Universidade de Aveiro
2013

Departamento de Eletrónica,
Telecomunicações e Informática

Nuno José
Coelho Almeida

Gateway de Ethernet-ZigBee

Ethernet-ZigBee gateway

Universidade de Aveiro
2013

Departamento de Eletrónica,
Telecomunicações e Informática

Nuno José
Coelho Almeida

Gateway de Ethernet-ZigBee

Ethernet-ZigBee gateway

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Eletrónica e Telecomunicações, realizada sob a orientação científica do Doutor Pedro Alexandre de Sousa Gonçalves, Professor adjunto na Escola Superior de Tecnologia e Gestão de Águeda, e do Doutor João Paulo Silva Barraca, Professor assistente convidado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

Dedico este trabalho aos meus pais, José e Filomena, por todo o esforço que tiveram para que este se pudesse realizar; ao meu irmão, Rodrigo, pelo encorajamento; à Anita pela presença e incansável apoio.

o júri / the jury

presidente / president

Prof. Doutor Paulo Bacelar Reis Pedreiras
professor auxiliar da Universidade de Aveiro

vogais / examiners committee

Prof. Doutor Joel Perdiz Arrais
professor adjunto convidado da Universidade de Coimbra

Prof. Doutor Pedro Alexandre de Sousa Gonçalves
professor adjunto da Escola Superior de Tecnologia e Gestão de Águeda (orientador)

Prof. Doutor João Paulo Silva Barraca
professor assistente convidado da Universidade de Aveiro (co-orientador)

agradecimentos / acknowledgements

Agradeço aos meus orientadores, os professores Pedro Gonçalves e João Paulo Barraca, pelo apoio, orientação e partilha de conhecimento, essenciais durante todo o período de execução da presente dissertação.

Ao grupo ATNoG (Advanced Telecommunications and Networks Group) bem como ao Instituto de Telecomunicações - pólo de Aveiro - por me terem fornecido as condições necessárias à elaboração desta dissertação.

À Bosch, pela oportunidade de desenvolvimento deste trabalho, em especial ao Eng. Nélson Ferreira pela constante disponibilidade e suporte.

Ao professor Rui L. Aguiar pela orientação aquando do processo de escolha do tema para esta dissertação, bem como pelo convite para integrar o grupo ATNoG.

Aos meus amigos e todas as outras pessoas com quem tive o prazer de me cruzar ao longo dos últimos cinco anos que contribuiram não só para esta dissertação mas também para a minha formação enquanto pessoa, das quais destaco o Rafael Gomes.

Por fim, agradeço à minha família que em todas as circustâncias me apoiou e incentivou.

Palavras Chave Machine-to-Machine, gateway, ZigBee, REST

Resumo

As telecomunicações estão há muito presentes no nosso dia-a-dia. O surgimento da Internet revolucionou o modo de comunicar, permitindo o estabelecimento de uma rede à escala global. Atualmente, caminhamos para um estado onde esta deixa de ser exclusivamente a "Internet das pessoas" para passar a ser a "Internet das coisas" ou mesmo a "Internet de tudo". O enorme aumento de dispositivos "inteligentes", ou seja, com capacidades de processamento, de comunicação, recolha de informação e memória, tem sido a principal causa desta mudança de paradigma. Os dispositivos referidos comunicam segundo o conceito denominado Machine-to-Machine; este permite a comunicação end-to-end entre dispositivos sem (ou com limitada) intervenção humana. A domótica é uma área em crescimento que faz uso dos conceitos descritos. Esta permite a utilização de dispositivos para automatizar as rotinas e tarefas de uma habitação. Um cenário habitual permite o controlo do sistema de iluminação, temperatura e som através de um dispositivo central que poderá estar conetado à Internet, permitindo assim o acesso e controlo remoto de todo o sistema. O dispositivo que permite a conexão com outras redes (por exemplo a Internet), denomina-se gateway. O presente trabalho propõe a implementação de uma gateway de Ethernet-ZigBee que permita a monitorização e controlo remoto de um sistema AVAC (Aquecimento, Ventilação e Ar Condicionado) através do Web browser.

Keywords

Machine-to-Machine, gateway, ZigBee, REST

Abstract

Telecommunications have been present in our daily lives for a long time. The emergence of the Internet has revolutionized the way of communicating, allowing the establishment of a global scale network. The state of the Internet has been evolving from the "Internet of people" to become the "Internet of things" or the "Internet of everything". The huge increase of "smart" devices, i.e. with processing and communication capabilities, memory and the ability of gathering information, has been the main cause of this paradigm shift. The referred devices communicate according to the Machine-to-Machine concept, which allows an end-to-end communication between them, without (or with limited) human intervention. Home automation has been growing and it uses the described concepts; it allows the use of devices in order to automatize home-related tasks. A usual scenario allows the control of both lighting, temperature and sound systems through a central device that can be connected to the Internet, allowing the remote access and control of the entire system. The gateway is the device that allows a connection with other networks (e.g. Internet). The present document proposes an implementation of an Ethernet-ZigBee gateway that enables the remote monitoring and control of an HVAC (Heating, Ventilation and Air Conditioning) system through the Web browser.

Contents

List of Figures

List of Tables

1 Introduction

This chapter provides a brief description of the developed system, as well as the motivations that led to the writing of this dissertation. Also, it introduces the topics that were further addressed, and it describes the entire document structure.

1.1 Motivation

The Internet has initially started as a global network, enabling services that allowed computers, and hence users, to communicate with each other and exchange information. The widespread use of the Internet and the increasing use of the Ethernet as a technology for IP networks convergence, caused a price reduction and the development of software and know-how.

In recent years, the processing power devices, and storage capacity, have been increasing, and network technologies have been evolving, allowing the devices to become more powerful, smaller and cheaper. This evolution allows the creation of environments where the devices can sense, compute, act and communicate, becoming part of the so-called Internet of Things (IoT) [1].

It is expected that the IoT will have a huge impact in the Future Internet, which is expected to become a global network that, besides connecting computational systems directly or indirectly used by humans, also allows sensory devices to communicate with the "physical world", as illustrated in Figure 1.1. Such devices are usually powered by batteries or renewable energy sources to ensure their use in remote places, or without electrical power. The simplicity

[1] The Internet of Things can be defined as a concept that describes a future where everyday physical objects will be connected to the Internet and will be able to identify themselves to other devices.

of these devices is a fundamental characteristic in order to reduce their prices, making them energetically more efficient, hence allowing their use in a large scale.

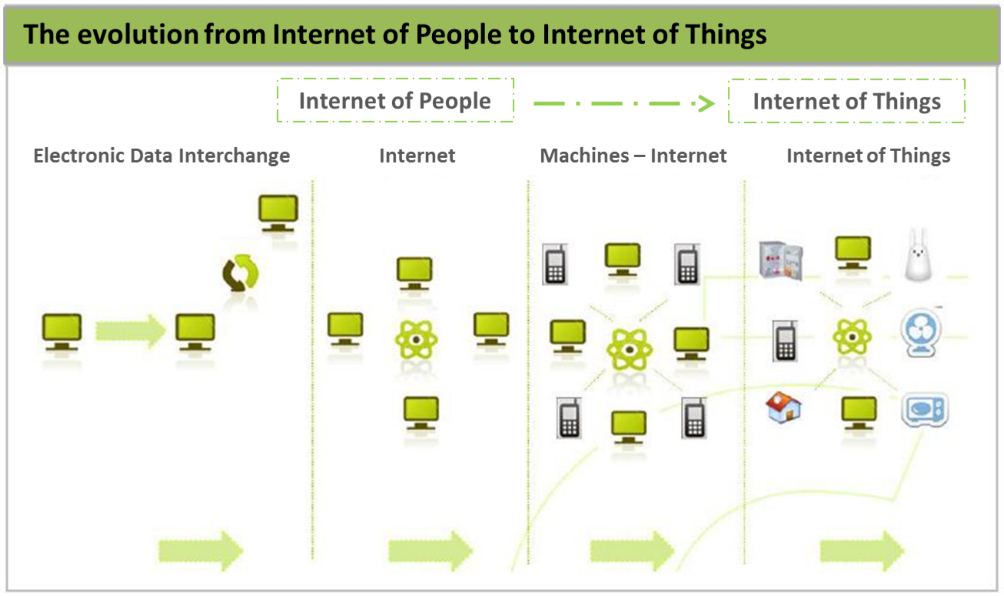

The evolution from Internet of People to Internet of Things
Internet of People
Internet of Things
Electronic Data Interchange
Internet
Machines – Internet
Internet of Things

Figure 1.1: The Internet evolution [1]

Since the IoT will mainly consist of machines talking between them while humans observe, analyze and act based on the received information, the Machine-to-Machine (M2M) paradigm will play an important role. M2M communications are responsible for establishing a connection between two devices, or a device and an application, allowing the exchange of information between them, with minimal human intervention. This communication model, besides increasing the number of connected devices, will result in a "data explosion" that will require massive and scalable, storage and processing capacity.

The described concepts have been contributed to the growth of markets such as home automation and control, energy management, security and surveillance, healthcare, etc. In order to develop solutions to be used within these environments, devices capable of processing data, such as the Arduino [2] or the Raspberry Pi [3] have played an important role, mainly due to their small size, memory capacity, low-cost and low-power consumption. These devices can also be easily connected to a large number of sensory and communication modules. The majority of environments where they are used, have several constraints that limit, for instance, their maintenance or removal; industrial processes or embedded systems are examples of applications where devices may be located outdoor or in hostile locations. Network technologies such as ZigBee [4], 6LoWPAN [5], Z-Wave [6], Bluetooth Low Energy [7], etc. have also been important in a way that they allow the establishment of low-rate and low-power networks within similar environments; these technologies enable the implementation of small, cheap and power-efficient solutions, that require low-maintenance.

The combined technologies and concepts have been contributing to the growth of "human-independent" applications, and consequently, to the explosion of the number of connected devices. Looking to the future, Cisco IBSG [8] predicts that there will be 25 billion devices connected to the Internet by 2015 and 50 billion by 2020 (study published in April 2011), as it can be seen in Figure 1.2. Although these numbers are not consensual between all entities, they all agree that sooner or later they will be achieved. This reflects the importance of the concepts addressed in the Future Internet, as well as the need of network technologies that allow the connection between all devices and the Internet.

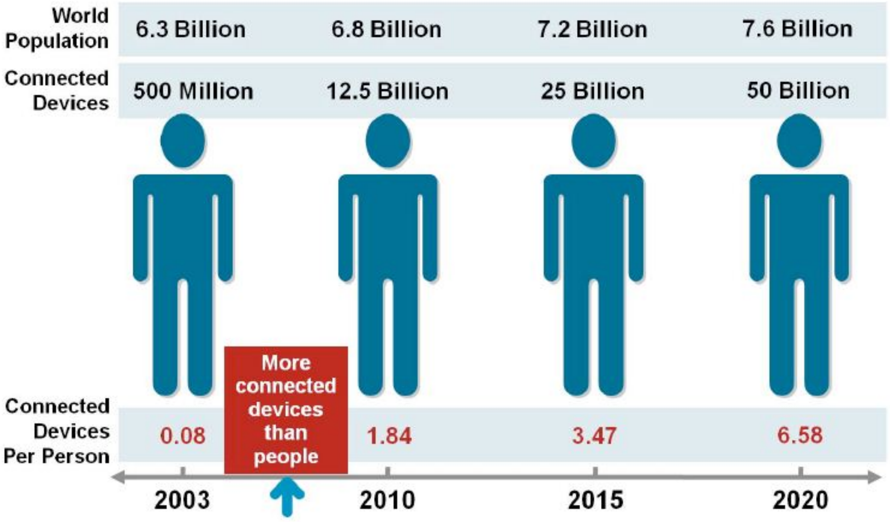


Figure 1.2: Number of connected devices per person [8]

1.2 Use case definition

The focus of the present dissertation is the proposal and construction of a service gateway, to be used within a residential environment, that enables users to remotely access information about the system they want to monitor.

In home automation scenarios, a gateway is responsible for the connection of home devices with other networks; it provides an interface to devices and it makes the system related data available to remote applications. Thus, this dissertation intends to propose the implementation of an Ethernet-ZigBee gateway that allows a direct remote communication between an Intelligent Electronic Device (IED) and a host.

The gateway is intended to monitor an Heating, Ventilation and Air Conditioning (HVAC) system, and it should enable an IED, such as an heat pump, which has an embedded ZigBee radio, to communicate with a host, such as a PC or a smartphone. This connection will allow a directly remote access to the IED, allowing the configuration of certain settings or even something simpler, such as reading its current status (always assuming that the equipments are installed at remote locations).

The system is controlled via the Web browser through a Representational State Transfer (REST) Web service interface (Subsection 2.2.1), and as an example, it can be monitored by a Munin service (Subsection 5.1.2). Figure 1.3 illustrates the overall architecture of the system.

Some general features of the gateway:

- Enable remote IP access, even with non-fixed IP addresses;
- Be based on an open platform;
- Enable the communication with more than one ZigBee device;
- Provide a control port to configure some network settings or devices parameters;
- Have an average power consumption equal to or less than 5W.

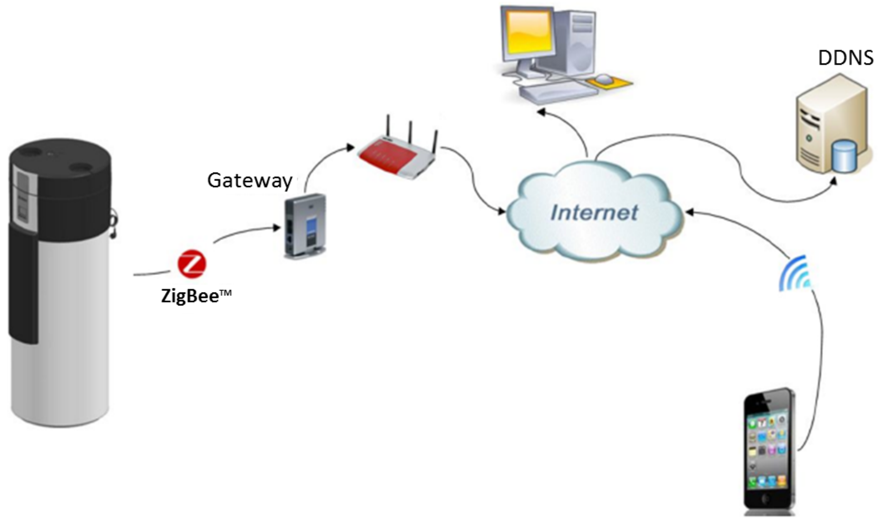


Figure 1.3: Schematic representation of the system

1.3 Document structure

This dissertation is structured in six chapters that are organized as follows:

The present chapter provides an introductory description of the adopted scenario, as well as the motivations that led to the development of this work.

The second chapter addresses some concepts behind Machine-to-Machine communications. It provides a brief analysis of related standards and it presents some usual M2M scenarios. An introduction of Web services and a comparison between REST and Simple Object Access Protocol (SOAP) [9] concepts is made.

The third chapter describes some communication protocols that can be used in sensor networks scenarios. It analyzes booth ZigBee and Bluetooth protocols and it establishes a comparison between them and the emergent Bluetooth Low Energy.

In the fourth chapter, it is presented a global description of the system and its components. Some technical details regarding the implementation of the developed prototype are also described.

The fifth chapter describes some more technical details regarding the prototype implementation and its behavior and performance when faced with different requests. It also presents some results and it makes an evaluation of the developed prototype's proper operation.

The sixth chapter presents the main conclusions regarding the work developed in this dissertation, as well as some issues still requiring further work.

2

Home automation scenarios

The increasing number of "intelligent" devices capable of communicating between them, have contributed to the growth of market segments mostly related to remote control, monitoring, and automation. The idea of reducing the need for human intervention boosted the development and growth of scenarios within those market segments. Such scenarios may have distinct requirements (*e.g.,* bandwidth, security issues, implementation and maintenance costs, energy constraints, etc.) and consequently, they may involve different technologies in order to implement solutions that fulfill those requirements.

Home automation (also known as domotics) represent a market segment that has been growing in the last years. Home Area Networks (HANs) are now becoming increasingly popular and have been evolving, with the including of smart metering devices such as valves, electricity and gas meters, etc. They have the potential to control consumers devices used in everyday life. This huge variety of possible applications makes imperative the identification of the users requirements; usually, they fall into the following groups: efficient automation of routine tasks; security of automation systems; ease of use; local and remote access; telemonitoring; or system cost and flexibility [10].

The field of home automation is expanding rapidly as electronic technologies converge. The growth of Low-Rate Wireless Personal Area Networks (LR-WPANs) technologies (see Chapter 3) have also played an important role in the development of these scenarios. Communication protocols such as Bluetooth Low Energy, ZigBee and other IEEE 802.15.4 [11] related protocols, provide the establishment of a low-rate network for relatively short distances, that allows the connection between devices, using little or even no infrastructure. These networks are ideal to the implementation of small, power-efficient, low-cost solutions.

Besides the support for basic devices communications, home networks may have to deal with different traffic requirements, since general home applications may include the monitoring and control of several systems, as shown in Figure 2.1.

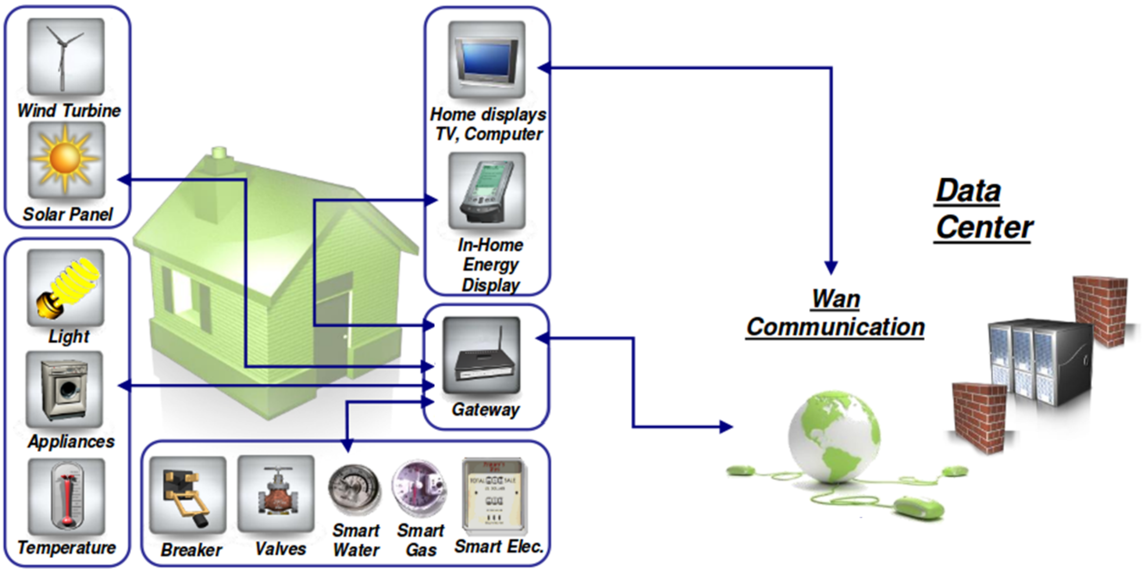


Figure 2.1: Typical smart metering configuration [12]

In this scenario, a gateway provides an interface to home automation devices such as sensors, displays, and appliances, and it makes the collected data available to remote applications. According to the user requirements, a controller, sensors and actuators may be coupled to home devices in order to monitor and to control them. Tasks such as turning off the lights, and activating the alarm system whenever no one is at home, are possible using this concepts; sensors will detect that everyone is out of the house, and they will communicate with a controller that will act on both the lighting and the alarm systems to perform the requested tasks. Also, the user may always remotely preform these or other control actions, or even monitor the overall system, as well as be informed if an alarm situation has occurred.

In addition to other systems, this scenario allows a user to control his domestic HVAC system, which is, in most cases, responsible for much of the total energy consumed. An HVAC system has the tasks of controlling the temperature processes within the residential environment; these systems have the goal to provide thermal comfort and acceptable indoor air quality. A typical residential HVAC system is presented in Figure 2.2.

HVAC systems are composed by several device types, as shown in Figure 2.2. Devices marked with numbers from 1 to 4, are responsible for home heating; devices 5 and 6 are responsible home cooling; devices with numbers from 7 to 11 improve the indoor air quality; devices 12 and 13 are hot water heaters; and finally, the thermostat is the controller device of the overall system. In the context of home automation, all enumerated devices have embedded sensors that gather information about the processes where they are involved (*e.g.*, the current temperature inside the house, the water temperature inside and outside the tank, etc.), and actuators that allow the device to respond properly to the users demands or when faced

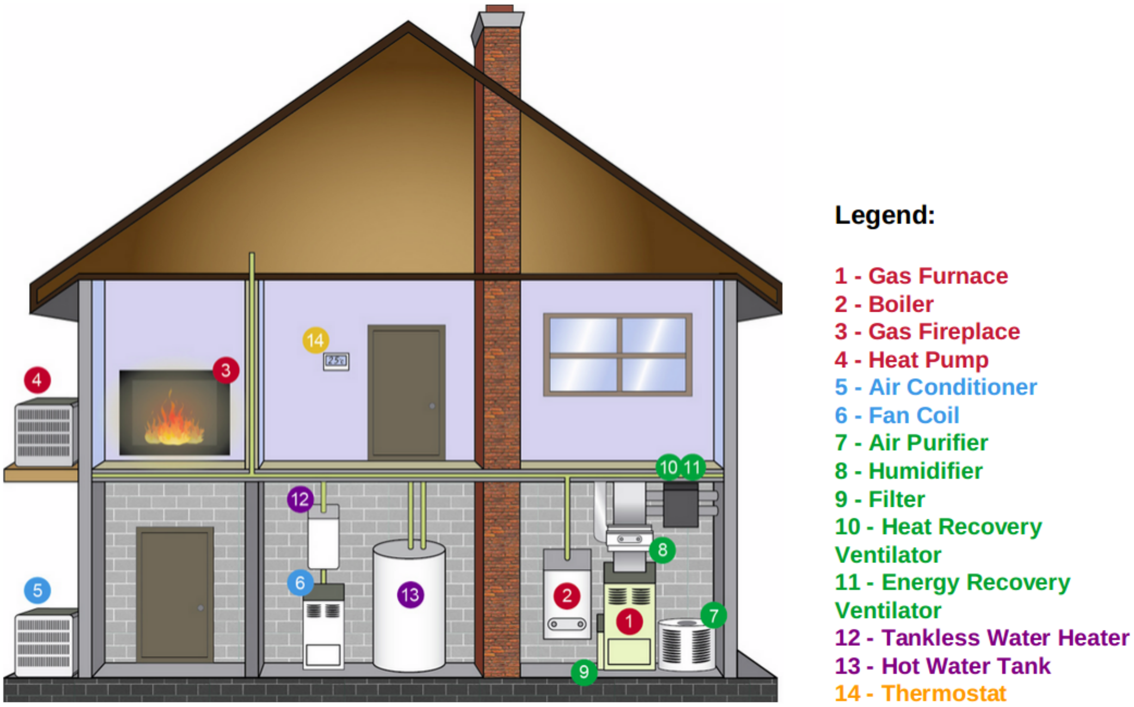


Figure 2.2: Typical residential HVAC system [13]

with alarm situations. The information gathered can be either transmitted to a centralized controller or to other devices in the system, enabling them to act according to the received information. A gateway can also be part of the system in order to allow the connection with external networks; it will enable users to remotely monitor and control the system. Within a domestic environment, the proper management of HVAC systems may increase the thermal comfort, reduce the energy consumption, and therefore reduce costs.

In the same way that a user can control its HVAC system, he is able to remotely perform the same actions in almost every system within his home, in a commode, quick and easy way. Such scenario may lead to the enhancement of energy-efficiency, as well as the increasing of the users comfort and buildings safety.

2.1 Machine-to-Machine

As already discussed, we are now witnessing the growth of a whole new communication paradigm, which is not only about connecting people between each other, but also connecting machines in order to provide extremely useful services to users. This communication paradigm is used to enable devices to communicate within the scenarios described above; it is called a Machine-to-Machine (M2M) communication and it means that no human intervention (or only limited) is needed whilst devices are communicating end-to-end.

This concept allows a wide variety of machines to become Personal Area Network (PAN) nodes, allowing the development of monitoring and remote control applications, that can operate over the machine itself or on the surrounding environment. These solutions will decrease the involved human resources costs and will make machines more intelligent and autonomous.

A usual M2M scenario uses a device (sensor, meter, etc.) to capture an "event" (*e.g.,* temperature exceeded a threshold value), which is relayed through a network (wireless, wired or hybrid) to an application (software program), that translates the captured event into meaningful information (*e.g.,* heating system must be turned off). Figure 2.3 shows a schematic representation of an M2M scenario.

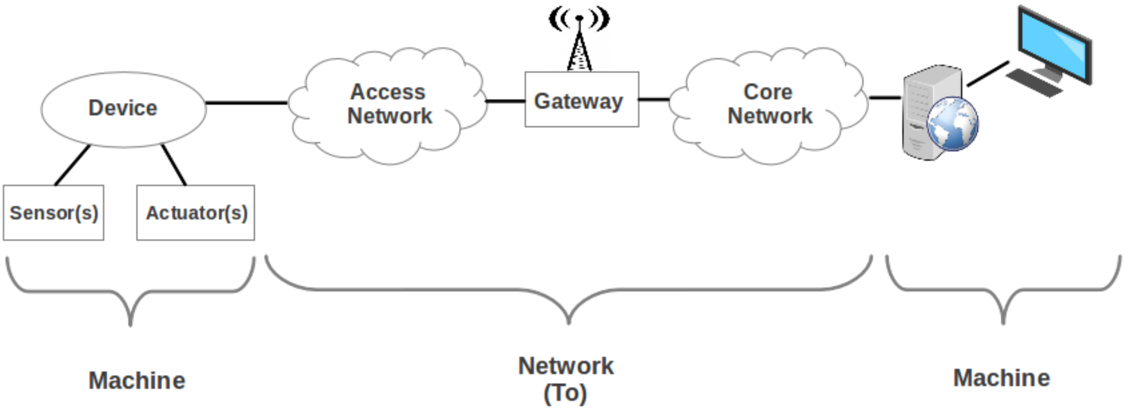


Figure 2.3: Example of an M2M scenario

The unique characteristic of M2M communications is largely due to the key role of the end-devices. Since this communication paradigm is intended to be used by devices to communicate end-to-end with no human interaction, these are responsible for every aspect regarding the information transaction between them, and they also must be as independent as possible when faced with unfamiliar situations. For instance, this concept may involve a huge number of devices with completely different characteristics, which can be used in distinct situations and environments. Distinct scenarios impose different applications and network requirements to which both the devices and networks have to adapt to. This way, multitude was probably the most significant change brought by M2M. Nowadays, the number of devices connected in M2M relationships are increasing and they will largely exceed the sum of all those that directly interact with humans, such as mobile phones, PCs, tablets, etc.

The concept behind M2M applications have already led to the emergence of a large variety of devices with extremely diverse requirements, in order to be able to adapt themselves, and to operate in distinct environments. The requirements imposed by some scenarios may represent a problem, since in some cases M2M devices may have to operate in environments with such stringent requirements (*e.g.,* latency, reliability), that they may exceed the capabilities of today's networks. Devices used in the health field, or in the monitoring of life-critical infrastructures, are examples of devices that might fit in the description. Besides fulfilling the environment requirements, these applications also need to adapt to a variety of contexts and business models.

As already discussed, the main characteristic of M2M communications is the ability of devices to routinely deliver their service with no, or very little, human control. To do so, the devices must take decisions and actions when faced with unfamiliar situations. For instance, when a critical problem appears in the system, the devices must be smart enough to take quick actions in order to fix or minimize its impact in the system; they must also be responsible to send alert messages to the user or system manager, with the current system status. This way, if needed, the user can take the proper action, in some cases avoiding the system degradation.

As many M2M devices are connected to batteries, either because they are located outdoors, in remote or hostile locations (*e.g.,* industrial processes and embedded systems) or because of their large number, they must be energetically efficient, thereby reducing the need for human intervention. In order to reduce their cost and extend their battery life, the devices computational capabilities are orders of magnitude below, when compared with a modern PC or smartphone. This may not be a problem in the majority of situations since, in general, most devices are not meant to be talkative or to perform computational intensive tasks.

The currently main problems behind M2M are related to security issues and the lack of globally accepted standards. Due to their inherent un-guarded, low cost and mass-deployed nature, M2M solutions would invite new threats in security. For instance, since their components spend most of the time unattended, it becomes easy to physically attack them. Issues regarding the users' privacy may also occur; for instance, the service providers' needs to better adapt the system to the end-users' requirements, may collide with their privacy rights. Finally, despite the growth of this market and the M2M systems evolution, standards for M2M are still lacking, and several of these market segments demand strong standards to ensure a long-term investment protection. Some work has been done for M2M standardization and European Telecommunications Standards Institute (ETSI) has been played an important role (Subsection 2.1.1).

With the evolution and consolidation of M2M communications in the market, a vast quantity of use cases were thought and developed. The variety of applications and technologies, as well as the lack of mature standards, make the business models of M2M highly complex. M2M plays a significant role in markets such as healthcare, energy, industrial and home automation and control, security and surveillance, transportation, and building control/management.

2.1.1 ETSI M2M standards

The European Telecommunications Standards Institute (ETSI) [14] is an independent, non-profit, standardization organization for Information and Communication Technologies (ICT). In order to provide an end-to-end view of M2M standardization, it was created in January 2009 the ETSI M2M Technical Committee (TC). Since then, it has become one of the main references to the development of global standards in the M2M field.

This TC made the decision of developing its work based in a set of use case families: smart metering, eHealth, connected consumer, automotive and city automation. Their goal is to cover enough use cases to ensure that all important requirements are captured, so that the proposed architecture may provide the foundation for a potentially large number of M2M applications.

ETSI M2M TC has adopted the high-level system architecture presented in Figure 2.4. This high-level system view provides an end-to-end representation of an M2M system, and it includes the concept of domains; the presented system architecture is divided in two domains: the M2M device domain (also referred to as gateway domain) and the network and applications domain. The first one is composed by the following elements [15]:

- **M2M Area Network:** refers to any network technology providing connectivity between M2M devices connected to the same M2M Area Network, or between them and M2M gateways;
- **M2M gateway:** acts as a proxy between M2M devices and the Network Domain. It may also run M2M applications;
- **M2M device:** a device that runs M2M applications using M2M Service Capabilities (SCs)[1]. M2M devices can connect to the M2M Network Domain in one of two ways: "Direct Connectivity", where they are connected via the access network, and "Gateway as a network proxy", where the connection is accomplished via an M2M Gateway. In the second method, the devices connect to the M2M Gateway using the M2M Area Network.

The network and applications domain is responsible for the transport network management, as well as for providing users interfaces to the M2M applications. It is composed by the following elements (see Figure 2.4): the access and core networks, M2M service capabilities, M2M applications, and both network and M2M management functions. The access network allows the M2M device and the gateway domain to communicate with the core network, which provides functions relating to network control, service, roaming and interconnection with other networks. Network management functions provide all the functions required to

[1] Service Capabilities (SCs) provide M2M functions that are intended to be shared by applications and exposed through a set of open interfaces.

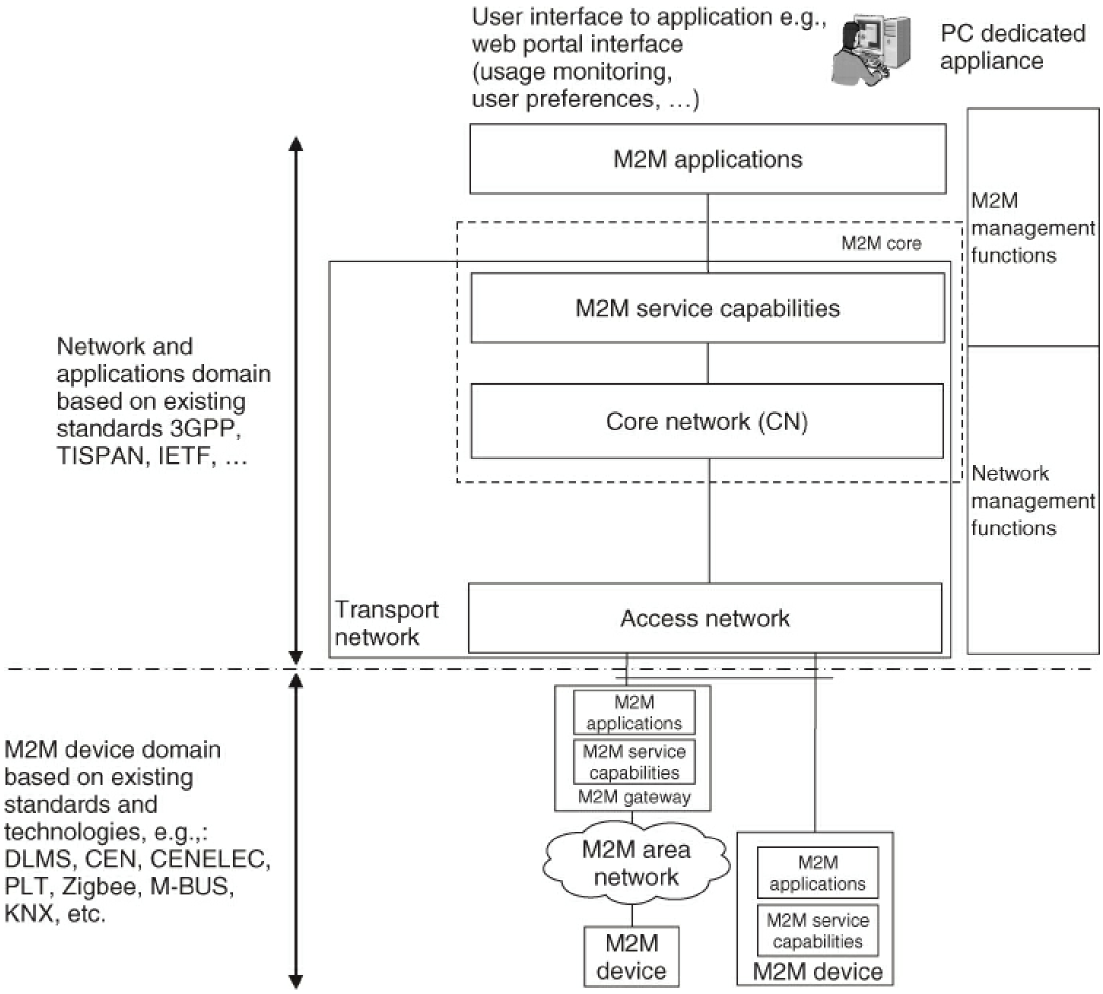

User interface to application e.g., web portal interface (usage monitoring, user preferences, ...)
PC dedicated appliance
M2M applications
M2M core
M2M management functions
M2M service capabilities
Network and applications domain based on existing standards 3GPP, TISPAN, IETF, ...
Core network (CN)
Network management functions
Transport network
Access network
M2M applications
M2M service capabilities
M2M gateway
M2M device domain based on existing standards and technologies, e.g.,: DLMS, CEN, CENELEC, PLT, Zigbee, M-BUS, KNX, etc.
M2M area network
M2M applications
M2M service capabilities
M2M device
M2M device

Figure 2.4: M2M high-level system architecture [16]

manage the access, transport, and core networks (*e.g.,* provisioning, supervision, and fault management), while M2M management functions provide the functions required to manage M2M applications and M2M SCs in the network domain. Finally, M2M applications run the service logic and use M2M SCs accessible via an open interface.

In order to allow the access to M2M Service Capabilities through an extendible and flexible structure, ETSI M2M TC developed a framework, which has its functional architecture presented in Figure 2.5. In this figure, MAS stands for M2M Authentication Server and MSBF stands for M2M Service Bootstrap Function. The role of MSBF is to facilitate the bootstrapping of permanent M2M service layer security credentials in the M2M Device (or M2M Gateway) and the M2M SCs in the network domain. The permanent security credentials that are bootstrapped using MSBF are stored in the MAS. Also, the M2M node is a logical representation of the M2M components in the M2M device, gateway or core.

This framework was created with extensibility in mind, since not all SCs are known at the time of the standards' specification, nor are all capabilities mandatory for an operational

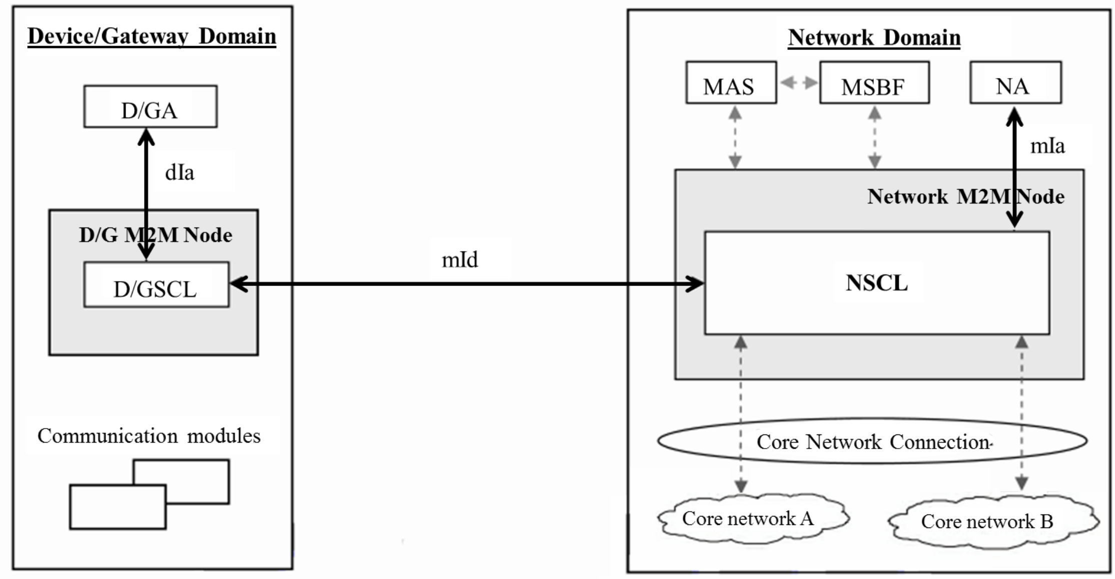


Figure 2.5: M2M Service Capabilities functional architecture framework [15]

deployment [16]. The framework is a skeleton for SCs and a set of reference points between the various entities of the system; these reference points are responsible for allowing the access to M2M SCs. The referred framework presents the following reference points:

- **mIa reference point:** allows a network application to access the M2M SCs in the network domain;
- **dIa reference point:** allows a device application to reside in an M2M device which enables its access to the different M2M SCs in the same device or in an M2M gateway; in the same way, it allows an application residing in an M2M gateway to access the different M2M SCs within the same gateway;
- **mId reference point:** allows an M2M SCs residing in an M2M device or gateway to communicate with the M2M SCs in the network domain and vice-versa.

In order to refer to SCs in the network, device and gateway, the following terms were adopted by ETSI: Network Service Capabilities Layer (NSCL), Gateway Service Capabilities Layer (GSCL), Device Service Capabilities Layer (DSCL), Service Capabilities Layer (SCL) and D/G SCL. NSCL, GSCL and DSCL refer to M2M SCs in the network, the gateway and the device, respectively, while SCL refers to any of the three. In the same way, D/G SCL refers to both DSCL and GSCL.

Service Capabilities provide M2M functions that are exposed to network applications through a set of open interfaces; they typically use core network functionalities through known and standardized interfaces. ETSI TC M2M identifies the list of SCs presented on Table 2.1 [15], where x stands either for N (Network), G (Gateway) or D (Device).

Table 2.1: List of M2M SCs

M2M SCs	**Description**
Application enablement (xAE)	provides a single Application Programming Interface (API) interface to applications.
Generic communication (xGC)	manages all aspects pertaining to secure-transport session establishment and teardown, as well as interfacing with bearer services provided by the core network.
Reachability, addressing, and repository (xRAR)	provides the capability of storing the state of applications, devices, and gateways. It also handles subscriptions to data changes.
Communication selection (xCS)	provides network and network bearer selection for devices or gateways that are reachable via multiple networks or multiple connectivity bearers.
Remote entity management (xREM)	provides functions pertaining to device/gateway life cycle management, such as software and firmware upgrade and fault and performance management.
SECurity (xSEC)	implements bootstrapping, authentication, authorization, and key management. Interfaces with a M2M authentication server.
History and data retention (xHDR)	stores records concerning the usage of the M2M SCs (optional).
Transaction management (xTM)	provides functions to manage the transactions, such as the aggregation of the individual operations results, and commits them when all individual operations have completed successfully (optional).
Compensation broker (xCB)	manages transactions compensation on behalf of applications (optional).
Telco operator exposure (xTOE)	provides access, via the same API used to access the SCs, to traditional network operator services (optional).
Interworking proxy (xIP)	allows a non-ETSI-compliant device to interwork with the ETSI standard.

The exchange of information between M2M applications and/or M2M SCL is accomplished according to REST principles. Besides some exceptions, it is assumed that all three reference points (mIa, mId and dIa) will use REST. A RESTful architecture is about the transfer of representations of uniquely addressable resources; ETSI M2M standardized the resource structure, which resides on a SCL. This structure will not be described in this document as it is already extensively described [15].

2.2 Web services

In recent years, Web services technologies have been successfully used for simplifying interoperability while providing flexibility and scalability to applications. A Web service (also known as an application service) is defined as a software system designed to support interoperable machine-to-machine interaction over a network [17]. They are mainly realized in two ways: SOAP-based services and RESTful services.

2.2.1 RESTful Web services

The term REST stands for *Representational State Transfer* and it identifies an architectural style for network-based systems. The term was introduced and defined in 2000 by Roy Fielding in his doctoral dissertation [18]. In essence, RESTful Web services use existing, proven Web standards – predominantly Hypertext Transfer Protocol (HTTP) [19] – and adhere to a number of principles for using these standards [20]. A representative example of a REST Web service is depicted in Figure 2.6.

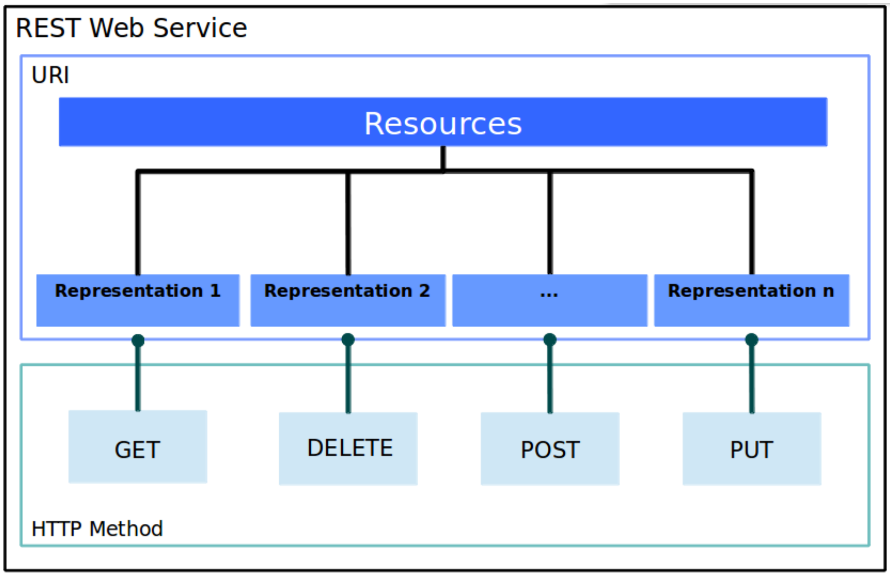


Figure 2.6: Representation of a REST Web Service

A Web service is called "RESTful" if it has a Web API conforming to the REST architectural style (REST API), which can be described as an assembly of interlinked resources.

In order to develop RESTful Web services it is normally used an architecture named Resource-Oriented Architecture (ROA) [21]. In ROA, REST principles are explicitly associated with Web protocols, such as Uniform Resource Identifiers (URIs), HTTP and eXtensible Markup Language (XML). In this approach, resources, which can be essentially any coherent and meaningful concepts that may be addressed, are unique, consumed by the clients using HTTP methods and accessed via a URI (see Figure 2.6). The state of a resource is transferred using its representation, which is typically any useful information about it.

A concrete implementation of a REST Web service follows four basic design principles:

- **Use HTTP methods explicitly:** One of the key characteristics of a RESTful Web service is the explicit use of HTTP methods in a way that it follows the protocol defined in RFC 2616 [19]: POST to create a resource on the server; GET to retrieve a resource; PUT to change the state of a resource or to update it and DELETE to remove or delete a resource.

- **Be stateless:** Each request from any client must contain all the necessary information to understand and serve the request, and it may not access any stored context on the server; session state is kept entirely on the client. This improves the Web service performance and simplifies the design and implementation of server-side components, because the absence of state on the server removes the need to synchronize session data with an external application [20];

- **Expose directory structure-like URIs:** One way to make an intuitive Web service is to define directory structure-like URIs. REST APIs use URIs to address resources. This type of URI is hierarchical, rooted at a single path, and branching from it are sub-paths that expose the service's main areas. RFC 3986 [22] defines the generic URI syntax as shown below:

  ```
  URI = scheme "://" authority "/" path [ "?" query ] [ "#" fragment ]
  Example: http://example.com:8042/over/there?name=ferret#nose
  ```

- **Transfer XML, JavaScript Object Notation (JSON), or both:** The data format that both application and service exchange in the request/response payload or in the HTTP body, must keep things simple and human-readable.

REST allows the resources manipulation through four basic interactions: CREATE, READ, UPDATE, and DELETE; these operations are also known by their acronym CRUD and they are mapped into the HTTP methods (CREATE is mapped on HTTP POST, READ on HTTP GET, UPDATE on HTTP PUT, and DELETE on HTTP DELETE). Since the same set of operations can manipulate the most diverse kind of resources, the same underlying architecture can be reused for multiple applications.

REST implementations are lightweight, which allows the presence of HTTP clients and servers even on the smallest platforms, what is therefore attractive for M2M applications. Additionally, the concept behind REST, maps naturally onto M2M devices, since every entity is a resource that has a particularly state that can be “manipulated”.

2.2.2 SOAP Web services

SOAP originally stood for *Simple Object Access Protocol* but this acronym was dropped with version 1.2 of the standard [23]. SOAP is a simple XML-based protocol that let applications exchange information over any transport protocol (mainly HTTP), and it provides a basic messaging framework upon which Web services can be built.

The messaging framework is defined by following the SOAP specifications [23]: SOAP processing model, extensibility model, underlying protocol binding framework and message construct. SOAP processing model defines the rules for processing a SOAP message; the extensibility model is responsible for defining the concepts of SOAP features and modules; the SOAP underlying protocol binding framework describes the rules for defining a binding to an underlying protocol that can be used for exchanging SOAP messages between SOAP nodes; and the SOAP message construct defines the structure of a SOAP message (see Figure 2.7).

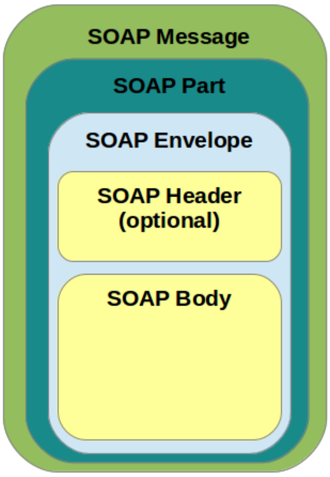


Figure 2.7: SOAP message structure

This protocol consists of an envelope, which defines what is in the message and how it should be processed, a convention for representing procedure calls and responses, and a set of encoding rules for expressing instances of application-defined data types.

The SOAP processing model defines several nodes, which participate in the transmission and reception processes. A SOAP node can transmit, receive, process and relay SOAP messages. The *initial SOAP sender* is the node that originates the SOAP message to be transmitted to the *ultimate SOAP receiver*. A message can be forwarded by *SOAP intermediary* nodes, and the set of nodes through which a single message passes is defined as *SOAP message path*. *SOAP senders* and *SOAP receivers* are nodes that transmit and accept SOAP messages, respectively.

This XML-based protocol is language and platform independent, and it can be used over any underlying transport protocol such as HTTP, Simple Mail Transfer Protocol (SMTP) or Transmission Control Protocol (TCP).

2.2.3 Comparison between REST and SOAP

As discussed above, REST uses standard HTTP, which simplifies and makes more easy to understand the process of creating classes or developing APIs. It was developed in order to create a set of principles that enable distributed systems to achieve greater scalability and allow distributed applications to grow and change over time [16]. One of its advantages is that REST allows many different data formats such as JSON, while SOAP uses predominantly XML. JSON can be an asset since it is recognized natively by JavaScript, and it is lightweight, highly portable, and human readable. In addition, REST allows that GET operations can be cached, while the same doesn't happen with SOAP.

Unlike REST, SOAP is an XML-based protocol that allows the use of different transport protocols. In terms of security, SOAP supports Secure Sockets Layer (SSL) (same as REST) but it also supports identification through intermediaries, not just point-to-point. Besides, it provides a standard implementation of data integrity and data privacy, and it supports ACID [24] transactions,[2] which can be needed for some enterprise applications. Another characteristic of SOAP is that it provides end-to-end reliability, while REST expects clients to deal with communication failures since it doesn't have a standard messaging system. SOAP can also easily tunnel firewalls and proxies, without modifications to the protocol and with the existing infrastructure.

Comparing the two concepts, REST has gained widespread acceptance across the Web as a simpler alternative to SOAP (see Figure 2.8), which was the predominant Web architecture. For service providers, RESTful services can improve system flexibility, scalability and performance compared to SOAP-based Web services [25].

REST-based architecture still offers several advantages that may be useful to M2M application developers, such as the possibility to visualize and manipulate the sensors data and calibration parameters through a Web browser, or even to create Web mash-ups[3] using one or more sensors of M2M devices as data sources [16]. The REST model is great to be used in M2M communications once it exposes resources via its URIs and allows remote clients to be able to perform CRUD (Create, Read, Update, Delete) operations over them. For instance, a command can be sent to a sensor, or an observation can be obtained from it, by accessing the proper URI with the correct HTTP method and payload (normally JSON or XML). In addition, REST provides a lighter tool that can reduce the effort need to develop applications when compared with SOAP.

[2] ACID stands for *Atomicity, Consistency, Isolation and Durability* which is a set of properties that guarantee that transactions are processed reliably.

[3] A mash-up is defined as a Web page, or Web application, that uses and combines data from different sources in order to create new services.

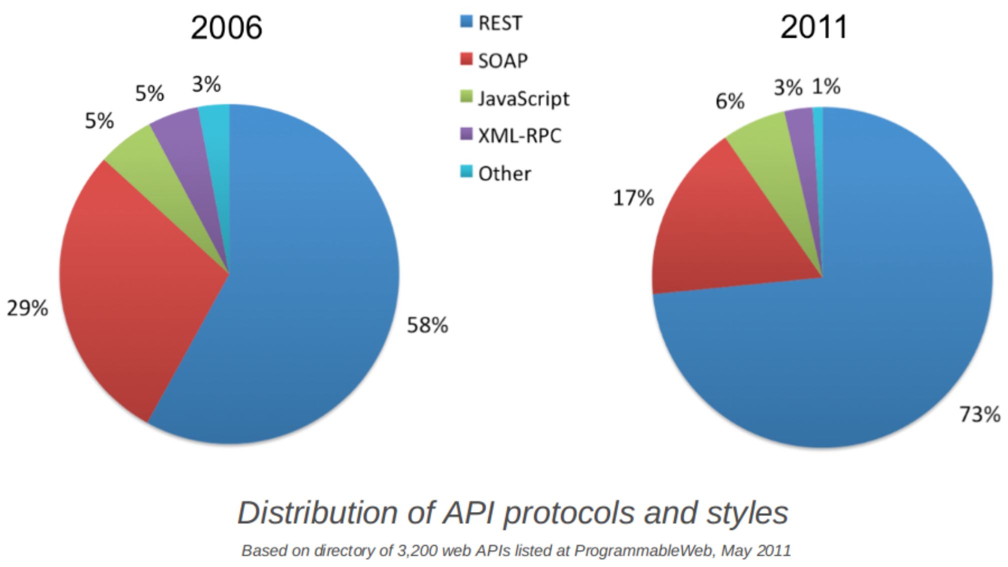


Figure 2.8: API market distribution [26]

2.3 Traffic critical constraints

Within the scenarios described throughout this chapter, applications exchange different kinds of traffic. Thus, they have the need to differentiate traffic according to its importance to the proper functioning of the system. Traffic differentiation refers to treating the packets of one flow differently than those of another flow; this differentiation may be due to the applications need to keep certain parameters such as the delay, within some specified limits, or to make sure that critical traffic is treated first.

The variety of scenarios brought by M2M communications, has been increasingly imposing the need of distributing the network resources according to the traffic priority. Figure 2.9 presents the huge amount of services that can take place within a home environment; it may be composed by subnetworks, which can use different technologies to connect services with similar purposes.

The fact that more network applications emerge within residential environments, increases the bandwidth demand, which along with the variety of services that users can have within their homes, require a careful bandwidth distribution in order to ensure the performance of the system for critical services.

Although some M2M applications do not impose stringent requirements on data delivery, there are a number of others in which the guarantee of some certain level of performance is of great importance. As discussed above, this guarantees may be different for the different traffic flows. For instance, in a home automation scenario, it may exist periodic traffic that carries the sensors measures, control and maintenance traffic, and alarm messages generated when an error or a critical situation occurs in the system. In this case, it is convenient that the

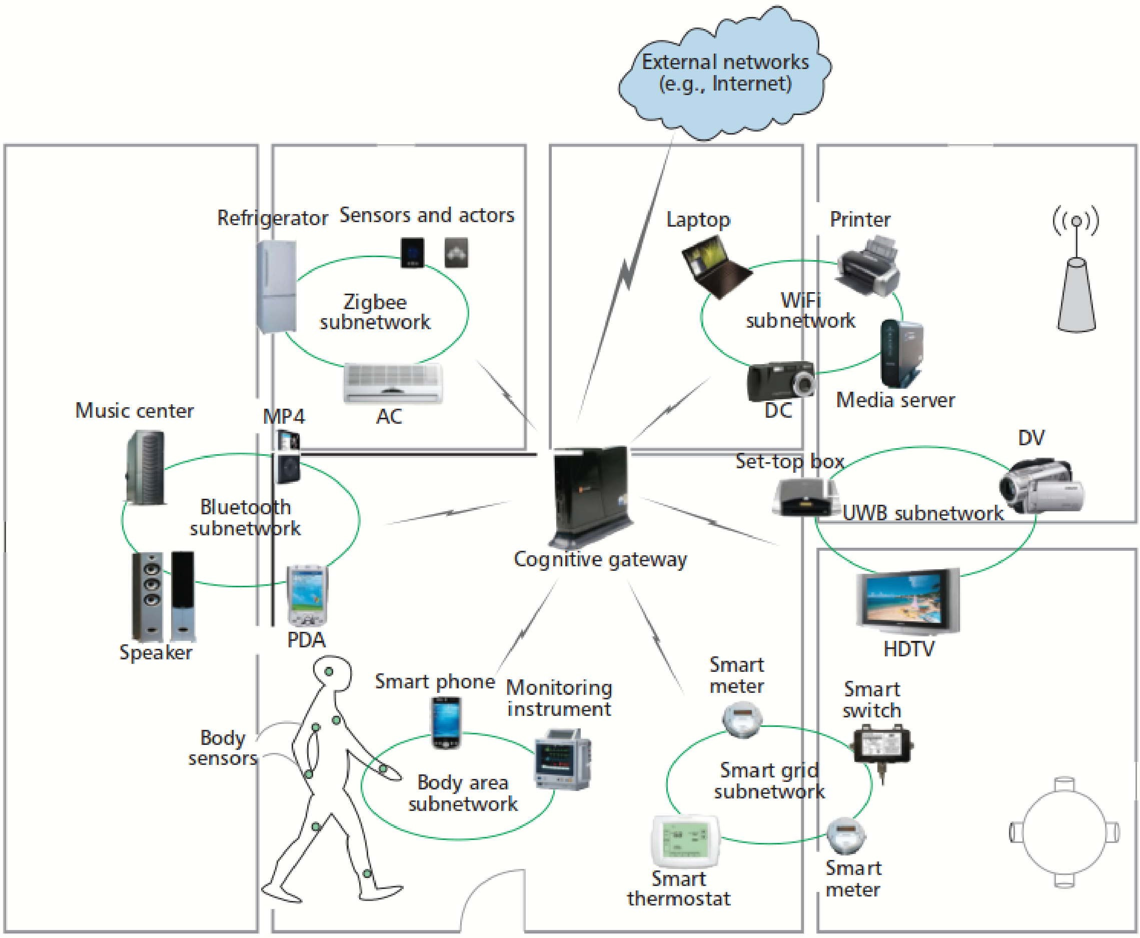


Figure 2.9: Different kinds of services/traffic in a home scenario [27]

traffic is prioritized, giving lower priority to the measurement messages and higher priority to the alarm ones. This way, it can be guaranteed that even during traffic congestion, alarm messages are successfully delivered to their destination.

The users' home network capacity has been growing, which made possible the provision of various multimedia services that require high bandwidth. Most of these services are mainly used by the user for entertainment purposes, so their quality should provide the satisfaction of users, however, they must have a lower priority when compared with real-time critical services, such as alarm messages sent by a monitoring system. The lack of adequate guarantees of bandwidth or delay for each service, may result in the network congestion and, consequently in the performance degradation of some services, due to the lack of network resources. This reflects the need of providing different guarantees of performance to different applications, according to their network resources requirements and their criticality on the system. The ability to provide some guarantees to data delivery is designed by Quality of Service (QoS).

2.3.1 QoS

The term QoS can be defined as the ability to guarantee a certain level of performance to a data flow. Aspects such as latency, error rate and jitter can be guaranteed, which is especially important in scenarios where critical applications are running and the network capacity is insufficient. Are example of applications where QoS mechanisms may be desired or required, Voice over IP (VoIP), videoconferencing, online games and safety-critical and industrial applications. These services require certain guarantees, which if not met can result in the degradation of its quality or even in the malfunction of the system. Such kind of services are becoming increasingly popular and the Future Internet will be faced with much more applications requiring Quality of Service.

In packet-switched networks, the following problems may have the need to fulfill some guarantees depending on the network traffic:

- **Latency:** a packet can take a long time to reach the destination which can be caused, for instance by waiting in long queues due to network congestion;
- **Error rate:** can be caused by noise and interference;
- **Low throughput:** the bit rate can be too low to support some applications guarantees, due to the load variation caused by other users sharing the same network resources;
- **Jitter:** packets from the source will reach the destination with different delays due to their position in the routers queues, which can vary unpredictably;
- **Dropped packets rate:** packets may have to be dropped by the router, which can happen because the buffers are already full or the data is corrupted.

A network or a service that does not support QoS is designated as "best-effort". Such service does not provide any guarantees that the data is delivered within certain delays, which makes its delivery time unpredictable. A user or application that required a best-effort service will obtain unspecified variable bit rate and delivery time, depending on the current traffic load.

In packet-switched IP networks, the two main approaches to provide QoS are: the Integrated services ("IntServ") [28] and Differentiated services ("DiffServ") [29].

IntServ

The IntServ architecture model provides a way to deliver end-to-end QoS, which was motivated by the needs of real-time applications. In order to establish and maintain QoS, it uses the mechanisms of resource reservation and admission control, and it is intended to provide the closest thing to circuit emulation on IP networks.

This model identifies three main categories of services that can be provided: best effort, which provides no QoS guarantees; controlled load services, which assure that users will get service that is, as close as possible, to the one received by a best-effort service in a lightly loaded network; and guaranteed services that provide users with an assured amount of bandwidth and firm end-to-end delay bounds.

The request of the required QoS guarantees is accomplished by using Resource ReSerVation Protocol (RSVP) [30] that apply a two way handshake to establish and maintain a sender-receiver connection that guarantees a certain level of service. If every network device along the path can reserve the necessary requested bandwidth, the transmission can take place.

Besides end-to-end signaling, IntServ requires that routers and switches are able to perform the following functions along the path: admission control, classification, policing, queuing and scheduling. Admission Control is responsible to determine whether a new flow can be granted with the requested QoS guarantees without impacting existing reservations; classification functions should recognize packets that IntServ provides for a rich end-to-end QoS solution, using end-to-end signaling, state-maintenance (for each RSVP-flow and reservation) and admission control at each network element; policing function are responsible to take action when traffic does not conform to its specified characteristics; queuing and scheduling functions must forward packets according to their QoS requests.

The main problem with the IntServ architecture is the lack of scalability. The requirements on the routers are very high (huge storage, capacity to process the overheads, must implement admission control, classification, policing, etc.) which increase their complexity.

DiffServ

Differentiated services or DiffServ is an architecture defined in order to overcome the disadvantages of IntServ. DiffServ is more scalable, manageable and easy deployable for service differentiation in IP networks. This approach pushes out the complexity to the edge routers while the core routers are maintained as simple as possible.

Unlike IntServ, this scheme uses relatively simple methods to categorize traffic into different classes, called Class of Service (CoS), and then apply QoS parameters to those classes. Packets are divided into classes by marking the Type of Service (ToS) byte in the IPv4 header or the Traffic Class (TC) in the IPv6 header.

When packets are classified at the edge of the network, specific forwarding treatments, formally called Per-Hop Behavior (PHB), are applied on each network element, providing packets with the appropriate delay-bound, jitter-bound, bandwidth, etc. This combination of packet marking and well-defined PHBs, results in a scalable QoS solution for any given packet, and any application. [31]

The PHB is determined by the Differentiated Service field (DS field) of the IP header. The DS field, which is present on the IPv4 Type of Service (ToS) byte and IPv6 TC byte, is composed by a 6-bit Differentiated Services Code Point (DSCP) value and a 2-bit Explicit Congestion Notification (ECN). The default PHB specifies that a packet marked with a DSCP value of "000000" gets the traditional best-effort service. The core routers forward packets by examining the DSCP field, and each possible value for it has a unique and specific meaning for how the packet should be treated.

As described above, under DiffServ, all the policing and classifying is done at the boundaries between DiffServ domains, which means that the complex operations are performed by the edge routers. In contrast to IntServ, DiffServ requires no advanced setup, no reservation, and no time-consuming end-to-end negotiation for each flow. However, it makes difficult to predict the end-to-end behavior, and mapping packets to pre-defined classes using DSCP produces more delay.

A comparison between Best-Effott, IntServ and DiffServ is presented in Table 2.2.

Table 2.2: Comparison between Best-Effort and the IntServ and DiffServ approaches [32]

	IntServ	**DiffServ**	**Best-Effort**
Granularity of QoS	Per-flow	Per-class	None (fair to all)
Services	- Guaranteed (quantitative) - Controlled Load (qualitative)	- Expedited Forwarding (quantitative) - Assured Forwarding (qualitative)	"best-effort"
Resource Allocation	Dynamic	Static or Dynamic	None
Control	At host/router	Marking at the edge and queue management at the core	FIFO only
Complexity	High	Medium	Low

3

Communication protocols for WSN

Within the line of thought of the previous chapter, the connectivity between different M2M devices or between them and a gateway or public router, is accomplished through the M2M area network. Wireless Personal Area Networks (WPANs), such as ZigBee, Bluetooth, Z-Wave, etc., are examples of network technologies that may be used to establish M2M area networks; they are of great importance in scenarios that require low maintenance, such as sensor network applications.

The present chapter does a theoretical approach of LR-WPANs focusing on the IEEE 802.15.4 standard and the ZigBee technology. It also presents a comparison between ZigBee and the Bluetooth technology, namely the emergent Bluetooth Low Energy.

3.1 IEEE 802.15.4 LR-WPAN

IEEE 802.15.4 is a standard, maintained by the IEEE 802.15 working group [33], that specifies the Physical layer (PHY) and the Media Access Control (MAC) layer for LR-WPANs. The second revision of this standard was approved in June 2011.

WPANs are used to convey information over relatively short distances and, unlike Wireless Local Area Networks (WLANs), connections involve little or no infrastructure, which allows the implementation of small, power-efficient, low-cost solutions. LR-WPANs support transfer rates until 250 kbps and they are design to typically operate in the Personal Operating Space (POS) of 10m, allowing wireless connectivity in applications with limited power and relaxed throughput requirements.

3.1.1 WPAN's components

The most basic component of a WPAN is the node or device. The standard classifies the devices in one of two categories: Full Function Devices (FFDs) and Reduced Function Devices (RFDs). These two types present different characteristics and each one is designed to perform different functions in the network. FFDs are capable of performing the function of network coordinator and they can communicate with any other devices; they have a greater processing capacity and they can be used in any topology, however, this type of devices consume more power. On the other hand, RFDs were designed to be extremely simple devices with very modest resources and communication requirements. They can only communicate with FFDs and they can never act as coordinators. An overview of the main characteristics of each device type is presented in Table 3.1.

Table 3.1: Comparison between FFDs and RFDs

Device	**Main characteristics**
FFD	- Can be used in any network topology - Can be used either as network or PAN coordinator - Capable of communicate with any other device - Capable of implement the full protocol stack - Greater processing power - Requires higher energy consumption
RFD	- Limited to act in the star topology or as an end device in a Peer-to-Peer (P2P) network - Not capable of being used as network or PAN coordinator - Reduced set protocol - Limited processing and memory capacity - Simple implementation - Requires lower energy consumption

3.1.2 Network topologies

Depending on the application requirements, an IEEE 802.15.4 LR-WPAN may operate in either of two topologies: star topology or Peer-to-Peer topology, as shown in Figure 3.1. P2P differs from the star topology in a way that a device is able to communicate with any other, as long as they are in range of one another; in the star topology the communication is established between devices and a single central controller, the PAN coordinator. Peer-to-Peer topology has the advantage of allowing more complex network formations to be implemented, such as mesh or tree topologies.

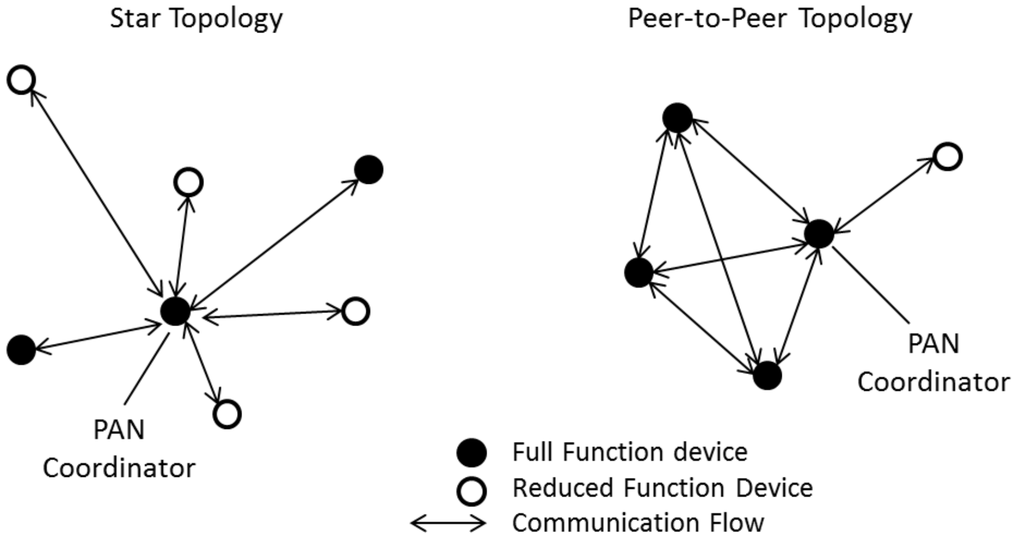


Figure 3.1: Comparison between star and P2P topologies in an IEEE 802.15.4 network [11]

3.1.3 Architecture

The IEEE 802.15.4 standard [11] specifies the two lower layers of the Open Systems Interconnection (OSI) model, as shown in the Figure 3.2: the PHY and the MAC layer. As represented in the Figure, the communication between the different layers are accomplished through a number of Service Access Points (SAPs). All the above layers are outside the scope of this standard and, in case of a ZigBee network, they are defined by the ZigBee Alliance [4].

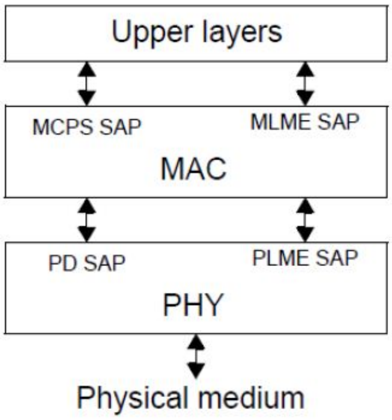


Figure 3.2: LR-WPAN device architecture [11]

Physical layer

The physical layer contains the Radio Frequency (RF) transceiver and it is responsible for its management, and for the transmission and reception of information across the physical radio channel. The radio transceiver can operate in one of three modes: transmission, reception or sleep; its activation and deactivation is performed by the PHY. Sleep mode allows the transceiver to suspend the reception and transmission of information, for a predefined period of time, thereby reducing their power consumption.

Once a device wants to transmit information, it first performs an Energy Detection (ED) within the current channel; it is an estimation of the received signal power within the bandwidth of an IEEE 802.15.4 channel. The quality of a received packet is characterized through the Link Quality Indicator (LQI), which may be implemented using the ED, a Signal-to-Noise Ratio (SNR) estimation, or a combination of both methods.

The Physical layer shall also provide the capacity to perform Clear Channel Assessment (CCA) for Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA). The standard defines six CCA modes that are used to check whether the medium is idle or busy. The first and third modes establish a threshold to the value obtained through the Energy Detection; in the first method CCA shall report a busy medium upon detecting any energy above the ED threshold; the second method reports a busy medium only upon the detection of a signal with the same modulation and spreading characteristics of the PHY that is currently in use by the device (this signal may be above or below the ED threshold); in the third mode, CCA shall report a busy medium upon the detection of CCA Mode 1 and/or CCA Mode 2. In mode 4, CCA shall always report an idle medium, and in the remaining two modes, the report of a busy medium is based on the Ultra Wide Band (UWB) preamble sense.

With the evolution of the standard, new PHYs were added in order to provide new features and to expand the number of allowed frequency bands. The PHYs defined in the standard are [11]:

- **Offset Quadrature Phase-Shift Keying (O-QPSK) PHY:** Direct Sequence Spread Spectrum (DSSS) PHY employing O-QPSK modulation, operating in the 780 MHz bands, 868 MHz, 915 MHz, and 2450 MHz;

- **Binary Phase-Shift Keying (BPSK) PHY:** DSSS PHY employing BPSK modulation, operating in the 868 MHz, 915 MHz, and 950 MHz bands;

- **Amplitude Shift Keying (ASK) PHY:** Parallel Sequence Spread Spectrum (PSSS) PHY employing ASK and BPSK modulation, operating in the 868 MHz and 915 MHz bands;

- **Chirp Spread Spectrum (CSS) PHY:** CSS employing Differential Quadrature Phase-Shift Keying (DQPSK) modulation, operating in the 2450 MHz band;
- **UWB PHY:** combined Burst Position Modulation (BPM) and BPSK modulation, operating in the sub-gigahertz and 3–10 GHz bands;
- **M-ary Phase-Shift Keying (MPSK) PHY:** MPSK modulation, operating in the 780 MHz band;
- **Gaussian Frequency-Shift Keying (GFSK) PHY:** GFSK operating in the 950 MHz band.

The multiple physical layers are defined in order to support a variety of frequency bands, as exposed in Table 3.2. This allows the use of this technology in a larger number of scenarios.

Table 3.2: Supported frequency bands

Frequency band [MHz]	**Use**
868 – 868.6	Europe
902 – 928	North America
2400 – 2483.5	Worldwide (ISM band)
314 – 316 430 – 434 779 – 787	China
950 – 956	Japan

MAC layer

The Media Access Control (MAC) layer provides the interface between the PHY and the network layer. It handles all accesses to the physical radio channel and it is responsible for providing a reliable link between two peer MAC entities. It is also responsible for both the generation of network beacons[1] (if the device is a coordinator) and the device synchronization according to beacons reception. Beacons define the superframe structure (Figure 3.3), which can contain Guaranteed Time Slots (GTSs) (allocated by the coordinator). Additionally, MAC is responsible for supporting PAN association and disassociation, for employing the CSMA-CA mechanism for channel access, and for providing some security mechanisms.

As commented above, MAC layer supports a structure designated superframe which is bounded by network beacons sent by the coordinator. There can be up to three types of periods in a superframe: the Contention Access Period (CAP), the Contention-Free Period (CFP), and the inactive period (optional). The combination of CAP and CFP is known as

[1] Beacons are control packets used to synchronize the attached devices, to identify the PAN, and to describe the superframe structure. The current ZigBee protocols support beacon and non-beacon enabled networks.

the active period, which is divided into 16 equal time slots. During the inactive period the coordinator may enter in low-power (sleep) mode, under which it can turn off its transceiver circuits to conserve battery energy.

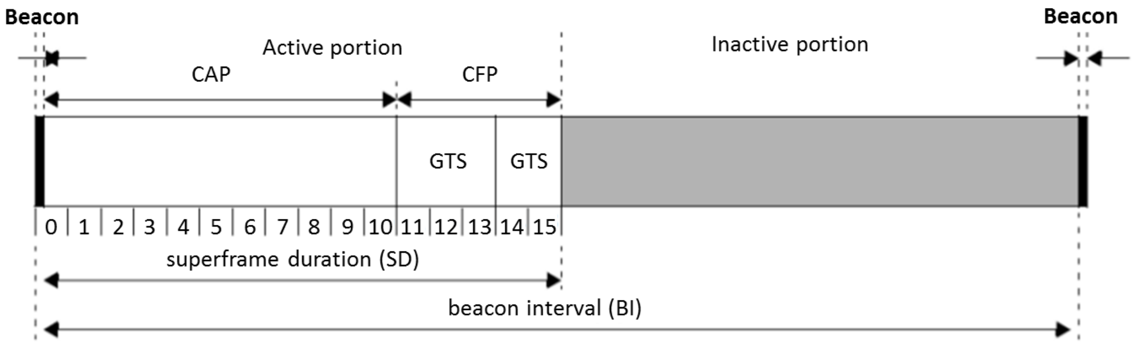


Figure 3.3: An example of the superframe structure [11]

During CAP, all the devices that want to transmit need to use the CSMA-CA mechanism to gain access to a frequency channel, and there is no guarantee for any device to be able to use a frequency channel exactly when it needs it. The CFP, in contrast, guarantees a time slot for a specific device, and therefore the device does not need to use CSMA-CA for channel access. For low-latency applications, or applications that require specific data bandwidth, the PAN coordinator can dedicate portions of the CFP period of the superframe to that application, known as GTSs. The PAN coordinator is able to allocate up to seven GTSs and each one is allowed to occupy more than one slot period.

3.2 ZigBee

ZigBee is defined as a low-cost, low-power, low-rate, wireless network standard. This standard was developed by ZigBee Alliance, who counts with more than 400 member companies [34]. The ZigBee Alliance was established in 2002 as a non-profit organization, with the goal of providing open global standards for low-power wireless networks focused on sensor applications.

ZigBee Alliance and Institute of Electrical and Electronics Engineers (IEEE) have been working closely to specify the entire protocol stack. While the IEEE 802.15.4 focuses on the specification of the two lower layers of the protocol stack, ZigBee Alliance aims to provide the upper layers (from network to the application layer). This partnership allowed ZigBee to become an open global standard and grow inside markets such as wireless control.

ZigBee technology has already a solid presence in the market and it has applications in several areas like home automation, industrial control, medical communication systems, among many others, as shown in Figure 3.4. It success is mainly due to its low-cost, power efficiency, high reliability and security features.

Figure 3.4: ZigBee market segments

3.2.1 Architecture

As shown in Figure 3.5, the ZigBee standard defines only the network and application layers of the protocol and it adopts IEEE 802.15.4 PHY and MAC layers as part of the ZigBee networking protocol. ZigBee also provides a set of security services that can be used for both the network layer and the application layer.

The network layer interfaces between the MAC and the application layer, and it is responsible for managing the network formation and routing. The application layer is the highest protocol layer in the ZigBee network and it hosts the ZigBee Device Objects (ZDOs)[2].

IEEE 802.15.4 was developed independently of the ZigBee standard, and it is possible to build short-range wireless networking based solely on IEEE 802.15.4 standard and not implement ZigBee-specific layers. In this case, the users develop their own networking/application layer protocol on top of IEEE 802.15.4 PHY and MAC. The decision of whether or not to implement the entire ZigBee protocol or just IEEE 802.15.4 PHY and MAC layers, depends on the application and the long-term plan for the product [35].

[2] ZDOs control and manage protocol layers in a ZigBee device. They are responsible for a number of tasks, such as the management of requests to join a network, device discovery and security.

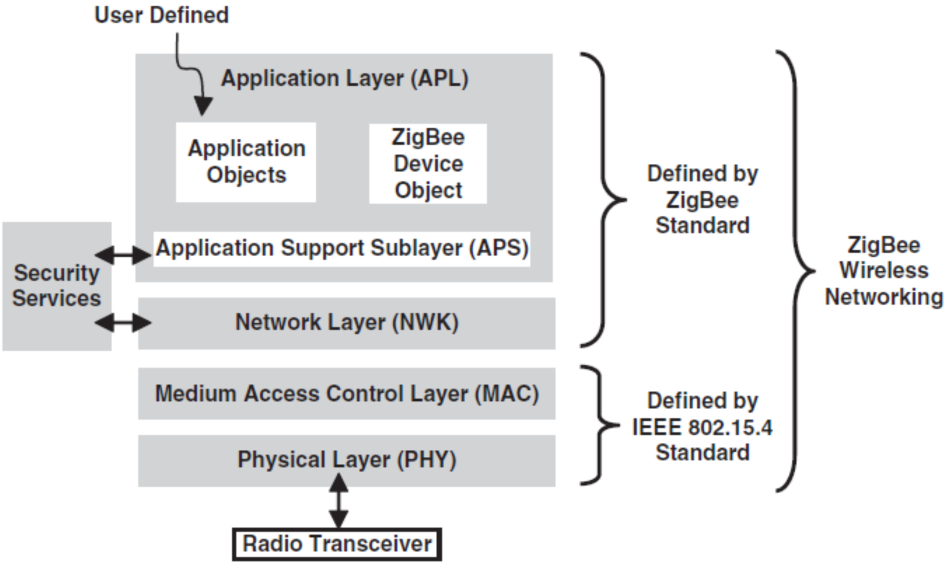

User Defined
Application Layer (APL)
Application Objects
ZigBee Device Object
Application Support Sublayer (APS)
Security Services
Network Layer (NWK)
Medium Access Control Layer (MAC)
Physical Layer (PHY)
Radio Transceiver
Defined by ZigBee Standard
Defined by IEEE 802.15.4 Standard
ZigBee Wireless Networking

Figure 3.5: ZigBee wireless networking protocol layers [35]

3.2.2 Device types

A ZigBee network supports three types of devices, according to their functionality in the network; a device can be either a ZigBee Coordinator, a ZigBee Router or a ZigBee End Device.

Each ZigBee network has a single coordinator, which is responsible for the starting and maintenance of the network's proper operation. It is also responsible for implementing security functions and storing relevant information related to the network.

A ZigBee Router is an optional device in the network and it has the ability to connect different networks as well as to send and to receive data messages. It allows multi-hop communications and, in beacon-enabled networks, these devices transmit periodic beacons in order to confirm their presence to other network nodes. Booth the ZigBee coordinator and ZigBee routers are FFDs.

ZigBee End Devices contain just enough functionality to talk to the parent node (either the coordinator or a router). They are RFDs and, in beacon-enabled networks, they may sleep between beacons, thus lowering their duty cycle and extending their battery life. An overview of the main characteristics of each device type is presented in the Table 3.3.

Table 3.3: Comparison between the different types of devices in a ZigBee network

Device	Main characteristics
Coordinator	- Starts the network - Is a FFD - Assigns addresses - Stores relevant information to the maintenance of the network - Responsible for network security functions - Each network has only one coordinator
Router	- Optional device - Is a FFD - Allows multi-hop communication - Allows the connection between different networks - Routing functions
End device	- Can be either a RFD (general rule) or a FFD - Allows the device to enter in sleep-mode - Only communicates with the parent node - Typically has sensors coupled to it - Optimizes its battery life

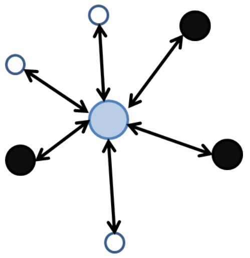
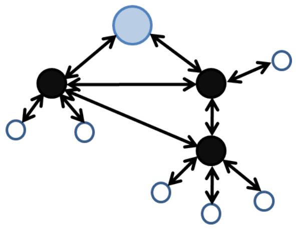
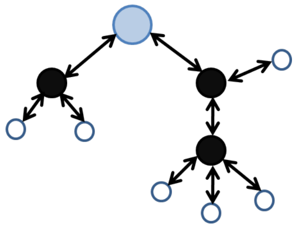
3.2.3 Network topologies

The ZigBee network layer allows the formation of three distinct network topologies: star, mesh, and tree.

In the star topology the network is started and maintained by a single device - the coordinator - with which all other devices directly communicate. In order to overcome the network range and size constraints imposed by the use of the star topology, it was developed the tree topology, where each end device maintains a single path between itself and the coordinator, however, multi-hop communications are allowed through routers. This topology has the disadvantage that if a router loses communication, all the nodes which depend on it, will also be inhibited of communicating. To overcome this problem, it is often used the mesh topology, which through the addition of redundant communication routes increasing the communication reliability. Thus, if a router loses connection, the network will redirect the data by an alternative path - hence the common designation of a ZigBee network as a "self-organizing mesh network". Although increasing the communication reliability, mesh topology does not allow sending beacons; this way, devices cannot enter in sleep mode, which may jeopardize the use of this topology in environments with energy constraints.

The network topology choice must take into account both the environment and application requirements, in order to establish a network that better fulfills these requirements. For instance, it may be useful to use the star topology in applications with a small number of devices that are located close to each other, and the mesh topology to implement highly flexible applications within environments with low energy constraints. On the other hand, within environments with high energy constraints, it may be useful to use the tree topology, since the devices are allowed to enter in sleep mode and therefore, they may reduce their power consumption. Table 3.4 describes the main characteristics of more frequent topologies in ZigBee networks.

Table 3.4: Comparison between the different network topologies

Topology	Characteristics	Example
Star	- Simple and low-cost - Devices communicate directly with ZC - Easy configuration and synchronization - Applicable in small networks - Low latency	
Mesh	- Allows full P2P communication - Regular beacons are not allowed - End devices can't enter in sleep mode - Robust multi-hop communication - Flexible, low latency network - More expensive routing (memory to keep the routing tables)	
Tree	- Low-cost routing - Regular beacons are allowed - End devices can enter in sleep mode - Allows multi-hop communication - Flexible, low latency network - Hierarchical routing strategy - Might have high latencies	

Legend:

🔵 Coordinator – FFD

● Router – FFD

○ End device – FFD ou RFD

↔ Communications flow

3.2.4 Security

In a wireless network, a transmitted message can be received by any nearby network device. This principle can be used by an intruding device that can listen the messages, modify them or even send his own messages. This can violate people's privacy, cause malfunctions on the system or even shut-down the entire system.

In order to avoid the problems described, the ZigBee standard defines the following methods for implementing security services: cryptographic key establishment; key transport; frame protection and device authorization.

ZigBee security architecture includes security mechanisms at MAC, Network and Application layers of the protocol stack. Each of these layers has services defined for the secure transport of their frames. The MAC layer is responsible for its own security management, but are the upper layers that determine which security level to use [35].

To operate securely within a network, a device must have a counterpart device, usually the ZigBee Coordinator, which it can trust to obtain keys and which performs ZigBee access control; this counterpart device is designed Trust Center (TC). The TC is responsible for: storing the network keys; using the security services to configure a device with its key(s); and using the security services in order to authorize a device onto the network.

ZigBee security is based on symmetric keys, in which both sender and receiver of a protected transaction need to share the same key. There are three types of keys:

- **Master key:** used to establish a link key between two devices as part of key establishment protocol;
- **Link key:** key which is uniquely shared between two and only two devices for protecting frames at the Application Support Sublayer (one of those devices is normally the Trust Center);
- **Network key:** global key which is used by all devices in the network.

A link key can get to both ends through one of three methods: pre-installation, key transport and key establishment. In the pre-installation method, the keys can be placed into devices using an out-of-band method (*e.g.,* by the manufacturer); in this way, when a device joins the network, it does not need to require a security key to the TC. In the key transport method, the device requests a security key and the Trust Center is responsible to send it (securely whenever possible). With this solution the system may be vulnerable for a moment, if the TC sends the key over an unsecured link.

Key establishment is the method of creating a key in both devices without communicating the key itself. ZigBee uses this method predominantly based on the Symmetric-Key Key Establishment (SKKE) [36] protocol. The devices that are establishing the key already have a master key (provided by means of pre-installation, key-transport or user-entered data such as passwords); the initiator device establishes a link key using the master key, and transfers the data to the responder, which uses the data to derive the link key. The initiator also derives the link key from the data and, if the derivation is done correctly, the two devices have the same link key that can be used in the symmetric key cryptography.

After establishing all keys needed, encryption, integrity and authentication can be applied at the Application, Network, and MAC layers to secure the frames at each of those levels.

Encryption

Encryption is the process of encoding messages such that they can only be understood by the authorized parties. The transmitter uses an algorithm to encrypt the message before the transmission and only the intended recipient can decrypt it and recover the original message. The unencrypted message is known as plaintext, while the encrypted one is known as cyphertext. The basic concept of encryption using symmetric keys is shown in Figure 3.6.

ZigBee supports the 128-bit Advanced Encryption Standard (AES) [37] encryption to encrypt data in a ZigBee network. In AES, the encryption algorithm is public knowledge and available to everyone, and it is associated with a secret key; the ZigBee standard uses a 128-bit key.

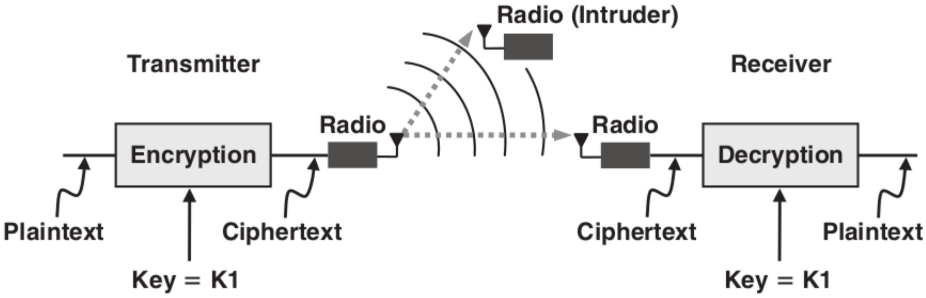


Figure 3.6: Encryption using symmetric keys [35]

Authentication and integrity

The ZigBee standard also supports both device authentication and data integrity. Those are the acts of confirming if either a new device that joins the network is the one that it claims to be, or the data received is the one that was sent without being modified, respectively.

The AES algorithm is not only used to encrypt the information but also to validate the data sent. This is the concept behind data integrity and it is achieved using a Message Integrity Code (MIC), also known as Message Authentication Code (MAC) or authentication tag, which is appended to the message. This code ensures integrity of both the header and payload data attached, and the more large the more secure it is (although less payload the message can take). The MIC is generated by a method known to both receiver and transmitter so that an unauthorized device won't be able to create it.

The device authentication procedure is performed by the Trust Center and while a new device is not authenticated within a secure network, it has the status "joined, but unauthenticated". If for any reason, the TC decides not to authenticate the newly joined device, it will request the device to be removed from the network.

ZigBee-2006 [38] and ZigBee-Pro [36], which are the second and the third versions of the ZigBee standard respectively, support additional security features when compared with the original version, released in 2004 [4]. In ZigBee-2006, the Trust Center can operate in either residential or commercial mode; the second one maintains a higher security level. ZigBee-Pro uses the same concept but renames the commercial and residential modes of operation to high security and standard security modes accordingly.

Residential and commercial modes have distinct procedures in order to authenticate a device. In residential mode, if a new device joins the network and does not have a network key, a moment of vulnerability is caused because the TC needs to send the network key over an unprotected link. On the other hand, if the new device already has the network key, it must wait until it receives a dummy network key from the TC, and it keeps the sender address of the message, which it associates to the TC address. In commercial mode, the TC never sends the network key over an unprotected link, however, if the new device does not have a shared master key with the TC, a master key may be sent unsecured. When the master key is received, the device and the Trust Center must start the key establishment protocol, where the device has a limited time to establish a link key with the TC. If the new link key is confirmed, the TC will send the network key over a secured connection and the device is considered authenticated inside the network.

3.2.5 QoS

In order to allow some QoS guarantees, the IEEE 802.15.4 standard provides the possibility to configure Guaranteed Time Slots on beacon-enabled networks (see 3.1.3). These guaranteed slots can be used by applications wanting to implement networks that require quality of service guarantees.

GTSs are a good option for low-latency applications in which the device cannot afford to wait for a random and potentially long period of time until the channel is available. One consequence of guaranteed time slots is that each slot only gives a fraction of the maximum bandwidth for the standard. Besides, a GTS shall only be allocated by the PAN coordinator, and it shall only be used for communications between the PAN coordinator and a device associated with the PAN through the coordinator.

Recent research works in Wireless Sensor Networks (WSNs) have focused on QoS support in order to improve the reliability and performance under severe energy constraints. In the network layer, improvements of QoS can be accomplished by adding and/or improving the QoS support to the routing algorithm. The QoS measurement and comparison using different routing protocols is presented in several papers such as [39], [40] and [41]. The same approach using different modulation schemes is presented in [42].

3.3 Bluetooth

Bluetooth is one of the most popular radio technologies for wireless devices. The choice of the radio technology in sensor network applications is particularly important since it has impact not only on the energy consumption, but also in the software design [43].

Bluetooth, which was first designed as a cable replacement technology, has now some interesting characteristics that allow it to be used in Wireless Sensor Networks. The mass production of Bluetooth radios ensures robustness and decreasing costs, while spread spectrum techniques make the radios almost immune to interferences in the free 2.4 GHz band.

With Bluetooth v4.0 and Bluetooth Low Energy protocol this technology took an important step in order to be used in low power applications.

3.3.1 Technical description

Bluetooth, also known as the IEEE 802.15.1 standard, is based on a wireless radio system designed for short-range and cheap devices, in order to replace cables for computer peripherals, such as mice, keyboards, joysticks and printers. As shown in Figure 3.7, this technology has adopted several enhancements to its core specification since its first version, v1.0, published in 1999 [44].

This is a master-slave technology managed by the Bluetooth Special Interest Group (Bluetooth SIG [46]), which is a privately held association founded in September 1998. Bluetooth connects a huge amount of devices allowing them to use a high-speed, low-power wireless

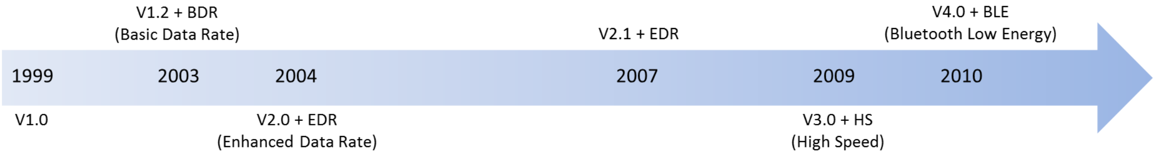


Figure 3.7: Bluetooth evolution [45]

technology. It is designed to automatically connect portable equipment together when one Bluetooth device is in close distance to another one, and using the same standard.

The Bluetooth technology is divided into two specifications: the core and the profile specifications. The core system architecture is presented in Figure 3.8 and it is composed by the RF Protocol, Link Control Protocol (LCP), Link Management Protocol (LMP) and Logical Link Control and Adaptation Protocol (L2CAP). In addition to the core specifications that discusses how the technology works, Bluetooth provides a set of profile specifications that focuses on how to build interoperating devices using the core technologies.

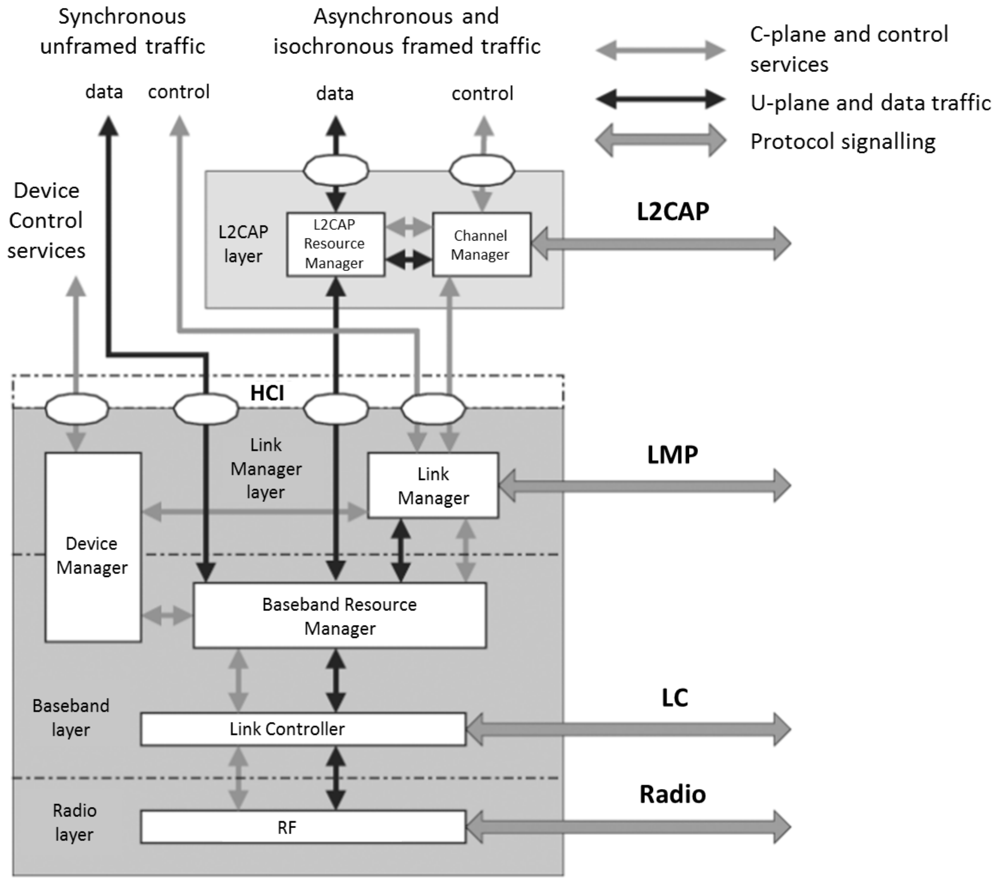


Figure 3.8: Bluetooth core system architecture [47]

Bluetooth works in the globally unlicensed Industrial, Scientific and Medical (ISM) 2.4 GHz short-range radio frequency band and it uses a radio technology called Frequency-Hopping Spread Spectrum (FHSS), which divides the whole frequency spectrum into several hop channels with nominal bandwidth of 1 MHz for each channel. During a connection, Bluetooth transceivers use all 79 channels in the range 2400-2483.5 MHz, and rapidly hop from one channel to another in a random manner across all the channels. Each channel is divided into time slots and only one channel can be used by the transceiver at a time.

To use the Bluetooth wireless technology, a device has to be able to interpret certain Bluetooth profiles, which are definitions of possible applications, and specify general behaviors that Bluetooth enabled devices use to communicate. The way a device uses Bluetooth technology depends on its profile capabilities, so these profiles include settings to parametrize and to control the communication from start. Profiles provide standards which manufacturers follow in order to allow devices to use Bluetooth technology in the intended manner.

Two connectivity topologies are defined in Bluetooth, as presented in Figure 3.9: piconet and scatternet.

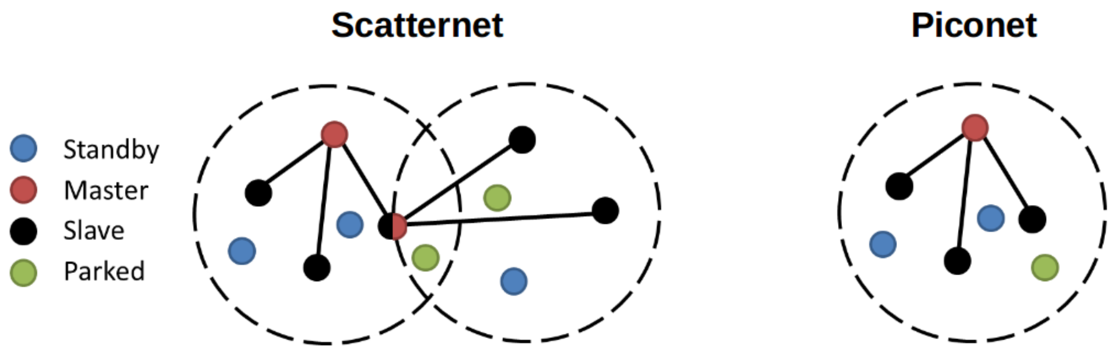


Figure 3.9: Example of a Bluetooth piconet (right) and scatternet (left)

A piconet is a WPAN formed by a Bluetooth device serving as a master and up to seven Bluetooth devices serving as slaves. A frequency-hopping channel based on the address of the master defines each piconet. All devices communicating in a given piconet are synchronized using the master's clock. Slaves communicate only with their master in a point-to-point fashion under its control, while the master's transmissions may be either point-to-point or point-to-multipoint. Also, besides working in active mode, a slave device can also be in the parked or standby modes, allowing a reduction in power consumption.

A scatternet is a collection of operational Bluetooth piconets overlapping in time and space. Two piconets can be connected to form a scatternet. A Bluetooth device may participate in several piconets at the same time, allowing the possibility of information flowing beyond the coverage area of a single piconet. A device in a scatternet could be a slave in several piconets, but master in only one of them.

3.3.2 Security

Like any wireless technology, Bluetooth potentially suffers from a number of serious security vulnerabilities that may compromise the device and those networks to which it is connected. In order to avoid situations like these, Bluetooth provides a set of security features that allow its use in a safest way.

As already referred, Bluetooth technology uses the FHSS technique, in which the devices communicate in a way that the transmission frequencies are alternated in a pre-determined order hopping pattern known only by the sending and receiving devices. Therefore, this technique has become itself a security feature in Bluetooth networks [48]; apart from making the system more reliable when faced with interferences, and to make unauthorized use more difficult, it also add security on data transmission between devices, since it makes it harder to eavesdrop [49].

In order to ensure a secure transmission, a Bluetooth system has defined several keys:

- **Link key:** is used in the authentication procedure and as a parameter to calculate the encryption key:
  - **Unit key, K_A:** the storage of this key requires little memory space and it is often used when the device has little memory or should be accessible to a large group of users;
  - **Combination key, K_{AB}:** is generated for each pair of devices and it is used when more security is needed;
  - **Master key, K_{master}:** is used when the master device wants to transmit to several devices simultaneously. It replaces the original link key temporarily;
  - **Initialization key, K_{init}:** is used as the link key during the initialization.
- **Encryption key:** is derived from the current link key and it is automatically changed each time encryption is needed;
- **PIN code:** may be a fixed number provided with the device or it can be chosen by the user. Its length is usually 4 digits, but it can be anything between 1 to 16 octets.

Bluetooth standard specifies three basic security services: authentication, confidentiality and authorization. Authentication is the process of verifying the identity of communicating devices based on their Bluetooth device address (Bluetooth does not provide native user authentication); the device authentication is accomplished by verifying the knowledge of a secret key — the Bluetooth Link Key. Confidentiality ensures that only authorized devices can access and view transmitted data, and it is accomplished through encryption; Bluetooth has three Encryption Modes [50]: Encryption Mode 1, where no encryption is performed on any traffic; Encryption Mode 2, where only point-to-point traffic is encrypted (broadcast traffic is not encrypted) and Encryption Mode 3, where all traffic is encrypted. It is important to note that only the second and third modes actually provide confidentiality. Finally, the authorization service ensures that a device is authorized to use a service before allowing it to do so. Three different levels of trust between devices are defined, where each level establishes the capacity of a remote device to access a particular Bluetooh service: a *trusted* device indicates that it is an authenticated device and its access to services is allowed; if a device is authenticated but its access to services is restricted, it is classified as *untrusted*; on the other hand, if the device has not been authenticated and it is considered untrusted, it is classified as an *unknown* device.

Bluetooth does not address other security services such as audit, integrity, and non-repudiation; if such services are needed, they should be provided through additional means [50].

Security modes

The family of Bluetooth BDR/EDR/HS (see Figure 3.7) specifications defines four security modes:

1. **Security Mode 1:** if a remote device initiates a security functionality, a Security Mode 1 device may participate, however, it can not initiate a security functionality and it do not employ any mechanisms to prevent other Bluetooth-enabled devices from establishing connections. Security Mode 1 devices are considered non-secure;

2. **Security Mode 2:** allows the initiation of security procedures after link establishment but before logical channel establishment. In this mode it is possible to grant access to some services without providing access to others and it introduces the notion of authorization;

3. **Security Mode 3:** a Bluetooth device initiates security procedures before the physical link is fully established. It is mandatory to authenticate and to encrypt all connections to and from the device;

4. **Security Mode 4:** on this mode (introduced in Bluetooth v2.1+EDR) security procedures are initiated after physical and logical link setup. The device authentication and encryption algorithms are identical to the earlier versions, however, this mode uses Secure Simple Pairing (SSP) [3] instead of LMP pairing (also known as PIN-code based). Security requirements for services protected by Security Mode 4 must be classified as

[3] SSP is a pairing technique that uses a much more elaborate mechanism, known as elliptic curve cryptography, which avoids the use of a Personal Identification Number (PIN) code as part of the Link Key calculation process. Using SSP, two devices will be capable of pairing without the need to transmit any critical information over-the-air, and without the need to share this information by an out-of-band mechanism (such as typing it on a keyboard) [51]

one of the following [50]: authenticated link key required; unauthenticated link key required or no security required.

The Bluetooth architecture allows the establishment of trust relationships in a way that even trusted devices could have access only to specific services. It also allows applications to enforce more granular security policies and, although core protocols can only authenticate devices, user-based authentication is still possible.

3.3.3 QoS

The Bluetooth protocol defines two link layer types: Synchronous Connection Oriented (SCO) and Asynchronous Connection-Less (ACL) links. Once a connection is established between two devices, an ACL link is formed between them, which provides packet-switched communication; it is the most common link used to handle data traffic. A master has the option to change an ACL link to an SCO link, which provides a QoS feature by reserving time-slots for transmission of time-critical information such as voice. With ACL links, Bluetooth can support various IP and non-IP protocol services, which have various QoS requirements, hence the importance to provide them different QoS guarantees.

The ACL link can be configured to provide Quality of Service by means of the HCI QoS Setup command. Through this command the traffic and QoS requirements are specified together with the required service.

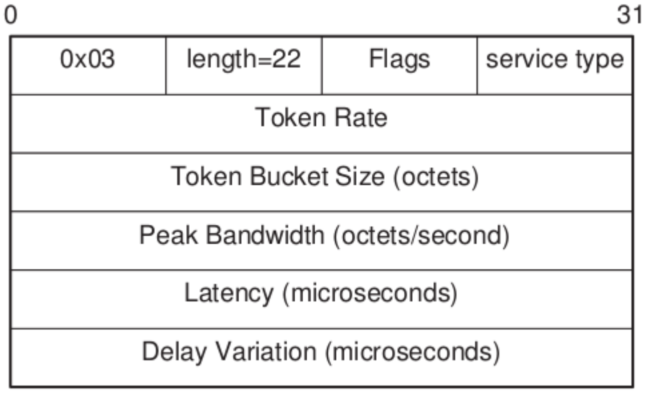
The conveying of QoS information is supported by the L2CAP layer (Figure 3.8). The L2CAP connection establishment process allows the exchange of information regarding the QoS expected between two devices. Each L2CAP implementation monitors the resources used by the protocol and ensures that QoS contracts are honored.

L2CAP channels have an associated QoS setting that defines constraints on the data frames delivery. These QoS settings may be used to indicate, for example, that the data has a limited lifetime after which it become invalid, or that the data should be delivered within a given time period, or also that it should be delivered without error, no matter how long it takes [47].

L2CAP implementations are required to support only best effort services by default and the support of any other service type is optional. If any QoS guarantees are required, then a QoS configuration request shall be sent; Figure 3.10 presents the QoS frame format.

The service type field presented on Figure 3.10 indicates the level of service required. It can be configured to one of the following values:

- **No traffic:** the uplink or the downlink should not transmit;



0			31
0x03	length=22	Flags	service type
Token Rate			
Token Bucket Size (octets)			
Peak Bandwidth (octets/second)			
Latency (microseconds)			
Delay Variation (microseconds)			

Figure 3.10: QoS option format containing flow specification [47]

- **Best effort (default):** does not satisfy any QoS guarantees;
- **Guaranteed:** satisfy the QoS guarantees requested in the QoS frame.

To require some QoS, the Service Type must be configured as Guaranteed Service and the remaining fields of the QoS request must be filled with the appropriated values. If best effort service is selected, the remaining parameters should be treated as optional by the remote device.

The Link Manager (Figure 3.8) can also provide QoS capabilities. For that purpose, a Protocol Data Unit (PDU) may be sent with the number of repetitions for broadcast packets, N_{BC}, which is negotiated between the master and the slave, and a poll interval, T_{poll}. The poll interval is defined as the maximum time between transmissions from the master to a particular slave on the ACL logical transport and it is used to support bandwidth allocation and latency control. These PDUs may be sent at any time after connection setup is completed.

Through the QoS features supported by the Bluetooth standard that were described above, it is possible to provide some guarantees to the data delivery, therefore improving the performance of the system.

3.3.4 Bluetooth Low Energy

The Bluetooth SIG completed the Bluetooth Core Specification version 4.0, which includes the Bluetooth Low Energy (BLE) protocol, and adopted it in July 2010 [7].

Part of the existing Bluetooth design was improved to enable a new technical version of low energy consumption, which provides technical support for a new set of applications. These new features enable Bluetooth devices to operate for months or even years on tiny batteries, which creates new opportunities for manufacturers of Bluetooth devices, as well as for applications' developers. This technology intends to cover a market segment occupied by applications that demand very low-power consumption, since Bluetooth classic was not appropriate for such scenarios. By allowing low duty-cycles, BLE can reduce its power consumption, which makes it able to be used in applications such as healthcare, fitness, security, and home entertainment.

BLE operates in the 2.4 GHz ISM band and defines 40 RF channels with 2 MHz channel spacing, as shown in Figure 3.11. There are two types of BLE RF channels: advertising channels (three) and data channels (thirty seven). Advertising channels are used for device discovery, connection establishment and broadcast transmission, whereas data channels are used for bidirectional communication between connected devices.

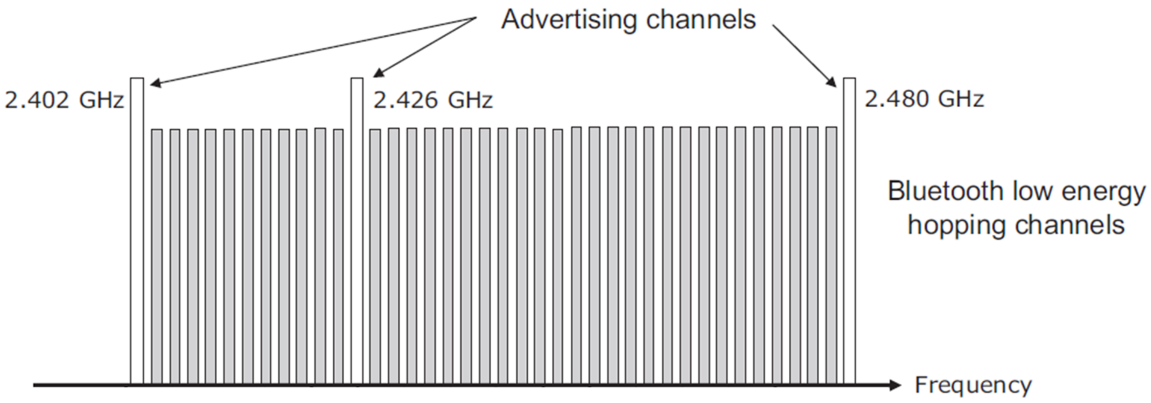


Figure 3.11: Bluetooth LE channels [52]

An adaptive frequency hopping mechanism [53] [54] is used on top of the data channels in order to face interference and wireless propagation issues. It is also responsible for the selection of one of the 37 available data channels for communicating, during a given time interval.

A distinctive feature of Bluetooth LE is the implementation of two types of devices: single mode and dual mode devices. Single mode supports only Bluetooth LE, it consumes less power and it cannot communicate with a device that only implements classic Bluetooth, while in dual mode, devices are implemented with both Classic and Bluetooth LE protocol stacks. Two topologies are defined for both single and dual modes; single mode only supports star topology, while dual mode implements a star-bus topology within a PAN.

Dual mode devices are expected to replace Bluetooth old devices within the next few years (Figure 3.12), while single mode devices are going to be a new type of equipment that requires much lower power consumption than regular Bluetooth and are targeted for control and monitoring applications.

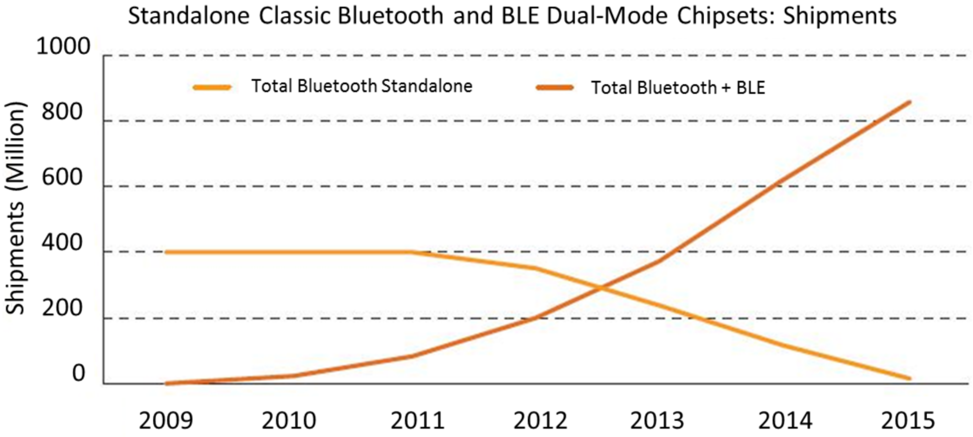


Figure 3.12: World market forecast of standalone Classic Bluetooth and BLE dual-mode shipments [45]

3.4 Technologies comparison

As already discussed, most target markets are characterized by periodic communications between sensor nodes and a central device, in order to exchange small amounts of data gathered by the sensors at a given moment. These applications are all constrained by the following critical key requirements: low-power, low-cost, and small physical size. The low-power requirement is related with the other two in a way that batteries limit either the devices cost and size; this requirement is accomplished through the adoption of technologies that allow low duty-cycles. In some cases, devices may also enter in sleep-mode, by suspend the reception and transmission of information, when they are not communicating, allowing the reduction of their power consumption.

Several technologies are able to perform solutions for the scenarios described. As discussed in subsection 3.3.4, Bluetooth Low Energy is an emerging technology that may play an important role in sensor applications. It may benefit from the widespread use of Bluetooth technology in order to achieve a "quickly" and significant presence in several market segments, however, while it is emerging, other low-power wireless technologies, such as ZigBee (Section 3.2), 6LoWPAN [5], enOcean [55] or Z-Wave [6], have already marked their presence.

Table 3.5 presents a comparison between the main features of ZigBee, Bluetooth and Bluetooth Low Energy. A more detailed comparative study of some WPAN technologies can be found in [56].

Bluetooth is presented as a technology that can achieve high bit-rates, as it can be seen in Table 3.5, and it was developed to be used for short distances. It is intended to connect a small number of devices and it is mainly used in personal devices such as smartphones, desktops and their peripherals, etc. This technology is great to perform the function for which was

Table 3.5: Comparison between ZigBee, Bluetooth and Bluetooth Low Energy [56]

	ZigBee	**Bluetooth Low Energy**	**Bluetooth Classic**
RF band (MHz)	868/915/2400	2400	2400
Bit rate (kbps)	20/40/250	1000	<= 721 (v1.2); 3000 (v2+EDR); <= 24000 (v3+HS)
Spreading technique	DSSS	FHSS (2 Mhz channel width)	FHSS (1 Mhz channel width)
MAC mechanism	TDMA+CSMA-CA (beacon mode) and CSMA-CA (beaconless mode)	TDMA	TDMA
Message size (bytes)	127 (maximum)	8 to 47	358 (maximum)
Error control	16-bit CRC. ACKs (optional)	24-bit CRC. ACKs	8-bit CRC (header); 16-bit CRC and 2/3 FEC (payload). ACKs
Identifiers	16 and 64 bit MAC addresses. 16 bit NWK identifiers	48-bit public device Bluetooth address or random address	48-bit public device Bluetooth address
Device types	Coordinator, Router and End device	Master and slave	Master and slave
Multi-hop solution	Tree and mesh topologies	Not currently supported	Scatternet (routing protocol out of the scope of the Bluetooth specifications)
Security	Integrity, confidentiality, access control (IEEE 802.15.4 security, using 128-bit AES)	Security Modes/Levels. Pairing. Key Generation/Distribution. Confidentiality, Authentication and Integrity	Pairing and Link Key Generation. Authentication. Confidentiality. Trust Levels, Service Levels and Authorization.
Implementation size	45-125 kB (ROM), 2.7-12 kB (RAM)	40 kB aprox. (ROM), 2.5 kB aprox. (RAM)	100 kB aprox. (ROM), 30 kB aprox. (RAM)

initially created, as a cable replacement technology, however, it was not developed to operate in environments with a large number of devices neither with as much energy constraints as WSN scenarios. In order to make its presence in this market segment, Bluetooth developed Bluetooth Low Energy protocol, which has been emerging as a strong low-power wireless technology for single-hop communications. Its low power consumption allows BLE to be used in control and monitoring applications. When compared with ZigBee, BLE can achieve higher bit-rates but it does not allow multi-hop communications, which may represent a problem in some scenarios that require a flexible network topology, in case of some node is not able for transmission.

ZigBee and Bluetooth Low Energy become closer to each other in low power characteristics and the choice of the technology to be used in a certain scenario, must take into account the scenario requirements and the technological features that each one present in order to fulfil those requirements.

3.5 QoS considerations

As it has been discussed, network services have changed dramatically in recent years in order to fulfill the requirements of modern mission-critical, delay-sensitive applications. The described network technologies provide the application's support, however, users are requesting guaranteed and reliable network services for business-critical applications; requirements that may exceed the network resources capacity.

When there is contention of resources, the QoS functions provide service/traffic differentiation. Service differentiation is the ability to provide a "better" service for one traffic flow at the expense of the service offered to another one. The service level is specified by means of QoS parameters such as bandwidth and delay.

Applications with distinct network requirements compete for bounded network resources. It should be avoid that critical applications or services suffer levels of performance degradation during network congestion, that may jeopardize the proper functioning of the system or even the user's safety. A level of QoS guarantees must be kept in order to avoid this type of undesirable situations.

Within the work proposal addressed in this dissertation, the communication between the gateway and an IED has also distinct classes of traffic that should be delivered with different QoS guarantees. This will allow, for instance, a user to act faster and reliably on an IED, in case of an alarm situation.

This way, it is proposed the implementation of a QoS mechanism to provide some guarantees to the data deliver. It is based on the work presented in [57], [58] and [59], and it provides a simple model where traffic is first classified according to its characteristics (bandwidth, priority) and then identified by a single DSCP. The packets will then be forwarded according to the PHB associated with the DSCP contained in the packet header. The proposed architecture is depicted in Figure 3.13.

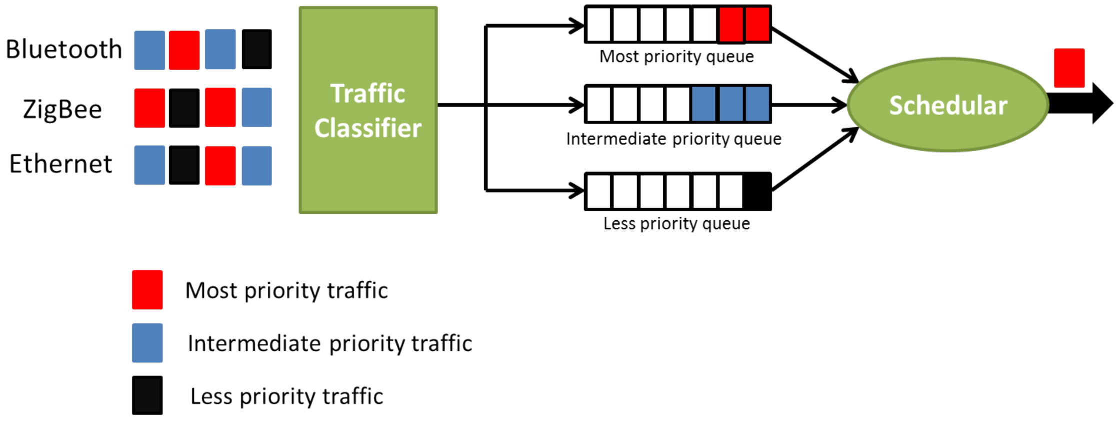


Figure 3.13: Proposed architecture for traffic differentiation

QoS should provide methods of allocating network resources (*e.g.,* bandwidth, priority) so that data gets to its destination consistently and within some time boundaries. A sufficient amount of bandwidth must be allocated for transactional applications and critical real-time traffic, thereby providing an improvement in response time and throughput for critical data. In order to accomplish that, less bandwidth has to be allocated for the remainder less-critical traffic. Thus, before bandwidth allocation can be performed, an identification/classification of all the network traffic must be done. The following classes are suggested to classify the traffic in the proposed scenario of this dissertation:

- **Broadcast messages:** in the proposed scenario, the IEDs send periodic broadcast messages to the gateway. This messages will have the lowest priority and they will be treated as best-effort traffic;

- **Measures requests/responses:** the messages sent when the user asks for a given IED measure will have an intermediate priority;

- **Command and alarm messages:** this class includes command messages sent from the gateway in order to perform some actions on the IED, and alarm messages sent from the IED when a given measure exceeds a preset safety value. This messages are relatively short so, a small bandwidth but high priority channel should be reserved.

The traffic classification into one of the three presented classes can be done through the command field of a frame (*e.g.,* Broadcast messages has the command field 0xA0 and measures requests/responses have the command field 0xD2). Once the traffic is classified, applications with higher priority of traffic get the right to deliver first.

Bandwidth management has to distribute the available bandwidth in a manner that it provides consistent performance and availability for all traffic applications. It is responsible to manage the bandwidth in order to guarantee that critical applications never starve from lack of bandwidth availability. Generally, bandwidth management obtains best performance when it integrates four functions, namely Class Based Queueing (CBQ) , TCP rate shaping, packet-size optimization and an algorithm for fair allocation of bandwidth by connection [57].

CBQ is a method of classifying, allocating and sharing network bandwidth among classes of traffic. After classifying the traffic into classes as described previously, it allocates a given amount of bandwidth to each one of them. Bandwidth management-CBQ guarantees bandwidth for priority mission-critical traffic, while blocking or limiting the bandwidth used by non-priority traffic.

Next it is presented the scenario behavior and the configuration of the QoS guarantees according to the traffic classification suggested above. For this proposal it is considered the use of Bluetooth technology since its standard provide a larger number of QoS mechanisms when compared, for instance, with ZigBee.

Bluetooth

The Bluetooth protocol supports some QoS features at various levels that may be configured in order to accomplish some QoS guarantees (see 3.3.3).

Bluetooth offers ACL and SCO channel types. While SCO links are intended to be used for transmission of “real-time” traffic such as voice (it behaves like a circuit-switched connection), ACL links provide packet-switched communication between a slave and the master.

Within the protocol stack shown in Figure 3.8, it is the Link Manager that configures and controls the baseband links. Bluetooth allows the observation and configuration of some LMP parameters; setting these parameters from higher-level protocols gives a high degree of control but it is time consuming. Therefore, Bluetooth specifies L2CAP layer QoS frame (see Figure 3.10) that can be sent to the Link Manager, who inspects it and decides if rejects the parameters or what channel parameters to set. A QoS setup command is followed by either a setup complete or a QoS rejected HCI event, as shown in Figure 3.14.

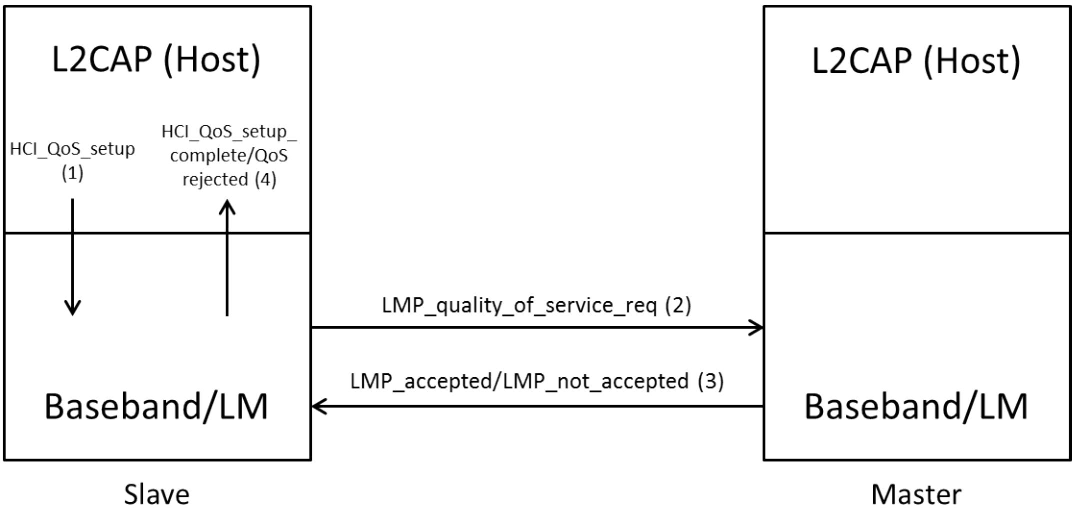

L2CAP (Host)
HCI_QoS_setup (1)
HCI_QoS_setup_ complete/QoS rejected (4)
Baseband/LM
Slave
LMP_quality_of_service_req (2)
LMP_accepted/LMP_not_accepted (3)
L2CAP (Host)
Baseband/LM
Master

Figure 3.14: Link Manager QoS signaling

If the Link Manager that receives the QoS setup command is a master, it sends the new QoS parameters to the slave, which in return has to set its link parameters accordingly. On the other hand, if the QoS setup is initiated by a slave, the master may accept or reject the parameters.

The service type field of the QoS frame has to be configured according to the traffic type and it can be set as no traffic, best effort or guaranteed. The guaranteed service type is based on the token bucket algorithm and it requires the following parameters:

- **Token rate:** the continuous data rate required (bytes/second);
- **Token bucket size:** the maximum burst data that can be sent (bytes);

- **Peak bandwidth:** the maximum data rate equivalent to a continuous transmission (bytes/second);
- **Latency:** the maximum acceptable delay for a packet of data to get from one designated point to another (microseconds);
- **Delay variation:** the variation in time delay between packets (microseconds).

The QoS frame is optional and it can be configured by means of the L2CAP_ConfigReq, when the L2CAP channel is being established. The Figure 3.15 presents the process diagram.

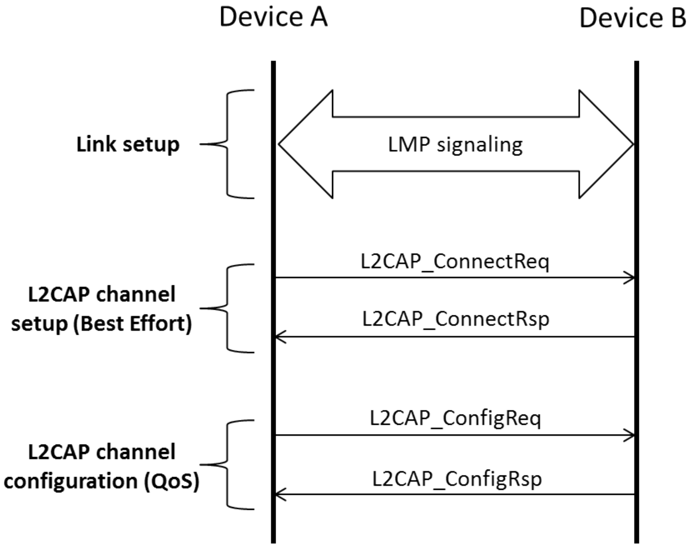


Figure 3.15: L2CAP channel configuration

According to the scenario described, the service type field of the QoS frame corresponding to both command or alarm messages and measures requests/responses, will be set as guaranteed (value 0x02). The broadcast messages will be configured as best effort traffic, thus, the service type must be set with the value 0x01, which is the default value.

In order to prioritize the command and alarm messages compared to measures requests and responses, the remain parameters of the QoS frame has to be carefully chosen. Command and alarm messages must have, for instance, lower latency and delay variation and higher peak bandwidth and token rate.

This way, it is possible to guarantee that the system will respond properly whenever it receives configuration message or it detects an alarm situation occurrence, no matter how much data traffic is flowing in the network at that given time.

4

System definition and development

Classified

5

Gateway evaluation

The present chapter intends to describe in more detail the developed prototype, as well as its proper operation. It will be presented the technology involved on its construction both at the hardware and the software levels. In order to evaluate the overall gateway performance, it will also be presented the results obtained after performing some tests on its individual modules (PAN, server and REST API).

5.1 Software and hardware review

This section presents a quickly overview through the software and hardware used to develop the prototype work presented in this dissertation.

5.1.1 Hardware

Raspberry Pi

The Raspberry Pi is a credit-card-sized, single-board computer developed by the Raspberry Pi Foundation[1] [3] with the intention of promoting the teaching of basic computer science in schools, providing a low-price piece of hardware available for hobbyists and programming enthusiasts.

[1] The Raspberry Pi Foundation is a UK charitable organization founded in 2009 to promote the study of computer science, especially at school level.

The Raspberry Pi presents some characteristics like its processing capacity, low-price and low-power consumption, that makes it a powerful tool to use in M2M applications. The available interfaces allow its connection and interaction with a large number of peripherals.

In the development of this work, the GPIO pins enabled a connection between the Raspberry Pi and a ZigBee communication module, while the RJ45 connector allowed an Ethernet network connection.

The Raspberry Pi fulfills all the proposed work requirements, which along with its powerful technical features (taking into account its small size and low-price) makes it a great hardware to develop the proposed Ethernet-ZigBee gateway.

Telegesis ETRX357 ZigBee Module

All hardware concerning the implementation of networking is based on the ETRX357 ZigBee module from Telegesis [60].

It is a low-power 2.4 GHz ISM ZigBee module that can be easily integrated with other devices to provide a wireless mesh network interface. Its integrated receive channel filtering allows for robust co-existence with other communication standards in the 2.4 GHz spectrum, such as IEEE 802.11 and Bluetooth [61].

This module is controlled using an AT command interface, which allows its configuration in a simple way. The connection between it and the Raspberry Pi is accomplished using a Z-Plate, which is an add-on that enables the Raspberry Pi to use its GPIO pins to establish a connection with the Telegesis ETRX357 module.

The use of these modules was imposed, in order to avoid incompatibility issues with the remaining elements of the system.

5.1.2 Software

Node.js

Node.js [62] is a server-side, JavaScript-based, Web server based on Google's high-performance V8 engine[2] for easily building fast, scalable network applications. It is a single-

[2] The V8 JavaScript Engine is an open source JavaScript engine developed by Google that is intended to be used both in a browser (Google Chrome and Chromium) and as a standalone high-performance engine that can be integrated into independent projects, for example server-side JavaScript in Node.js

thread, event-driven server where requests run in parallel, firing events when done, so they don't block the server (non-blocking I/O model) and they are handled one at a time by a single common process. Thus, the code should be written in order to run asynchronously allowing Node to handle a huge number of simultaneous connections without commensurate Random Access Memory (RAM) requirements [63].

Node provides a variety of tools like database access, file access or process forking, to perform all the essential Web serving duties. It also has a package manager, npm [64], which manages dependencies for an application, making tasks such as the addition of new modules much easier.

Its non-blocking I/O model combined with JavaScript and a large number of functions and modules, such as Express [65], which is a Web application framework for Node.js, makes it an interesting tool for building lightweight REST/JSON APIs. In the development of the proposed work, it was used Node.js v0.8.12 and Express v3.0.0.

Node.js is currently used by a number of large companies including Microsoft, LinkedIn, Walmart, Mozilla and Yahoo! [63].

Munin

Munin [66] is a network monitoring application that presents all information in graphics through a Web interface. It allows monitoring the performance of computers, networks and applications, and it is designed to be very plug and play [66]. Plugins are very easy to crate and, although Munin's framework is written in Pearl, they can be written in any language. After completing its installation, a large number of monitoring plugins will be automatically available.

Munin has a master/node architecture in which the master connects to all the nodes periodically. Once connected with a node, the master asks it for data, stores that data in RRD files[3], and updates the graphs.

This application provides to system managers the possibility to monitor a given system through a set of graphs that display the data variation with time; for each plugin, Munin provides daily, weekly, monthly and yearly graphs. One of Munin major advantages is its simplicity of installation and development of new plugins.

[3] Munin uses the RRDtool which is the OpenSource industry standard, high performance data logging and graphing system for time series data. It includes tools to extract and present data from an RRD file in a customizable graphical format.

5.2 Home gateway testbed scenario

The gateway was implemented using a Raspberry Pi model B revision 1, running the Raspbian wheezy[4] operating system using kernel 3.6.11+; model B revision 1 has a 700 MHz CPU, 256 MB (shared with GPU), a maximum 700 mA current and a power consumption less than or equal to 3.5 W. Connected to the Raspberry Pi, is a Telegesis ETRX357 ZigBee module with the firmware revision number 308C. The Electronic Control Unit (ECU) used to simulate the HVAC system, which can be seen in Figure 5.1, allows the simulation of a Domestic Hot Water circuit. Figure 5.1 presents the global scenario.

Code was developed using Node.js and, as already referred in 5.1.2, it was used the version 0.8.12. All modules should also work with any Node version above 0.7.0, however no tests were performed using other versions.

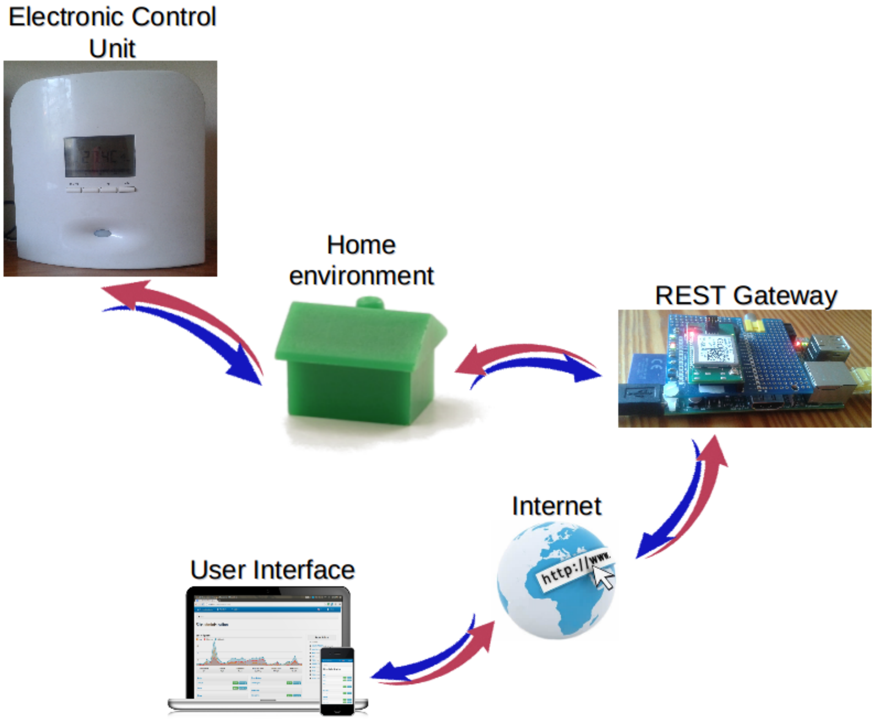


Figure 5.1: Scenario overview

[4] Raspbian "wheezy" is a special version of Debian Linux distribution, optimized for the ARMv6 Raspberry Pi.

5.3 Results

Using the testbed scenario described, it will be presented the results obtained upon performing a series of tests to evaluate the gateway performance. Those tests focused mainly in the setup times and in the HTTP server response times. Whenever it is mentioned a Confidence Interval (CI), it is stated at the 95% confidence level.

5.3.1 Gateway initialization

Whenever the gateway is initialized, it stars both an HTTP and an Hypertext Transfer Protocol Secure (HTTPS) servers, it opens the serial port with the required configuration parameters, and it creates and starts a new PAN where it becomes the coordinator. The start-up times of each process are presented in Table 5.1.

Table 5.1: Start up times

	Start server	Open serial port	Start PAN[a]	Start PAN[b]	Start gateway[a]	Start gateway[b]
Max [ms]	67	11	4445	3274	4523	3352
Min [ms]	63	7	4322	3194	4392	3264
Avg [ms]	64	9	4338	3200	4411	3273

[a] when the gateway is already part of a PAN

[b] when the gateway is not part of a PAN yet

As the results demonstrate, the average time for the gateway to start is 4411 ms if it is already part of a PAN, or 3273 ms otherwise. Both situations can happen because it was established that the gateway is always responsible for starting the network, becoming their coordinator; this way, if the gateway is part of a PAN, it must disassociate itself from that PAN and start a new one. Despite obtaining an average time for starting a new PAN of 3273 ms, according to the manufacturer, this time can take up to 16 seconds.

5.3.2 Recording information in the database

The gateway is responsible for register all relevant information to the system in a database, and to keep that information updated. Some tests were performed in order to evaluate the gateway behaviour when it is faced with those tasks. In order to simulate a large number of ECUs, it was developed a script in Node.js that sends broadcast messages to the gateway; these messages were sent with a period of ten seconds and, according to simulation test, they were configured with different parameters (sender address, system failure notification, temperature measures, etc.).

Registration of a new ECU

Whenever the gateway receives a broadcast message from an ECU that is not registered in the database yet (which can be verified through the existence or not, of the message sender address in the “addresses” table), it proceeds to their register. Fifty new ECU registers were performed and their execution time can be observed in Figure 5.2.

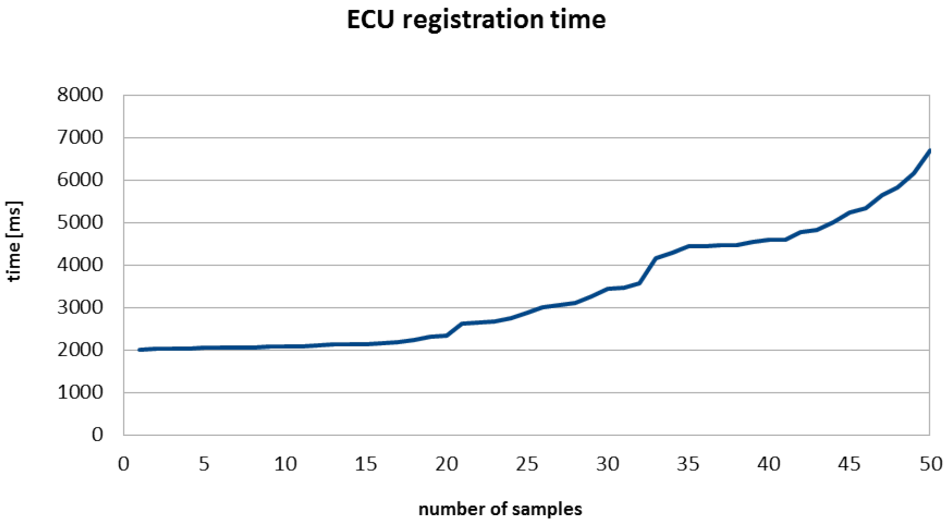


Figure 5.2: Registration time of new ECUs in the database sorted in ascending order

Through the graphic obtained and represented in Figure 5.2, it can be observed that the registration time of a new ECU varied between two and seven seconds, approximately. The same variation order by the time in which the registrations were made (shown in Figure 5.3), led to the conclusion that the number of registrations already in the database, have influence on the time that takes for register new ECUs. This is caused by the increasing amount of information that the gateway has to process in order to create new entries in the database tables; for example, the gateway takes longer to check the existence of the device, if the database has already a large number of devices registered.

Making a statistical analysis of the results, it was obtained a maximum value of 6696 ms, a minimum value of 2008 ms, an average of 3445 ms, and a 95% CI of 3445±365 ms. Also, along with the ECUs registration, it was verified that each one occupied 512 bytes of memory.

The results obtained enabled us to estimate for instance, the minimum time allowed

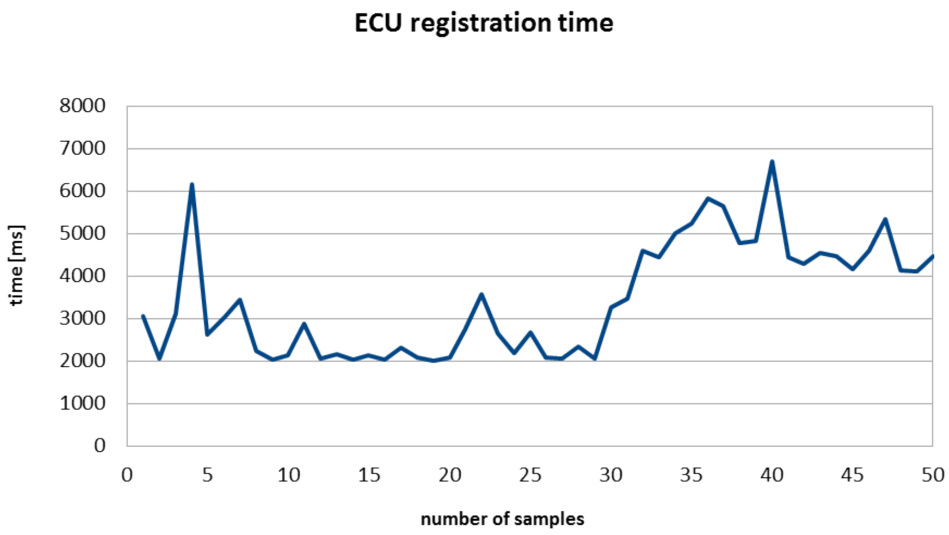


Figure 5.3: Registration time of new ECUs in the database

between two consecutive registers, and the maximum number of ECUs that can be registered in the database.

Temperature measures recording

As already referred, all temperatures transmitted by the ECUs in their broadcast messages, can be periodically recorded in the database. This feature allows the creation of historical records. Figure 5.4 presents the time variation obtained after 100 records; for each one are saved in the database all six temperature measures referred to the ECU.

After a more detailed statistical analysis, it was obtained a minimum time value of 590 ms, a maximum value of 3179 ms, and a CI of 1009±116 ms. Through the analysis of these results, it can be concluded that the time period between recordings must never be less than one second. Within the scenario of monitoring of an HVAC system, these times may not be a problem since the water temperature variation is a slow process. Also, since the recordings are performed according to the reception of broadcast messages, the minimum periodic time between recordings have to necessarily be greater or equal to one second, which is the minimum period value with which the broadcast messages can be configured.

It was also verified that each recording (of six temperature measures), occupied 256 bytes in memory. This value is a reference to know how many recordings are allowed in the database, considering the gateway's free memory.

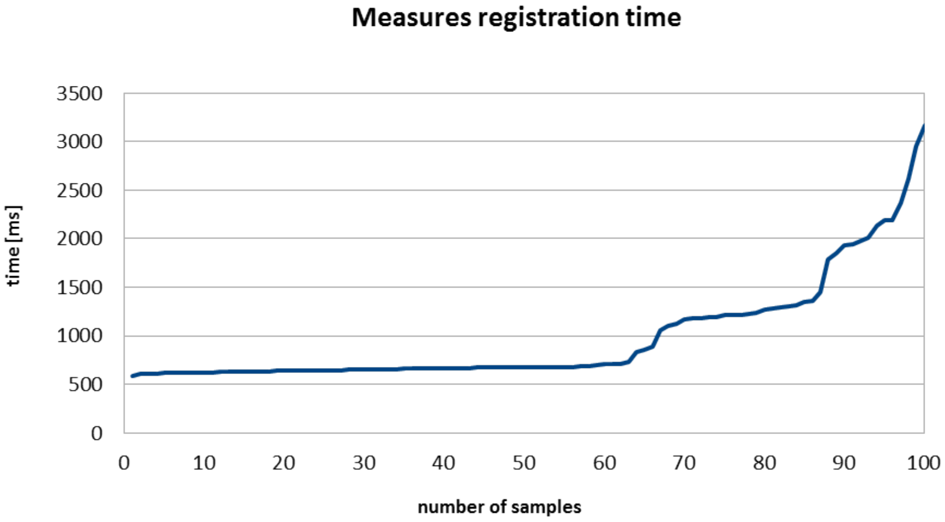


Figure 5.4: Recordings time

ECU's information update

Every time the gateway receives a broadcast message from an ECU that is already registered in the database, and that don't report a system failure, it proceeds to the update of their information. It includes all temperature measures, power consumption, current, and operation mode. The times obtained to update the non-static information about an ECU are presented in Figure 5.5.

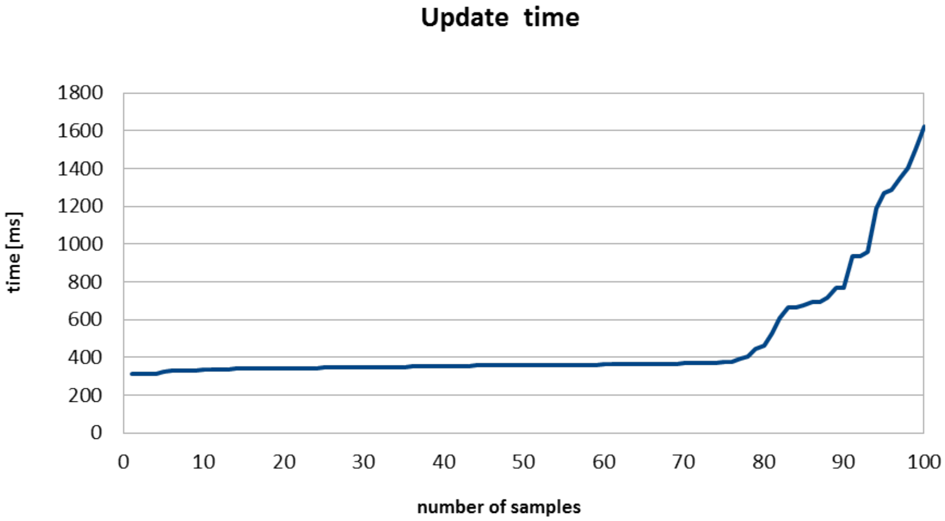


Figure 5.5: ECU's update time

For this experiment, it was obtained a maximum and a minimum time values of 1656 ms and 310 ms, respectively. Additionally, it was obtained a Confidence Interval of 487±60 ms. Through these values, it can be estimated an important parameter, which is the minimum time between broadcast messages. This has a huge impact on the system's proper operation, since it will somehow limit the number of ECUs that the system may support. For instance, the periodic reception of broadcast messages with a time interval less than, approximately 500 ms[5], may result in data loss, which consequently may jeopardize the proper system operation.

Although it was obtained an average time of 487 ms, this may not represent an appropriate value to guarantee the required level of service. Figure 5.6 presents the error rate obtained for different time intervals between broadcast messages, when a recording is active and it is configured with its default value (1 minute). The error rate for each time interval, was obtained from 200 samples.

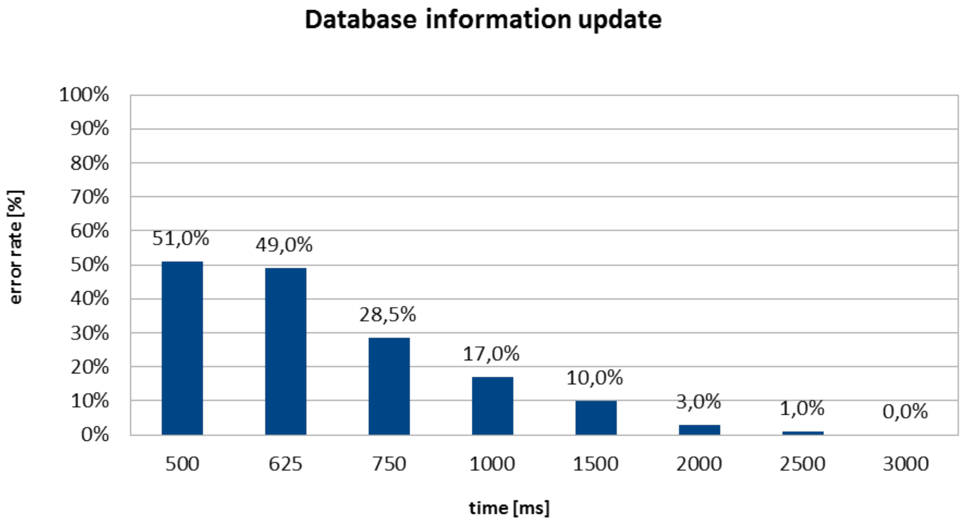


Figure 5.6: Verified error rate for ECU's information update

Through the figure analysis, it can be seen that for a time of 500 ms, it was obtained an error rate of 51%, which represents that in average, for each two received broadcast messages, only one of them updates the ECUs's information in the database. This effect is the result of the gateway not being able to update the ECU's information received in the broadcast messages, because the database is occupied either still performing an information update, or recording the temperature measures. The latter may led to the gateway's inability of performing several database accesses, since it may take more that one second (as already analysed in the present Section).

[5] assuming that ECUs are already registered and that the recordings are disabled

In the Figure 5.6, it is also visible that, as expected, with the time interval increasing, the error rate tends to zero. The error rate decreasing is not more pronounced, because for larger time intervals, the number of times that the temperature measures are recorded increases as well. However, for a 3 seconds interval between broadcast messages, it was verified that all database accesses were successfully performed.

Finally, it can be concluded that, to ensure that the system performs all database accesses, the time interval between the received broadcast messages, should be equal or greater than 3 seconds. It is important to notice that the error rates were obtaining assuming that all ECUs were already registered in the database, otherwise, the occurrence of registration events would increase the obtained values.

Notifications recording

The notifications are recorded in the database whenever an error is reported in the received broadcast messages, even if the ECU reports the same error in more than one message. This enables the system manager to know which error occurred, when it occurred, in which ECU, and for how long. In case a notification is recorded, the measures are not updated, since the failure may have jeopardized their values. The results presented in Figure 5.7 represent the time variation upon the registration of 100 system notifications.

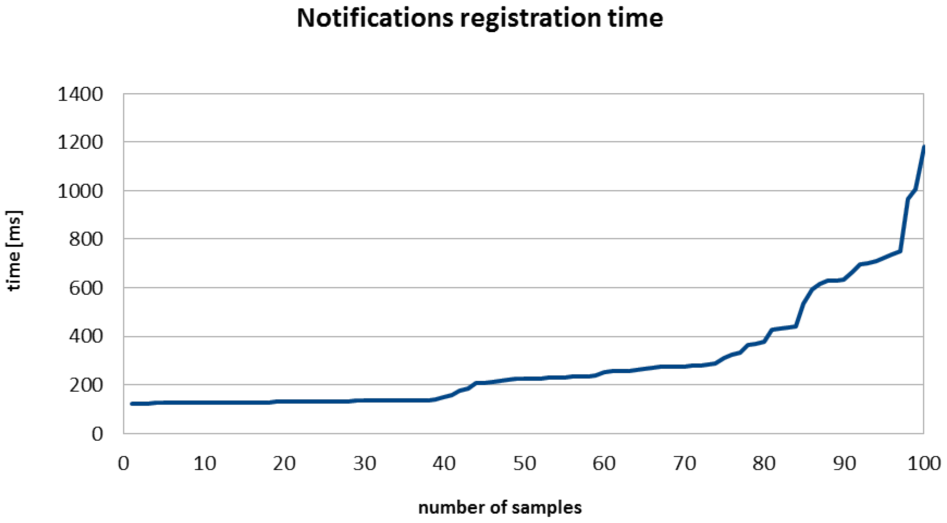


Figure 5.7: Notifications recording time in the database

Through a more detailed statistical analysis were obtained the following values: maximum of 1181 ms, minimum of 122 ms, and a CI of 289±43 ms. Along with those values, it was observed that the gateway memory increases 1 kB after recording 24 notifications; this value corresponds, approximately to 43 bytes of memory per notification.

5.3.3 Server response evaluation

In order to evaluate the HTTP server's response time to different requests, were performed 2000 requests to the resources represented in Figure 5.8. Those requests allowed us to evaluate and to compare the server's response time, when it has to respond with static data, with data that has to be retrieved from the database, and with data that it has to request to an ECU. To perform the described test, it was used the version 2.3 of the ApacheBench (ab)[6] tool.

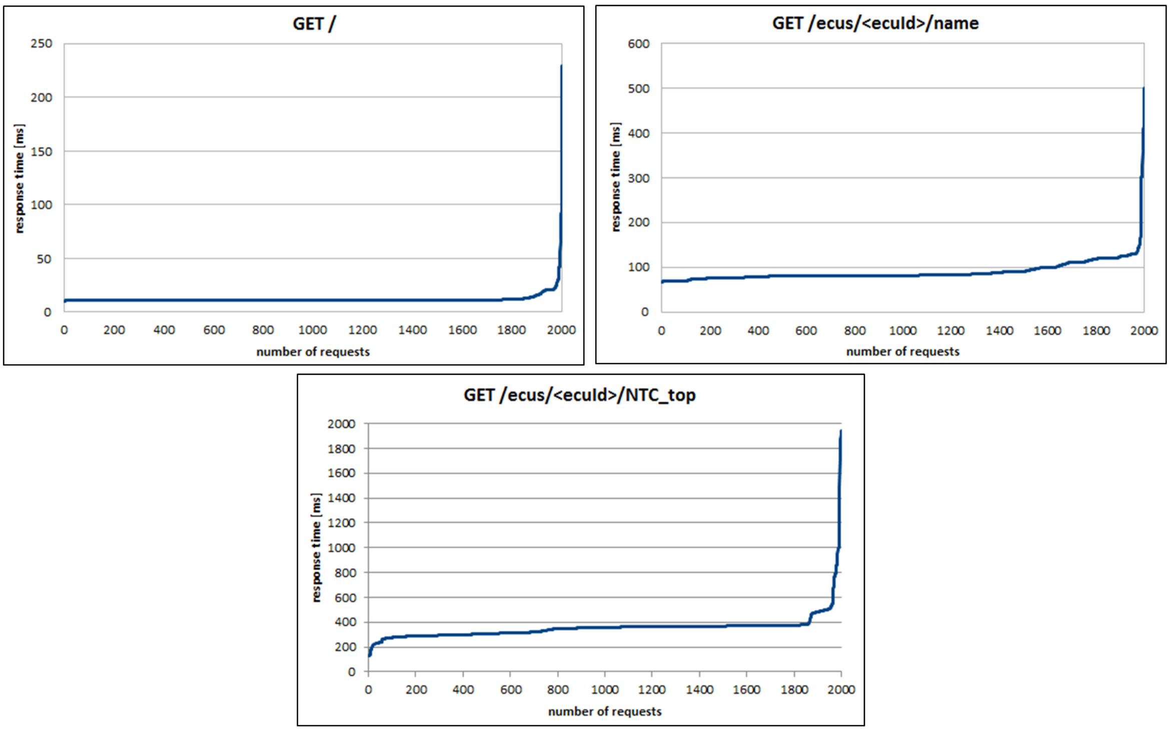


Figure 5.8: Response time of different data requests: static (upper left), from the database (upper right) and requested to the ECU (bottom center)

From the experimental results, were obtained the statistical values presented in Table 5.2. The table, in addition to the obtained values considering all time measures, also presents the results when it is considered that the last 100 values (which correspond to 5% of all samples) are despised. This way, the largest response times, sporadically caused, for instance by the CPU's occupation by other processes, are not considered.

[6] The Apache HTTP server benchmarking tool, also referred to as "ab", is an open-source, single-threaded command line computer program for measuring the performance of HTTP Web servers.

Table 5.2: Statistical analysis' results obtained from: a) all measures; b) 95% of the measures

	GET "/"		GET "name"		GET "NTC_top"	
	a)	b)	a)	b)	a)	b)
Min [ms]	10	10	66	66	130	130
Max [ms]	229	16	500	123	1943	481
Avg [ms]	11.99	11.13	89.72	85.97	355.43	336.80
CI [ms]	11.99±0.31	11.13±0.02	89.72±1.21	85.97±0.59	355.43±5.44	336.80±1.96

Through an analysis of those values, it can be seen that the highest average response time corresponds to a request that requires a communication with the ECU; on the other way, a request to a resource that is only represented by static data (*e.g.,* the root resource "/"), presents the smallest response time.

The results obtained are in agreement with the expected ones, since in the first case (analysing the results represented in Table 5.2 from left to right) the gateway already has all information needed to send the response; in the second case, it has to access the database for data before sending the response; finally, the third request requires that the gateway send a message to the ECU and wait for their answer before it can process and send the response.

5.3.4 System remote monitoring

The gateway has the Munin software configured in order to allow the display of the gathered temperature values in a graphical way. An example of the daily temperature variation is presented in Figure 5.9.

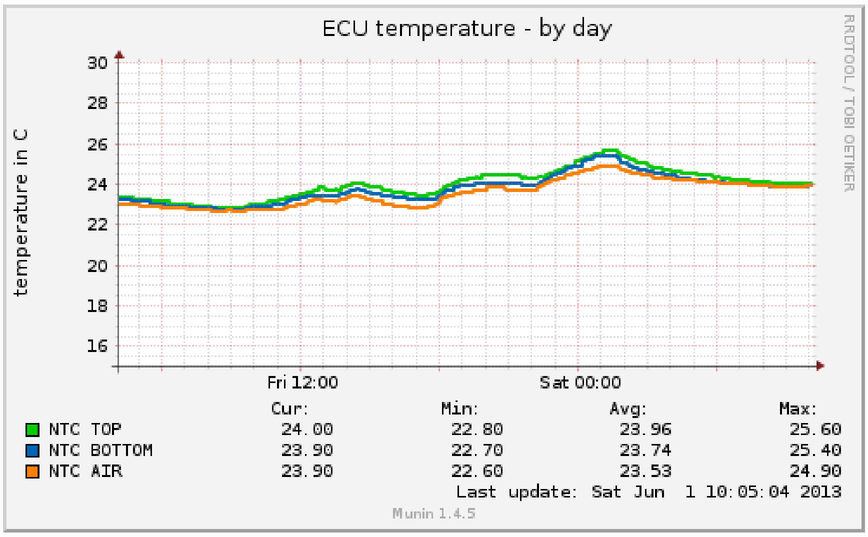


Figure 5.9: Munin graphic that displays the daily temperature variation

In the figure are only represented the measures obtained by the available embedded sensors in the ECU, however, as already referred, a regular ECU transmit six different temperature values from its sensors, in their broadcast messages, which will all be graphically represented by Munin.

The graphical representation of the system temperatures, allows a quicker perception of the current system state, as well as the comparison between the temperatures gathered by the different sensors in a more efficient way. Along with the graphics, it is also displayed the current, the minimum and the maximum values, as well as the overall average. This values represent useful information to a more detailed system status analysis.

Munin provides not only a graphic visualization by day but also by week, by month and by year. This tool allows users or system managers to remotely monitor the proper operation of the system through the Web browser, in a simple and quick way.

6 Final conclusions and future work

Machine-to-Machine communications are of much importance in the Future Internet. The number of devices capable of connecting to the Internet has greatly increased and it is expected that this growth will continue. In the past, devices with Internet connectivity were mainly PCs; now, devices such as smartphones, TVs, refrigerators, vehicles, etc. are part of this global network. Some studies [8] reveal that already are more devices connected to the Internet that people in the whole world. The Internet is no longer a network that is exclusively used by humans to communicate, but also to machines communicate with each other. It is believed that the concept of an Internet of Things will define the Future Internet. In the IoT, “things” are expected to be able to interact and communicate among themselves and with the surrounding environment, while reacting in an autonomous way to “physical world” events. Within this scenario, “things” are also expected to become active participants in business, information and social processes.

Many M2M applications are already implemented and it is believed that their number will enormously increase. This growth has not been bigger due to several factors such as the lack of standards and security issues. Although some work has been done in order to create standards to M2M, they are not mature enough and several market segments demand strong standards to ensure a long-term investment protection. In addition to standardization, some work has to be done in order to provide the exchange of information in a more secure way; some M2M applications are safety-critical and therefore they must be made robust against a large variety of security threats, which demands that security requirements must be precisely understood and developed through standards and certification.

M2M will bring a huge number of market opportunities for various related stakeholders, such as devices and components manufacturers, service providers, and communication network operators. It will also increase the users commodity, allowing them to perform tasks in an easier way and to have access to more detailed information about the current status of the system.

The goal of the present dissertation can be analysed in two parts: firstly the proposal for a REST services gateway and secondly the implementation of this proposal. It was presented and described a solution for the development of a gateway that enables the users remote access to devices through a REST Web server. In the proposed system, “smart” devices have sensing and communication capabilities, which allow them to gather information about the surrounding environment and send it to the gateway. In the gateway, the information is processed and displayed according to the REST principles; users may access this information via the URI that uniquely identifies the resource, using the standard HTTP methods.

In order to verify the correctness of the involved concepts, a functional implementation of this proposal was developed. The prototype implementation used ZigBee technology to enable the gateway to communicate with a real Electronic Control Unit, which allowed the simulation of an HVAC system.

The developed solution proved to be capable of implementing properly the concepts described in the proposal, allowing the remote control and access to information about the HVAC system, through a REST API. This information is formatted as JSON and it can be accessed through the Web browser using the standard HTTP GET method. Along with the HTTP server, through the HTTPS port, it can be configured some system parameters, as well as it can be performed actions on both the server and the remote devices. The gateway is also configured to automatically save all relevant system information in a database, making the overall process more user independent.

The results show that the system’s performance is directly dependent from the time that takes to maintain the database updated (with all ECUs registered and updated, and for instance, with the temperature measures recorded). Those combined times impose the minimum time period between broadcast messages, which if not met, it may result in undesirable situations, such as data loss. It was verified that, with the ECU already registered and without temperature measures to record, the average time to update the information was 487 ms. However, through the error rate analysis, it was verified that this value may not fulfill the service requirements, since it causes an error rate of 51%, when the temperature measures of an ECU are recorded with a periodicity of one minute. This way, it is concluded that for guarantee that the database accesses are always successfully performed, the time between broadcast messages must be close or greater than 3 seconds. Even so, the gateway may still lose some data when a new ECU becomes part of the system, however, since it is a sporadic event, and it is expected to occur most likely at the network’s formation, this may not represent a relevant problem.

Future work may focus on the gateway support of QoS guarantees. One approach may be the implementation of a network gateway that maps Quality of Service between different communication technologies (Ethernet and ZigBee). For instance, configuration messages would have higher priority that the periodic broadcast messages. It would ensure that, no matter how congested the traffic was, these messages would be delivered. Although, it does not meet the objectives proposed for the work presented in this document, the support of QoS guarantees has proven to be of great importance in systems with different kinds of traffic or traffic from different services. For instance, this would also allow, the support of other features, such as an alarm service that notifies the system manager whenever an error is reported.

Glossary

ACL	Asynchronous Connection-Less
AES	Advanced Encryption Standard
API	Application Programming Interface
ASK	Amplitude Shift Keying
BLE	Bluetooth Low Energy
BPM	Burst Position Modulation
BPSK	Binary Phase-Shift Keying
CAP	Contention Access Period
CBQ	Class Based Queueing
CCA	Clear Channel Assessment
CFP	Contention-Free Period
CI	Confidence Interval
CoS	Class of Service
CRC	Cyclic Redundancy Check
CSMA-CA	Carrier Sense Multiple Access with Collision Avoidance
CSS	Chirp Spread Spectrum
DHW	Domestic Hot Water
DQPSK	Differential Quadrature Phase-Shift Keying
DSCL	Device Service Capabilities Layer
DSCP	Differentiated Services Code Point
DSSS	Direct Sequence Spread Spectrum
ECU	Electronic Control Unit
ED	Energy Detection
ETSI	European Telecommunications Standards Institute
FEC	Forward Error Correction
FFD	Full Function Device
FHSS	Frequency-Hopping Spread Spectrum
GFSK	Gaussian Frequency-Shift Keying
GSCL	Gateway Service Capabilities Layer
GTS	Guaranteed Time Slot
HAN	Home Area Network
HCI	Host Controller Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
HVAC	Heating, Ventilation and Air Conditioning
IED	Intelligent Electronic Device
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
ISM	Industrial, Scientific and Medical
JSON	JavaScript Object Notation
L2CAP	Logical Link Control and Adaptation Protocol
LMP	Link Management Protocol
LQI	Link Quality Indicator
LR-WPAN	Low-Rate Wireless Personal Area Network
MAC	Media Access Control

MIC	Message Integrity Code
MPSK	M-ary Phase-Shift Keying
M2M	Machine-to-Machine
NSCL	Network Service Capabilities Layer
O-QPSK	Offset Quadrature Phase-Shift Keying
OSI	Open Systems Interconnection
PAN	Personal Area Network
PDU	Protocol Data Unit
PHB	Per-Hop Behavior
PHY	Physical layer
PIN	Personal Identification Number
POS	Personal Operating Space
PSSS	Parallel Sequence Spread Spectrum
P2P	Peer-to-Peer
QoS	Quality of Service
RAM	Random Access Memory
REST	Representational State Transfer
RF	Radio Frequency
RFD	Reduced Function Device
ROA	Resource-Oriented Architecture
ROM	Read-Only Memory
RSVP	Resource ReSerVation Protocol
SCL	Service Capabilities Layer
SCO	Synchronous Connection Oriented
SKKE	Symmetric-Key Key Establishment
SMTP	Simple Mail Transfer Protocol
SNR	Signal-to-Noise Ratio
SOAP	Simple Object Access Protocol
SSL	Secure Sockets Layer
SSP	Secure Simple Pairing
TC	Trust Center
TCP	Transmission Control Protocol
TDMA	Time Division Multiple Access
ToS	Type of Service
URI	Uniform Resource Identifier
UWB	Ultra Wide Band
VoIP	Voice over IP
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network
XML	eXtensible Markup Language
ZDO	ZigBee Device Object

References

[1] Casaleggio Associati, *The Evolution of Internet of Things*, White Paper, Feb. 2011.

[2] *Arduino - Home Page*, `http://www.arduino.cc/`, Accessed February, 2013.

[3] *Raspberry Pi Foundation - Home Page*, `http://www.raspberrypi.org/`, Accessed February, 2013.

[4] *ZigBee Alliance*, `http://www.zigbee.org/`, Accessed October, 2012.

[5] Z. Shelby and C. Bormann, *6LoWPAN: The Wireless Embedded Internet.* Wiley, 2009.

[6] *Z-Wave Alliance - Home Page*, `http://www.z-wavealliance.org/`, Accessed February, 2013.

[7] *Bluetooth Low Energy - Home Page*, `http://www.bluetooth.com/Pages/low-energy.aspx`, Accessed February, 2013.

[8] D. Evans, *The Internet of Things - How the Next Evolution of the Internet is Changing Everything*, Cisco IBSG, Apr. 2011.

[9] *SOAP specifications*, `http://www.w3.org/TR/soap/`, Accessed March, 2013.

[10] M. Zamora-Izquierdo, J. Santa, and A. Gómez-Skarmeta, "An Integral and Networked Home Automation Solution for Indoor Ambient Intelligence", *Pervasive Computing, IEEE*, vol. 9, no. 4, pp. 66–77, 2010, ISSN: 1536-1268. DOI: `10.1109/MPRV.2010.20`.

[11] "IEEE Standard for Local and Metropolitan Area Networks - Specific Requirements Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)", Tech. Rep., 2011, pp. 1–334.

[12] *Machine-to-Machine communications (M2M); Smart Metering Use Cases*, ETSI, ETSI TR 102 691 v1.1.1, 2010.

[13] *Stan's, Heating and air conditioning*, `http://stanshvac.ca/ottawa-products/`, Accessed May, 2013.

[14] *ETSI - Home Page*, `http://www.etsi.org`, Accessed April, 2013.

[15] *Machine-to-Machine communications (M2M); Functional architecture*, ETSI, ETSI TS 102 690 V2.0.12, 2012.

[16] D. Boswarthick, O. Elloumi, and O. Hersent, *M2M communications : a systems approach.* WILEY, 2012.

[17] *Web Services Glossary - W3C Working Group note 11 february 2004*, http://www.w3.org/TR/ws-gloss/, Accessed March, 2013.

[18] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures", PhD thesis, University of California, 2000.

[19] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, *Hypertext Transfer Protocol – HTTP/1.1*, RFC 2616 (Draft Standard), Updated by RFCs 2817, 5785, 6266, Internet Engineering Task Force, 1999.

[20] A. Rodriguez, *RESTful Web services: The basics*, IBM Corporation, 2008.

[21] L. Richardson and S. Ruby, *RESTful Web Services.* O'Reilly, 2007.

[22] T. Berners-Lee, R. Fielding, and L. Masinter, *Uniform Resource Identifier (URI): Generic Syntax*, RFC 3986 (Standard), Internet Engineering Task Force, 2005.

[23] *SOAP version 1.2 part 1: Messaging framework (second edition)*, http://www.w3.org/TR/soap12-part1/, Accessed March, 2013.

[24] J. Gray and A. Reuter, *Transaction Processing: Concepts and Techniques*, 1st. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1992, ISBN: 1558601902.

[25] B. Upadhyaya, Y. Zou, H. Xiao, J. Ng, and A. Lau, *Migration of SOAP-based Services to RESTful Services*, 2011.

[26] J. Musser, *Open APIs: state of the market*, Glue Conference 2011, Accessed March, 2013.

[27] Y. Zhang, R. Yu, S. Xie, W. Yao, Y. Xiao, and M. Guizani, "Home M2M networks: Architectures, standards, and QoS improvement", *Communications Magazine, IEEE*, vol. 49, no. 4, pp. 44–52, 2011, ISSN: 0163-6804. DOI: 10.1109/MCOM.2011.5741145.

[28] R. Braden, D. Clark, and S. Shenker, *Integrated Services in the Internet Architecture: an Overview*, RFC 1633 (Informational), Internet Engineering Task Force, 1994.

[29] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, *An Architecture for Differentiated Service*, RFC 2475 (Informational), Updated by RFC 3260, Internet Engineering Task Force, 1998.

[30] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, *Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification*, RFC 2205 (Proposed Standard), Updated by RFCs 2750, 3936, 4495, 5946, Internet Engineering Task Force, 1997.

[31] *DiffServ — The scalable ent-to-end Quality of Service model*, Cisco Systems, White paper, 2005.

[32] Y. Lin, *Internet QoS: IntServ and DiffServ*, May 1999. [Online]. Available: http://speed.cis.nctu.edu.tw/~ydlin/course/cn/part2/InternetQoS.pdf.

[33] *IEEE 802.15*, http://www.ieee802.org/15/, Accessed October, 2012.

[34] *ZigBee members*, http://zigbee.org/imwp/web3/reports/ecosystems.aspx, Accessed October, 2012.

[35] S. Farahani, *ZigBee Wireless Networks and Transceivers.* Elsevier, 2008.

[36] *ZigBee Specification 053474r17*, ZigBee Alliance, available from www.zigbee.org, Jan. 2008.

[37] National Institute and Technology of Standards, "Advanced encryption standard", *NIST FIPS PUB 197*, 2001.

[38] *ZigBee Specification 053474r13*, ZigBee Alliance, available from www.zigbee.org, Oct. 2006.

[39] V. Kumar and S. Tiwari, “Performance of Routing Protocols for Beacon-Enabled IEEE 802.15.4 WSNs with Different Duty Cycle”, in *Devices and Communications (ICDeCom), 2011 International Conference on*, 2011, pp. 1–5. DOI: 10.1109/ICDECOM.2011.5738549.

[40] B. Nefzi and Y. Song, *Performance analysis and improvement of ZigBee routing protocol.*

[41] G. Kaur and K. Ahuja, *QoS Measurement of ZigBee Home Automation Network using Various Routing Protocols*, International Journal of Computer Applications (0975–8887) Volume 13, No.2, 2011.

[42] ——, *QoS Measurement of ZigBee Home Automation Network using Various Modulation Schemes*, International Journal of Engineering Science and Technology (IJEST), 2011.

[43] M. Leopold, M. B. Dydensborg, and P. Bonnet, *Bluetooth and Sensor Networks: A Reality Check*, 2003.

[44] *Bluetooth - Home Page*, http://www.bluetooth.com/Pages/Bluetooth-Home.aspx, Accessed February, 2013.

[45] M. G. Zanchi, *Bluetooth Low Energy*, LitePoint Corporation, 2012.

[46] *Bluetooth SIG*, https://www.bluetooth.org/apps/content/, Accessed February, 2013.

[47] IEEE Std, “Part 15.1: Wireless Medium Access Control (MAC) and Physical layer (PHY) specifications for Wireless Personal Area Networks (WPANs)”, Tech. Rep., 2005, pp. 1–580.

[48] K. Morsi, X. Huagang, and G. Qiang, “Performance estimation and evaluation of Bluetooth frequency hopping selection kernel”, in *Pervasive Computing (JCPC), 2009 Joint Conferences on*, 2009, pp. 461–466. DOI: 10.1109/JCPC.2009.5420142.

[49] S. Bosworth, E. Kabay, and E. Whyne, *Computer Security Handbook.* Wiley, 2012, ISBN: 9780470413746. [Online]. Available: http://books.google.pt/books?id=2yPcGF5HhaoC.

[50] J. Padgette, K. Scarfone, and L. Chen, *Guide to Bluetooth Security*, NIST, Jun. 2012.

[51] Ellisys, *Secure Simple Pairing Explained*, Ellisys Bluetooth Expert Notes, 2012.

[52] N. Hunn, *Essentials of Short-Range Wireless.* Cambridge University Press, 2010.

[53] J. S. Mander, D. Reading-Picopoulos, and C. Todd, *Evaluating the Adaptive Frequency Hopping Mechanism to Enable Bluetooth – WLAN Coexistence.*

[54] N. Golmie, O. Rebala, and N. Chevrollier, *Bluetooth Adaptive Frequency Hopping and Scheduling.*

[55] *enocean - Home Page*, http://www.enocean.com/en/home/, Accessed February, 2013.

[56] C. Gomez, J. Oller, and J. Paradells, “Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology”, *Sensors 2012*, 2012.

[57] W. Hwang and P. Tseng, “A QoS-aware residential gateway with bandwidth management”, *IEEE Trans. on Consum. Electron.*, vol. 51, no. 3, pp. 840–848, Aug. 2005, ISSN: 0098-3063. DOI: 10.1109/TCE.2005.1510493. [Online]. Available: http://dx.doi.org/10.1109/TCE.2005.1510493.

[58] P. Tseng and W. Hwang, “Toward the ubiquitously networked society: QoS-aware residential gateway with ZigBee-based network”, in *Wireless and Pervasive Computing (ISWPC), 2011 6th International Symposium on*, 2011, pp. 1–6. DOI: 10.1109/ISWPC.2011.5751322.

[59] B. Lei, A. L. Ananda, and T. S. Teck, “QoS-Aware Residential Gateway”, in *Proceedings of the 27th Annual IEEE Conference on Local Computer Networks*, ser. LCN ’02, Washington, DC, USA: IEEE Computer Society, 2002, ISBN: 0-7695-1591-6. [Online]. Available: http://dl.acm.org/citation.cfm?id=648047.788597.

[60] *Telegesis Home Page*, http://www.telegesis.com, Accessed March, 2013.

[61] *ETRX3 ZigBee Module*, http://www.telegesis.com/product_range_overview/etrx3_zigbee_module.htm, Accessed March, 2013.

[62] *Node.js - Home Page*, http://nodejs.org/, Accessed January, 2013.

[63] *Using Node.js to write RESTful web services*, http://www.openlogic.com/wazi/bid/267042/Using-Node-js-to-write-RESTful-web-services, Accessed March, 2013.

[64] *Npm - Home Page*, https://npmjs.org/, Accessed January, 2013.

[65] *Express - Home Page*, http://expressjs.com/, Accessed January, 2013.

[66] *Munin - Home Page*, http://munin-monitoring.org/, Accessed March, 2013.