POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Elettrica

Tesi di Laurea Magistrale

Design of a Test Bench to Emulate the Control of Smart Electrical Loads Using a Smart Meter

Relatori

Prof. Maurizio Repetto

Prof. Jean-Claude Maun

Ing. Benoit Mattlet

Candidato

Claudio Alberti

Ottobre 2017

Abstract

Nowadays, the power grid is called to face new challenges. The total energy consumption is rising steadily by several percent per year, as a consequence of demographical growth and a constant process of electrification of devices. Indeed, as fossil fuels become more expensive and consumer awareness grows, the rate of use of electric energy will rise accordingly, also in the fields of transport and heat production.

Moreover, as consciousness on environmental risks caused by pollution has risen, Countries have become more and more concerned about fostering energy generation from renewable sources. All these factors contribute to augment the complexity of the power system management, since distribution system operators (DSOs) have to face more variable and less predictable power flows.

A transition towards actively managing, monitoring and controlling the distribution network is needed, in order to enhance the integration of the distributed generation capacity and permit the steep increase in energy consumption. This process entails the development of reliable communication infrastructures, as a real-time coordinated control of demand and supply requires.

Furthermore, a well-designed layer of intelligence necessitates tight integration of smart meters, smart devices and distribution transformers, requiring an interoperable communication infrastructure that employs standardized protocols .

In this context, this master thesis targets the design of a test bench that could emulate the control and monitoring of smart electrical loads, the aim being to bring benefits to both the consumer and the distribution grid. The project include the design of a controller and the creation of a communication network.

Contents

List of figures

List of acronyms

API Application Programming Interface

APL Application layer

BAN Building Area Network

CCA Clear Channel Assessment

CP Constant Price

CS Carrier Sense

CSMA-CA Carrier Sense Multiple Access with Collision Avoidance

DA Distribution Automation

DAP Day Ahead Pricing

DP Dynamic Pricing

DR Demand Response

DSL Digital Subscriber Line

DSM Demand Side Management

DSO Distribution System Operator

ED Energy Detection

EM Energy Manager

EMS Energy Management System

EV Electric Vehicle

FAN Field Area Network

FFD Full Function Device

HAN Home Area Network

HEMS Home Energy Management System

IAN Industrial Area Network

IP Integer Program

LAN Local Area Network

LP Linear Program

MAC Medium Access Control layer

MILP Mixed Integer Linear Program

NAN Neighborhood Area Network

NWK Network layer

PAN Personal Area Network

PAR Peak to Average Ratio

PHY Physical layer

PLC Power Line Carrier

RTP Real-Time Pricing

SG Smart Grid

SLP Synthetic Load Profile

SM Smart Meter

SO System Operator

TB Test Bench

TOU Time Of Use

TSO Transmission System Operator

WAN Wide Area Network

WiMAX Worldwide Interoperability for Microwave Access

WLAN Wireless Local Area Network

1 Introduction

1.1 Background

Environmental concerns, demographical growth and the constantly increasing electrification of transports and devices in general, constitute nowadays a call for the exploitation of renewable forms of energy. Indeed, traditional primary resources result unsuitable to accompany the foreseen increment of energy demand if, in parallel, the goal is to abate the emission rate of polluting agents. Many are the initiatives that various countries are undertaking in order to contrast pollution while trying to keep high level of life quality for people [1]. Most of these initiatives involve the increase of the adoption of sustainable forms of energy, so renewable resources[2].

However, as well known, renewable sources of energy furnish an highly unpredictable generation capability since, for most of them, power production is intimately tied with weather conditions. Furthermore, this type of generation necessarily entails that the unities of production were distributed all over the territory, and this goes against the traditional vision of the power system functioning. Therefore, the integration of such a distributed and unpredictable generation represents a great challenge for the power grid administration, which needs to be kept safe and reliable even in such an unsettled framework.

In this sense many studies investigate the benefits can be obtained in terms of efficiency when flexibility options are introduced in the power system in order to dynamically match demand and generation of electricity [3][4]. Among those, a solution is represented by the adoption of Demand Side Management (DSM) programs, above all Demand Response (DR), by which the electrical loads become actively involved in the power system balancing operations.

1.1.1 Environment for the EMS

The EMS works in full synergy with the environment in which it is located. Focusing on a domestic framework, EMS coordinates the functioning of the so-called *smart appliances (SA)* and furnishes an user interface to communicate with the residents as well. The SAs are electric devices which have been made adapted to be managed remotely by an EMS. The communication between them is realized through a *Home Area Network (HAN)*.
Enlarging the focus outside the house, EMS is supposed to exchange information with the utility. This in order to provide useful information for the management of the distribution grid. This time the communication requires a wider coverage, and thus it will be a *Neighborhood Area Network (NAN).*
Further information about the communication protocols will be provided in Chapter 3. The working environment of the EMS is depicted in Fig.3:

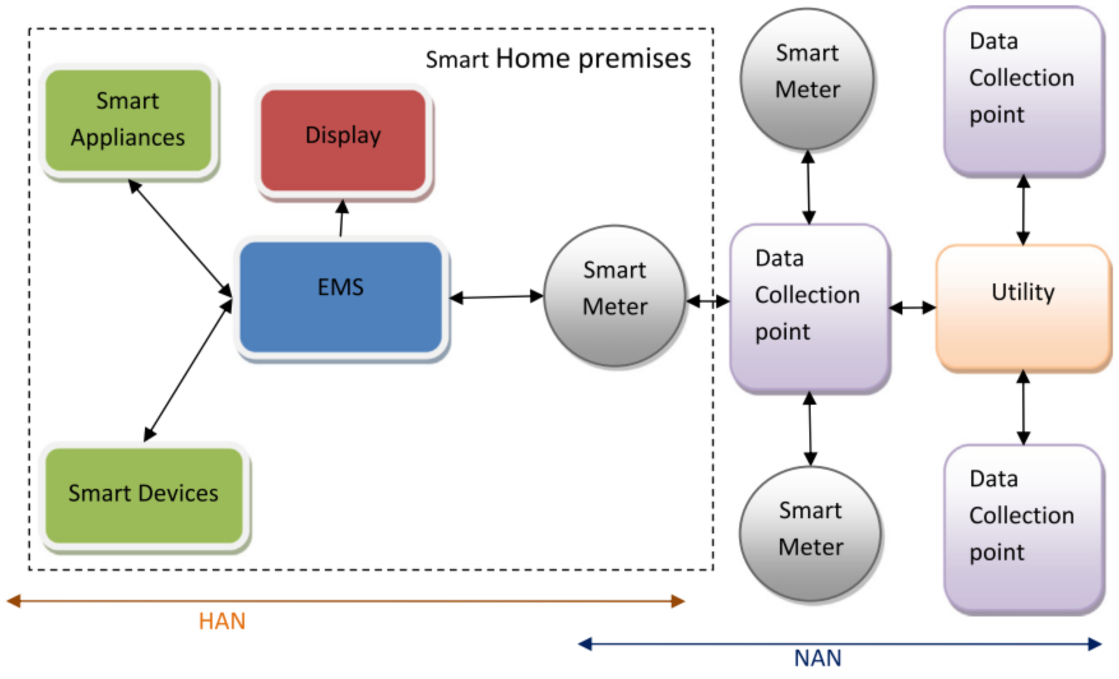


Figure 1: the EMS environment [5]

1.2 Thesis objective

The realization of DR programs entails a close cooperation between the various element of the grid. As complexity of control and management systems of the power grid rises, new, rapid, reliable and effective communication structures are needed.
In this context, this thesis aims the realization of a Test Bench (TB) to support the real-time simulation of possible scenarios of smart electrical loads control, in a DSM perspective. The TB exploits the optimization formulation previously developed in [6] and gives to it a physical, casual and real-time implementation.
Particularly, the attention is focused on the smart management of charging stations for Electric Vehicles (EVs) which, thanks to their characteristics, potentially offer to the DSO opportunity of flexibility for the implementation of DR programs. In this regard, the objectives of the optimization are two: minimizing the costs of the recharge and flatten the power profile delivered by the MV/LV transformer that supplies the charging stations.
The TB designing involves both a software and a hardware part. The first one focuses on the programming of a code that reliably executes the functions of creating the scenarios, handling the simulations in real time and performing the optimizations. The second one instead involves the physical realization of a communication network that allows robust interoperability between the diverse entities of the TB.

1.3 Thesis outline

This paper is organized in this way:

- Chapter 2 introduces the Energy Management System (EMS). Going through a short overview on the state of the art of DSM applications it arrives to a TB structure formulation
- In Chapter 3 it is discussed the creation of the communication network. Firstly they are presented the main issues related to the implementation of Smart Grids (SG), and then they are discussed the choices of the communication protocol and the radio modules for this project. Furthermore, they are illustrated all the countermeasures taken to

strengthen the reliability of the communications, in order to contrast the problem of the transmission failure. Lastly the network topology chosen for the TB is depicted

- In Chapter 4 it is deeply described the design of the test bench. The working principle of the main entities and their reciprocal interations are delineated. They are stated all the assumptions and the algorithms implemented to create the scenario and to reliably perform the optimizations in real time. The methods to set the various parameters are delineated as well
- After having justified the choice of the mathematical tool, Chapter 5 presents the formulation of the optimization problem. Therefore it describes the variables, parameters and constraints used. Lastly they are shown and commented the results coming from a typical day of simulation
- Finally, Chapter 6 draws the conclusions of the work and discusses possible future developments of the project.

2 Energy Management System

Nowadays, technologies to store electric energy are not yet enough developed to permit a practical and cost-effective implementation [7]. As a consequence the electricity produced and delivered to the grid has to be consumed instantaneously. Thereby it arises the need to control, in real-time, that the balance between the quotas of power injected and withdrawn from the grid were respected. The challenging role of managing such a control is responsibility of the System Operators (SO), that act both at the transmission (TSO) and distribution levels (DSO).
As the complexity of power injections and withdrawals framework increases, the difficulties for SOs rise as well (Fig.1). Such a complexity augmentation is highly predictable, since, as aforementioned, the production is always more and more oriented towards the exploitation of renewable sources.

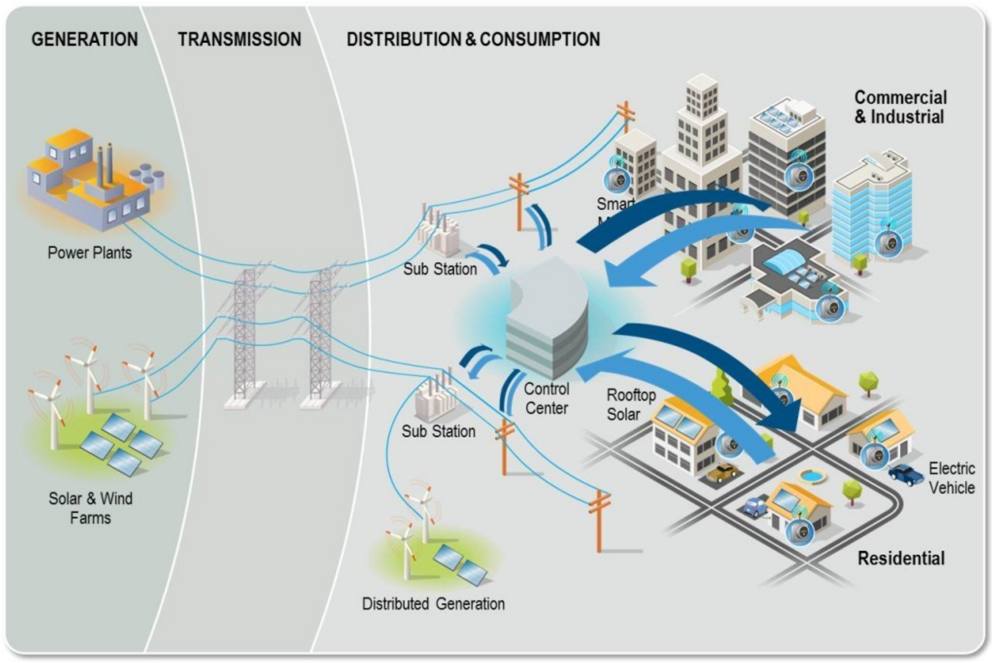


Figure 2: smart grid environment for SOs control [8]

The traditional concept of power grid must be revolutionized in order to permit to these unpredictable and distributed generation units the participation to the electricity market [9]. Indeed, if historically the power market have been based on the supply-side viewpoint, the arising idea nowadays is to get more involved the demand-side in the balancing operations [4].

2.1 Demand Side Management

The studies in literature that investigate the benefits of DSM are already numerous. It is to register a general agreement on the fact that when loads provide flexibility on their consumption patterns, they can be obtained advantages in optimally scheduling their functioning.
It is possible to cluster the main profits attainable with DSM programs as [10]:

- *Load levelling*: the electricity dispatch can be performed more easily when the load profile is flat. Furthermore, the power system facilities (lines, transformers, etc.) can be undersized respect to the present ones, as the possibilities to incur in demand peaks decrease
- *Electric transport:* the development of electric vehicles can be fostered as the power consumption of these can be optimally scheduled
- *Participation on the ancillary service market:* the load flexibility can surely constitutes a resource for the SOs. Therefore, contrary to the traditional way to conceive the power system, they can be provided services also in the opposite direction from the load to the grid. Thus, such services can be sold on the regulation market
- *Cut on the electricity cost:* when peaks of electricity demand are reached, in order to satisfy the request the supplier have to turn on also the less efficient power plants. Since these entail higher production costs, this action, although unavoidable, implies an increase on the cost of electricity. As the peak demand periods are supposed to be drastically shrunk, the electricity price is likely to be diminished consequently.

Although previsions are florid, a success on the actualization of DSM cannot be attained regardless of a change on the behaviour of customers. Indeed, if no profits are foreseeable people might be reluctant to switch from the well-established way of energy provision to the new one.

2.1.1 Dynamic Pricing

The study conducted in [6] individuates the current structure of the retail market of electricity as the main obstacle to the advancement of DSM programs. Indeed, customers are used to stipulate contracts with suppliers on a fixed tariff per megawatt hour basis and they are not incentivized to modify their habits.

According to [11] an active involvement of the demand can be achieved by adopting *Dynamic Pricing (DP)* methods, such as *Real-Time Pricing (RTP).* The latter envisages to make the retail prices of electricity dynamically linked to the real-time ones established by the wholesale market. This pricing method would allow revealing the underlying cost of electricity, making people more concerned about provision costs oscillations. In the meanwhile, also the electricity market would improve in efficiency.

However, two main factors prevent the implementation of DP:

1) skepticism about the *responsiveness of consumers* to DP, above all if this is equipoised with the implementation costs of such programs
2) *lack of metering, billing and communication systems* adequate to support DP [12].

2.1.2 Demand Response

DR is among the DSM programs. It is defined as the consumption behaviour that changes its patterns in response to DP structures or payments of incentives, the aim being the efficient and reliable operations of the electric system [13].

Therefore, there exist two groups of DR programs that can be distinguished (Fig.2). They differ by the manner the consumer is involved in its consumption patterns modification :

1) *incentive-based:* in this group lie those programs that involve payments to the consumers for reducing their power consumptions during critical hours. Hence, they are involved on the basis of contractual agreements with the supplier
2) *price-based:* this category includes those programs that adopt DP. Time-varying tariffs pull the consumers shift their energy demand from high price hours (peak demand periods) to low price hours (out-of-peak times). In this manner, consumers voluntarily adapt their consumptions depending on their rational and economic preferences.

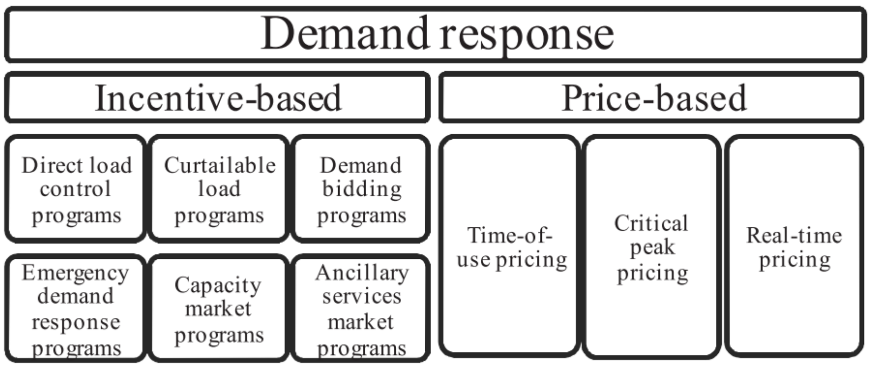


Figure 3: DR types of programs[11]

For the purposes of this thesis only *price-based DR* programs are of interest. Indeed, as it will be explained later, one of the optimization performed by the TB entails a rescheduling of the electrical loads based on the research of the minimum cost solution.

2.1.3 Types of DP for price-based DR

In this paragraph they are listed the possible DP tariff methods [14]. It is worth to remember that all of them vary in time reflecting the variations of the wholesale market prices. Therefore one has:

- *Real-Time Pricing (RTP):* with this method the electricity price of each hour in a day is known one hour in advance
- *Time Of Use (TOU):* this method is similar to a bi-hourly tariff. It differs because of the fact that in TOU the day is divided in more than 2 periods to which different tariffs apply. The diverse tariffs are assigned depending on the hours of the day they refer to, the type of day (weekend/week), the season, etc. and are normally established for a long time period (i.e. several weeks)
- *Critical Peak Pricing (CPP):* it is substantially a TOU method but with the difference that price during periods of demand peak can be modified in time. Normally, whenever a peak is foreseen, the consumer is warned a day in advance that price will be changed on a specified peak time.

However, these concepts will be taken up in Chapter 4, where it will be more thoroughly discussed how electricity prices are formed within the simulations.

2.2 Home Energy Management System

The Home Energy Management System (HEMS) is an intelligent system that performs monitoring, planning and controlling functions on electricity utilization within domestic environment [15]. Therefore, thanks to its functions the HEMS enables the actuation of DR programs within premises.

The TB realized in this thesis performs the same functions of a HEMS but, as it actually refers to a numerous set of households, it has been called simply Energy Management System (EMS).

At this point, it is worth to highlight the difference existing between a real world implementation of the EMS and the emulator developed in this thesis.

Indeed, as a real world implementation was not among the purposes of this work, the entire environment the EMS refers to has been created 'virtually' within the software. Then, the only physical realization of the TB is the communication network: all the electric loads (the EVs in this case) are simulated by computers and are allowed to talk to the EMS by means of radio modules. Since all the computers are operating within a household environment, the EMS actually operates as a HEMS.

This implies that all the communication protocols and radio devices have been chosen to operate in an in-home environment, as it will be deeper discussed in chapter 3, being the application actually a HEMS. Nevertheless the acronym EMS is maintained throughout the paper in order to do not lose the viewpoint of the real world implementation.

2.3 A model of the EMS environment

In this paragraph it is described the residential area where the EMS is supposed to work. They are envisaged three agglomerates of households called *block areas*, each comprising both habitations and a parking station for EVs (Fig.4). In each block area they are present 64 households and the same amount of vehicles. In this environment a price-based DR program is performed, since it suits well for the adaptation of the consumption patterns in response to price signals.

Indeed, EVs require considerable quantities of energy, and managing their charge by optimally scheduling the power delivered is very important .

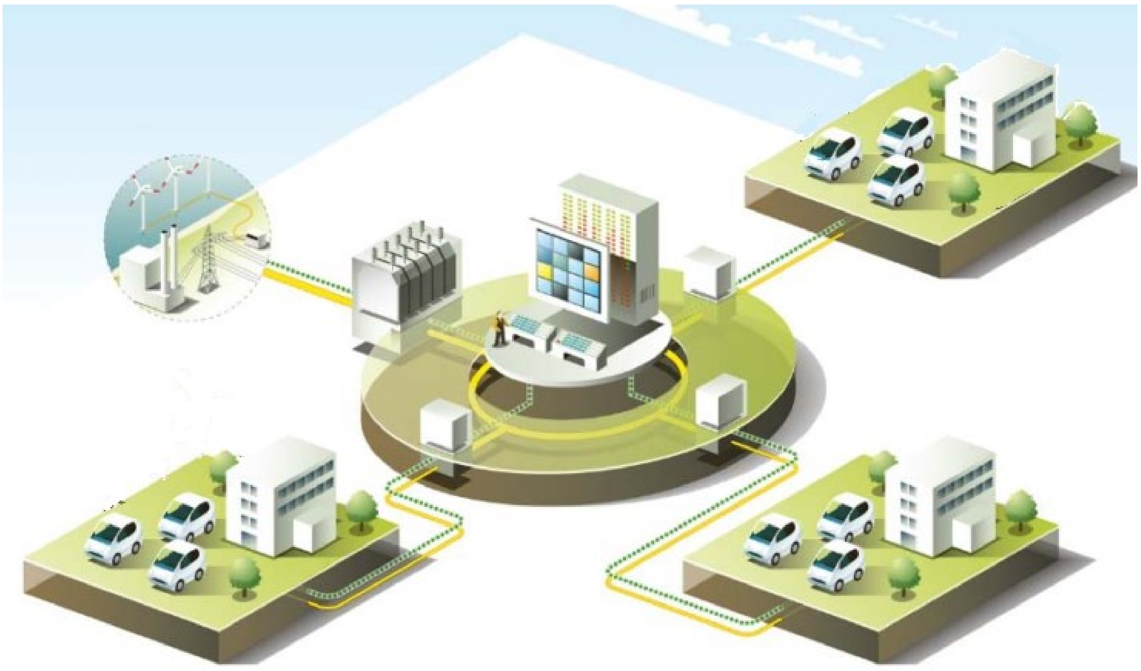


Figure 4: an overview of the block areas [6]

All the agglomerates are supplied by the same MV/LV transformer and managed by the same *Energy Manager (EM).* This controls the entire area and, to fulfill its functions, it can counts on information coming from the power grid (electricity prices, emergency signals, etc.), the charging stations (energy requests, security signals, etc.) and the households as well.

Yet, every parking is handled by another entity called *Aggregator*, which does not have a control role but acts simply as a data collector. Its duties are to cluster the energy requests of the EVs belonging to the related fleet and to send them to the EMS. The Aggregator behavior has been simulated by the software, although its communication with the EM has been physically realized via radio.

The information exchange between EMS and houses regards the consumptions and scheduled power profiles, and it is made possible by the installation of a smart meter on each on them. Therefore, it is foreseen that every house in the block areas is managed internally by an own HEMS that controls the in-home appliances, whereas externally the EMS, thanks also to the support provided by the smart meters, clusters the energy requests of the EVs and performs an optimal charge for them.

The communication patterns between EMS and the smart meters of each house has not been investigated in this work, since the behavior of the households has been virtually simulated. The communication routing principle is shown in Fig.5:

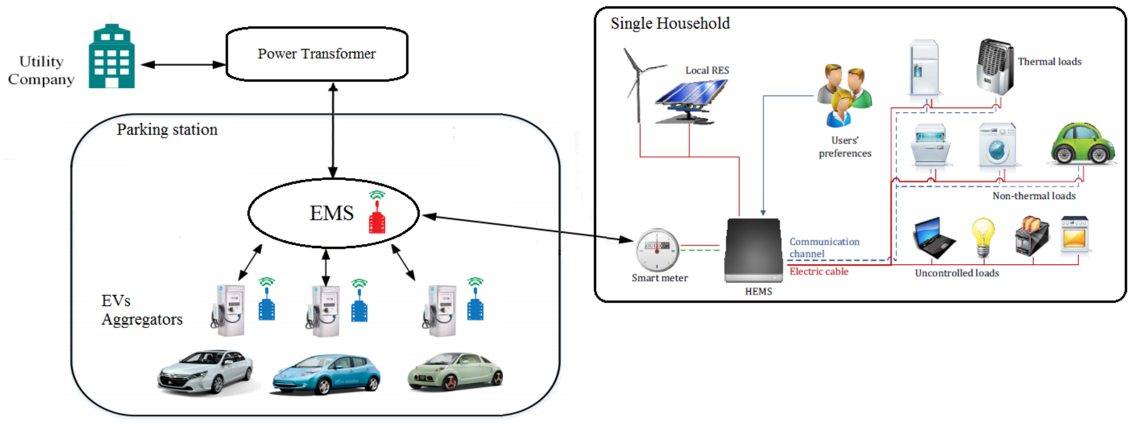


Figure 5: the block area communications routes

2.4 Electric load classification

To properly apply a DR program it is necessary to define the types of electrical loads can be involved on that. First of all, they must be surely *smart loads*, where the concept of smart appliance has been already introduced in 2.2.1. As a remind, they are electrical loads which are adapted to be controllable and monitorable remotely.

However, the concept of 'smartness' of this kind of devices can include also other foreseeable possibilities, such as the ability to automatically arrange and optimize their operations based on information received from the users or other sources.

Therefore, depending on their controllability, the electric loads can be categorized in three macro groups [6][16] as:

- *Non-Shiftable, Non-Interruptible (NSNI)* loads
- *Shiftable, Non-Interruptible (SNI)* loads
- *Shiftable and Interruptible (SI)* loads.

2.4.1 Non-shiftable, Non-Interruptible loads

NSNIs are those loads whose consumption profile cannot be anyhow modified. It includes lighting, cooking appliances, IT devices, irons, vacuum cleaners, etc. Managing the scheduling of such devices is unthinkable, since it would result highly discomfort for the user, given the fact of not being allowed to use these appliances when wished.

2.4.2 Shiftable, Non-Interruptible loads

This category contains all those loads whose consumption pattern is composed by a set of phases which can be shifted in time, but not interrupted once started. Furthermore, the power level of each phase cannot be changed. SNIs for example are washing machines, dishwashers and dryers, whose typical power consumption profiles are shown in Fig.6 [17].

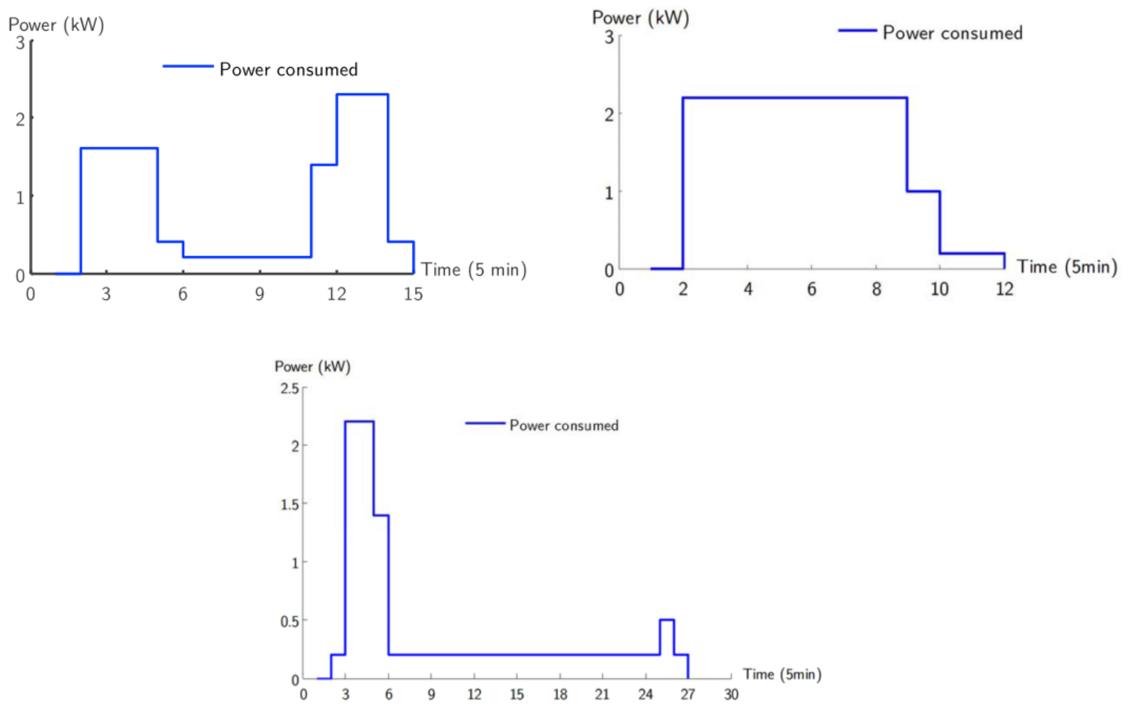


Figure 6: consumption profiles of a dishwasher (top left), a dryer (top right) and a washing machine(bottom) [6]

According to [6], the stepwise power consumption profile that characterizes these loads can be modelled with a series of parameters (Fig.7) as: number of phases, maximum and minimum process time intervals (D_{max} , D_{min}), maximum and minimum delay between two adjacent phases (T_{max} , T_{min}) and the energy required by each phase ($E_{i_1,j}$)

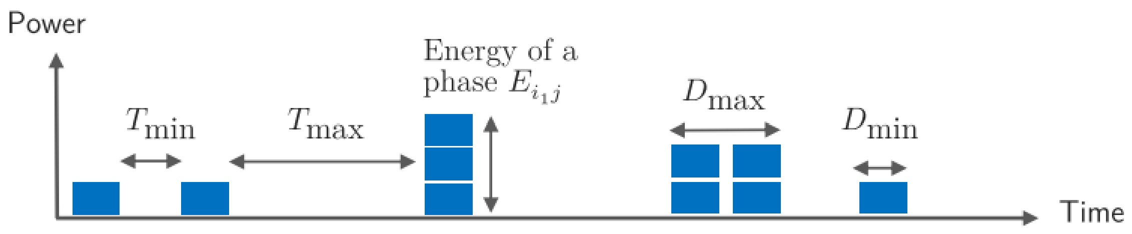


Figure 7: a model of the power consumption profile for SNI loads [6]

The possibility to opportunely arrange these parameters offers a certain degree of freedom for optimal scheduling operations.

2.4.3 Shiftable and Interruptible loads

In these category lie all the types of loads whose power consumption profile can be modified, up to a certain extent, both in time scheduling than in power. The loads belonging to SIs are basically energy storage devices. Depending on the quality of energy stored we can have electric battery loads (e.g. EVs) and thermal loads (e.g. boilers, heat pump, refrigerators, etc).
As there is not the presence of well-specified working phases, these loads offer the largest degree of freedom for the implementation of DR programs. The main constraint they pose is the achievement of a desired quantity of energy stored within a specified time interval.

Therefore, the EVs that will be simulated in the test bench fall into this category. We will see that all the optimizations performed by the EM are actually the realization of a DR program where, thanks to the degree of freedom offered by such a SI load, the charging profile of EVs are always rescheduled in order to match the optimization objective.

3 Communication network

In a context of highly unpredictable and distributed generation it has emerged overwhelmingly the idea of Smart Grid (SG), a way to conceive the power grid where classical elements of the existing systems are supported by completely new components of measurement and control.
The implementation of the concept of SG necessarily implies a tight interoperability between electricity suppliers and users, extended to every level of the generation-transmission-distribution chain.
Therefore, a reliable and well-structured communication network results of vital importance for the control entities of the system (TSO/DSO), the aim being to provide effective two-ways communication channels between the different parts of the grid and the operators in charge for controlling them.
As the focus is enlarged to the whole system, there is no more a unique solution of communication network, since each utility presents diverse regulatory regimes, topographies and even different communication systems inherited from the classical concept of power system [18].

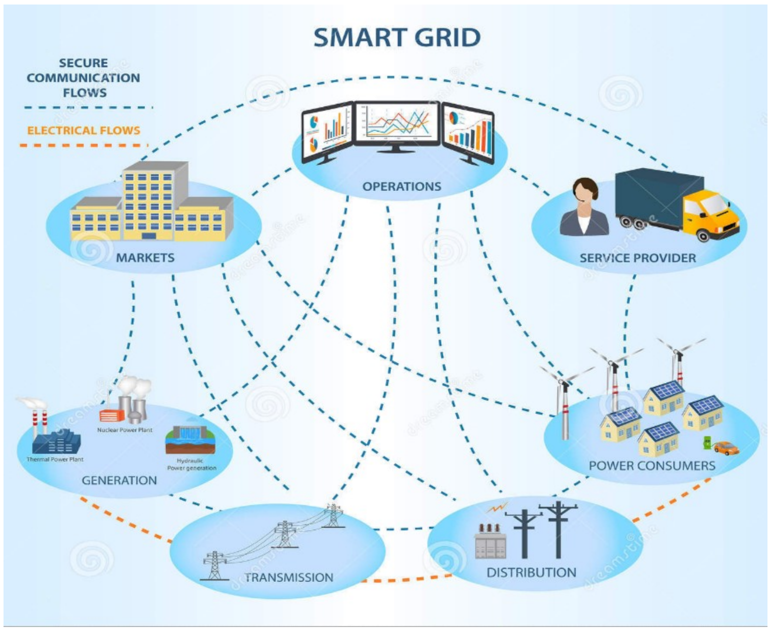


Figure 8: a Smart grid communication environment [19]

Hence, the great challenge of nowadays is to upgrade the existing communication infrastructures by adapting them to the wide set of smart grid applications foreseeable.

In this sense, accordingly with [20] the main issues are:

- Scarce level of interoperability between power system's elements communication standards
- Backwardness of existing power grid's communication networks, based on an old fashioned concept of power system
- Latency on data transmission.

The creation of a communication network has been the only physical realization of this work, and so it constitutes the hardware part of the test bench.

However, before the description of such a creation it is given an overview on the existing and implementable networks suitable for SG applications.

3.1 Levels of network for SG

Such a complex and articulated system can only be operated through a multi-layer communication network, where several communication technologies coexist in synergy to proffer reliable and effective access to each section of the power system, even in different environments.
Whence in [20] and [18] the authors propose a possible hierarchical multi-tier architecture for the communication network. The classification is basically made on the data-transmission rate and the coverage range most appropriate for each portion of the SG and it is divided in three main areas, as depicted in Fig.9.

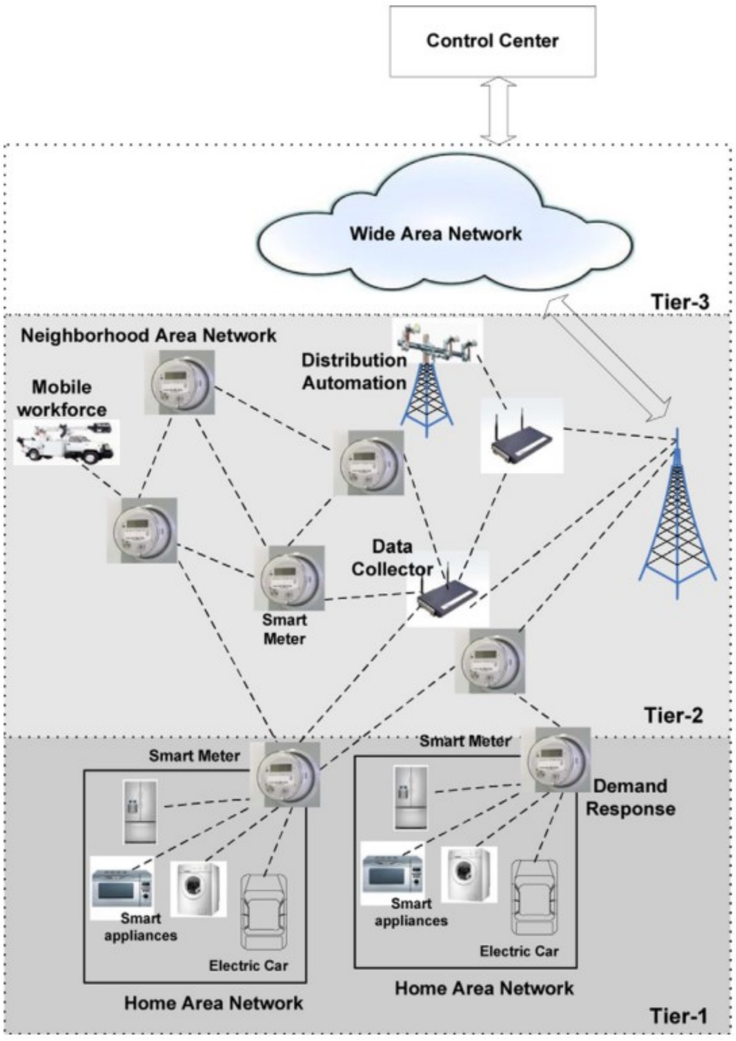


Figure 9: SG multi-layer communication network [1]

3.1.1 Wide Area Network (WAN)

At the level of transmission/distribution WAN serves to grant the control on the stability of the power system, and thus it envisages the monitoring, control and protection of extensive areas. It requires an ample data throughput at high frequency, consequently the communication technology must support high data rate (10 – 1000 Mbps) and offer considerable coverage distance (until to 100 km). Commonly used technologies at this level are fiber optical, cellular and WiMAX.

3.1.2 Neighborhood Area Network (NAN)

NAN is also called FAN (Field Area Network) where it is used for power grid monitoring. It is designed for applications like smart metering (SM), distribution automation (DA) and demand response (DR). This network requires data rate between 0,1 to 10 Mbps and a coverage distance up to 10 km. Usable technologies are coaxial cables, DSL, cellular, WiMAX but also ZigBee or WiFi meshed networks.

3.1.3 Home Area Network (HAN):

HAN is used in home, building (BAN) or Industrial (IAN) automation to provide communication channels between sensing/actuator/measurement devices and a controller. Conversely to the previously discussed networks, HAN are designed to operate in a smaller environment where the security of power system is not involved. Thus, in this case electrical signals going to/coming from sensors or actuators do not necessitate high performances in terms of data rate and coverage distance. It follows that communication technologies such as ZigBee, Bluetooth, Wifi, Z-Wave, Ethernet or PLC are sufficient to support these kind of applications.

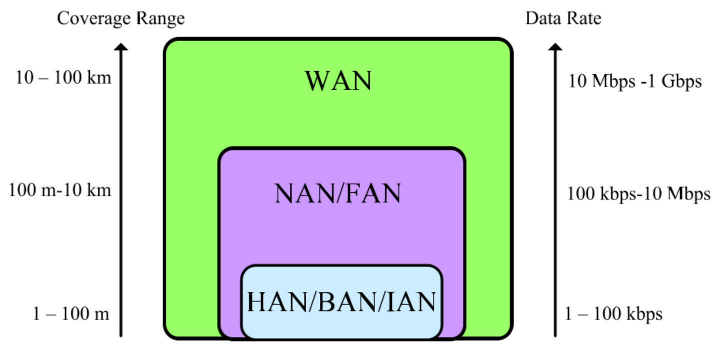


Figure 10: communication range and data rate requirements for SG network hierarchy [20]

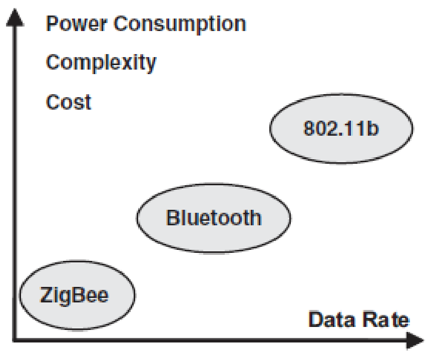
It is worth to remark that the structure above descripted aims to allow interactive cooperation among the different areas of communication, the aim being to work in synergy to accomplish the various requirements of the SG operations.
Nonetheless, the application studied in this thesis is certainly within the domain of HAN networks.

3.2 ZigBee

In HAN networks wireless communication is preferable over the wired counterpart, due to its flexibility in adding/removing devices, its low cost of installation and its low power consumption. Moreover, significant advantages come to light when the nodes present in the network become numerous, making the wired choice impracticable.

3.2.1 ZigBee versus Bluetooth and WLAN

ZigBee, based on IEEE 802.15.4 standard, is one of the principal wireless communication protocols tailored for low-data-rate, short-range networks. It envisages three possible operating frequency bands: 868 MHz, 915 MHz and 2.4 GHz. Compared with other existing standards for short-range wireless networking, such us Bluetooth and IEEE 802.11 WLAN, ZigBee has the lowest complexity and consequently the lowest data exchanging rate[21] (Fig. 11). Nevertheless, due to its cost and power efficiency, it is particularly suitable for applications where the wireless communication between nodes involves the exchange of simple commands or measurements from sensing devices. Indeed this protocol finds space in home, industrial and hospital applications where the need is for low-cost, low-power consumption and low data throughput [22].



	Data Rate	Typical Range	Application Examples
ZigBee	20 to 250 Kbps	10–100 m	Wireless Sensor Networks
Bluetooth	1 to 3 Mbps	2–10 m	Wireless Headset Wireless Mouse
IEEE 802.11b	1 to 11 Mbps	30–100 m	Wireless Internet Connection

Figure 11: Comparison between ZigBee, Bluetooth and IEEE 802.11b [21]

Because of the aforementioned features, ZigBee and its compliant XBee modules have been chosen to support the operations of the Energy Management System (EMS).

3.2.2 ZigBee features

The standard 'ZigBee' has been designed by ZigBee Alliance [23]. It is structured in several networking layers, each of which is apt to accomplish specific functions and exchange data/commands exclusively with the adjacent layers below and above it [24]. Fig.12 illustrates schematically the structure of the protocol layers:

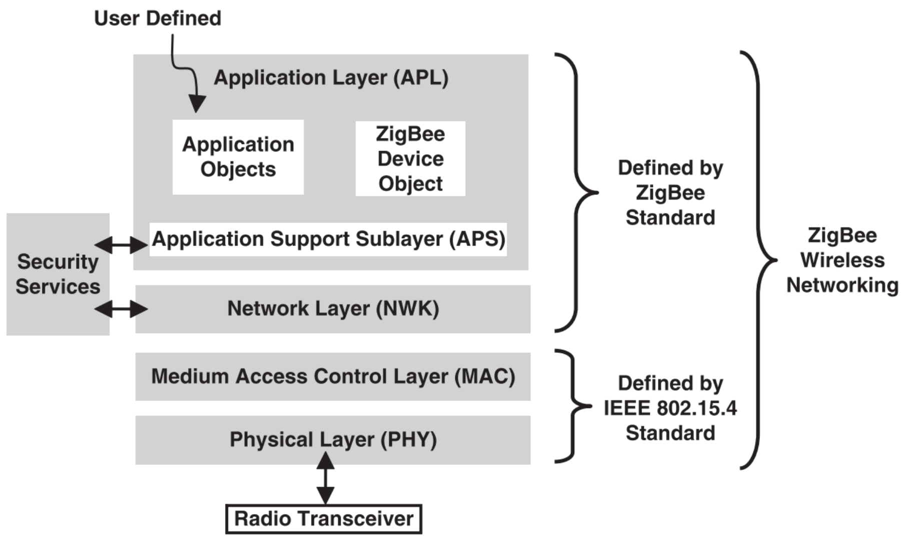

User Defined
Application Layer (APL)
Application Objects
ZigBee Device Object
Application Support Sublayer (APS)
Security Services
Network Layer (NWK)
Medium Access Control Layer (MAC)
Physical Layer (PHY)
Radio Transceiver
Defined by ZigBee Standard
Defined by IEEE 802.15.4 Standard
ZigBee Wireless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

Figure 12: ZigBee networking layers [21]

As it can be noticed in the precedent figure, the last two layers are compliant with IEEE 802.15.4 specification. Let us describe briefly the functions each layer fulfills:

- Physical Layer (PHY): it is in charge of activate the radio when it has to receive or transmit a message packet. It performs the selection of the channel frequency, verifying as well if this channel is busy or not
- Medium Access Control Layer (MAC): it serves as an interface between PHY and NWK layers. Moreover it accomplishes services of synchronization and association/dissociation
- Network layer (NWK): it constitutes an interface between MAC and APL layers and is responsible for the network formation
- Application layer (APL): it is the head layer and contains the application objects. These are in charge of managing and controlling the protocol layers in a ZigBee device
- Security: this section has the role to assure the confidentiality of the data transmitted by operating an encryption.

3.2.3 ZigBee devices roles and network topologies

After having described the ZigBee network structure, let us now describe the devices that are part of it, the roles they can take and the way they interact. The standard IEEE 802.15.4 foresees mainly two types of devices:

- Full function devices (FFD): they are able to perform every role or duty envisaged by the IEEE 802.15.4 standard. Furthermore they are allowed to communicate with all the devices of the network
- Reduced function devices (RFD): as the name suggests, they have restricted capabilities compared with FFDs. Indeed in the network they cannot talk each other, but the only communication allowed is toward an FFD.

For what regards the roles they can take, in a ZigBee network one can have:

- Coordinator: it is a FFD device, and its main function is to establish the network and to be the principal controller of it
- Router: it is also an FFD device like the Coordinator, but it is not set to be the controller of the network
- End device: it is an RFD device that, contrary to the first two, cannot relay messages but can only talk to an FFD.

It is worth to remark that between ZigBee and IEEE 802.15.4 standards the terminology about these roles slightly differs. Therefore, in IEEE 802.15.4 'Coordinator', 'Router' and 'End devices' become respectively 'PAN coordinator' (since what it actually sets is a Personal Area Network), 'Coordinator' and 'Device'.

These kind of entities are exploitable in two possible network topologies: star or peer-to-peer, as shown in the Fig.13 and Fig.14:

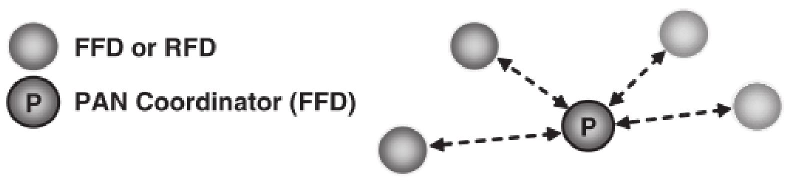


Figure 13: star topology [21]

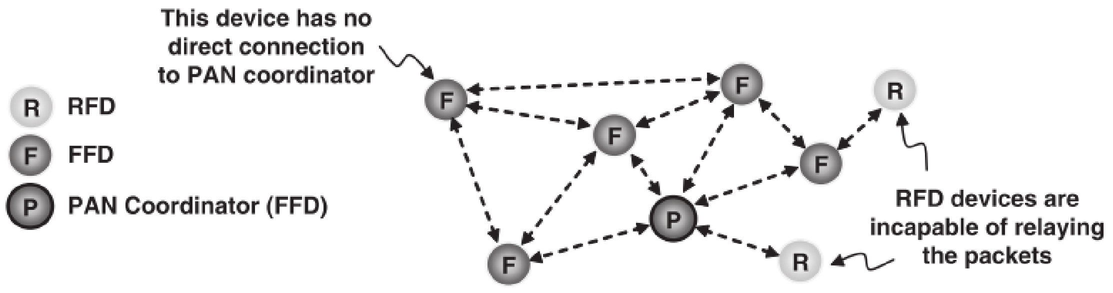

This device has no direct connection to PAN coordinator
R RFD
F FFD
P PAN Coordinator (FFD)
RFD devices are incapable of relaying the packets

Figure 14: peer-to-peer meshed topology [21]

In a star topology every Routers or End devices can only convey messages to the coordinator. Instead, in a peer-to-peer topology FFD devices can communicate each other, while again RFD can do it only with an FFD.

3.2.4 ZigBee collision avoidance method

Once network is set, problems might arise when different devices have to use the same frequency channel to perform their communication. This represents a key issue for the work of this thesis since, as it will be explained later, the radios that participate to the network have to send their messages to a single coordinator every fifteen minutes ‘almost’ at the same time.
For this reason, the standard under analysis has developed two simple methods to permit several devices to utilize a single channel while avoiding the collision of messages.
A first one is called ‘contention-free’ method. It implies that the Coordinator keeps synchronized all the radios in the network, giving a specified time interval to each of them to perform communication. This method has not been taken into account in this thesis since it is more power-consuming.
The method utilized is instead a ‘contention-based channel access’ and is called Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA). It consists in a particular mechanism through which before getting access to the channel, a device that wants to transmit operates firstly a channel availability assessment. This process is named Clear Channel Assessment (CCA) and thus assures that the frequency channel is not currently used by another transmission.

There exist two ways to operate CCA. The first one is Energy detection (ED), whereby a device measures the spectral energy of the channel it is interested on, without trying to decode the signal. The second way to do a CCA is called Carrier Sense (CS). Again it is done a signal energy estimation, but, contrary to what happens in ED, in this case the type of signal is ascertained in order to verify if it is compliant with IEEE 802.15.4.

The difference between ED and CS is that in the former the channel is declared busy everywhen signal energy is detected, while in the second it is possible to discern if that signal is an IEEE 802.15.4 and, in that case, consider the channel as busy.

The important thing to underline is that in both cases if the channel is declared as 'not clear' then the device waits for a random time period before trying again the transmission. Afterwards it keeps iterating this procedure until the channel turns clear or a maximum number of tries (defined by the user) is reached.

3.3 Test bench network structure

After having introduced the structure of ZigBee standard and its working principle, in this last paragraph a description of the network designed to create the test bench is given. In addition to the topology, a brief description of the radio devices chosen to set up the network is provided.

3.3.1 XBee modules

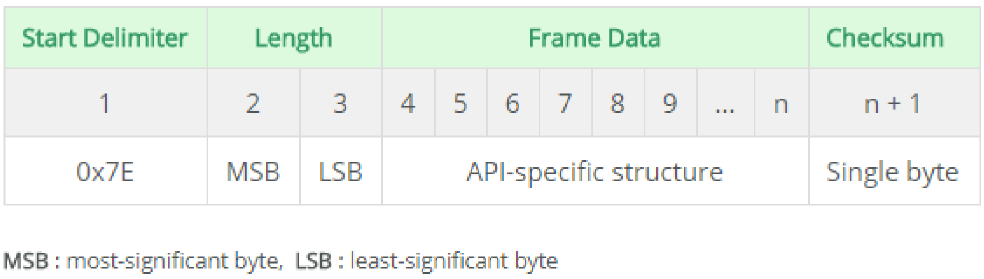
XBees are radio modules produced by Digi International [25]. They are based on IEEE 802.15.4 standard, so they are ZigBee compliant. In the test bench realized in this thesis, XBees have the role to support the Aggregators in their operations, by providing them a mean to communicate to the Coordinator the energy requests they gather from their respective fleet. The setting of every radio module has been done through XCTU, a software made available by Digi International that allows either tuning the radios' parameters and realizing every sort of communication.

Within a PAN XBees operate in conformity with ZigBee standard protocol: there will be a Coordinator that forms and controls the network, Routers and End Devices that communicate with it following the procedures discussed in 3.2.4. What is still to be presented is the way these modules exchange data.

These devices have basically two modes of transmitting data: API and AT mode [26]. The latter is to be used to send simple commands to the end device (e.g. to change some parameter of the end device). API mode instead is helpful to transmit sets of data grouped in packets and called 'API frames'. Considering that Aggregators have to send numerous information about the state of their fleets, only API mode has been used in this work.

3.3.2 API frame

Let us describe the structure of an API frame, with the help of Fig.15:



Start Delimiter	Length		Frame Data								Checksum
1	2	3	4	5	6	7	8	9	...	n	n + 1
0x7E	MSB	LSB	API-specific structure								Single byte

MSB : most-significant byte, **LSB** : least-significant byte

Figure 15: API frame structure [27]

One can recognise:

- Start Delimiter: it indicates the beginning of the frame and it has always the same value 0x7E (hexadecimal numeral system)
- Length: this field specifies the number of bytes forming the data frame. In this count, it excludes itself, the start delimiter and the checksum
- Frame data: in this section they are contained the data to be transmitted and the API frame type identifier as well. The packet of data to be sent can have a maximum length of 100 bytes
- Checksum: the last byte serves to test data integrity. Checksum is computed by summing only the bytes which are bunched in 'frame data'.

3.3.3 Network topology

Finally, one has all the means to define the network structure of the test bench. The latter is a star topology constituted of 3 End Devices and one Coordinator at the service, respectively, of the Aggregators and the Energy Manager (Fig.16). Radio modules have been mounted on specific XBee adapters through which it is possible to handle them from a computer by means of an USB connection. So, virtually we have four computers in total, each of which runs a distinct code to play the role that has been foreseen for it within the test bench.

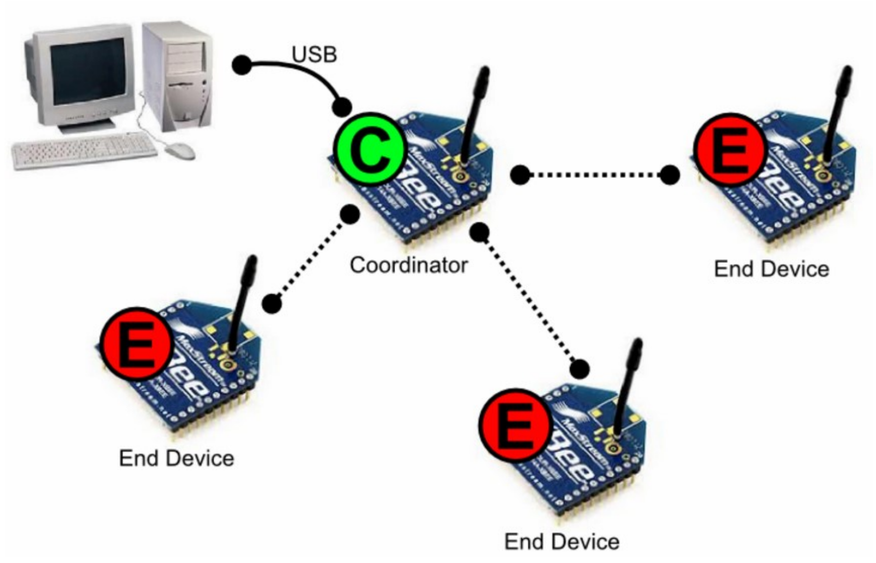


Figure 16: test bench network topology [28]

4 The Test Bench

4.1 Emulator structure

The emulator has been entirely created in Matlab language. The whole program is articulated in several scripts, each concerning a specific part of the same. The code run day by day and the user is allowed to choose the electricity price determination method among constant price, day-ahead pricing, time of use and bihourly tariff.

The daytime is discretized in intervals of fifteen minutes, during which all the operations of communication and optimization are carried out. It is worth to underline that the time scale of the emulator could not correspond to the real time: indeed, for the sake of velocity, the time of the simulation can be opportunely scaled to make a day last in some minutes, instead of having to wait the real time.

As a first overview of all the scripts we have:

- 'Fleet_Characteristics': it sets up, at the beginning of every day of simulation, the schedule all the EVs arrive with and also their energy requests. This is done by following criteria of randomness explained later on
- 'Aggregator': as suggested by the name, this script emulates the behavior of the aggregator and thus it acts, every fifteen minutes, as an interface between the coordinator and the electric vehicles. The aggregators can decide to leave the optimization process at the end of the day or to continue with the next date
- 'Coordinator': this is the core script since it emulates the operations performed by the EM. It is continuously ready to receive energy requests from the aggregators, by then it run the optimization. Hence, it contains all the processing and post-processing parts and works as an interface for the user

- 'Optimization_bill', 'Optimization_PAR': at each iteration they operate sequentially the optimizations, firstly at the level of single houses (bill optimization) and then at the grid level (peak to average ratio optimization).
- Set parameters: these are a set of scripts called once a day to properly configure the electric load profile of all habitations, the electricity price, the time discretization and the figure parameters for the current day
- 'Plot_Electricity_Price', 'Plot_Power': they are the scripts for the post-processing operations. Hence, they show in graphs the electricity price profile for the current day and the updates that occur in the transformer power profile at each iteration.

Let us proceed with a deeper insight on the scripts one by one in order.

4.2 Fleet set up

The idea behind the fleet generation is to create a scenario which best reflects a realistic situation of vehicles arriving/leaving a parking lot. Therefore, they must be simulated all the probable features constituting a daily routine of a charging station: arrival times, energy requests and charging times made available by every EV. These ones are light duty electric cars, and they can charge in a normal home outlet at the typical home power range, although the philosophy the test bench is designed with is implementable for vehicles of any size and charging method.
Each of them, once arrived at home and plugged into the network, is allowed to set its own preferences about energy it needs and time it is willing to wait to recharge the battery.
Therefore, every EV communicates to the coordinator four information concerning its status:

1) Identification number
2) Start charging time
3) Time interval allowed to charge
4) Energy needed

The number of EVs that constitute a fleet is 64. The latter comes from the following considerations: as mentioned in 3.3.2, a packet of data exchanged between two devices through an API frame has a maximum length of 100 bytes. A Coordinator could receive more than one message at a time though, and it must store all the requests coming from all the Aggregators without losses of messages. If one considers that every EV's information counts for 1 byte and that each EV sends 4 information, by supposing all vehicles communicating simultaneously to the Coordinator the maximum number of EVs allowed would have been

$$N_{EV,fleet} = \frac{Packet\ size\ maximum}{Number\ of\ Aggregators * EV\ request\ size} = \frac{100}{3 * 4} = 8{,}33$$

However, such a simultaneity of energy requests sent by the 3 Aggregators is unlikely to happen, since the start charging time of EVs has been opportunely spread throughout the all day time. So, it has been chosen that number of 64 in order to have a considerable number of EVs in a fleet, while maintaining still a margin respect to the maximum size of the data packet.
Let us proceed with the description of how the different features of the EVs' charging patterns have been set in the simulations. As it will be explained, all these settings follow principles of randomness.

4.2.1 Start charging time scheduling

The choice of the start charging time for the vehicles is made with a random process. By considering a normal working day, it has been assumed that people mostly leave home at 8:00 in the morning and come back at 19:00 in the evening. Another portion of drivers instead could arrive and plug his car at around 13:00. Therefore the fleet results as split in 2 portions:

- a first part (whose number is randomly selected from a normal distribution with mean=13 and standard deviation=2) is composed by EVs that start charging at a time taken from a normal distribution with mean 13 and a standard deviation of 2 hours;
- a second part where the remaining number of EVs start charging at a time picked up again from a normal distribution, but with mean 19 and standard deviation of 2 hours.

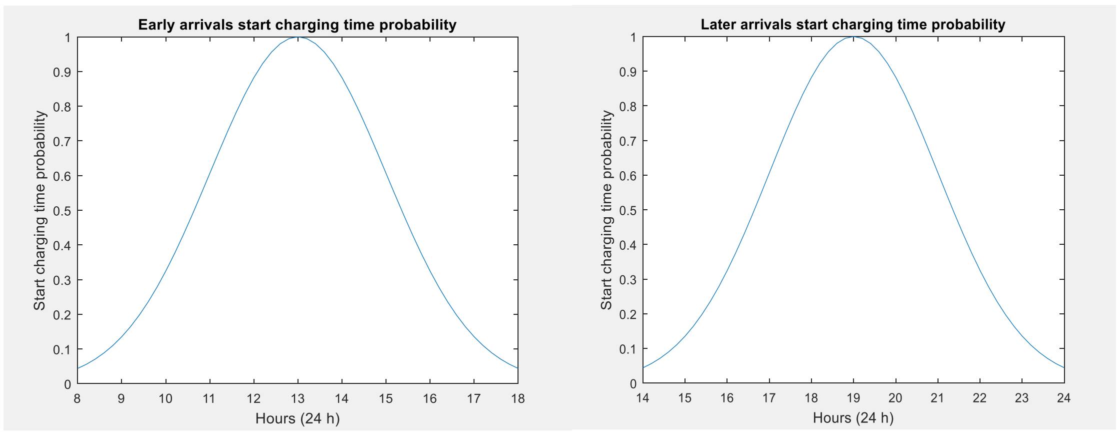


Figure 17: probability of start charging time for the early arriving EVs (left) and for the last arriving EVs (right)

At the end of each day, at midnight to be precise, the fleet is updated in order to obtain new values of time scheduling and energy requests for the next day of simulation.

4.2.2 Energy request formulation

The energy requested by every EV is gotten again in a probabilistic manner. In [29] authors estimated an average energy consumption rate (ECR) of 160 Wh/km for mid-sized EVs, while the European report [30] shows that the mean daily distance traveled by drivers in the European countries ranges from 40 to 80 km. Therefore, as a reference value, it has been assumed an average energy consumption of 6,5 kWh per day for the drivers.

Hence, the energy consumed in a day for each EV of the fleet is drawn from a normal distribution with mean 6,5 kWh and standard deviation 3 (Fig.18).

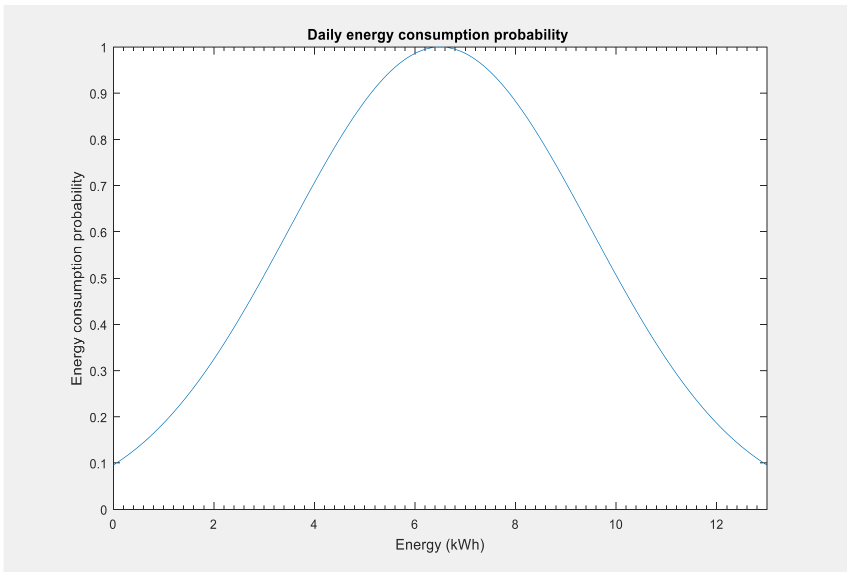


Figure 18: daily energy consumption probability curve

4.2.3 Time available to charge

The last part of the fleet generator script aims the formation of the time interval the drivers make available to refill the battery. This process has been implemented again with a certain degree of randomness: firstly a minimum time range is computed, by basing the calculation on the energy requested and the charger power as follows:

$$T_{min} = \frac{E_{needed}}{P_{max,charger}}$$

Secondly, an extra-time is randomly added (up to 10 hours) to the first amount, to take into account the factors listed below:

- not always the maximum power of the charger is exploitable at the socket. Other appliances may consume power during the EV's battery refill, reducing in this way the margin with respect to the power available by contract in the household;

- not necessarily the time a driver is willing to wait to recharge its battery coincides with the minimum time achievable by charging at maximum power. It is surely the case, for instance, of a typical during-night recharge.

4.3 Aggregator

The Aggregator is the entity that manages the communication between the Coordinator and the electric vehicles of the fleet. It collects and sends every fifteen minutes the requests gathered from the EVs arrived during the previous quarter hour. If new EVs have arrived during this time range this script activates the ZigBee communication, otherwise it notifies the user that nothing has changed.
In a real implementation this entity might act as a Router in the network, by actively communicating with every EV of the fleet. Its status could be turned on whenever a vehicle arrives, plugs in to the charger and requests to receive energy. EVs instead would operate as End Devices in this case (Fig.19), giving rise to a tree topology network.

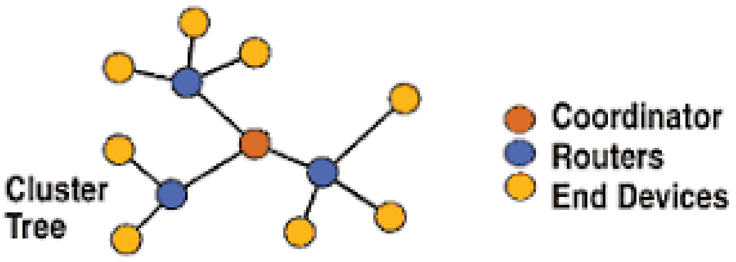


Figure 19: real implementation network topology [31]

Nevertheless, since in the test bench the EVs behavior has been entirely virtually created, no communications are necessary between Aggregators and related Fleets. Thus, such a node can be easily handled as an End Device that wakes up, at every iteration, only if there are new requests to be sent.

4.3.1 Transmission failure management

The main problem that has been tackled in this part of code is surely the data transmission failure. The purpose was to create a reliable code able to react in case of missed reception, in order to guarantee a proper communication between the parts.
Many are the causes that can represent an obstacle to a correct radio transmission. These are mostly related to the presence of materials in the space which act as electromagnetic shields [21] or the proximity of other LAN/PAN networks (e.g. a router WiFi) whose signals may represent electromagnetic interferences [32].
Moreover, it is to be taken into account the issue related to the coordinator's input channel congestion. The way ZigBee protocol counteracts the crowdedness of transmission channels has been already discussed in 3.2.4 [22].
Because of the way to work of the test bench, channel congestion is likely to occur: the three Aggregators are set to send energy requests every fifteen minutes, and the action of sending messages results synchronized for all of them with the internal clock of Matlab. Therefore, after having performed CCA, it can happens that more than one Aggregator finds the channel either free (and so it start transmitting) or busy (then it delays the transmission). In the first case, to avoid information losses, the Coordinator has been designed to be able to store energy requests coming from more than one Aggregator at a time. The code has been implemented to recognize if more than one sender is present within the same API frame, then it discerns the sources and runs the optimizations separately for each fleet. In the second case instead the CCA procedure could be repeated several times up to the maximum number of reiterations set by the user. If this border is exceeded then the transmission is erased.
In order to overcome such a transmission failure, XBee modules make available a helpful function that can be asked to the coordinator when setting: a feedback on the state of the reception called 'Acknowledgment frame' (ACK) (Fig.20).

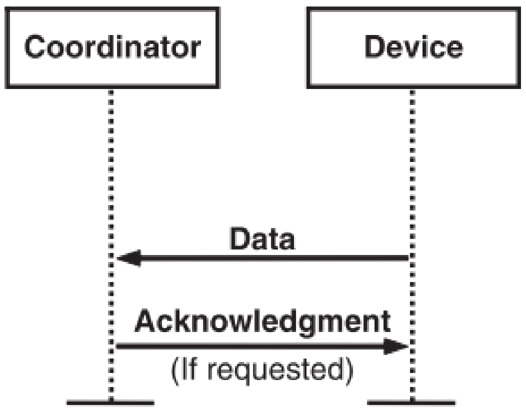


Figure 20: Acknowledgment frame exchange [21]

Thereby, in a bilateral communication, XBee receivers send back to the sender this kind of feedback as soon as they receive a message [33]. If an acknowledgement is not received, the packet will be resent up to three extra times. After that, if the acknowledgement is still not received by the sender, an ACK failure is recorded.
However, since this procedure could not yet guarantee a successful transmission, it has been created in the code an algorithm on purpose to strengthen the overall probability to attain a satisfactory communication. The algorithm is depicted in Fig.21.

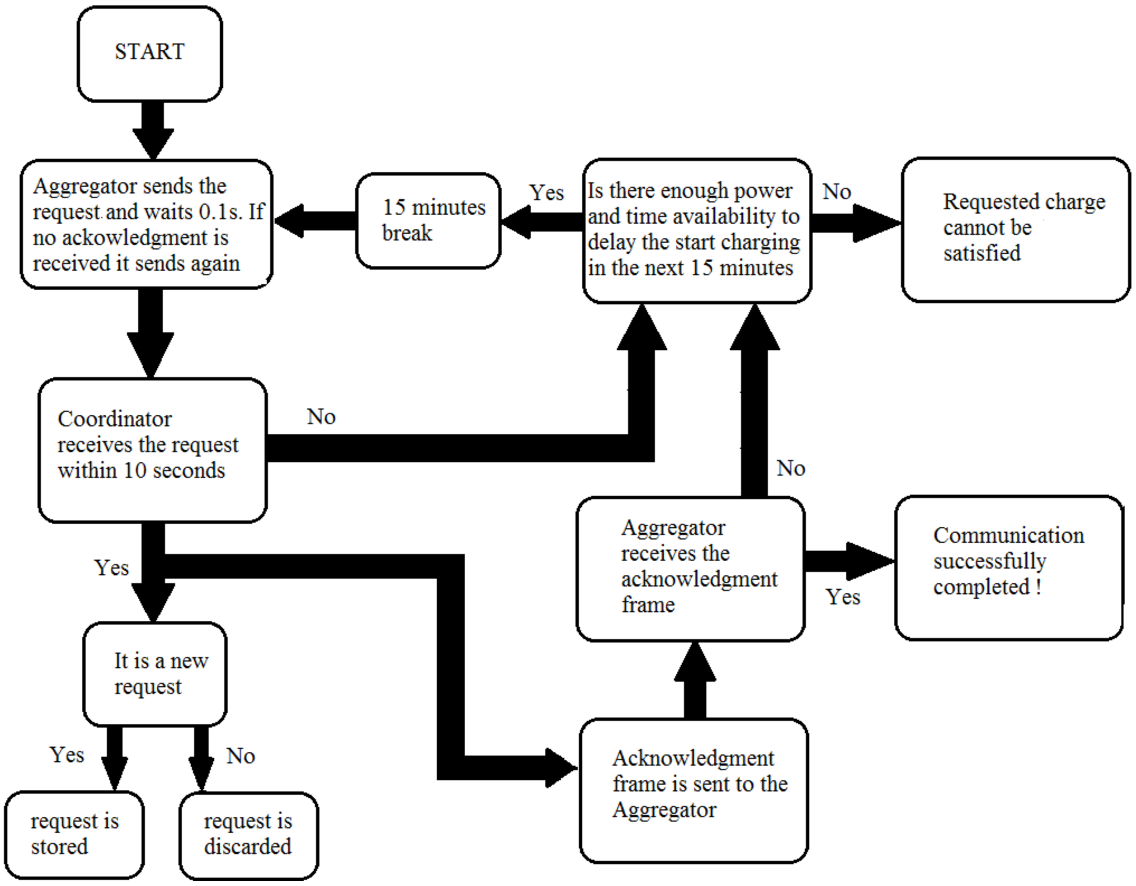

START
Aggregator sends the request and waits 0.1s. If no ackowledgment is received it sends again
15 minutes break
Yes
Is there enough power and time availability to delay the start charging in the next 15 minutes
No
Requested charge cannot be satisfied
Coordinator receives the request within 10 seconds
No
No
Aggregator receives the acknowledgment frame
Yes
Communication successfully completed !
Yes
It is a new request
Yes
No
request is stored
request is discarded
Acknowledgment frame is sent to the Aggregator

Figure 21: scheme of the algorithm to enhance the probability of reception success

The working principle of the algorithm follows these steps:

1) One End Device, after having assessed the channel is free, sends a message to the Coordinator. During the next 400 µs it goes in ‘receiving mode’ waiting for one ACK [21]. It is to remember that it resends the message 3 extra-times automatically, so at most this procedure takes 1.6 ms. If within 0.1 second the Aggregator does not receive any ACK it keeps repeating this procedure up to 10 seconds.
2) If Coordinator receives the message within 10s then it can either hold or discard it, depending if that request has already been received before or not. Indeed, it can also happens that the Coordinator sends an ACK but the End Device doesn’t receive it! Thus, from the Aggregator point of view, the message is to be sent again in the next 15 minutes, as it will be explained later. In any case, as soon as Coordinator receives, it sends an ACK. Otherwise it does not.

3) If the End Device receives the ACK within 10 s the communication is to be considered as successfully accomplished. In the other case the Aggregator behaves like if the message has not been received at all by the Coordinator. When this occurs an extra procedure is triggered, since the transmission is judged as failed: for every one of the latest-plugged EVs contained in that message, it is verified if there are both enough power capability and time slots remaining to complete the charge within the user-defined time. If this is possible, the start charging time is shifted to the next 15 minutes, together with the following possible requests. If it is not, the user is warned that its desired charge cannot be satisfied and it has to change the terms.

It is worth noting that, although this procedure already yields satisfactory results , this timing so narrow has been designed for the time scale of the emulator. In reality such an execution velocity is not strictly necessary, since times are wider and one can afford more margin. So, in a real implementation, in case of transmission failure one might think to reiterate the algorithm for more than 10 s in order to enhance the likelihood of success.

4.4 Coordinator

The Coordinator is the part of the test bench that emulates the behavior of the Energy Manager. Thereby later on we will speak equivalently about Coordinator or Energy Manager by referring to the same entity. This script is the principal one, from which the entire test bench can be started.

At the first launch it loads the following data:

- Whole Sale Price (€/MWh) from Belpex[1] website
- Syntetic Load Profile curves from Synergrid[2]
- Irradiance (W/m^2) from IRM[3]
- Electricity tariffs from Electrabel[4]

[1] Belgian electricity market operator.
[2] Belgian federation for electricity and gas networks.
[3] Institut royal météorologique. It is a Belgian federal institution for meteorological research.
[4] Belgian utility company for electricity.

All of these data are referred to the year 2014. They have been gathered for the entire year in a discrete time manner, day by day, by samples of 15 minutes each.

Successively, Coordinator randomly selects a day of the year 2014 and, by exploiting the data previously loaded, it calls the scripts that set the parameters needed to run the simulation for the current day. These are:

- *Time parameters*: essentially this script operates the time discretization of the day. It creates all the time vectors and parameters which will be used all throughout the program, both in the processing than in the post-processing part
- *Block parameters*: this script generates the load profile for all the households which are, in a broader perspective, grouped in blocks and take part to the optimization. Indeed, every EV's recharge is considered as a part of the total electricity consumption of the respective house. Each fleet corresponds to a block and so to an Aggregator
- *Electricity price parameters*: this is the script responsible for the formation of the electricity cost vector
- *Figure parameters*: this script sets the graphical parameters useful for the post-processing.

It is worth to highlight that parameters are not only set for the current day but partially for the day subsequent as well. In fact, in a daily optimization they must be included also EVs arriving late in the evening, whose scheduled charging power profile might extend to the early hours of the successive day.

The EM keeps continuously open the reception channel during a day. Everywhen it receives a message it then triggers sequentially the optimizations.

4.4.1 Multiple reception management

As mentioned in section 4.3.1, despite of all the countermeasures taken to prevent the congestion of the communication channel to the Coordinator, messages coming from the 3 Aggregators may arrive almost synchronously or, at least, within a small time interval.

To properly face these eventualities, the EM has been designed to wait as far as its input channel detects the presence of incoming signals before launching the optimizations. In this way it has been enabled to store, one after the other, messages arriving from more than one source.

Thanks to the knowledge of the API frames structure (already introduced in section 3.3.2), it has been possible to pinpoint within a message the *start delimiters* and the *source identifiers* of the diverse energy requests. This possibility of discerning has allowed to properly distinguish and separate the optimizations for the three fleets of EVs.

4.4.2 Time organization

As aforementioned, in DSM applications time is discretized in intervals of 15 minutes. Consequently the daytime is subdivided in a number of time slots (N_{slots}) equal to

$$N_{slots} = \frac{24 * 60}{15} = 96$$

It is to emphasize that the emulator has been designed to make the Aggregators drive the time during the simulations. Whereas in a real time implementation they are supposed to wake and transmit synchronously at every 15 minutes, we remember that in the test bench this time discretization can be opportunely scaled in order to make the iterations faster. By default, the real and simulation discretization intervals are in the relationship

$$15\ min \ \leftrightarrow 30\ sec$$

even if the user is allowed to tune differently the scaling factor. This means that if in a real application each iteration lasts 15 minutes, in the simulations these terminates in 30 seconds.
It is important to highlight that this operation of scaling must be done carefully in order to respect the test bench working times. Indeed, the time slots cannot expire before both Coordinator and Aggregators have completed their operations!
In this regard, if CPU computation time is neglected, the Aggregator operations take at most 11.1 seconds to be accomplished. This time interval includes also the maximum delay can be reached in case of transmission failure, as discussed in section 4.3.1. To facilitate the transmission operations by alleviating the communication channel congestion problem, the start sending instants of the 3 Aggregators have been delayed of 2.5 seconds each other. So, at most, at each iteration the transmissions from Aggregators can last 16.1 seconds.
For what concerns the Energy Manager instead, this is kept constantly in receiving mode and it operates its computations as soon as it receives energy requests. Normally, the optimization

processes require no more than 10 seconds to be accomplished, from when the message is received until the results are found and shown.
Hence, if iterations last 30 second there is enough time to complete both operations of communication and optimization. Nevertheless, overlapping issues may arise whenever YALMIP does not find an optimal solution within a few number of iterations and keeps reiterating the calculations. In those cases it has been experimentally observed that the optimizations can take even more than a minute to terminate! Then, the scaling factor should be properly regulated in order to permit their completion.
A Coordinator that warns the End Devices that their discretization intervals are too short, with respect to the optimization time, has not been implemented yet. Also because the concept of time intervals scaling would not apply to reality, where such iterations velocity is not required.

4.4.3 Baseload formation

In this work they are considered only Electric Vehicles as SI loads, and only photovoltaic production as local generation. To properly conduct the simulations it is also needed an estimation about the behavior of single houses in terms of power consumption profiles during the day. The latter have to be as realistic as possible in order to validate the model.
In this regard, on the traces of the work done in [6], the power load profiles have been generated on the basis of statistic data of energy consumption, made available by the already cited Synergrid. By means of these data, Synergrid formulates what it calls *Synthetic Load Profile* (SLP*,* Fig.22)): it is a theoretical model based on measures of electricity consumption conducted on thousands of residential meters, which takes into account several variables such as weather conditions, irradiance and period of the year. SLP curves are traced for every day of the year and they are discretized on a 15 minutes basis. Moreover, they are available on the Synergrid website [34].

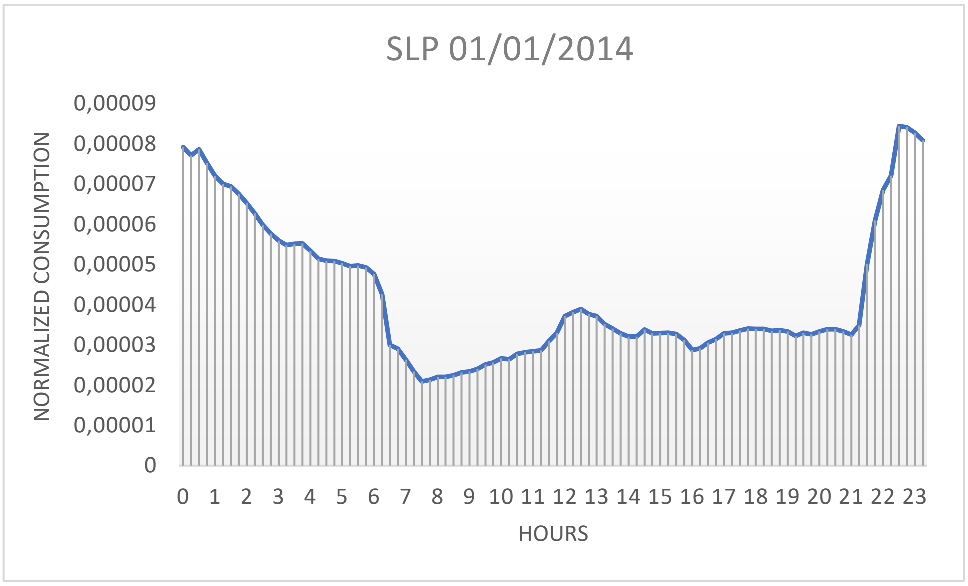


Figure 22: synthetic load profile for January 1, 2014

Therefore, SLP constitutes a valuable model to estimate a normalized consumption for one household. The normalization factor of these curves is W/kWh. To obtain the power consumption profile of all the households, SLP curves are multiplied by the annual energy consumption of each house. Therefore, if the annual consumption of one household is equal to 3500 kWh, for a generic day x of the year, the power consumption profile of this house (Pow_x) is computed, as:

$$SLP_x\left[\frac{W}{kWh}\right] * 3500[kWh/year] = \; Pow_x\;[W]$$

The annual energy consumed by each household is obtained in a random way. Indeed, it can be extracted from three normal distributions with:

- average 2000 and standard deviation 500
- average 3500 and standard deviation 500
- average 5000 and standard deviation 1000.

4.4.4 Price formation

Households involved into the optimizations are obviously to be considered residential customers. Most commonly they are charged with flat tariffs by the suppliers [35]. However, such a pricing method does not reflect directly the actual cost of electricity and, more importantly, it does not allow the implementation of a DR program. In order to be able to perform DR taking advantage from load flexibility, it is necessary to adopt *Dynamic pricing (DP)* tariffs, such as *Real-time pricing (RTP)*. Indeed, these ones allow adapting the consumption profile on the basis of the knowledge about how electricity prices vary throughout the day [36]. Nonetheless, nowadays such forms of DP are not yet existing for residential customers, so there is not data availability. In this thesis it has been adopted a method to dynamically define prices which has been studied in [11].

First of all it is necessary to define which part of the electricity bill is subject to variability. The costs charged to the final user are composed of 4 elements, each referring to a specific service: transmission, distribution, taxes and energy consumed. Among these, only the energy component is directly subjected to variations under DP.

This variability is linked to the production costs faced by the producers and the subsequent market dynamics that lead to the definition of the final wholesale electricity price.

Therefore, to reflect this variability [11] proposes to link the retail market price to the wholesale one. However, price cannot be directly the wholesale market one, since retailers will adjust it by adding a quota to protect themselves by the risks of the price instability[16].

Hence, RTP will be derived from the day-ahead wholesale market prices through a proportionality coefficient deduced according to the so-called principle of *revenue neutrality*: under the hypothesis that a consumer maintains unvaried its consumption pattern, the idea is that, on a yearly basis, the electricity bill resulting from DP would equals the one ensuing from the former pricing method. Thereby, it is supposed that the idea of changing tariff would be accepted by customers.

The above mentioned coefficient (K) is called *pricing factor* and it is calculated as follows:

$$K = \frac{AP}{\sum_i \left[\sum_j \left(CL_{ij} \times WP_{ij} \right) \right]}$$

where i and j are the indices referring, respectively, to the days of the year and to the time blocks by which the days are discretized. Yet

- CL_{ij} is the consumption level in the time block j of day i. It is estimated by making an average of the SLP
- WP_{ij} is the wholesale price in the time block j of day i. It is calculated by an average of the wholesale price of each hour
- AP is the average yearly energy tariff computed with the former electricity pricing method.

Therefore, the calculation of the pricing factor has made possible the generation of valid cost vectors, which have been used in the simulations as a reference to perform RTP.

4.4.5 Tariffs

As formerly mentioned, the EM asks the user to select the pricing method it wants to adopt in the simulations. The tariffs implemented are:

- *Constant price (CP)*: simply, the price is kept fixed to the retailer value. Besides the flat electricity price it includes the costs of contract, meter, connection, distribution and taxes (DSO,TSO and green energy taxes) as set by Electrabel
- *Day-Ahead Price (DAP)*: it is generated by multiplying the hourly-based wholesale market price by the pricing factor K, as introduced in the previous paragraph. Furthermore, they are added the same costs of contract, meter, connection, distribution and taxes as in CP (Fig.23)

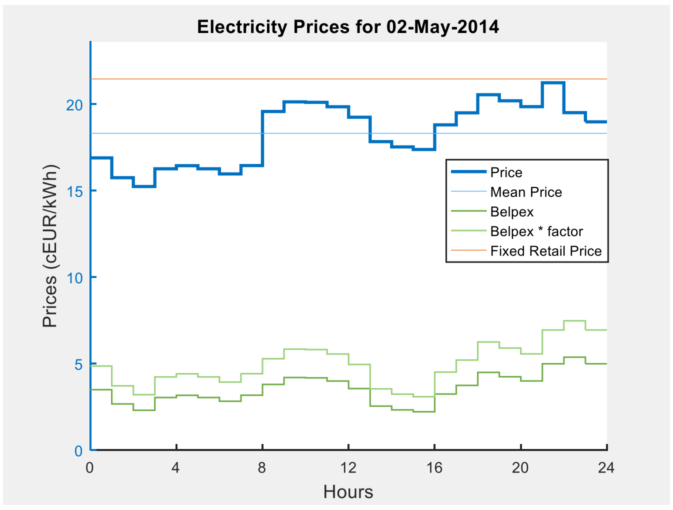


Figure 23: DAP profile for May 2, 2014

- *Time of Use (TOU)*: it is organized in 5 time blocks per day. For each of them the price value is obtained by averaging the wholesale price contained into the interval. Again the average is then scaled by the pricing factor K and the same extra costs are included as in CP and DAP (Fig.24)

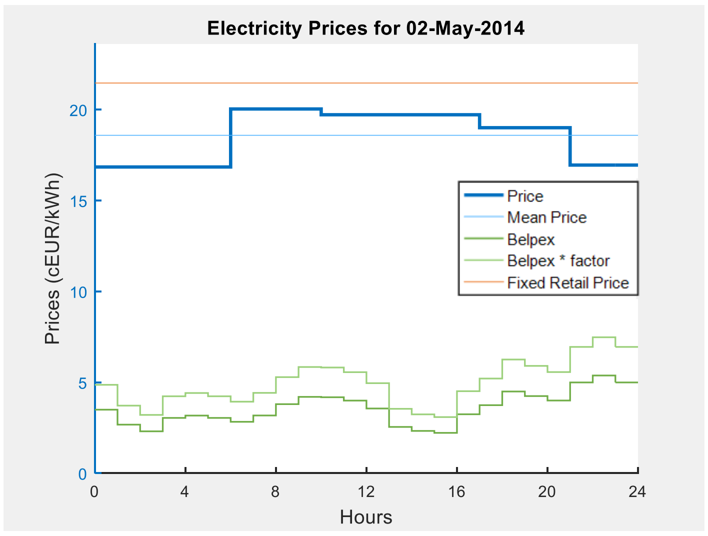


Figure 24: TOU profile for May 2, 2014

- *Bihourly*: it is split in a during day tariff (valid from 8:00 to 23:00) and in a during night one (valid for the remaining hours) (Fig.25). It is computed by adding to the flat electricity tariffs the same extra costs as in the previous cases

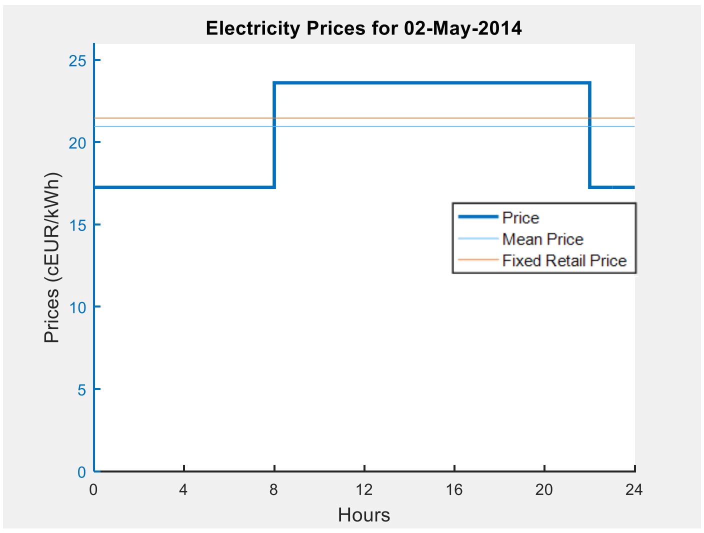


Figure 25: Bihourly profile for May 2, 2014

4.5 Bill and PAR Optimizations routine

As it will be explained more deeply later, the optimizations are organized in two levels. The first one is called *Bill optimization* and it can be considered a local optimization. In this one, they are searched solutions for each household to minimize the cost of the total energy needed to refill the EV's battery. These optimizations are executed only once a day for each EV. They return the power profile, which goes from the start to the end charging time selected by the user, that minimizes the cost of the recharge.

The second level instead is called *PAR optimization* and operates a global optimization. Indeed, the results coming from the previous level are handled together and partially adjusted in order to flatten the power profile of the MV/LV transformer that supplies the zone (i.e. Peak to Average Ratio minimization). This optimization encompasses all the vehicles currently in charge and corrects their charging profiles from the current up to the latest end charging time. The results are updates anytime a new EV makes a request for start charging.

4.5.1 An example of optimization

A clearer explanation about how the optimizations are carried on can be furnished with the help of the simple scenario created in Fig.26:

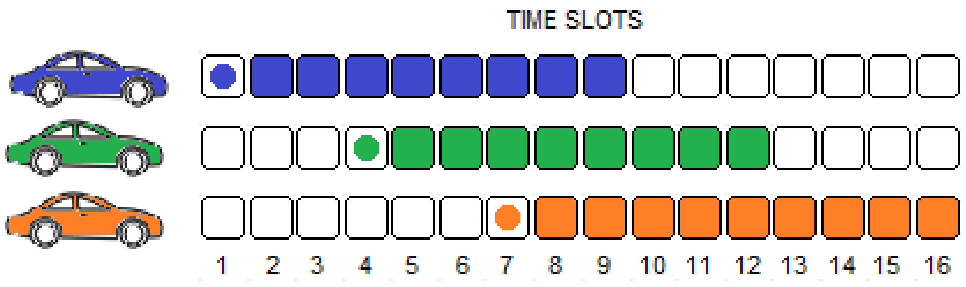


Figure 26: an example of optimization scheme

Let’s suppose we have only three EVs instead of three fleets. Time is split in 16 slots of 15 minutes each. The scenario would evolves as follows:

1) The blue EV arrives in time slot 1 and requests to be charged within 9 slots. The Aggregator records this request and sends it to the Coordinator. This one runs the two level optimization and authorizes the charge of the blue EV on the basis of the power profiles resulted. These refer to the whole time interval made available by the blue EV user, so from time slot 2 to 9
2) Nothing changes until the arrival of the green EV in time slot 4. After having received the request this vehicle sent through its related Aggregator, the Coordinator computes firstly the green EV minimum cost power profile. Secondly, it performs the *PAR optimization* which now includes both vehicles blue and green. In particular the green profile is rescheduled entirely from time slot 5 to 12, whereas the blue one is possibly readjusted only from slot 5 to 9. Indeed power from time slot 2 to 4 has been already supplied

3) The same steps are repeated for the orange EV as soon as it arrives in time slot 7. This time *PAR optimization* operates a rescheduling from time slot 8 to 16. In doing so, the charging power profiles of blue and green EVs are rearranged respectively from time slot 8 to 9 and from 8 to 12, whereas the orange one is entirely rescheduled.

4.5.2 Two levels of optimization management

The goals of the two optimizations have internally a contrasting nature. Indeed, the first level of *Bill optimization* tends to concentrate the charging profiles of EVs in the time slots where electricity cost is lower. This attitude yields automatically an increase on the PAR, as all vehicles are pushed to charge during the hours of minimum electricity cost.
Conversely, *PAR optimization* pulls the results towards the levelling of the total power profile, and substantially it would disregard the cost optimization problem.
Hence, the power profiles originated by the *Bill optimization* must not be altered too much by the second level of optimization, otherwise they would have been calculated in vain. For this purpose the study in [35] assesses the benefit can be obtained in terms of PAR minimization by deviating from the minimum cost solution of a certain value δ. Namely the charging power profile of each EV can be readjusted in such a way to cause, at most, an increment of the electricity bill of a certain delta with respect to the solution of minimum cost.
This philosophy has been thus implemented on the test bench, although it introduces a criticality. Indeed, this δ constitutes an additional constraint for the second level of optimization and not always the Matlab solver succeeds in finding a solution. Hence, the problem is that if delta is excessively enlarged the results might differ too much from the optimal cost solution, but, in the other hand, if δ is too strict the optimizer could not find a solution.
Thereby the necessity to opportunely adapt δ depending on the case. Then, in the Coordinator script it has been included a part of code that, if during an iteration the *Par optimization* fails, it automatically increases the value of delta in order to relax the constraints of the optimization. The initial value of δ, as well as the step by which it is increased in case of failure, can be chosen by the user before the beginning of the simulation.
It is worth to remark that if this step is kept short, then finding a solution might require several reiterations of *Par optimization*. Although the possibilities to remain in the neighborhood of the

minimum cost solution is higher, this choice may results impractical whenever it is required a certain velocity of the iterations (as discussed in section 4.4.2). Therefore, if the Coordinator operations have to be accomplished rapidly it is better to make δ increase by wider steps, in order to enhance the likelihood of finding an optimal PAR solution within few reiterations.

5 The optimizations

The process of optimizing involves the research of a minimum (or equivalently a maximum) of a specific function called *Objective function*. As mentioned before the optimization conducted within this work has two goals:

1) minimizing the price that drivers have to pay to fill their EVs' batteries
2) smoothing the load profile curve as seen by the transformer that supplies the area.

Such a problem has been treated as a *bilevel optimization*, where the internal level solution represents actually a constraint for the upper level. The decision variables are mainly constituted by the power to be supplied to the EVs during the daytime, which represent surely time-shiftable loads. The type of variables influences the choice of the optimization solving method, since depending on it one formulation results more suitable than others.
An important aspect to highlight is that in this work the optimizations performed are of type *deterministic*, since the future consumption and generation power profiles of each household are supposed to be known in advance when starting the optimizations, for the whole time interval these refer to.

5.1 MILP choice for DSM applications

For what regards the problem formulation, the comprehension of the type of variables involved in the case is essential. As the target is to find an optimal schedule for EVs' charging profiles, power will constitute a decision variable for the optimization problem. These power profiles will result as **curves discretized in time** on a basis of 15 minutes ($\Delta T = 15$), which might be as the one depicted in Fig.27:

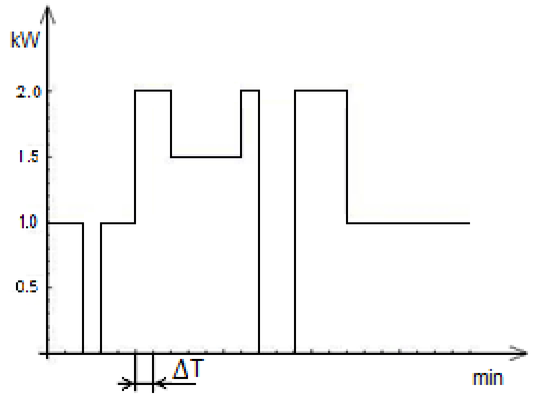


Figure 27: typical shape of a charging power profile

At each discretization interval, the action can be taken on the power delivered to the EVs is either to set a positive real value or zero. By then the necessity to introduce other decision variables to represent the state of the charge (On/Off), which will take discrete values 0/1 at each time slot.

Several studies in literature propose an optimal power profile scheduling either for DSM [37][38][35][39] or for Unit Commitment [40], and all of them agree in choosing to formulate the optimization as a Mixed-Integer Linear Programming (MILP) problem. Indeed all of the cited applications share the necessity to deal with electric devices of well time-defined power pattern. In the DSM case for instance, this means that some electrical appliances have necessarily to consume, according to their working patterns, quantities of power stepwise constant [38] (as a remind we refer to section 2.4). As some of the decision variables are integers values and some can evolve linearly in time, MILP ensues the best adequate optimization method.

5.1.1 Linear, Integer and Mixed-Integer Linear programming

The objective function is generally defined on a domain that can be either finite or infinite. When the optimization problem is constrained it can be formulated as follows:

$$min\{f(\boldsymbol{x}) : \ \boldsymbol{h}(\boldsymbol{x}) = 0, \boldsymbol{g}(\boldsymbol{x}) \leq 0\} \qquad (5\text{-}1)$$

where $\boldsymbol{x} \in R^n$ represents the finite set of variables which vary on the domain of the objective function $f : R^n \rightarrow R^1$. The dimension of such a domain is 'n'.

Yet $\boldsymbol{h} : R^n \rightarrow R^m$ and $\boldsymbol{g} : R^n \rightarrow R^q$ constitute respectively the equality and inequality constraints set of functions. When $f(\boldsymbol{x})$, $\boldsymbol{h}(\boldsymbol{x})$ and $\boldsymbol{g}(\boldsymbol{x})$ are linear on $\boldsymbol{x}$ then the model is definable as a *Linear Program (LP)* [41].

In this sense *Integer Programming (IP)* represents a subset of LP. This is due to the fact that IP maintains all the characteristics of the linear case except for one point: the solution of LP has to be constrained to integer values. Therefore, in IP $\boldsymbol{x}$ is exclusively composed by integers. The advantages in using IP are in terms of flexibility and capability of rigorous constraints modeling [42]. The drawback instead is due to the precision of the findable solution: the best integer programming solution can only place near the best linear one.

So, as an example, in case of the 2 dimension optimization problem shown in Fig.28, where x_1 and x_2 are the decision variables and the grey zone represents the feasible region, the best IP solution (green point) can only approximate the best LP solution (red square) since it has to be integer.

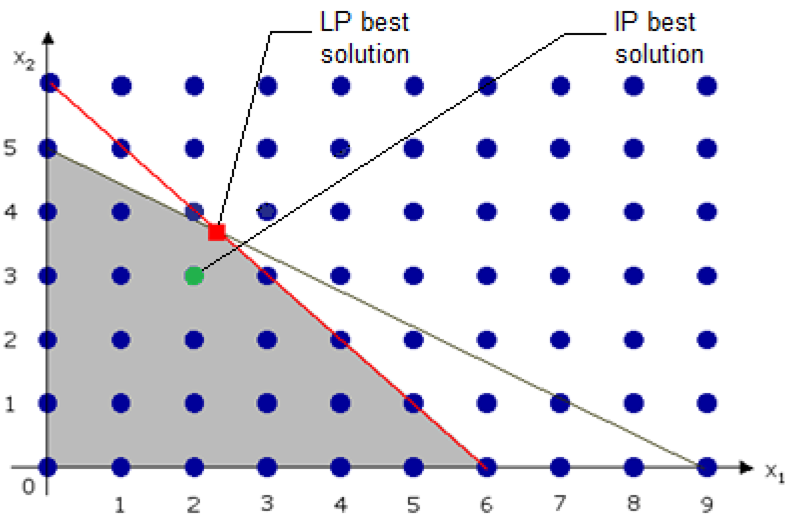


Figure 28: an example of IP and LP best solutions[43]

However, by means of *Lagrangian Relaxation* it is possible to 'relax' the integer variables as continuous ones, allowing finding the best LP solution starting from the corresponding IP problem[44][45].

Hence, finally *mixed-integer linear programming* relates to problems where a part of variables are restricted to be integers, whereas another part is not. In matrix notation ($f(\boldsymbol{x}) = c\boldsymbol{x}$) it can be formulated as follows [41]

$$\begin{gathered}\min c\boldsymbol{x} + h\boldsymbol{y} \\ subject\ to\ A\boldsymbol{x} + G\boldsymbol{y} \leq b \\ x \geq 0 \\ y \geq 0\ and\ y \in \mathbb{Z}\end{gathered} \tag{5-2}$$

where

- c is a row vector of dimension *n*, h is a row vector of dimension *p*
- A and G are respectively an *m* by *n* and an *m* by *p* matrices
- $\boldsymbol{x}$ is a column vector of dimension *n* including non-integer variables and unknowns
- $\boldsymbol{y}$ is a *p*-dimension columns vector of integer variables
- b is a column vector of dimension *m.*

5.1.2 YALMIP - CPLEX

The Matlab toolbox used to conduct the optimization is YALMIP. This was originally aimed for semi-definite programming (SDP) and linear matrix inequalities (LMI). Thanks to successive developments the toolbox has been made adequate to support many mathematical control problems, among which the MILPs [46]. To accomplish its calculations YALMIP makes use of different solvers, and, depending on the type of problem it has to deal with it, it is able to select automatically the best suitable one. In case of MILP the most fitting solver results to be CPLEX.

5.2 Bilevel optimization problem formulation

A *bilevel optimization* problem is characterized by having an outer problem which contains an inner one. The external level refers to the upper level task, whereas the internal level points to the lower level task [6].

A generic formulation for bilevel optimization problem can be done as [41][6]:

$$\begin{aligned}
&\min F(x_u, x_l) \\
&subject\ to\ G_k(x_u, x_l) \leq 0, \quad k \in K \\
&\qquad \min f(x_u, x_l) \\
&\qquad subject\ to\ g_j(x_u, x_l) \ \leq 0, \ \ j \ \in J \\
&\qquad\qquad x_u \in X_u \ , \ x_l \in X_l
\end{aligned} \tag{5-3}$$

where

- x_u and x_l are respectively the upper and the lower level decision variables vectors
- F and f represent the objective functions of the upper and lower level respectively
- G_k and g_j are the inequality constraints again at the external and internal level
- X_u and X_l represent the sets of bound constraints respectively for the outer and inner level decision vectors.

5.2.1 Decision variables

The decision variables of the problem are substantially related to the behavior of the EVs' recharges. Since the model is built in discrete time they are easily determined as:

- $E_{bat} \in \Re$: quantity of energy stored in EV's battery
- $P_{chg} \in \Re$: charging power
- $x_i \in \{0,1\}$: binary variable denoting the charging state On/Off .

5.2.2 Parameters

A set of parameters have been set in order to proper configure the optimization problem. Hereunder a list of those is reported:

1) *Time slots*: it is a vector that establishes the timing of the simulation

$$J = [j_1, j_2, \dots, j_T] \qquad j_{i+1} = j_i + 1, \qquad T = 180$$

As the time is discretized in slots of 15 minutes every t_i represents a specific quarter hour of a day, starting from 00:00. It is to note that the length of J is T > 96, which is the number of time slots contained in a day. This has been done to include the optimizations which start late and extend till the following day;

2) *Block areas:* they are three matrices that group the households power profile of each block area for a specific day. They can be described as

$$B_i = \begin{bmatrix} P_{i,1}^1 & \cdots & P_{i,H}^1 \\ \vdots & \ddots & \vdots \\ P_{i,1}^T & \cdots & P_{i,H}^T \end{bmatrix} \qquad i = 1,2,3 \qquad H = 64$$

where H is the number of households in a block and i is the block area index. Every term of the matrix is the sum of the power consumed and generated $[kW]$ by a specific home in a specific time slot;

3) Cost of electricity: it is a vector containing the electricity price profile $[c€/kWh]$ of a day, according to the criterium introduced in section 4.4.5

$$C = [c_1, c_2, \dots, c_T]$$

5.2.3 Lower level objective function

The lower level objective function to be minimized is, for every household, the price of the energy needed to recharge the EV's battery. It can be formulated as follows [6]:

$$BILL_{i,k} = \min \sum_{t=1}^{T_{i,k}} c_t(P_{i,k}^t)\Delta t \qquad 1 \leq T_{i,k} \leq N$$

where k is the index of the household of the block area i, t the index of the time slot and Δt is the time slot duration. $T_{i,k}$ is the number of time slots the user i, k makes available to recharge the battery.

5.2.4 Upper level objective function

The upper level objective function to be minimized is the PAR. A mathematical formulation of PAR can be stated as the standard deviation of the power profile supplied by the transformer. Furthermore, this global optimization must respect the results of the lower level optimization, which will constitute an additional constraint to the problem. Since the latter could be too restrictive, as discussed in 4.5.2, a parameter δ is introduced to relax such a constraint in case the solver fails in finding an optimal solution to the problem. Therefore, the global objective function is

$$\min \sum_{i=1}^{3} \left\{ \sum_{t=1}^{T_{hor}} \left[\sum_{k=1}^{K_i} P_{i,k}^{t} \, \Delta t \right]^2 \right\} - \left(P_{avg}\, \Delta t\right)^2 \qquad 1 \le K_i \le H, \qquad 1 \le T_{hor} \le T$$

$$subject\ to \quad \sum_{t=1}^{T_{hor}} c_t(P_{i,k}^{t})\Delta t \le BILL_{i,k} + \delta \qquad \forall\, i,k$$

where

- T_{hor} is the time horizon set by the EV that lasts the recharge the latest
- K_i is the number of EVs in charge during T_{hor} in the block area i
- P_{avg} is the average power delivered by the transformer during T_{hor}
- $BILL_{i,k}$ is the electricity bill of household i,k which is recalculated, at each iteration, from the current time slot till $T_{i,k}$.

It is worth to remember that the global optimization is performed every time new EVs' requests arrive to the coordinator. The latter has been designed to update, at each iteration, the parameters, the objective functions and the constraints related to each household, in order to properly take into account only the EVs which are actually on state of charge from the current time till their end of charge.

5.2.5 Constraints

In this paragraph they are described all the constraints of the optimization problem. It is to notice that the set of constraints apply equally to the lower and the upper level functions. For the sake of simplicity and clarity they are formulated solely for one household. Hence, the set of constraints is listed below:

1) *Power constraints:* they are two limits and they refer both to the household and the transformer level. For what concerns the domestic power, it must not exceed the maximum value allowed by the electric installation (P_{max}). Indeed, for safety reasons the rated power of all the security circuits (i.e. breakers, cables) must be respected. Thus, one has:

$$P_{i,k}^{t} \leq P_{max} \qquad \forall t, i, k$$

The same considerations are valid for the transformer. Though, the power constraint on the transformer applies only to the global function:

$$\sum_{i=1}^{3}\left[\sum_{k=1}^{K_i} P_{i,k}^{t}\right] \leq P_{tr,max} \qquad \forall t$$

where $P_{tr,max}$ is the rated power of the transformer.

2) *State of charging:* this is the constraint that reflects the user preferences in terms of charge timing. Wherefore, it bounds de decision variable x_i to the user selected time slots for charging:

$$x_{i,t} \leq U_{i,t} \qquad \forall i, t$$

with U_i being the vector of the preferences. Its components are 1 where the user decides to be available to the recharge and 0 otherwise

3) *Energy constraints:* for each EV these constraints impose the final energy recharged in battery to be equal 'at least' to the one requested by the user. For the upper level function one has:

$$E_{req_{i,k}} \leq \sum_{t=1}^{T_{i,k}} P_{i,k}^{t}\,\Delta t \leq E_{req_{i,k}}(1+\alpha) \qquad \forall i,k$$

where $E_{req_{i,k}}$ is the energy request of the EV k of the fleet i, whereas α is a maximum margin the solver can take to minimize PAR. This margin is chosen by the user, which could accept to charge a little more its vehicle to support the EM operations. It would constitute a service the customers may provide to the grid, for which economic agreement can be foreseen. This do not applies to the lower level function where the constraint is simply:

$$E_{req_{i,k}} \leq \sum_{t=1}^{T_{i,k}} P_{i,k}^{t}\,\Delta t \qquad \forall i,k$$

4) *Evolution of energy:* this constraint fixes the evolution with which the energy supplied to the batteries can evolve:

$$E_{i,k}^{t+1} = E_{i,k}^{t} + P_{i,k}^{t}\,\Delta t\,\eta_{i,k} \qquad \forall i,k \quad \forall t \in \{t_c+1,\dots,T_{i,k}\}$$

where $\eta_{i,k}$ is the efficiency of the charger and $E_{i,k}^{t}$ is the quantity of energy stored in the battery of the EV k of the fleet i at the time t. Yet, t_c indicates the current time slot of the simulation and corresponds to the start-charging time slot, for a generic i,k EV, only for the first level optimization and for the first iteration of the second level one. Note that the energy evolution begins always from the second time slot with respect to t_c and ends at the last time slot set by the user ($T_{i,k}$)

5) *Charging power:* this constraint reflects the fact that the charging rate can neither exceed the maximum power fixed by the type of connection nor going below a certain threshold. The latter agrees with the standard IEC 61851 [47] which states that when an EV is in charge there is a minimum value of current that must be supplied:

$$x_{i,k}^t \, P_{ch,min} \leq P_{i,k}^t \leq x_{i,k}^t \, P_{ch,max} \qquad \forall t, i, k$$

where $P_{ch,min}$ and $P_{ch,max}$ are the minimum and maximum charging power rate allowed by the specific charging station.

Lastly, it is to mention that no constraints have been set on the state of charge of the batteries, since the users are supposed to require autonomously proper quantities of energy not harmful for the state of health of the cells.

5.3 A simulation example

In this paragraph they are reported and commented the processes and results of an optimization. Before that, a description of the hardware components utilized to realize the TB is given.

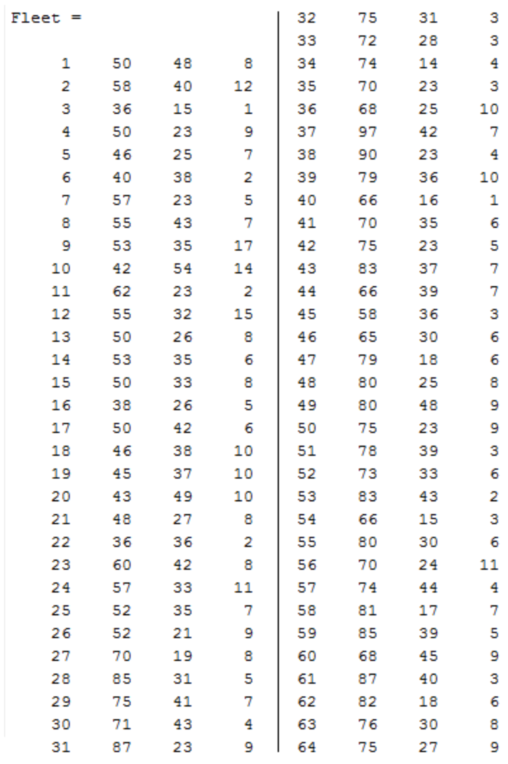
5.3.1 Hardware description

The simulations have been run on a Intel® Core™ i7-7500U CPU @2.70GHz 2.90GHz, 8 GB RAM computer with Windows 10 Home, 64-bit as operating system.
The network created is a WPAN, whereas the radio modules used are XBee Series 1, from Digi International [25].
The four radios supporting the operations of the Coordinator and the Aggregators are connected to the computer through serial ports and USB cables.
Simulations are performed on Matlab R2016a and optimizations are computed by the Matlab tool YALMIP, through its solver CPLEX.

5.3.2 Process and results

The first step to start the TB is to run the Coordinator. After being started, Coordinator loads the SLP, the wholesale market prices and the irradiance profile for the whole year 2014. Subsequently, it asks the user to set its preferences for what regards the pricing method (CP,

DAP, TOU, Bi-hourly). Once set the tariff program, the Coordinator randomly selects a day of the year 2014 and creates the vectors of price and power consumption for all the households of the area. Furthermore, it creates the time vector (we remind that each day is subdivided in 96 time slots of 15 minutes each) and the parameters for the post processing part.
At this point Coordinator enters in 'receiving mode', and keeps waiting for messages arrivals.
The second step to do is running the 3 Aggregators. This action should be done synchronously, since the aggregators update the current time slot everywhen the Matlab internal clock marks a time multiple of 30 seconds (as already said, this interval is modifiable though).
The first action each Aggregator performs is the creation of the matrix of the fleet characteristics, following the criteria discussed in 4.2. An example of fleet generated is shown in Fig.29



```
Fleet =                          |   32    75    31     3
                                 |   33    72    28     3
     1    50    48     8         |   34    74    14     4
     2    58    40    12         |   35    70    23     3
     3    36    15     1         |   36    68    25    10
     4    50    23     9         |   37    97    42     7
     5    46    25     7         |   38    90    23     4
     6    40    38     2         |   39    79    36    10
     7    57    23     5         |   40    66    16     1
     8    55    43     7         |   41    70    35     6
     9    53    35    17         |   42    75    23     5
    10    42    54    14         |   43    83    37     7
    11    62    23     2         |   44    66    39     7
    12    55    32    15         |   45    58    36     3
    13    50    26     8         |   46    65    30     6
    14    53    35     6         |   47    79    18     6
    15    50    33     8         |   48    80    25     8
    16    38    26     5         |   49    80    48     9
    17    50    42     6         |   50    75    23     9
    18    46    38    10         |   51    78    39     3
    19    45    37    10         |   52    73    33     6
    20    43    49    10         |   53    83    43     2
    21    48    27     8         |   54    66    15     3
    22    36    36     2         |   55    80    30     6
    23    60    42     8         |   56    70    24    11
    24    57    33    11         |   57    74    44     4
    25    52    35     7         |   58    81    17     7
    26    52    21     9         |   59    85    39     5
    27    70    19     8         |   60    68    45     9
    28    85    31     5         |   61    87    40     3
    29    75    41     7         |   62    82    18     6
    30    71    43     4         |   63    76    30     8
    31    87    23     9         |   64    75    27     9
```

Figure 29: an example of fleet characteristics creation. It is actually one only matrix, it has been split in two parts for graphic needs

As it can be noticed it is a 4x64 matrix, each row representing a specified EV of the fleet.

The first column represents the EV identification index, the second the arrival time (in time slots), the third the time made available for the charge (again in time slots) and the fourth the quantity of energy needed (kWh). Now, as aforementioned, the Aggregators update the current time slot every 30 seconds, starting from 1. If in the arrivals time column of the matrix 'fleet' it is found a correspondence with the current time slot, then that EV is included in the set of the energy requests to be sent to the Coordinator.

This process keeps iterating for the whole day simulated, within which they are performed all the algorithms described in Chapter 4 (e.g. *transmission failure management* (4.3.1) and *two level optimization management* (4.5.2)). At the end of every iteration the Coordinator shows to the user the updates on the transformer power profile.

At the end of the day, once all the power profiles of all the EVs have been set, the Coordinator shows the last resulting transformer power profile. Furthermore it depicts for every EV:

- the data relating to the charge: arrival/departure time, energy requested/energy actually recharged
- the comparison between the two levels of optimizations: the bill resulting from *Par optimization* is compared with the one resulting from *Bill optimization*.

Results coming from the simulation on October 24 are shown in the following figures:

- Electricity prices:

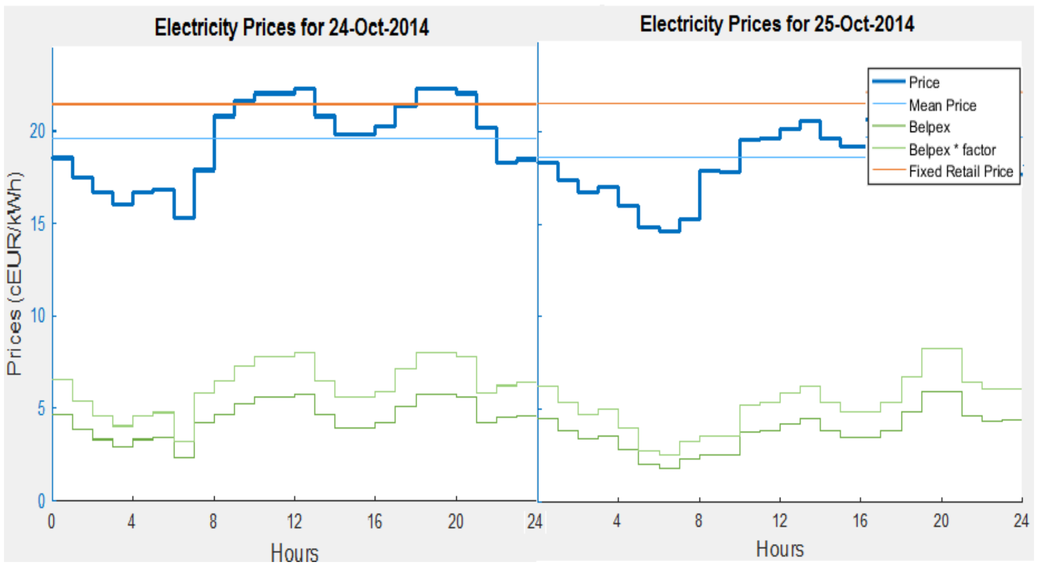


Figure 30: electricity price profiles in days 24-25 October 2014

To remember the meaning of the various curves in Fig. 30 we refer to the section 4.4.5.

- Transformer power profile:

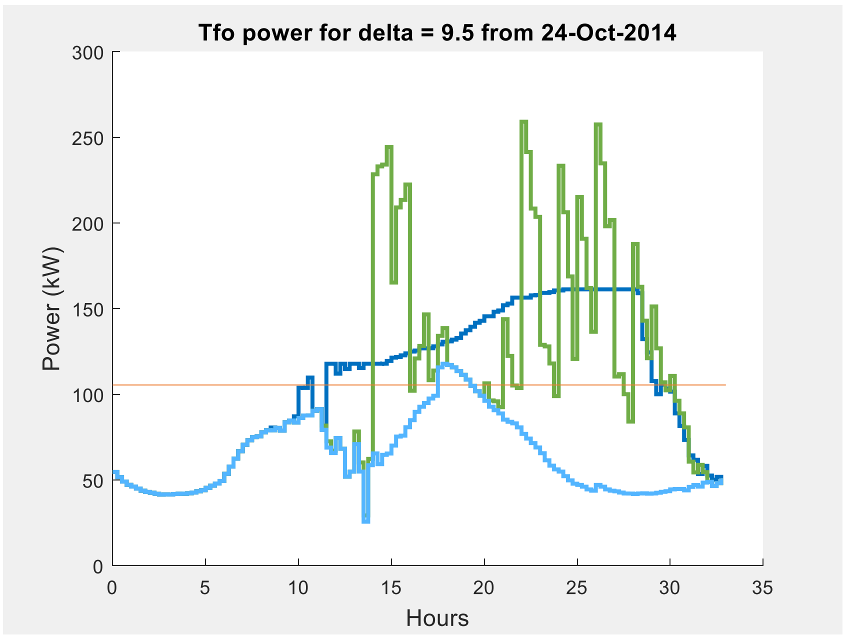
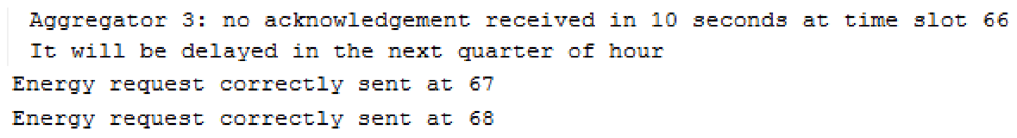


Figure 31:MV/LV transformer power profiles. In light blue the baseload, in green the profile obtained with the first level of optimization (cost optimization), in dark blue the profile obtained with the second level of optimization (PAR optimization) and in orange its average value.

By comparing Fig.31 and Fig.30 it can be noticed how the cost optimization actually pushes the recharge to be done during the hours of minimum cost (green curve of Fig.31). Then the second level of optimization flattens the cost-optimized profile, aiming the PAR reduction (dark blue curve in Fig.31). The light blue curve in Fig.31 instead represents the baseload profile, namely the sum of the power profiles of every household in the area which does not take into account the recharge of the EVs. Note also that the parameter δ (section 4.5.2) has increased until 9.5. This is due to the fact that, in a particular iteration within the day simulated, CPLEX did not succeed in finding a solution to *PAR optimization* with a lower value of delta. So it has been operated the constraint relaxation discussed in section 4.5.2. Although this value seems to be high, as often has happened in these cases, the addition of the following energy requests from the EVs arrived later has permitted a wider opportunity of rescheduling for *PAR optimization*. Indeed, the arrival of new EVs constitutes itself a constraint relaxation for the problem, as the time slots number available to operate the rescheduling increases. Hence finally, despite the fact that δ has assumed a high value during the iterations, the costs of the recharge resulting from the two levels of optimization are maintained close, being the rate of increase of the bill equal to 5% at most (Fig.33).

However, experimental trials have revealed that it could happen that a particularly unlucky combination of fleets' recharge requests and baseload profile led to a larger difference of bills between the two levels of optimization, even 100% or more! Although it has happened rarely in the simulations performed, this is surely an aspect that can be further cared and investigated in future possible developments of the TB.

Yet, they are interesting the results showed to the user by the Aggregators during the iterations Fig.32:

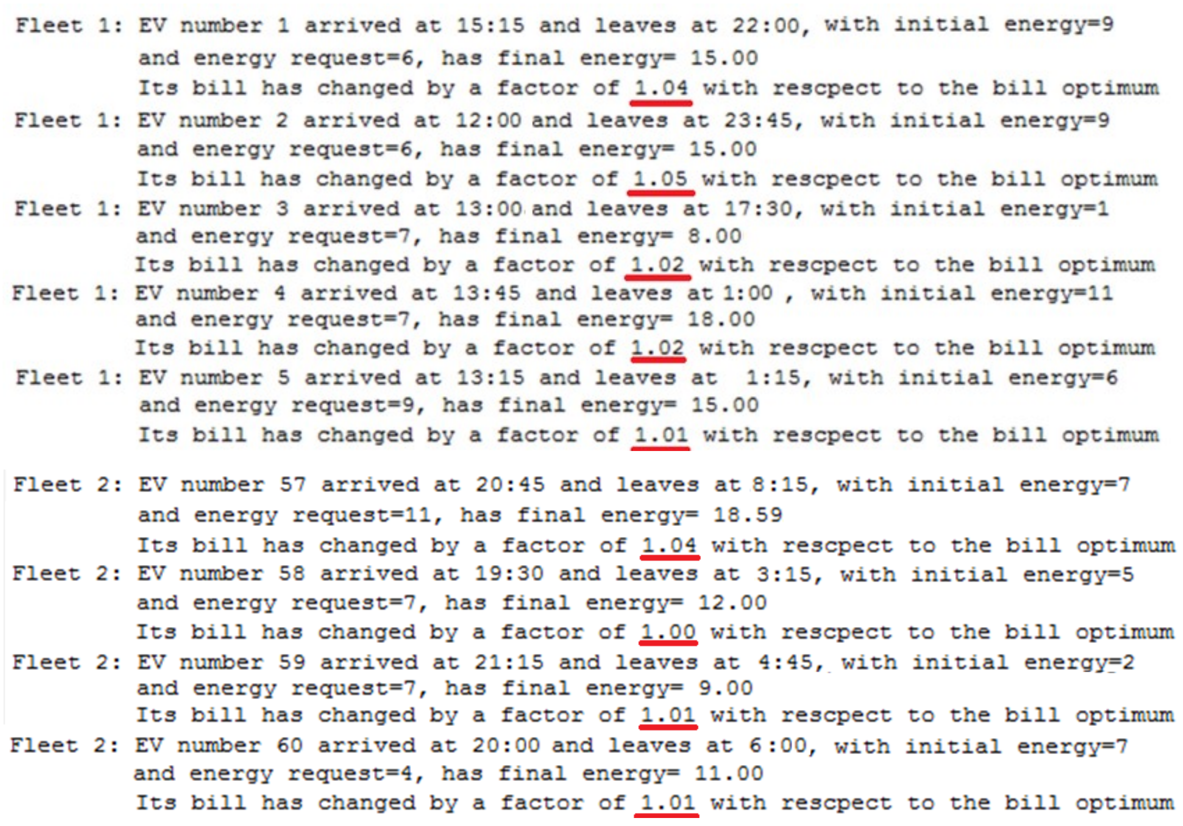


```
 Aggregator 3: no acknowledgement received in 10 seconds at time slot 66
 It will be delayed in the next quarter of hour
Energy request correctly sent at 67
Energy request correctly sent at 68
```

Figure 32: Aggregator displayed results. The time is expressed in time slots (section 4.4.2)

In Fig. 32 it is visible how one Aggregator displays the result of the *transmission failure management* algorithm discussed in section 4.3.1.

Finally, in Fig.33 they are worth to be shown the results displayed by the Coordinator at the end of the day of simulation.



```
Fleet 1: EV number 1 arrived at 15:15 and leaves at 22:00, with initial energy=9
         and energy request=6, has final energy= 15.00
         Its bill has changed by a factor of 1.04 with rescpect to the bill optimum
Fleet 1: EV number 2 arrived at 12:00 and leaves at 23:45, with initial energy=9
         and energy request=6, has final energy= 15.00
         Its bill has changed by a factor of 1.05 with rescpect to the bill optimum
Fleet 1: EV number 3 arrived at 13:00 and leaves at 17:30, with initial energy=1
         and energy request=7, has final energy= 8.00
         Its bill has changed by a factor of 1.02 with rescpect to the bill optimum
Fleet 1: EV number 4 arrived at 13:45 and leaves at 1:00 , with initial energy=11
         and energy request=7, has final energy= 18.00
         Its bill has changed by a factor of 1.02 with rescpect to the bill optimum
Fleet 1: EV number 5 arrived at 13:15 and leaves at  1:15, with initial energy=6
         and energy request=9, has final energy= 15.00
         Its bill has changed by a factor of 1.01 with rescpect to the bill optimum

Fleet 2: EV number 57 arrived at 20:45 and leaves at 8:15, with initial energy=7
         and energy request=11, has final energy= 18.59
         Its bill has changed by a factor of 1.04 with rescpect to the bill optimum
Fleet 2: EV number 58 arrived at 19:30 and leaves at 3:15, with initial energy=5
         and energy request=7, has final energy= 12.00
         Its bill has changed by a factor of 1.00 with rescpect to the bill optimum
Fleet 2: EV number 59 arrived at 21:15 and leaves at 4:45, with initial energy=2
         and energy request=7, has final energy= 9.00
         Its bill has changed by a factor of 1.01 with rescpect to the bill optimum
Fleet 2: EV number 60 arrived at 20:00 and leaves at 6:00, with initial energy=7
         and energy request=4, has final energy= 11.00
         Its bill has changed by a factor of 1.01 with rescpect to the bill optimum
```

Figure 33: results displayed by the EM at the end of a day of simulation for fleet 1 and fleet 2. The rates of bill increase between the two levels of optimization are underlined

From the above figure it can be noticed how the EM reports the information about the EVs' recharge and the rates of increase on the bill caused by the second level of optimization, with respect to the optimal cost solution.

6 Conclusions and future works

The integration of small-size distributed generation and the increase of electricity consumption have made very challenging the operations of the power system. In this regard, the actuation of price-based DR programs in residential areas can considerably assist the DSO in its function of real-time balancing demand and production. Nevertheless, the implementation of DR programs necessitates the development of new communication infrastructures which, contrary to the existing ones, could reliably guarantee a tight interoperability between the various elements of the system.

The objective of this thesis was to design a test bench able to emulate the control of smart electrical loads within a price-based DR program. The aim were firstly the investigation of the possible communication protocols suitable for residential areas DSM programs and, secondly, the physical and real-time implementation of the optimization principles developed in the work of B. Mattlet in [6]. The latter implementation has involved in a first step the creation of a communication network and, successively, the programming of a software able to emulate, in continuous time and in a stand-alone fashion, the operations of an Energy Management System. For what concerns the hardware part, the adoption of the ZigBee communication protocol and the compliant Xbee radio modules has constituted a valuable solution.

Moreover, for what regards the software, satisfactory results have been obtained considering that:

- test bench realized represents a valid tool to support the simulations of DSM scenarios. Indeed, even if the software has been exclusively developed for emulate the recharge of EVs' fleets, the same hardware structure and control philosophy can be implemented for any type of smart load. However, if in the case of other SI loads the adaptation is straightforward, in case of SNI loads the code necessitates to be adjusted accordingly with the specificities of the particular application

- the problem of the transmission failure has been appropriately tackled by exploiting the study of the working principle of XBee modules in an algorithm created on purpose. Indeed, the experimental trials reveal a very low rate of fails in communication even when the system is stressed by the velocity of the iterations
- the test bench operations have been scaled in time in order to permit a fast execution of the simulations. The ways to opportunely select the scaling factor, avoiding misfunctioning of the code, have been discussed as well
- the optimization principles developed in [6] have been profitably implemented in real-time. The way to configure all the settings, parameters, constrains and variables has been transformed from static to dynamic, in order to adapt them during the succession of the iterations in time.

Future developments of the software are foreseen. First of all the optimization philosophy can be changed from deterministic to stochastic, as it would make the EM usable in reality. Indeed, in the current version of the EM optimizations are based on a previously acquired knowledge of the future consumption profiles. A stochastic approach could instead make a prevision of those profiles based on statistical data on the consumer behaviour collected in the past.
Secondly, the emulator effectiveness could be enhanced by including models of other appliances in the scenario. Additionally, new ways to compute the optimizations in case YALMIP fails to find a solution within a reasonably small value of δ (section 4.5.2) can be explored.
Lastly, other interesting developments could arise in case the working philosophy of the test bench would apply to the other levels of the power system.

Looking to the next future, it is clear that the energy world is evolving towards a more and more effective exploitation of renewable resources. For this reason the power network is going to incorporate a huge number of decentralised, unpredictable and small-size production units. In this framework, the field of DSM clearly deserves further commitment and research.

Bibliography

[1] European Commission, “DIRECTIVE OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL on the deployment of alternative fuels infrastructure,” *Off. J. Eur. Union*, vol. 12, no. April, pp. 1–38, 2014.

[2] European Commission, “COP21 UN Climate Change Conference, Paris.” [Online]. Available: https://ec.europa.eu/commission/priorities/energy-union-and-climate/climate-action-decarbonising-economy/cop21-un-climate-change-conference-paris_en.

[3] D. Müller *et al.*, “Demand side management for city districts,” *Build. Environ.*, vol. 91, pp. 283–293, 2015.

[4] G. Strbac, “Demand side management: Benefits and challenges,” *Energy Policy*, vol. 36, no. 12, pp. 4419–4426, 2008.

[5] A. Mahmood, N. Javaid, and S. Razzaq, “A review of wireless communications for smart grid,” *Renew. Sustain. Energy Rev.*, vol. 41, pp. 248–260, 2015.

[6] B. Mattlet, “Optimal Scheduling of Electric Loads in Residential Area using a Bi-level Optimization under Real Time Pricing.”

[7] J. Tant, F. Geth, D. Six, P. Tant, and J. Driesen, “Multiobjective battery storage to improve PV integration in residential distribution grids,” *IEEE Trans. Sustain. Energy*, vol. 4, no. 1, pp. 182–191, 2013.

[8] Trilliant, “Smart grid.” [Online]. Available: https://trilliantinc.com/smart-grid.

[9] Creg, “Evolution of ancillary services needs to balance the Belgian control area towards 2018,” no. May, p. 56, 2013.

[10] B. Mattlet and J. C. Maun, “Assessing the benefits for the distribution system of a

scheduling of flexible residential loads," *2016 IEEE Int. Energy Conf. ENERGYCON 2016*, 2016.

[11] B. Dupont, C. De Jonghe, K. Kessels, and R. Belmans, "Short-term consumer benefits of dynamic pricing," *2011 8th Int. Conf. Eur. Energy Mark. EEM 11*, no. May, pp. 216–221, 2011.

[12] S. Stoft, *Power System Economics: Designing Markets for Electricity*. New York: Wiley-IEEE Press, 2002.

[13] U. D. of Energy, "Benefits of Demand Response in Electricity Markets and Recommendations for Achieving Them - A Report to the United States Congress Pursant to Section 1252 of the Energy Policy Act of 2005," *United States Congr. Purs. To Sect. 1252 Energy Policy Act 2005*, no. February, p. 122, 2006.

[14] F. Ivan, "Controlling Smart Loads using ZigBee," 2015.

[15] A. A. Khan, S. Razzaq, A. Khan, K. Fatima, and Owais, "HEMSs and enabled demand response in electricity market: An overview," *Renew. Sustain. Energy Rev.*, vol. 42, pp. 773–785, 2015.

[16] M. Perilleux, "Simulation of optimal scheduling of interruptible and shiftable household loads under real-time pricing," p. 68, 2014.

[17] Laborelec, "SHEL 2012 : flexibility potential of domestic appliances Verified by," pp. 1–44, 2013.

[18] N. Saputro, K. Akkaya, and S. Uludag, "A survey of routing protocols for smart grid communications," *Comput. Networks*, vol. 56, no. 11, pp. 2741–2771, 2012.

[19] Dreamstime, "Conceptual model of Smart Grids." [Online]. Available: https://www.dreamstime.com/stock-illustration-conceptual-model-smart-grid-concept-industrial-devices-connected-network-image83214338.

[20] M. Kuzlu, M. Pipattanasomporn, and S. Rahman, "Communication network requirements for major smart grid applications in HAN, NAN and WAN," *Comput. Networks*, vol. 67, pp. 74–88, 2014.

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

[22] M. Zeghdoud, P. Cordier, and M. Terr??, "Impact of clear channel assessment mode on the performance of ZigBee operating in a WiFi environment," *2006 1st Work. Oper.*

Community Networks, OpComm 2006, vol. 2, 2006.

[23] Z. Alliance, “Official Web Page.” [Online]. Available: www.zigbee.org.

[24] KnowledgeBase, “ZigBee Network Concepts.” [Online]. Available: https://mmbnetworks.atlassian.net/wiki/spaces/SKB/pages/39518242/Zigbee+Network+Concepts.

[25] Digi International, “Xbee product.” [Online]. Available: https://www.digi.com/xbee.

[26] Digi International, “XBee API mode.” [Online]. Available: http://docs.digi.com/display/RFKitsCommon/XBee+API+mode.

[27] Digi International, “RF kits common.” [Online]. Available: http://docs.digi.com/display/RFKitsCommon/Frame+st.

[28] “Arduino-Xbee.” [Online]. Available: http://wiki.t-o-f.info/Arduino/Xbee.

[29] G. M. Fetene, “A Report on Energy consumption and Range of Battery Electric Vehicles Based on Real - World Driving Data,” no. September, pp. 1–42, 2014.

[30] G. Pasaoglu *et al.*, *Driving and parking patterns of European car drivers - a mobility survey*. 2012.

[31] “Implementation of Mesh Networking.” [Online]. Available: https://mmutrig2.wordpress.com/2016/01/24/implementation-of-mesh-networking-explained/.

[32] Y. Tang, Z. Wang, D. Makrakis, and H. T. Mouftah, “Interference aware adaptive clear channel assessment for improving zigbee packet transmission under Wi-Fi interference,” *2013 IEEE Int. Conf. Sensing, Commun. Networking, SECON 2013*, pp. 336–343, 2013.

[33] Digi International, “XBee ® /XBee-PRO ® RF Modules,” *Prod. Man. v1.xEx-802.15.4 Protoc.*, pp. 1–69, 2009.

[34] Synergrid, “Synthetic load profile.” [Online]. Available: http://www.synergrid.be/index.cfm?PageID=16896&language_code=FRA.

[35] B. Mattlet and J. C. Maun, “Assessing the benefits for the distribution system of a scheduling of flexible residential loads,” *2016 IEEE Int. Energy Conf. ENERGYCON 2016*, 2016.

[36] A.-H. Mohsenian-Rad, V. W. S. Wong, J. Jatskevich, R. Schober, and A. Leon-Garcia, “Autonomous Demand Side Management Based on Game-Theoretic Energy

Consumption Scheduling for the Future Smart Grid," *IEEE Trans. Smart Grid*, vol. 1, no. 3, pp. 320–331, 2010.

[37] L. Yao, Z. Damiran, and W. H. Lim, "Optimal Charging and Discharging Scheduling for Electric Vehicles in a Parking Station with Photovoltaic System and Energy Storage System," *Energies*, vol. 10, no. 4, p. 550, 2017.

[38] Ziming Zhu, Jie Tang, S. Lambotharan, Woon Hau Chin, and Zhong Fan, "An integer linear programming based optimization for home demand-side management in smart grid," *2012 IEEE PES Innov. Smart Grid Technol.*, pp. 1–5, 2012.

[39] M. Fischetti, G. Sartor, and A. Zanette, "MIP-and-refine matheuristic for smart grid energy management," *Int. Trans. Oper. Res.*, vol. 22, no. 1, pp. 49–59, 2015.

[40] T. Li and M. Shahidehpour, "Price-based unit commitment: A case of Lagrangian relaxation versus mixed integer programming," *IEEE Trans. Power Syst.*, vol. 20, no. 4, pp. 2015–2025, 2005.

[41] J. F. Bard, "Practical bilevel optimization : algorithms and applications," *Nonconvex Optim. its Appl.*, vol. v. 30, p. xii, 473 , 1998.

[42] X. Guan, Q. Zhai, and A. Papalexopoulos, "Optimization based methods for unit commitment: Lagrangian relaxation versus general mixed integer programming," *Power Eng. Soc. Gen. Meet. 2003, IEEE*, vol. 2, pp. 1095–1100, 2003.

[43] M. Lin, "No Title." [Online]. Available: https://www.quora.com/What-is-the-difference-between-integer-programming-and-linear-programming.

[44] B. Gendron, "Lagrangian Relaxation in MIP," 2016.

[45] M. L. Fisher, "The Lagrangian Relaxation Method for Solving Integer Programming Problems," *Manage. Sci.*, vol. 50, no. 12_supplement, pp. 1861–1871, 2004.

[46] J. Lofberg, "YALMIP : a toolbox for modeling and optimization in MATLAB," *2004 IEEE Int. Conf. Robot. Autom. (IEEE Cat. No.04CH37508)*, pp. 284–289.

[47] A. Di Giorgio, F. Liberati, and S. Canale, "Electric vehicles charging control in a smart grid: A model predictive control approach," *Control Eng. Pract.*, vol. 22, no. 1, pp. 147–162, 2014.

Annex A : Coordinator_Xbee

```
clc
close all
clear all
```

6.1 Set up configuration

```
load_config = 0;
switch load_config
    case 0
            load WP_2014;   % Whole Sale Price EUR/MWh from Belpex webiste,
                            % hourly based (24) for one year (365 or 366)
                            % Matrix [365x24]
            load SLP_2014;  % SLP curves from webiste Synergrid quarter
                            % based : 96 sample/day*365 days = Matrix [35040x1]
            load I_2014;    % Irradiance (W/m²) at Uccle from the IRM
                            % quarter based : 96 sample/day*365 days =
                            % Matrix [35040x1]
            WP              = WP_2014;
            SLP             = SLP_2014;
            Irr             = I_2014;
            Cost.tariffs = readtable('Electrabel_tariffs_09_2015.dat');
            Cost.pricing = input ('Choice of pricing method: 0 = Constant (or Free), 1 =
DAP, 2 = ToU, 3 = Bihourly\n');

            Date.day = datenum('2014,1,1')+round((datenum('2014,12,31')-
datenum('2014,1,1'))*rand);  % it generates randomly the number of a day of year 2014.
            Date_str = datestr(Date.day);     % It generates a string in format
day/month/year  hour/minutes/seconds
            Date_day = Date_str(1:11);
            fprintf('Optimization starts running from %s \n', Date_day);
            Date.day = Date.day - datenum(2014,1,0);

     % Set Parameters %
        Set_Time_Parameters
        Set_blocks_parameters
        Set_Electricity_Price
        Set_Figure_Parameters
        save parameters;
    case 1
        load parameters;
end
```

6.2 Coordinator

```
delete(instrfindall);
c = serial('com4');
set(c,'BaudRate', 9600);
set(c,'DataBits' , 8 );
set(c,'StopBits', 1);
set(c,'InputBufferSize',1000);
set(c, 'Terminator', '');
set(c, 'FlowControl', 'none');
fopen(c);

Set_Matrices_Vectors_Xbee
Plot_Electricity_Price
Elapsed_time = [];
```

6.3 Optimize bill

```
delta =0.5;

while run==0;
% Inquiring if new EV are connected
    message = [];

    if c.BytesAvailable ~= 0
      tic
        receiving = [];
        pause(0.5);
        receiving = [receiving;fread(c,c.BytesAvailable)];

        pack1 = [];
        pack2 = [];
        pack3 = [];
        if length(receiving)>= 13              % to avoid to consider bad messages who
are shorter than the minimum length of 13
        message = [message, receiving];
        start_delimiter = find (message==126); % It may happen more messages arrive at
```

```
same time!

        Number_of_requests = length (start_delimiter);
        if message(length(message))== 126      % It may happen that the checksum is
equal to the stard delimiter
            Number_of_requests = Number_of_requests-1;
        end
        if Number_of_requests > 1
            for j = 1:(Number_of_requests-1)
                split_message = message(start_delimiter(j):start_delimiter(j+1)-1);
                identifier = split_message (6);
                switch identifier
                    case 1
                           for i = 9:4:(length(split_message)-4)
                                pointer = split_message(i);
                                if Fleet1 (pointer,1)==0   % It may happen that
                                %the message is arrived although the aknowledgment
                                %frame is not sent back to the sender, so
                                %it sends twice.In this way the message is
                                %stored only if the slot is vacant
                                Fleet1 (pointer,:) = split_message (i:(i+3));
                                pack1 = [pack1; Fleet1(pointer,:)];
                                end
                            end
                    case 2
                            for i = 9:4:(length(split_message)-4)
                                pointer = split_message(i);
                                if Fleet2 (pointer,1)==0
                                Fleet2 (pointer,:) = split_message (i:(i+3));
                                pack2 = [pack2; Fleet2(pointer,:)];
                                end
                            end
                    case 3
                            for i = 9:4:(length(split_message)-4)
                                pointer = split_message(i);
                                if Fleet3 (pointer,1)==0
                                Fleet3 (pointer,:) = split_message (i:(i+3));
                                pack3 = [pack3; Fleet3(pointer,:)];
                                end
                            end
                end
            end
            split_message =
message(start_delimiter(Number_of_requests):length(message));
            identifier = split_message (6);
                    switch identifier
                            case 1
                                    for i = 9:4:(length(split_message)-4)
                                        pointer = split_message(i);
                                        if Fleet1 (pointer,1)==0
                                        Fleet1 (pointer,:) = split_message (i:(i+3));
                                        pack1 = [pack1; Fleet1(pointer,:)];
                                        end
                                    end
                            case 2
                                    for i = 9:4:(length(split_message)-4)
                                        pointer = split_message(i);
                                        if Fleet2 (pointer,1)==0
                                        Fleet2 (pointer,:) = split_message (i:(i+3));
                                        pack2 = [pack2; Fleet2(pointer,:)];
                                        end
                                    end
                            case 3
                                    for i = 9:4:(length(split_message)-4)
                                        pointer = split_message(i);
                                        if Fleet3 (pointer,1)==0
                                        Fleet3 (pointer,:) = split_message (i:(i+3));
                                        pack3 = [pack3; Fleet3(pointer,:)];
                                        end
                                    end
                        end


        else
        identifier = message(6);
        switch identifier
            case 1
                    for i = 9:4:(length(message)-4)
                        pointer = message(i);
                        if Fleet1 (pointer,1)==0
                        Fleet1 (pointer,:) = message (i:(i+3));
                        pack1 = [pack1; Fleet1(pointer,:)];
                        end
                    end
            case 2
                    for i = 9:4:(length(message)-4)
                        pointer = message(i);
                        if Fleet2 (pointer,1)==0
                        Fleet2 (pointer,:) = message (i:(i+3));
                        pack2 = [pack2; Fleet2(pointer,:)];
                        end
                    end
            case 3
                    for i = 9:4:(length(message)-4)
                        pointer = message(i);
                        if Fleet3 (pointer,1)==0
                        Fleet3 (pointer,:) = message (i:(i+3));
                        pack3 = [pack3; Fleet3(pointer,:)];
                        end
                    end
        end
        end
        end
%Fleet1
        if ~isempty (pack1)
             for p = 1:size(pack1,1)
                h = pack1(p,1);
                current_time = pack1(p,2);
                Full_charge_time_1(h) = pack1(p,2)+ pack1(p,3);
                if Full_charge_time_1(h)>= time_horizon
                    time_horizon = Full_charge_time_1(h); % Time horizon in slot
                end
               Occupancy_1(h,Fleet1(h,2):Full_charge_time_1(h)) = 1;
               [Aggregator.house_1(h).bill, Aggregator.house_1(h).EV_x,
Aggregator.house_1(h).EV_E, Aggregator.house_1(h).EV_P]=
Optimization_bill(Fleet1(h,:),Time, Cost, Aggregator.house_1(h));

               % Create the profile for the whole time span

               Before_arrival_1 = zeros(1,Fleet1(h,2)-1);
               After_arrival_1 = zeros(1,Time.span-Full_charge_time_1(h));
               Aggregator.house_1(h).EV_E = [Before_arrival_1,
Aggregator.house_1(h).EV_E, After_arrival_1] ;
               Aggregator.house_1(h).EV_P = [Before_arrival_1,
Aggregator.house_1(h).EV_P, After_arrival_1] ;
               Power_profile_bill_1(h,:) = Aggregator.house_1(h).EV_P;
               Energy_profile_bill_1(h,:)= Aggregator.house_1(h).EV_E;
               Bill_cost_optimization_1(h) = Aggregator.house_1(h).bill;

               % Reiterate in case of failure

               if Energy_profile_bill_1(h,Full_charge_time_1(h))==0
                 fprintf('Bill optimization failure fleet 1 for EV num %d \n', h);
                 pause (1);
                 while Energy_profile_bill_1(h,Full_charge_time_1(h))==0
                      [Aggregator.house_1(h).bill, Aggregator.house_1(h).EV_x,
Aggregator.house_1(h).EV_E, Aggregator.house_1(h).EV_P]=
Optimization_bill(Fleet1(h,:),Time, Cost, Aggregator.house_1(h));
                       Aggregator.house_1(h).EV_E = [Before_arrival_1,
Aggregator.house_1(h).EV_E, After_arrival_1] ;
                       Aggregator.house_1(h).EV_P = [Before_arrival_1,
Aggregator.house_1(h).EV_P, After_arrival_1] ;
                       Power_profile_bill_1(h,:) = Aggregator.house_1(h).EV_P;
                       Energy_profile_bill_1(h,:)= Aggregator.house_1(h).EV_E;
                       Bill_cost_optimization_1(h) = Aggregator.house_1(h).bill;
                 end
               end
               pause (1);
       end

             end
% Fleet 2
```

```
        if ~isempty (pack2)

                for p = 1:size(pack2,1)
                h = pack2(p,1);
                current_time = pack2(p,2);
                Full_charge_time_2(h) = pack2(p,2)+ pack2(p,3);
                if Full_charge_time_2(h)>= time_horizon
                    time_horizon = Full_charge_time_2(h); % time horizon in slot
                end
               Occupancy_2(h,Fleet2(h,2):Full_charge_time_2(h)) = 1;
               [Aggregator.house_2(h).bill, Aggregator.house_2(h).EV_x,
Aggregator.house_2(h).EV_E, Aggregator.house_2(h).EV_P]=
Optimization_bill(Fleet2(h,:),Time, Cost, Aggregator.house_2(h));

               % Create the profile for the whole time span

               Before_arrival_2 = zeros(1,Fleet2(h,2)-1);
               After_arrival_2 = zeros(1,Time.span-Full_charge_time_2(h));
               Aggregator.house_2(h).EV_E = [Before_arrival_2,
Aggregator.house_2(h).EV_E, After_arrival_2] ;
               Aggregator.house_2(h).EV_P = [Before_arrival_2,
Aggregator.house_2(h).EV_P, After_arrival_2] ;
               Power_profile_bill_2(h,:) = Aggregator.house_2(h).EV_P;
               Bill_cost_optimization_2(h) = Aggregator.house_2(h).bill;

               % Reiterate in case of failure

               if Energy_profile_bill_2(h,Full_charge_time_2(h))==0
                 fprintf('Bill optimization failure fleet 2 for EV num %d \n', h);
                 pause (1);
                 while Energy_profile_bill_2(h,Full_charge_time_2(h))==0
                      [Aggregator.house_2(h).bill, Aggregator.house_2(h).EV_x,
Aggregator.house_2(h).EV_E, Aggregator.house_2(h).EV_P]=
Optimization_bill(Fleet2(h,:),Time, Cost, Aggregator.house_2(h));
                       Aggregator.house_2(h).EV_E = [Before_arrival_2,
Aggregator.house_2(h).EV_E, After_arrival_2] ;
                       Aggregator.house_2(h).EV_P = [Before_arrival_2,
Aggregator.house_2(h).EV_P, After_arrival_2] ;
                       Power_profile_bill_2(h,:) = Aggregator.house_2(h).EV_P;
                       Energy_profile_bill_2(h,:)= Aggregator.house_2(h).EV_E;
                       Bill_cost_optimization_2(h) = Aggregator.house_2(h).bill;
                 end
               end
               pause (1);
                end
         end
% Fleet 3
        if ~isempty (pack3)

                for p = 1:size(pack3,1)
                h = pack3(p,1);
                current_time = pack3(p,2);
                Full_charge_time_3(h) = pack3(p,2)+ pack3(p,3);
                if Full_charge_time_3(h)>= time_horizon
                    time_horizon = Full_charge_time_3(h); % Time horizon in slot
                end
               Occupancy_3(h,Fleet3(h,2):Full_charge_time_3(h)) = 1;
               [Aggregator.house_3(h).bill, Aggregator.house_3(h).EV_x,
Aggregator.house_3(h).EV_E, Aggregator.house_3(h).EV_P]=
Optimization_bill(Fleet3(h,:),Time, Cost, Aggregator.house_3(h));

               % Create the profile for the whole time span

               Before_arrival_3 = zeros(1,Fleet3(h,2)-1);
               After_arrival_3 = zeros(1,Time.span-Full_charge_time_3(h));
               Aggregator.house_3(h).EV_E = [Before_arrival_3,
Aggregator.house_3(h).EV_E, After_arrival_3] ;
               Aggregator.house_3(h).EV_P = [Before_arrival_3,
Aggregator.house_3(h).EV_P, After_arrival_3] ;
               Power_profile_bill_3(h,:) = Aggregator.house_3(h).EV_P;
               Energy_profile_bill_3(h,:)= Aggregator.house_3(h).EV_E;
               Bill_cost_optimization_3(h) = Aggregator.house_3(h).bill;

               % Reiterate in case of failure

               if Energy_profile_bill_3(h,Full_charge_time_3(h))==0
                 fprintf('Bill optimization failure fleet 3 for EV num %d \n', h);
                 pause (1);
                 while Energy_profile_bill_3(h,Full_charge_time_3(h))==0
                      [Aggregator.house_3(h).bill, Aggregator.house_3(h).EV_x,
Aggregator.house_3(h).EV_E, Aggregator.house_3(h).EV_P]=
Optimization_bill(Fleet3(h,:),Time, Cost, Aggregator.house_3(h));
                       Aggregator.house_3(h).EV_E = [Before_arrival_3,
Aggregator.house_3(h).EV_E, After_arrival_3] ;
                       Aggregator.house_3(h).EV_P = [Before_arrival_3,
Aggregator.house_3(h).EV_P, After_arrival_3] ;
                       Power_profile_bill_3(h,:) = Aggregator.house_3(h).EV_P;
                       Energy_profile_bill_3(h,:)= Aggregator.house_3(h).EV_E;
                       Bill_cost_optimization_3(h) = Aggregator.house_3(h).bill;
                 end
               end
               pause (1);
               end
        end
```

6.4 Optimize Fleet

```
           EV_in_charge_1 = find(Occupancy_1(:,current_time)==1);
           EV_connected_1 = [];

           if ~ isempty (EV_in_charge_1)
                   for n = 1:length(EV_in_charge_1)
                    if Occupancy_1(EV_in_charge_1(n),current_time+1)~=0    % Last time
slot is excluded
                        EV_connected_1 = [EV_connected_1;Fleet1(EV_in_charge_1(n),:)];
                    end
                   end

                   % Compute the bill for the reamining time.

                   for y = 1:size(EV_connected_1,1)
                    h = EV_connected_1(y,1);
                    Aggregator.house_1(h).bill =
sum((Power_profile_bill_1(h,current_time:Full_charge_time_1(h))+
Aggregator.house_1(h).p_baseload(current_time:Full_charge_time_1(h)))*Cost.c(current_tim
e:Full_charge_time_1(h))*Time.tau/60);

                   end

           end

           EV_in_charge_2 = find(Occupancy_2(:,current_time)==1);
           EV_connected_2 = [];

           if ~ isempty (EV_in_charge_2)
                   for n = 1:length(EV_in_charge_2)
                    if Occupancy_2(EV_in_charge_2(n),current_time+1)~=0    % Last time
slot is excluded
                        EV_connected_2 = [EV_connected_2;Fleet2(EV_in_charge_2(n),:)];
                    end
                   end

                   % Compute the bill for the reamining time.

                   for y = 1:size(EV_connected_2,1)
                    h = EV_connected_2(y,1);
                    Aggregator.house_2(h).bill =
sum((Power_profile_bill_2(h,current_time:Full_charge_time_2(h))+
Aggregator.house_2(h).p_baseload(current_time:Full_charge_time_2(h)))*Cost.c(current_tim
e:Full_charge_time_2(h))*Time.tau/60);
                   end
           end
           EV_in_charge_3 = find(Occupancy_3(:,current_time)==1);
           EV_connected_3 = [];

           if ~ isempty (EV_in_charge_3)
                   for n = 1:length(EV_in_charge_3)
                    if Occupancy_3(EV_in_charge_3(n),current_time+1)~=0    % Last time
slot is excluded
                        EV_connected_3 = [EV_connected_3;Fleet3(EV_in_charge_3(n),:)];
                    end
                   end

                   % Compute the bill for the reamining time.

                   for y = 1:size(EV_connected_3,1)
                    h = EV_connected_3(y,1);
                    Aggregator.house_3(h).bill =
sum((Power_profile_bill_3(h,current_time:Full_charge_time_3(h))+
Aggregator.house_3(h).p_baseload(current_time:Full_charge_time_3(h)))*Cost.c(current_tim
e:Full_charge_time_3(h))*Time.tau/60);

                   end

           end


             optimization_failure = 0 ;
            [Diff, EV_x_1, EV_E_1, EV_P_1, EV_x_2, EV_E_2, EV_P_2, EV_x_3, EV_E_3,
EV_P_3] = Optimization_PAR(EV_connected_1, EV_connected_2, EV_connected_3, Time, Cost,
Aggregator, delta, current_time, time_horizon, Occupancy_1, Occupancy_2, Occupancy_3);
             pause (2);

             if ~isempty(EV_E_1) && any (EV_E_1(:,time_horizon+1-
current_time)==0)||~isempty(EV_E_2) && any (EV_E_2(:,time_horizon+1-
current_time)==0)||~isempty(EV_E_3) && any (EV_E_3(:,time_horizon+1-current_time)==0)
                 fprintf ('Optimization failure with delta = %.1f \n', delta);
                 optimization_failure = 1;
                     while optimization_failure == 1
                         [Diff, EV_x_1, EV_E_1, EV_P_1, EV_x_2, EV_E_2, EV_P_2, EV_x_3,
EV_E_3, EV_P_3] = Optimization_PAR(EV_connected_1, EV_connected_2, EV_connected_3, Time,
Cost, Aggregator, delta, current_time, time_horizon, Occupancy_1, Occupancy_2,
Occupancy_3);
                         pause(2);
                             if ~isempty(EV_E_1) && any (EV_E_1(:,time_horizon+1-
current_time)==0)||~isempty(EV_E_2) && any (EV_E_2(:,time_horizon+1-
current_time)==0)||~isempty(EV_E_3) && any (EV_E_3(:,time_horizon+1-current_time)==0)
                                 optimization_failure = 1;
                                 fprintf ('Optimization failure with delta = %.1f \n',
delta);
                                 delta = delta + 3;           % Increasing delta to
lighten the burden of contraints
                                 else
```

```
                                        optimization_failure = 0;
                                end
                        end

                end

                for d=1:size(EV_connected_1,1)
                    j=EV_connected_1(d,1);
                    Energy_arrival_time_1(j) = Aggregator.house_1(j).EV_E(Fleet1(j,2));
                    fprintf('Fleet 1 : EV number %d arrived at %d:%d and leaves at %d:%d,
with initial energy=%d and energy request=%d, has final energy= %f \n', Fleet1(j,1),
floor(Fleet1(j,2)/4),(Fleet1(j,2)/4-
floor(Fleet1(j,2)/4))*60,floor(Full_charge_time_1(j)/4),(Full_charge_time_1(j)/4-
floor(Full_charge_time_1(j)/4))*60,Energy_arrival_time_1(j), Fleet1(j,4),
EV_E_1(d,Full_charge_time_1(j)+1-current_time));
                    Aggregator.house_1(j).EV_E(current_time:Full_charge_time_1(j)) =
EV_E_1(d,1:(Full_charge_time_1(j)+1-current_time));
                    Aggregator.house_1(j).EV_P(current_time:Full_charge_time_1(j)) =
EV_P_1(d,1:(Full_charge_time_1(j)+1-current_time));
                    Power_profile_PAR_1 (j,EV_connected_1(d,2):Full_charge_time_1(j)) =
Aggregator.house_1(j).EV_P(EV_connected_1(d,2):Full_charge_time_1(j));
                    Energy_profile_PAR_1 (j,EV_connected_1(d,2):Full_charge_time_1(j)) =
Aggregator.house_1(j).EV_E(EV_connected_1(d,2):Full_charge_time_1(j));
                end

                for d=1:size(EV_connected_2,1)
                    j=EV_connected_2(d,1);
                    Energy_arrival_time_2(j) = Aggregator.house_2(j).EV_E(Fleet2(j,2));
                    fprintf('Fleet 2 : EV number %d arrived at %d:%d and leaves at %d:%d,
with initial energy=%d and energy request=%d, has final energy= %f \n', Fleet2(j,1),
floor(Fleet2(j,2)/4),(Fleet2(j,2)/4-
floor(Fleet2(j,2)/4))*60,floor(Full_charge_time_2(j)/4),(Full_charge_time_2(j)/4-
floor(Full_charge_time_2(j)/4))*60,Energy_arrival_time_2(j), Fleet2(j,4),
EV_E_2(d,Full_charge_time_2(j)+1-current_time));
                    Aggregator.house_2(j).EV_E(current_time:Full_charge_time_2(j)) =
EV_E_2(d,1:(Full_charge_time_2(j)+1-current_time));
                    Aggregator.house_2(j).EV_P(current_time:Full_charge_time_2(j)) =
EV_P_2(d,1:(Full_charge_time_2(j)+1-current_time));
                    Power_profile_PAR_2 (j,EV_connected_2(d,2):Full_charge_time_2(j)) =
Aggregator.house_2(j).EV_P(EV_connected_2(d,2):Full_charge_time_2(j));
                    Energy_profile_PAR_2 (j,EV_connected_2(d,2):Full_charge_time_2(j)) =
Aggregator.house_2(j).EV_E(EV_connected_2(d,2):Full_charge_time_2(j));
                end

                for d=1:size(EV_connected_3,1)
                    j=EV_connected_3(d,1);
                    Energy_arrival_time_3(j) = Aggregator.house_3(j).EV_E(Fleet3(j,2));
                    fprintf('Fleet 3 : EV number %d arrived at %d:%d and leaves at %d:%d,
with initial energy=%d and energy request=%d, has final energy= %f \n', Fleet3(j,1),
floor(Fleet3(j,2)/4),(Fleet3(j,2)/4-
floor(Fleet3(j,2)/4))*60,floor(Full_charge_time_3(j)/4),(Full_charge_time_3(j)/4-
floor(Full_charge_time_3(j)/4))*60,Energy_arrival_time_3(j), Fleet3(j,4),
EV_E_3(d,Full_charge_time_3(j)+1-current_time));
                 Aggregator.house_3(j).EV_E(current_time:Full_charge_time_3(j)) =
EV_E_3(d,1:(Full_charge_time_3(j)+1-current_time));
                 Aggregator.house_3(j).EV_P(current_time:Full_charge_time_3(j)) =
EV_P_3(d,1:(Full_charge_time_3(j)+1-current_time));
                 Power_profile_PAR_3 (j,EV_connected_3(d,2):Full_charge_time_3(j)) =
Aggregator.house_3(j).EV_P(EV_connected_3(d,2):Full_charge_time_3(j));
                 Energy_profile_PAR_3 (j,EV_connected_3(d,2):Full_charge_time_3(j)) =
Aggregator.house_3(j).EV_E(EV_connected_3(d,2):Full_charge_time_3(j));
             end


           tfo_total_power =
sum(Power_profile_PAR_1(:,1:time_horizon))+sum(Power_profile_PAR_2(:,1:time_horizon))+su
m(Power_profile_PAR_3(:,1:time_horizon)) + Aggregator.tfo.p_baseload(1:time_horizon);
           Not_optimized_PAR =
sum(Power_profile_bill_1(:,1:time_horizon))+sum(Power_profile_bill_2(:,1:time_horizon))+
sum(Power_profile_bill_3(:,1:time_horizon)) + Aggregator.tfo.p_baseload(1:time_horizon);
           Aggregator.tfo.p_average = mean(tfo_total_power);
           Aggregator.tfo.PAR = max(tfo_total_power)/Aggregator.tfo.p_average;
           pause(1);
           Plot_Power
           Elapsed_time = [Elapsed_time;toc];
           Elapsed_time

        end
```

6.5 Verify new day

```
    if ~any (Fleet1(:,1)==0) && ~any (Fleet2(:,1)==0) && ~any (Fleet3(:,1)==0)

            Date.day = Date.day + 1;
            Date_str = datestr(Date.day);
            Date_day = Date_str(1:11);
            fprintf('Optimization continues running in %s \n', Date_day);
            if Date.day >365
                 fprintf('Last day of the year! End program. \n', current_time);  % /!\
The possibility to change year is not implemented!
                 return
            else

                % Comparing the bill variations from both optimization

                for f = 1: Aggregator.H

Difference_bills_1(f)=(Power_profile_PAR_1(f,Fleet1(f,2):Full_charge_time_1(f))+Aggregat
or.house_1(f).p_baseload(Fleet1(f,2):Full_charge_time_1(f)))*Cost.c(Fleet1(f,2):Full_cha
rge_time_1(f))/4-Bill_cost_optimization_1(f);

Percentage_difference_1(f)=((Power_profile_PAR_1(f,Fleet1(f,2):Full_charge_time_1(f))+Ag
gregator.house_1(f).p_baseload(Fleet1(f,2):Full_charge_time_1(f)))*Cost.c(Fleet1(f,2):Fu
ll_charge_time_1(f))/4)/Bill_cost_optimization_1(f);
                fprintf('Fleet 1: EV number %d arrived at %d:%d and leaves at %d:%d,
with initial energy=%d and energy request=%d, has final energy= %f \n Its bill has
changed by a factor of %.2f with rescpect to the bill optimum \n', Fleet1(f,1),
floor(Fleet1(f,2)/4),(Fleet1(f,2)/4-floor(Fleet1(f,2)/4))*60,
floor(Full_charge_time_1(f)/4),(Full_charge_time_1(f)/4-
floor(Full_charge_time_1(f)/4))*60,Energy_arrival_time_1(f), Fleet1(f,4),
Energy_profile_PAR_1 (f,Full_charge_time_1(f)),Percentage_difference_1(f));

                end

             for f = 1: Aggregator.H

Difference_bills_2(f)=(Power_profile_PAR_2(f,Fleet2(f,2):Full_charge_time_2(f))+Aggregat
or.house_2(f).p_baseload(Fleet2(f,2):Full_charge_time_2(f)))*Cost.c(Fleet2(f,2):Full_cha
rge_time_2(f))/4-Bill_cost_optimization_2(f);

Percentage_difference_2(f)=((Power_profile_PAR_2(f,Fleet2(f,2):Full_charge_time_2(f))+Ag
gregator.house_2(f).p_baseload(Fleet2(f,2):Full_charge_time_2(f)))*Cost.c(Fleet2(f,2):Fu
ll_charge_time_2(f))/4)/Bill_cost_optimization_2(f);
                fprintf('Fleet 2: EV number %d arrived at %d:%d and leaves at %d:%d,
with initial energy=%d and energy request=%d, has final energy= %f \n Its bill has
changed by a factor of %.2f with rescpect to the bill optimum \n', Fleet2(f,1),
floor(Fleet2(f,2)/4),(Fleet2(f,2)/4-floor(Fleet2(f,2)/4))*60,
floor(Full_charge_time_2(f)/4),(Full_charge_time_2(f)/4-
floor(Full_charge_time_2(f)/4))*60,Energy_arrival_time_2(f), Fleet2(f,4),
Energy_profile_PAR_2 (f,Full_charge_time_2(f)),Percentage_difference_2(f));
             end

             for f = 1: Aggregator.H

Difference_bills_3(f)=(Power_profile_PAR_3(f,Fleet3(f,2):Full_charge_time_3(f))+Aggregat
or.house_3(f).p_baseload(Fleet3(f,2):Full_charge_time_3(f)))*Cost.c(Fleet3(f,2):Full_cha
rge_time_3(f))/4-Bill_cost_optimization_3(f);

Percentage_difference_3(f)=((Power_profile_PAR_3(f,Fleet3(f,2):Full_charge_time_3(f))+Ag
gregator.house_3(f).p_baseload(Fleet3(f,2):Full_charge_time_3(f)))*Cost.c(Fleet3(f,2):Fu
ll_charge_time_3(f))/4)/Bill_cost_optimization_3(f);
                fprintf('Fleet 3 : EV number %d arrived at %d:%d and leaves at %d:%d,
with initial energy=%d and energy request=%d, has final energy= %f \n Its bill has
changed by a factor of %.2f with rescpect to the bill optimum \n', Fleet3(f,1),
floor(Fleet3(f,2)/4),(Fleet3(f,2)/4-floor(Fleet3(f,2)/4))*60,
floor(Full_charge_time_3(f)/4),(Full_charge_time_3(f)/4-
floor(Full_charge_time_3(f)/4))*60,Energy_arrival_time_3(f), Fleet3(f,4),
Energy_profile_PAR_3 (f,Full_charge_time_3(f)),Percentage_difference_3(f));
             end
```

```
            Set_blocks_parameters
            Set_Electricity_Price
            Set_Figure_Parameters
            Set_Matrices_Vectors_Xbee
            delta = 0.5;
            Plot_Electricity_Price
            Elapsed_time = [];
            end
    end
end

fclose(c);
delete(c)
clear c
```

Published with MATLAB® R2016a

Annex B : Aggregator

6.6 Aggregator2

```
delete(instrfindall);
s = serial('com3');
fopen(s);

Start_delimiter = '7E';      % It indicates the beginning of a data frame
MSB = '00';                  % Most and Least significant bytes are used to
                             % compute the length of the API frame
Frame_type = '00';           % Tx transmit request frame
Frame_ID = '01';             % The sender asks for a 'Transmit Status' frame
Coordinator_address = ['00';'13';'A2';'00';'40';'C3';'41';'2E'];
Option = '00';               % Supported transmission option, when '00' no options are
selected

new_day = 0;
Set_Time_Parameters
P_charge_max = 3.68;
Fleet_Characteristics_Xbee_2;
Fleet
Number_of_EV = size(Fleet(:,1),1);
while new_day == 0

last_EV_day = max(Fleet(:,2));
time_slot = 35;
interval_check = tic;
time_division = 20;           % To make real 15 minutes last in 'time_division' seconds
while time_slot <= last_EV_day
    tic
        b = datevec(now);
        c = floor(b(6));
        if  floor(c/time_division)==ceil(c/time_division) % so that 15 min corresponds
with 20 sec
            toc(interval_check)
            pause (0.5); % needed to avoid enter in the next time slot
            EV_plugged_in = any (Fleet(:,2) == time_slot);
            switch EV_plugged_in
                case 0
                    fprintf('Fleet_2:No new vehicles are plugged at %d\n',time_slot);
                    time_slot = time_slot + 1;
                case 1
                    RF_data = [];
                    for j = 1: Number_of_EV
                        if Fleet(j,2)== time_slot
                            RF_data = [RF_data;
dec2hex(Fleet(j,1),2);dec2hex(Fleet(j,2),2);dec2hex(Fleet(j,3),2);dec2hex(Fleet(j,4),2)]
;
                        end
                    end
                        % Compute and add checksum to frame
                        sum=0;
                        n = size(RF_data,1);
                        LSB = dec2hex(n + 11,2);     % in the length calculation are
included all the bytes of the API frames,
                                                     % except for the star delimiter,
MSB and LSB and the checksum
                        API_frame = [Start_delimiter ;MSB ;LSB;Frame_type; Frame_ID;
Coordinator_address; Option; RF_data];
                        for j=4:n+14
                              sum = sum + hex2dec(API_frame(j,:));
                        end
                        sum = dec2hex(sum);
                        m = length(sum);
                        if m > 2
                            last_2_bytes_of_sum = sum(m-1:end);
                            API_frame(n+15,:) = dec2hex(hex2dec('FF')-
hex2dec(last_2_bytes_of_sum));
                        else
                            API_frame(n+15,:) = dec2hex(hex2dec('FF')-hex2dec(sum(m-
1:m)));
                        end
```

6.7 Sending message

```
                         pause(2.5);       % create a delay among the three aggregator
                      while s.BytesAvailable ==0 && toc <= 10
                            API_frame;
                          % L'API frame lo invia tante volte fintanto che bytesav è 0 e
timeout<50.
                          % e in queste condizioni non si entra dentro gli if.
                            for x=1:n+15
                              fwrite(s,hex2dec(API_frame(x,:)),'uint8');
                            end

                        pause(0.1);    % a proper time is needed to receive the
aknowledge frame.
```

```
                                if s.BytesAvailable ~= 0
                                    Get_aknowledge_frame =
dec2hex(fread(s,s.BytesAvailable));
                                    if Get_aknowledge_frame (6,:) == '00'
% Delivery status 'success'
                                       fprintf ('Energy request correctly sent at %d \n',
time_slot);           % The message has been correctly sent
                                    break
                                    end
                                end

                        if toc >= 10
                          fprintf (' Aggregator 2: no acknowledgement received in 10
seconds at time slot %d \n It will be delayed in the next quarter of hour \n',
time_slot);
                          % If the request is not correctly sent it is stored in the next
quarter
                          % of a hour message
                          for h = 1: Number_of_EV
                              if Fleet(h,2)== time_slot
                                  % The time available to get charged is lowered by 15
minutes. It
                                  % should be verified that this time is still compatible
with the power of the charger
                                  if (Fleet(h,4)/((Fleet(h,3)-1)*(Time.tau/60))) <=
P_charge_max
                                  Fleet(h,2)=time_slot + 1;      % the message is set to
be stored on the next quarter of hour
                                  else
                                      fprintf(' Warning! Aggregator 2: the request of
vehicle %d cannot been accomplished!\n', Fleet(h,1));
                                  end
                              end
                          end
                        end
                        end
                        time_slot = time_slot + 1;
              end
          end
          pause (0.5);
          end
      new_day = input('Press 0 to continue the optimization for the next day \n');
      if new_day ~=0
          break
      else
Fleet_Characteristics_Xbee_2;
      end
end
fclose(s)
```

```
delete(s)
clear s
```

Published with MATLAB® R2016a

Annex C : Fleet_characteristics

6.8 Defining EV fleet characteristics

```
%function [Fleet,Full_charge_time,P_charge_max]= Fleet_Characteristics
% Battery parameters. The charger is supposed to be the same for all fleet's vehicles
P_charge_min = 1.38;   % kW
P_charge_max = 3.68;   % kW
Eff_ch = 0.9;          % charging efficiency
SoC_min = 0.2;
SoC_max = 0.9;

% Every EV has to send: 1)
% energy needed, 2) time to get charged, 3) time of start charging/plug in
% and 4) idendification index
% Therefore every EV sends 4 bytes

Number_of_EV = 64;                     % number of EVs in the fleet

Fleet = zeros (Number_of_EV,4);        % initialize the fleet matrix
% (col 1: index; col 2: start charging time;  col 3:time available to charge;  col 4:
energy request)
Fleet(1:Number_of_EV,1) = (1:Number_of_EV);  % indexing


Fleet (1:Number_of_EV,4)= round(abs(6 + 3*randn(Number_of_EV,1)));     % Energy request
(kWh)

% Defining the time range allowed to charge by every EV
for i = 1:Number_of_EV

    Minimum_time_frame_needed = Fleet(i,4)/P_charge_max;                 % hours
    Minimum_time_slots_needed = ceil (Minimum_time_frame_needed * 4);    % minimum time
slots needed to charge
    % time slots allowed to charge, it is a minimum time due to the
    % charger capacity + a time related to user's preference (randomly
    % chosen in a range of 10 hours: 40 time slots) + two hours of extra time
    % considering that max power is not always available. The extra time is
    % up to 10 hours.
    Fleet(i,3) = Minimum_time_slots_needed + 8 + ceil(rand(1)*(32));
```

```
end
% Defining the start charging time:
% Assuming that most of the EVs come back home at 7 pm and some at 1 pm, I can assume
that
% the start charging/plug in time will have a bimodal distribution with a
% peak at 7 pm and  another peak at 1pm, both with standard deviation of 2 hours.

Earlier_plug_in = randi ([0,(Number_of_EV/2)]);   % it generates a random
% number from 0 to 32, this will represent the number of vehicles arriving home at about
1 pm
Fleet (1:Earlier_plug_in,2)= abs(13 + 2*randn(Earlier_plug_in,1));        % A random
number (up to 32) of EV arrive at about 13 h
Fleet ((Earlier_plug_in + 1):Number_of_EV,2)= abs(19 + 2*randn((Number_of_EV -
Earlier_plug_in),1));

% Start charging time in time slots, every plug in instant is rounded to the subsequent
time slot
Fleet (1:Number_of_EV,2)= ceil(Fleet(1:Number_of_EV,2)*4);  % 4 = 60/Time.tau

% Full charge time scheduled
Full_charge_time = zeros (Number_of_EV,1);
for i=1:Number_of_EV
    Full_charge_time (i) = Fleet(i,2)+Fleet(i,3);
end
Full_charge_time;
max(Full_charge_time);
```

Published with MATLAB® R2016a

Acknowledgments

In the conclusion of my work, I would like to express my deep gratitude to my supervisors, Pr. Jean-Claude Maun and Pr. Maurizio Repetto, for giving me the opportunity to develop this topic that profoundly fascinates me and has attracted me always more and more as the prosecution went on. Furthermore, I would like to thank my co-supervisor, Ir. Benoit Mattlet, for the vital and precious help he provided my, and the willingness by which he has been always ready to support me in my work during these years, even by distance.

Yet I would like to thank my family, above all my parents, for having always supported and believed in me, even more than I did. I will always be grateful for the strength and the courage they transmit to me and the love that always bind us.

I want also to thank my friends Loris, Biagio and Max. They are lovely and loyal friends and they have made me feel at home during these years of university, even if I was actually in totally new environment.