDP	TCP
RESTful	✓	✓	✗
Publish/Subscribe	✗	✓	✓
Request/Response	✓	✓	✗
QoS	✗	✓	✓

Table 3: Comparison between the main enabling application IoT protocols.

3 Evolution of IoT architectures

The rapid proliferation of gadgets and solutions for the everyday problems and situations is creating chaos inside the Internet of Things. The lack of agreement between companies for the usage of architectural and communication standards is making it impossible to create heterogeneous systems in which devices from different manufacturers interchange information smoothly. This is known as the vertical silos problem, in which every manufacturer creates his own close and private solution ranging from the sensors up to the end user application. This verticality does not allow the interoperability between company solutions, disabling the possibility to intercommunicate gadgets for different information sources inside the same scenario. Following sections present an overview of the evolution of IoT architectures since the appearance of such term up until the current ones making emphasis on how they coped with the stated *vertical silos* problem.

3.1 Initial models

The lack of architectural standards and protocols during the initial stages of the IoT development hampered the creation of systems with the current minimum requirements such as scalability, interoperability, security and reliability. During the first years, Intranet of Things was a more accurate term to define the situation. Devices were only provided with physical wireless communication protocols such as Bluetooth or ZigBee, with no possibility to transmit through the Internet. Moreover, the connection between those devices and the application was directly performed without any intermediate layer to decouple the system. Figure 1 shows an abstraction of such architecture.

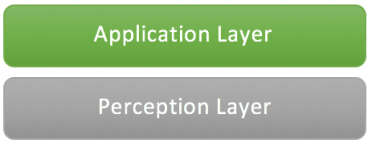

Figure 1: Initial IoT 2 layer architecture.

As it can be seen, this architecture can only be divided into two separate layers, namely perception and application layer. Although it was possible to use multiple devices inside the same system, they acted as individual elements, without communicating between them or helping each other during transmission. That is, a Network Layer combining them into a Wireless Sensor Network was not used. Instead, data being generated by the perception layer was directly sent to the application layer without any intermediate decoupling which made impossible the scalability or interoperability of such system.

Then, the architecture that can be seen as the birth of the Internet of Things appeared, which was comprised by three layers namely perception, network and application layer [12]. The network layer grouped all the sensors and actuators of the system forming a Wireless Sensor Network, in which devices were aware of each other. Additionally, gateways were added to gather and forward all the raw messages generated by the perception layer devices. Even though initial systems continued using only physical wireless communication protocols for its devices, the insertion of gateways as a more powerful intermediate element allowed for Internet communication between the network and the application layers.

The introduction of the network layer helped to slightly cope with the scalability problem by means of the placement of more gateways if necessary to handle all the device connections. However, this did not completely solved the problem. Regarding the interoperability and heterogeneity, the lack of standardization between companies for the usage of protocols and message structure hampered the combination of several devices into the same system.

At this point, researchers and manufacturers agreed in the necessity of having an abstraction layer to completely decouple the physical network from the

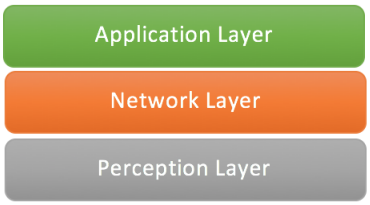

Figure 2: IoT 3-layered architecture.

application, in order to allow the creation of device-agnostic applications.

3.2 The appearance of a *Middleware*

Many efforts have been put to provide an abstraction and standardization layer from the WSN perspective. European Union projects such as SENSEI [13] and Internet of Things - Architecture (IoT-A) [14] have been addressing this problem by means of creating and defining the architecture for different applications. However, there is still a lack of agreement when it comes to overall architectural standards in regard to upper layers.

A middleware generally abstracts the complexities of the system and hardware allowing the application developer to fully focus all his efforts on the task to be solved without the distraction of concerns regarding system or hardware level. A middleware provides a software layer between physical and application layers. As it has been seen before, IoT interacts with many infrastructure and application technologies. Therefore, a middleware must provide almost full compatibility.

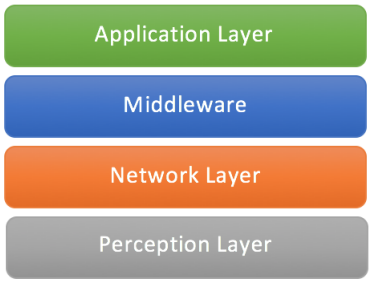

Figure 3: IoT 4-layered architecture.

Even though of the agreement in the necessity of a middleware as an abstraction layer, during the last years diverse solutions have appeared in terms of their design approach, such as event-based, database-oriented, application-specific or

service-oriented [15, 16]. However, the usage of a single design approach might not be sufficient. Instead, successful middlewares have been built upon the combination of multiple designs.

Since all the generated data must traverse the middleware for abstraction, the inclusion of the database as an element of such layer seems the right decision. This is why many current middleware solutions include a database-oriented design. However, the connectivity to the database may vary depending whether data is directly exposed to the end users or it is privately stored and instead, events or services are offered. The former case follows a mere database-oriented design, whilst the later is a combination of database and either event or service-oriented design.

Middlewares based on events with database storage are gaining popularity due to the easiness of deployment and lightness of resource utilization. Since individual sensor messages can be seen as events, the storage is straightforward. Regarding event communication, this type of middlewares usually use the publish/subscribe pattern, in which a set of subscribers acquire events from a set of publishers. Protocols such as MQTT (Section 2.2.3) or CoAP (Section 2.2.2) are designed to this aim.

Service-oriented middlewares are based on Service-Oriented Architectures (SOA) that has been traditionally used in corporate IT systems. Characteristics such as service reusability, composability or discoverability are also beneficial for IoT scenarios. However, large scale networks, constrained devices and mobility make this approach challenging.

With this approaches, applications connected to the middleware benefit from the abstraction and are agnostic to the underlying hardware.

3.3 Towards Intelligent IoT systems

Up to this point, IoT applications started to exploit the benefits of new and well designed architectures to solve daily problems or make lives easier. Moreover, many monitoring applications for different scenarios appeared such as health monitoring systems, building energy monitoring or city resource monitoring. However, the essence of such systems was merely informative.

New IoT or Future Internet is meant to go beyond that informative perspective. Instead, creation of intelligent and autonomous systems is what companies and researchers are aiming for. To this aim, new elements are added to the previous defined architecture.

Specifically, a new layer appears between the middleware and the application, commonly named knowledge-based layer, context awareness layer or cognitive layer. It is responsible for requesting data and extracting valid information for acquiring new knowledge and act upon it.

Depending on the purpose of the application, many techniques can be used such as rule based programming, machine learning or predictive analysis. Rule-based applications are meant to modify the status of the scenario if certain events occur. Usually, rules are static. However, the combination of rules with machine learning techniques offers a richer system in which rules are modified

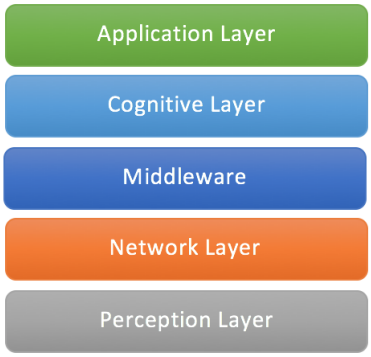

Figure 4: IoT 5-layered architecture.

depending on past actions. Predictive analysis is also being used to anticipate future actions and thus, increase comfort by adjusting the system to the desirable state beforehand.

4 Cloud-based IoT architecture presentation

This section presents an architecture developed to try to cope with some of the problems stated in the previous sections. It offers interoperability in regard to the type of sensors and protocols that can be used. Moreover, reliability and data persistence is achieved by means of a cloud middleware capable of replicating services on demand. The cloud also allows for data exposure and possibly, utilize it as a service for third parties.

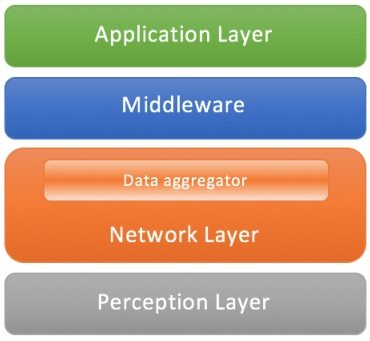

Figure 5: Cloud-based IoT architecture abstraction.

Figure 5 shows the architecture decomposed in the different layers. Starting

from the bottom, the perception layer includes all the sensors and actuators of the network. It is responsible for sensing the environment and also for executing the necessary actions that are received from the above layer.

Network layer comprises and groups the gateways of the platform. Since this devices are resource constrained in regard to the number of established connections, it is necessary to study the scenario under development in order to know how many of these devices need to be deployed. In terms of energy usage, gateways need a more powerful source than sensors. That is why these devices are usually placed inside buildings to maintain them fully operable. Moreover, received data might need to be uploaded to the Internet, which is another reason to locate them inside Internet-reachable buildings.

As previously mentioned, one of the main issues regarding IoT and WSN is the heterogeneity at the physical level. The lack of agreement for communication and message structure make it necessary to endow the system with the possibility to upgrade gateway software to allow compatibility with new devices.

Even though the south gate (i.e. the communication between sensors and gateways) of the gateways may be heterogeneous, the north gate (i.e. the communication between gateways and the above layer) maintains its homogeneity by exclusively using one communication protocol.

The data aggregator and processing layer, as its name says, is responsible for receiving every sensing message in raw format. These messages are then processed in order to modify their structure to a standard one. Since JSON is the standard de facto inside the Big Data world and is also used by the middleware, raw sensing data is transformed into JSON formatted files. This layer can be seen as a module of the network layer, and that is why it has been located inside it. This decision is explained later more in depth.

Regarding the middleware, it has been decided to use an external cloud platform called ServIoTicy [17] that allows for data upload, storage and retrieval using standard protocols such as HTTP and MQTT. Thanks to the usage of big data technologies, it also offers the possibility for server and database replication in case of necessity due to an increase in the number of connection requests.

Finally, the application layer contains the actual application. It is fed with standard data incoming from the middleware which allows the developer not to worry about the underlying hardware and communication protocols.

This architecture has been used for the development of a simulator for smart buildings that tries to reduce building energy consumption by avoiding unnecessary device states. For instance, switching off room lights when it is empty or adjusting room temperature depending on the environmental one and also on occupant desires.

Following sections explain more in detail the development of every layer and the communication between them.

4.1 Perception Layer

The perception layer is formed by all the sensors and actuators of the system. The main task of this layer is gathering data from the elements of the scenario

under monitoring. In the case of a Smart Building, sensors are usually placed to monitor environmental conditions such as temperature, humidity, luminosity, air quality, and also device state such as doors, windows, blinds, computers, etc. Moreover, actuators are deployed to allow state modification of those elements. For instance, if the system detects that a room has been left with lights on, it can send the signal to switch them off in order to avoid wasting energy unnecessarily. To achieve this feature, the communication between devices and above layer is bi-directional.

The system can also take advantage of the communication bi-directionality to interact with the sensors. Since sensors can sometimes be located in hard-reachable locations, this feature is needed to allow for software upgrade without having to manually access to them, commonly known as Over The Air (OTA) programming.

As it can be seen, there is a plethora of characteristics to monitor and actuate with, allowing for wide market opportunities for companies. The *vertical silos* problem previously stated starts in this layer. Companies usually specialize themselves in a single scenario or problem, without commonly agreeing standards in design or communication. However, when a more general system is developed such as a Smart Building, it is necessary to combine multiple sensors to fulfill all the requirements stated above. Therefore, it is needed a layer in which all this differences are solved by allowing the transmission and communication using different protocols.

4.2 Network Layer

The network layer groups and manages the gateways and it is responsible for creating a WSN between sensors, actuators and gateways. Due to the heterogeneity of the perception layer, gateways must be rich in terms of protocol compatibility. To this aim, they must be endowed with multiple interfaces depending on the type of sensors they are managing. Due to this necessity, their requirement in terms of energy usage is higher and the usage of batteries is not sufficient. Instead, they are deployed inside buildings in order to be powered with electric current.

Up to this point, the communication between the devices of the system is locally performed. However, once messages are received by the gateway, the connectivity can vary. Local systems can opt to maintain a private network with no Internet connection in which gateways locally connect to the above layer to standardize and store messages. Another approach could be to endow the gateways with Wi-Fi interfaces for directly upload the data to the Internet.

Similarly to sensor software programming, gateways can also be enhanced with new compatibilities if needed. However, this can be more time and money consuming if the compatibility also needs to place new physical interfaces in every deployed gateway.

4.3 Data aggregator Layer

The data aggregator layer can be seen as the standardization message layer. It is responsible for receiving raw messages from every gateway and transforming them into a valid message format. It has been decided to use JSON as the data standard because of the compatibility with the above layer and the friendliness that it offers with big data technologies.

This layer can be deployed into multiple places inside the architecture. Specifically, these are valid locations for it:

- Multiple instances distributed across the gateways.
- Central server with replicability.
- Middleware module.

Depending on the power of the gateways, this layer can be deployed in each of them in order to avoid the necessity of a central server gathering the data from every gateway to later transform and upload it. However, this decision has some drawbacks. Firstly, it requires that all the gateways of the platform are capable of connecting to the Internet in order to upload the data. Secondly, processing power and storage for these gateways would need to be higher. To conclude, in the case of needing to make a modification in the data aggregator to allow new data structures, it would be needed to completely flash all the gateways of the platform.

Another alternative is to develop a module for the middleware under usage in order to have a unique layer capable of standardizing the data and storing it. This is a good design approach but in our case it has not been followed in order to maintain the external middleware as it is.

The design approach finally followed has been to deploy this layer into a central server capable of creating multiple replicas if needed to cope with incoming connections. With this design, gateways do not need to have Internet connection and their processing power can be reduced to also reduce consumption, allowing a possible gateway deployment by means of battery power sources if strictly necessary.

4.4 Middleware

As previously defined in section 3.2, a middleware is an abstraction layer that hides the complexities of the system and hardware underneath.

Moreover, this middleware can have additional features depending on the requirements, which in our case are:

- Cloud storage with standard technologies.
- Big data oriented with high scalability.
- Standard communication protocols for data upload and download.

- Public and private virtual objects for data scope control and sharing.

After reviewing current available IoT platforms, ServIoTicy was finally chosen since it covers all the requirements stated above. ServIoTicy is an online platform developed by the Barcelona Supercomputer Center (BSC) [18] during the COMPOSE project [19]. It allows for fast and simple composition of IoT data streams, offering multi-tenant data architecture. As for its communication capabilities, both for data upload and download, it allows REST and publish/subscribe communication.

The transmitted content must be formatted into JSON data-objects. The extension and acceptance of this format as a standard allows to homogenize all the data independently of the transmitting platform, completely hiding the hardware from below.

4.5 Application Layer

The application layer, as its name indicates, contains the application responsible for interacting with the user or showing the desired information. Inside the IoT world, current developed applications are firstly focused on monitoring the environment and acting as an information panel in which the user can read, in real time, the values of the different sensors of the system, such as the inside temperature, power usage, outside luminosity, etc. There also exist interactive applications in which the user, apart from being able to see the sensor information, can also interact with the environment by sending actions to perform such as close the door, lower the inside temperature or switch elements ON or OFF.

One of the main advantages of the architecture presented is the freedom that the developer has when creating a specific application. By having a middleware with standard formats and transmission protocols, it allows him to fully focus their efforts into the use case without having to take into consideration hardware specifications.

The only coupling element between the middleware and the application is the message reception module. As it has been previously mentioned, messages can be requested via REST API or subscriptions thanks to the publish/subscribe protocol. The former allows for synchronous data requests, which can be necessary when a specific value needs to be obtained. However, the later is the standard widely used. The publish/subscribe protocol allows for asynchronous message reception by the application without the necessity to constantly query the middleware. Instead, when a new sensor message is stored inside the middleware, it is directly forwarded to the application by means of the subscription previously performed.

5 Use Case: Smart Building automation

This section presents a Building Management System (BMS) for Smart Building automation using the architecture explained in section 4. The main purpose of a

BMS is to increase people's comfort by maintaining the building in the desired state every time and also reduce energy consumption by avoiding situations in which elements are being overused. For instance, by predicting the time of entrance of a person in a room, it is possible to adjust the temperature beforehand, or by detecting that a room is empty, lights can be turned *off* if they have been left *on* by mistake.

First section explains how the data is generated and which elements are being monitored inside the building. Since the deployment of many sensors inside a building is costly, some of them are simulated in order to scale the system and create a more realistic scenario. Then, how this data is transformed into a standard format and later uploaded to the cloud platform is explained. After, the consumption of the data by means of the building management application is described. Finally, conclusions of the performance of the architecture and the benefits of this specific case are extracted.

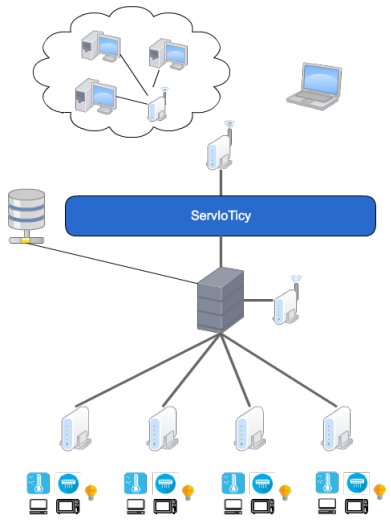

Figure 6: Cloud-based IoT architecture.

Figure 6 shows the elements composing the system divided into the different layers stated in figure 5

5.1 Data Generation

The first step towards the enhancement of a building with smart features is the monitoring of all the necessary elements inside it. In the case of a building, important elements are lights, HVAC systems, computers, doors and windows. Moreover, the environment also needs to be monitored to know whether we can take advantage of it. For instance, light can be turned *off* if outside luminosity is high enough for indoor working.

Some of the aforementioned elements are endowed with small sensors capable of acquiring the necessary data to deduce their state. In the case of lights, HVAC systems and computers, potentiometers are used to read the amount of energy being consumed and thus, know their state. Alternatively, doors and windows make use of electromagnetic sensors to know if they are opened or closed. In the case of environmental data, temperature, humidity and luminosity sensors allow us to exactly read the respective values. For mimicking the rest of the elements of the system, data is generated by means of a software capable of creating the exact same packets as the physical ones.

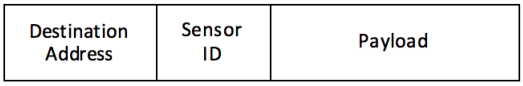

Figure 7: Data packet abstraction.

Even though the data generation may vary from one type of sensor to another, the packet containing the data and the corresponding headers for a transmission is equal. Figure 7 shows the abstraction of the structure of such packets. As it can be seen, it is merely formed by the destination address, the sensor identifier and the payload containing sensor readings.

Data generation rate varies depending on the device under monitoring. In the case of environmental conditions and power usage, samples are taken every 5 minutes. Regarding doors and windows, the sample rate remains the same but additionally, if a change in their state is detected, a message is also generated.

Once the sensor data is read and encapsulated in a packet with the shape seen in figure 7, it is transmitted to the closest gateway in each case. Gateways receive the messages by means of different protocols such as ZigBee and Bluetooth Low Energy. Since gateways are enhanced with Internet connection, raw messages are directly forwarded to a central server responsible for the data aggregation and standardization.

As it can be seen, the architecture is capable of combining the usage of real sensors with software defined sensors, which allows for easy scalability and also fast adaptation testing on other possible scenarios.

5.2 Data transformation and storage

When the messages reach the central server, it firstly reads the sensor identifier to know the type of message contained inside the payload. Once the type has been detected, the message is transformed into JSON standard format with a structure of *"type":"value"* for each value. Additionally, each message contains a time stamp to know when the value has been generated.

The standardized data is then pushed using the REST API to the cloud service explained in section 4.4. In this cloud platform, each physical sensor corresponds to a virtual sensor. That means, each physical identifier is assigned

to a virtual identifier. This relationship is privately stored inside the central server in order to be able to correctly push the messages to the corresponding virtual sensor. However, virtual identifiers along with sensor information such as model, type of sensor and location is publicly available thanks to an additional cloud database that stores this data. By doing so, external entities can take advantage of the platform and query specific sensors without the necessity of deploying their own ones.

As it has been mentioned before in section 4.4, not all the sensors registered into the system are publicly available. The necessity to privatize some sensors is directly related to the security and privacy of the users under the monitored environment. For instance, if proximity and movement sensors are publicly available, third persons would be able to know whether the room containing the sensors is empty or not, and take advantage of such information for social hacking.

In order to avoid this, the only sensors that are shared correspond to environmental monitoring such as temperature, humidity and light. For the rest of the sensors, a password is needed to receive the updated values.

5.3 Data consumption

The application developed is composed by two differentiated elements. Firstly, the BMS is responsible for directly receiving sensor information via the subscriptions performed to the different sensors inside the building. By using a lightweight publish/subscribe client, once a new message is stored in the middleware database, it is also forwarded to the application, allowing the BMS to act accordingly if necessary. However, as it has been previously said, since the deployment and testing of such scenario in a real environment is too costly, simulation has been chosen as the second element for acquiring close-to-real results of the benefits of the building enhanced with smart capabilities.

The behavior of the whole system is as follows. The architecture developed feeds the BMS with both real and software generated sensor data. Once this data reaches the application, simulated elements modify their state in order to be synchronized with the corresponding sensor. For instance, if a sensor from a specific location tells that the lights are *off*, the simulated element must also be *off*. By using this pattern, the simulator maintains the synchronism between the real and virtual sensors with the building simulated elements. Consequently, if the simulator detects that an actuation must be performed, it automatically changes the state of the element and updates all the required software defined sensors in order to maintain the synchronism.

In addition to the simulation of the building elements, people inside it are also simulated to be able to repeat the tests multiple times. People are defined by a set of actions that can be performed inside the building along with the probability over time of this actions to actually be performed. For instance, if the building under simulation corresponds to an office, people are more susceptible to perform the action *enter* during the initial morning hours. The combination of such definitions is stored as the *profile* of the user, allowing different user

profiles across the simulated people.

Last feature of the simulator corresponds to the smart capabilities of the BMS. That is, the system must be able to detect whether an action that increases comfort and possibly reduces energy consumption can be carried on by looking at the state of the building at each moment. The implementation is developed by means of a rule-based system that monitors conditions corresponding to every possible actuation to activate. For instance, to know whether the light of a room can be switched *on* or *off* directly depends on the presence of people inside the room, the current indoor light state, outdoor luminosity and windows position.

By comparing the results of a building structure enhanced of the smart features previously explained it is possible to increase the comfort of the people. Even though there exists no metric capable of quantifying the comfort of a person, it is possible to deduce that by entering in a room with the desired temperature or by not having to interact with elements such as lights or HVACs, his comfort is increased. Moreover, energy consumption of the building can be reduced thanks to the activation of rules under wasteful scenarios.

5.4 Conclusions

The potential benefits of the designed architecture after analyzing how it behaves under a close-to-real simulated scenario are:

- Hardware abstraction at the lowest possible level. Network layer gateways completely hide the protocols underneath.
- Cloud middleware allows for rapid database and server replication under demand.
- Homogenization both in terms of file format and communication protocols.
- Possibility for sensor data sharing thanks to the public database and flexibility in the visibility of sensors.

Regarding the scalability of the system, the simulations show that under a standard office building, the number of sensors is no issue and does not affect on the performance of the system. Reliability is also another important factor to take into account. Even though the cloud middleware can be seen as a central attack point that could hamper the functioning of the system, the replicability offered substantially increases the reliability.

The testing of the platform by means of simulations shows that the platform is viable for a physical deployment. Moreover, the level of abstraction eases the development of the application by allowing the developers to fully focus their efforts in it, without worrying about heterogeneity of protocols and formats.

The future of IoT will be directly related to the design of architectures aiming for a homogeneous Internet and the usage of abstraction middleware layers capable of interconnecting many solutions for the creation of more sophisticated

systems. Additionally, the placement of such middlewares inside the Internet is gaining popularity due to the possibility to access data from everywhere and the integration of the *things* with the *Internet*, enabling the creating of multi-located systems.

References

[1] LitePoint. *Bluetooth Low Energy Whitepaper*, 2012. Rev. 1.

[2] R. Frank, W. Bronzi, G. Castignani, and T. Engel. Bluetooth low energy: An alternative technology for vanet applications. In *Wireless On-demand Network Systems and Services (WONS), 2014 11th Annual Conference on*, pages 104–107, April 2014.

[3] M. Siekkinen, M. Hiienkari, J. K. Nurminen, and J. Nieminen. How low energy is bluetooth low energy? comparative measurements with zigbee/802.15.4. In *Wireless Communications and Networking Conference Workshops (WCNCW), 2012 IEEE*, pages 232–237, April 2012.

[4] Wi-fi alliance introduces low power, long range wi-fi halow. `https://www.wi-fi.org/news-events/newsroom/wi-fi-alliance-introduces-low-power-long-range-wi-fi-halow`.

[5] N. Sornin, M. Luis, T. Eirich, T. Kramp, and O. Hersent. *LoRaWAN Specification*. LoRa Alliance, 2015. Rev. 1.

[6] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 7540, Network Working Group, January 1997. `https://tools.ietf.org/html/rfc2068`.

[7] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 7540, Network Working Group, June 1999. `https://tools.ietf.org/html/rfc2616`.

[8] M. Belshe, R. Peon, and M. Thompson. Hypertext Transfer Protocol Version 2 (HTTP/2). RFC 7540, Internet Engineering Task Force (IETF), May 2015. `https://tools.ietf.org/html/rfc7540`.

[9] CoAP RFC 7252 Constrained Application Protocol. `http://coap.technology`.

[10] Z. Shelby, K. Hartke, and C. Bormann. The Constrained Application Protocol (CoAP). RFC 7252, Internet Engineering Task Force (IETF), June 2014. `https://tools.ietf.org/html/rfc7252`.

[11] Information technology: Message Queuing Telemetry Transport (MQTT) v3.1.1. `http://www.iso.org/iso/catalogue_detail.htm?csnumber=69466`.

[12] Miao Wu, Ting Jie Lu, Fei Yang Ling, Jing Sun, and Hui Ying Du. Research on the architecture of Internet of Things. *ICACTE 2010 - 2010 3rd International Conference on Advanced Computer Theory and Engineering, Proceedings*, 5:484–487, 2010.

[13] Vlasios Tsiatsis, Alexander Gluhak, Tim Bauge, Frederic Montagut, Jesus Bernat, Martin Bauer, Claudia Villalonga, Payam Barnaghip, and Srdjan Krco. The SENSEI real world internet architecture. *Towards the Future Internet: Emerging Trends from European Research*, pages 247–256, 2010.

[14] Martin Bauer, Mathieu Boussard, Nicola Bui, and Francois Carrez. Project Deliverable D1.5 Final Architectural Reference Model for IoT. pages 53–59, 2013. `http://www.iot-a.eu`.

[15] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys and Tutorials*, 17(4):2347–2376, 2015.

[16] Luka Milić and Leonardo Jelenković. A novel versatile architecture for Internet of Things. *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2015 - Proceedings*, pages 1026–1031, 2015.

[17] ServIoTicy: IoT streaming made easy. `http://www.servioticy.com`.

[18] Barcelona Supercomputing Center. `http://www.bsc.es`.

[19] COMPOSE: Collaborative Open Market to Place Objects at your Service. `http://www.compose-project.eu`.