UNIVERSITÀ
DEGLI STUDI
FIRENZE

UNIVERSITÀ DEGLI STUDI DI FIRENZE
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE (DINFO)
CORSO DI DOTTORATO IN INFORMATICA, SISTEMI E TELECOMUNICAZIONI
CURRICULUM: TELEMATICA E SOCIETÀ DELL'INFORMAZIONE

SMART HOME: ENERGY MONITORING AND EXPLOITATION OF NETWORK VIRTUALIZATION

SSD: ING-INF/03

Candidate
Ing. Francesca Paradiso
[illegible]

Supervisors
Prof. Dino Giuli [illegible]
Ing. Federica Paganelli [illegible]
Prof. Paola Cappanera [illegible]

PhD Coordinator
Prof. Luigi Chisci Luigi Chisci

CICLO XXVIII, 2012-2016

Università degli Studi di Firenze, Dipartimento di Ingegneria dell'Informazione (DINFO).

A Cristian e Diego

Acknowledgments

I would like to firstly express my sincere gratitude to my supervisor, **Professor Dino Giuli** for his valuable advice, his optimistic approach and for this precious study opportunity in his laboratory. I am also deeply grateful to **Ing. Federica Paganelli** for her kind guidance as well as her trust, great support and encouragement. I also would like to express my great appreciation to **Prof. Paola Cappanera** who gave me a lot of insightful advice and enormous help to approach mathematical models. A big thank goes to **Luca Capannesi** for its crucial technical support. Furthermore, I would like to thank those people who, over these years and in different ways, have helped me to improve these contributions: **Ing. Iacopo Girolami, Ing. Samuele Capobianco, Prof. Antonio Luchetta, Prof. Franco Pirri and Pino Castrogiovanni**. I thank my fellow labmates for the stimulating discussions and for all the fun we have had in these years; a special mention for **Ing. Stefano Turchi** and **Ing. Clio Mugnai**: colleagues and friends.

Special thanks is due to my ***parents*** for supporting me throughout writing this thesis and to my friends **Vale** and **Viola** for keeping me... me.

Last but not least, you two. The loves of my life, **Cristian** and **Diego**. Nothing would have been possible without you.

Contents

Contents **v**

1 Introduction **1**
1.1 Towards a Virtual Smart Home 1
1.2 The multidisciplinary context 5
1.3 Contributions 9

2 Energy savings applications **13**
2.1 Energy Consumption Monitoring 14
2.2 Background 18
2.3 Related Work 21
2.3.1 Appliance Profiling and Classification related work 25
2.3.2 Context-based Non-intrusive Load monitoring related work 27
2.4 Appliance profiling and Classification 28
2.4.1 The logical classification architecture 28
2.4.2 The HEMS architecture 31
2.4.3 Classification Algorithm 35
2.4.4 Evaluation 36
2.4.5 Evaluation and Testing Results 38
2.5 Context-based Non-Intrusive Load Monitoring 42
2.5.1 The probabilistic data model 42
2.5.2 NILM Inference 43
2.5.3 Context-based disaggregation 44
2.5.4 Evaluation 48
2.6 Conclusions and Further Improvements 60

3 Multi-user VNF Placement optimization in a SDN-based multi-stakeholder NFV architecture 65
3.1 Introduction . . . 66
3.2 Background . . . 69
3.2.1 Software Defined Networking . . . 70
3.2.2 Network Function Virtualization . . . 74
3.2.3 Service Function Chaining . . . 77
3.3 Related Work . . . 79
3.3.1 Motivation of this work . . . 82
3.4 Problem Formulation . . . 88
3.5 Pre-Processing phase and Optimization Model . . . 95
3.5.1 Pre-processing phase model . . . 96
3.5.2 Optimization model . . . 102
3.6 Evaluation . . . 107
3.6.1 Evaluation scenario . . . 107
3.6.2 Evaluation metrics . . . 113
3.6.3 Testing Results . . . 114
3.7 Conclusions and Further Improvements . . . 120

4 Conclusion 123

A Appendix 127

B Appendix 129

C Publications 131

Bibliography 133

Chapter 1

Introduction

"Homes are unique. They define the occupants, their preferences and lifestyles. In addition to the purely functional aspects, homes also provide the boundary and interface to the rest of the world. They contain personal possessions, memories and familial relationships and they provide security and privacy for everything contained within them." [1]. This statement simply underscores the importance of the home in people's lives. Now more than ever We are witnessing a radical change in the way a house is conceived. The increasing use of devices with connectivity and capabilities to enhance and automate everyday actions that describe our existence, promotes the natural transition to an expanded concept of home, a Virtual Smart Home.

1.1 Towards a Virtual Smart Home

The *Smart Home* refers to a domestic environment properly designed and equipped with advanced technologies to generally improve inhabitants quality of life. These advanced technologies are able to react to environmental parameters changes or to perform specific functions as a result of a direct (e.g. the user activates a smart device), indirect (e.g. the user programs the activation of a smart device) or "learned" (e.g. the system activates a smart device according to user habits) input. Considering the house as a whole, the change brought by this development allows the coordinated and integrated management of both technological systems and communication networks in order to improve administration flexibility, security and comfort as well as to reduce energy waste, networking conflicts and management costs. The Smart Home infrastructure, is nowadays mainly composed by four components:

- **Smart Objects**: it generally refers to technologically advanced objects of everyday life (e.g. washing machine) which are connected to the Management and Control System or to other objects, able to gather and/or share data and, depending on their functionality and intended use, perform certain actions.

- **User interfaces**: the inhabitant can get information and control the Smart Home and the devices that it contains, through user interfaces such as remote controls, touch screens, keyboards, voice commands.

- **Home Management and Control System**: this component has the task to manage and control the Smart Environment in order to provide the users with meaningful information by collecting data from smart objects (e.g. home temperature, security alarm status, energy consumption, etc.) and perform specific actions according to user's needs or habits.

- **Communication System**: one or more communication systems allow the user interaction with both the internal (i.e. Home Area Network) and external (i.e. Wide Area Network) area. The Home control and Management System on the one hand interacts with smart object by collecting data or performing actions and, on the other hand, communicates with the user inside or outside the domestic area. The core of user's home network is represented by the Residential Gateway who has a double duty: it allows both the connection of network-enabled devices intra-home and users to access the Internet.

Nowadays houses are frequently equipped with a broadband Internet connection usually supplied through the Residential Gateway (RGW), also known as Home Gateway (HGW), which also provides users with a Home Network. A typical RGW is able to provide several services; some of them have always been within its competence such as the Dynamic Host Configuration Protocol - DHCP (to provide each home network with a dynamic Internet Protocol - IP - address) and Network Address Translation - NAT (e.g. to enable the sharing of a single Internet connection between home users). Other features have been gradually introduced later such as Internet Gateway Device - IGD (to allow the server and peer-to-peer applications to function properly behind NAT), Dynamic Domain Name Server - DynDNS (to provide DNS servers with the updated IP) [2], etc., however this services are constantly changing depending on the user requirements that arise with technology advances. So far the Home Network has been used to connect devices that were used for compute and communication purposes such as PCs, Tablets, Smart phones to the Internet [1] but, recently the number (and the typology) of appliances which exploit the Internet connection is becoming countless. Indeed, the Internet of Things (IoT) [3] paradigm, which refers to the networked interconnection of everyday objects (becoming Smart), has lead to a distributed network of devices communicating with both users and other devices. Creating, sharing and consuming information is becoming therefore more and more immediate, easy and fast, thus causing an increase in user-generated shared content in the Internet. This improved connectivity as well as the increasingly number of new smart appliances, robotic devices and electronic assistants, has favored the emergence of new services to the home domain (e.g. health care, home automation, media content management, etc.), often introducing separate devices with their own protocols, user interfaces and networks [2]. We are therefore witnessing a huge revolution in the history of homes: a "life-automated" future in which home devices will cooperate to provide cognitive digital assistance, is predicted. Ervin Six and Jan Bouwen [1] envisage a classification of future home assistance functions mainly composed by these four groups:

- **Infrastructure and environment management**: this includes all those services that consist in monitoring, controlling and automating the home environment. The Smart Home may be provided with a Home Energy Management System which can inform the user about energy consumption and suggest a usage behavior changing; with the same purpose it can automatically turn off all the unused devices if no-one is at home as well as alert the user if an appliance malfunction is detected. Security issues can also be taken into account for both the intrusion detection, access control and unexpected adverse events such as fire or gas leak.

- **Information and entertainment management**: this set concerns the information scheduling and sharing among home inhabitants. It provides family calendar as well as social, news, alerts information control and delivery; it also controls media content such as personal video/photo or streaming media content and online gaming.

- **Housekeeping management**: this sector simplifies and automates all the actions that typically concerns the domestic management operations such as cleaning, pet feeding, replenish of consumable, etc.

- **Health management**: this set includes everything that can monitor inhabitants' health namely motion sensors, smart wearables (to further analyze gathered data), child/infant monitors, etc.

Although the development trend seems to outline the increasingly wide-spread transition to Smart Home, at present the gap with the real home is still significant; Dixon et al. [4] identify two main causes. Firstly, for ordinary consumers, it has become difficult to manage their growing number of Smart Objects and, secondly, homes heterogeneity as regards devices, inter-connectivity, user preferences, etc. prevent to ease the development of a Home Management and Control System capable of favoring the coexistence of totally different devices. Therefore, by considering the totality of the elements that share the Home Network, the increase in management complexity becomes evident as well as the clear need for home networking solutions. The envisaged automation trend, dictated by the emergence of the above-mentioned new services, will be achievable only with a drastic conceptual shake-up, which, in my opinion, is represented by the use of the emerging technologies that rely on *virtualization*. One of these main technology is the Software Defined Networking (SDN) [5] [6] which is an emerging network architecture where control plane is decoupled from data plane and directly programmable; this revolutionary approach promises to overcome the innovation difficulties of the traditional network where the coupling between control and forwarding planes makes modifications difficult to achieve. Another emerging technology is the one represented by Network Function Virtualization (NFV) [7] which consists in the implementation of Network Functions (NF) in an open and standardized IT virtualization environment as Virtual Network Functions (VNF) by performing a separation between the NF (software) and vendor-specific hardware they are built on. Due to this mechanism of separating resource, these technologies offer different options in terms of flexibility, portability or performance, to suit intended use cases. Recently this new paradigm, which can be implemented for platforms, applications, storage, networks, devices, etc., is becoming increasingly important due to its proliferation in Data Centers (DCs) and with our dependency on cloud computing. By taking advantage of the emerging new network paradigm depicted by the SDN (see Section 3.2.1) together with the *softwarization* proposed by the NFV (see Section 3.2.2), this new services will be created as chained virtual instances in the edge clouds with a networking separated control inside and outside the Home Environment. Moreover Network Operators are evolving their systems towards SDN and NFV and will envisage an involvement of the house that is considered as an end-point of the operator network. This shift will represent a great opening in the home concept, since it provides for the exploitation of resources that will not even be in the house, but as close as possible to be ready to meet the inhabitants' needs in a more flexible and economic way. For this reason the future Smart Home (a Virtual Smart Home) will be composed of a new element: a **Home Virtualization Control** capable of dynamic partitioning of resources into secure access places (i.e. edge data centers) in order to, for instance, recreate the home space and digital content anywhere as virtual.

1.2 The multidisciplinary context

This thesis investigates the Virtual Smart Home concept in a multidisciplinary context and provides an architectural proposal that foresees the use of virtualization-based technologies. Figure 1.1 graphically resumes the reference architecture for the Virtual Smart Home that is provided in this thesis.

As it is immediately evident, two independent structures are emphasized: on the left side the Smart Home and, on the right side, a simplified reference architecture for the virtualization-based service provisioning.

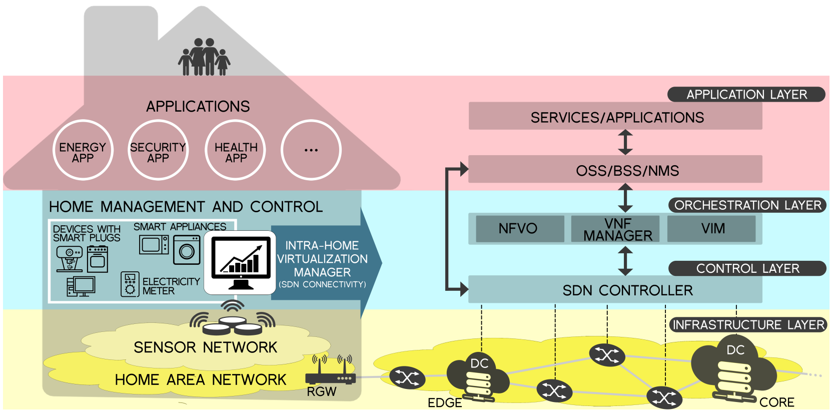


Figure 1.1: Proposed architectural solution for a Virtual Smart Home

The Figure is organized in layers, separated by different colors, which contain elements that are comparable for their intended uses.

- The **Infrastructure Layer** (the yellow one): has been represented as the lower level as it "envelops" all the higher layers. At Home side it is composed from the Home Area Network (HAN) and the Sensor Network (SN). The SN light yellow cloud represents the network (or more than one) created by sensors collocated in the whole house in order to monitor one or more specific parameters. For instance a motion Sensor Network can monitor the movement of house inhabitants e.g. in order to advice the control system that the security system can be disabled or to collect motion data to learn users' habits. A temperature/humidity Sensor Network can instead achieve the heating/air conditioning automation as well as avoid energy waste. As mentioned above, the HAN is provided through the Residential Gateway (RGW) which is also responsible to connect the inhabitants with the Internet. Hence is through the RGW that happens the connection between HAN and Wide Area Network (WAN). Outside the house the yellow level continues with the wide area network infrastructure which is composed by forwarding elements and different-sized Data Centers (DCs). In the Software Defined Network (SDN) (Section 3.2.1) concept, the Data Plane and the Control plane are separated to favor service provision flexibility. This thus implies that traditional switches are deprived of route control and then have the single task to steer traffic flows through determined paths which are chosen by higher level elements (i.e. SDN Controller). The DCs are general-purpose centralized repositories, either physical or virtual, used to accommodate computer, server and networking systems for storing, processing or serving large amounts of data. As shown in Figure 1.1, edge DCs usually have smaller size and capacity than core DCs, but, due to the proximity to the user, in this case the house, they are suitable to host services with low-latency requirements. It is reasonable to assume that, if for example, the service to be provided must guarantee a certain maximum tolerable End-to-end delay (e.g. an online gaming or a movie streaming), it is convenient to instantiate the virtual functions as close as possible to the end-user. On the contrary, if the service requires high capacity resources to be accomplished, a core DC, with higher computational capacity, may be more suitable.

- The **Management and Control Layer** (the blue one): this layer provide the architecture with the necessary control. The Home Management and Control System on the one hand deals with the collection and processing of data gathered by SNs, and, on the other hand, interfaces with applications (and therefore users) to provide them with organized informative data. The digital representation of Networked Devices is the white rectangle. Figure 1.1 only shows those type of appliances that are suited to this work (i.e. first case study deepen in Section2), but it is evident that this group can contain all the others devices that share the home network connection such as smartphones, tablets, gaming consoles, etc. The Management and Control System is high-level graphically represented by the white panel depicting a monitor.
  The external side of the Management and Control Layer is represented by the *Orchestration Layer* and the *Control Layer*, used as complementary elements. The Orchestration Layer is the management element of the Network Function Virtualization (NFV)(Section 3.2.2) which deals with the Virtual Network Functions (VNFs) lifecycle. It is composed by three main blocks which, together,

arrange the Management and Orchestration (ManO) [8] of the NFV platform namely Network Function Virtualization Orchestrator (NFVO), the VNF Manager and the Virtual Infrastructure Manager (VIM). It has a global view of the compute nodes (i.e. NFVI-based Data Centers) and is aware of instantiated VNFs, their position, their workload status, etc. It has the task to turn off unused instances and to deploy new VNFs to be used for service provision. In a complementary way the SDN Controller is the element that controls the SDN-forwarding devices of the Network Infrastructure. Once the Orchestrator has performed the instantiation and activation of the VNFs to accomplish a specific service request, the SDN Controller decides the best network path to allow the traffic flow to traverse the VNFs in a specific order.
To provide the home environment with the possibility to exploit the external virtualization-based architecture, the Home Management and Control must be SDN/NFV integrated. In fact it has to communicate to the external infrastructure the needs to instantiate/use a virtual service on an Edge DC.

- The **Application Layer** (the red one): this layer provide services to the users leveraging the layers below.
  Intra-home applications use the data provided by the Management and Control System to show the user information of interest from inside and outside. An energy monitoring app may show to the inhabitants the whole-house energy consumption together with the incidence of each single device. A security app may provide the user with alarm status, doors/windows locking status as well as sensor-detected environmental parameters. A health app may detect an illness (e.g. a user is detected suspiciously motionless) by controlling user movements or may control vital signs through user wearable. Moreover applications send user commands to the Management and Control System in order to perform/program specific actions (e.g. turn off the lights when no motion is detected).
  Also the external architecture is characterized by an Application Layer. In this case this is composed by the Services/Applications block which is in charge of users (or service providers) and the Operational Support System (OSS)) (or rather Business Support System or Network Management System). From the Services/Ap-

plications block specific service chain requests arrive to the OSS which is responsible of the accomplishment of the request, it processes requests and communicates on the one hand with the Orchestration layer to get the instantiated VNFs on the most suitable DCs, and, on the other hand, with the SDN Controller to make it aware about where (i.e. VNF instantiated in a specific DC), and in which order (i.e. the order of the chained VNF that compose a service), the traffic flow has to pass.

In this context, thus, the house can be seen as a closed and protected environment with a great window on the world, the virtual world, from which it may itself take advantage. It is only at the end of this description that the parallelism between levels (i.e. bands of different colors) assumes a crucial importance: home inhabitants could be the same users which ask for virtualized service to the outside infrastructure. Thus, Home Management ad Control System would be integrated with by intra-home SDN Controller and NFV Orchestrator; the former with the task of performing intra-home devices networking control and the latter to orchestrate the virtual services whose functions are instantiated inside (in the RGW) or outside (in the edge DCs). Thus Edge DCs act as extensions of the functionality of the house itself, facilitating service provisioning with hardware abstractions and faster software updating/upgrading.

1.3 Contributions

In the multidisciplinary context depicted in figure 1.1 this thesis is concerned with the evolution of the Residential Gateway (RGW) toward a Virtual Smart Home Environment by mainly dealing with two topics.

- According to the application perspective, focus is made on the crucial issue of improving user awareness on power consumptions for achieving energy savings. This work can be contextualized in the Infrastructure and Environment Management, that has been introduced above, which is also responsible to encourage energy conservation and favor the reduction of expenditures. The rational use of energy has recently become one of the most pressing research topic because of the constantly growing consumptions in contrast with the scarcity of resources. In a Smart Home scenario the recent progress of technology, along with lower costs, has made it possible to perform energy monitoring and management actions through the distribution of smart meters and environmental sensors capable of providing information to a Home Energy Management System (HEMS). Recent studies have shown that informing users about the actual appliances consumption as well as device-usage habits, can help to obtain energy consumption reduction in private households. In order to achieve this goal a supervised classification algorithm for detecting and identifying consuming appliances has been implemented. Then a Non-Intrusive Load Monitoring (NILM) approach has been investigated to reduce the cost of attaching a single meter (i.e. smart plug) to each device; the proposed algorithm aims at recognizing the power consumption of a specific device from the whole-house consumption profile and from the input of context information (i.e. the user presence in the house and the hourly utilization of appliances).

- On the technological infrastructure perspective, this work provides a discussion of one of the fundamental aspects in NFV orchestration that is VNF Placement. The Network Function Virtualization (NFV) technology, together with the complementary Software Defined Networking (SDN) paradigm, let envisage a revolution in the traditional concept of network service delivery and availability. This evolution promises to enable on-demand and flexible services provision as it allows to separate the network functions from the hardware they run on by leveraging the virtualization abstraction. According to this concept the Smart Home Environment could be considered as a peripheral computation/access node which, through the RGW, can take advantage of computational resources provided by edge DCs to host Virtual Network Functions (VNFs). The reference architecture that has been considered is mainly composed by a VNF-Orchestrator and an SDN controller which are in charge of managing respectively VNFs lifecycle and the network forwarding path for the service provision across multiple DCs. In a realistic scenario, a VNF Placement optimization model has been formulated by considering privileged and standard users asking for service provision through a multi-stakeholder network; the problem is formalized and its model implemented and tested.

This work is organized as follows. Chapters 2 and 3 respectively describe the first and the second contributions that have been mentioned above. Chapter 2, after a focused introduction, firstly reports a highly detailed background (Section 2.2) as well as a wide-ranging related work (Section 2.3) also separated between the two provided contributions (i.e. Appliance Classification and Context-aware NILM). In Sections 2.4 and 2.5 the Appliance Classification and the Context-aware NILM are detailed. Each Section reports model description, algorithm implementation and evaluation. For each of them, detailed testing results are provided together with discussions on further improvements. Chapter 3 is concerned with the second contribution of this thesis, namely the VNF Placement optimization problem. It is organized as the previous chapter by providing, after a detailed introduction, a synthetically background (Section 3.2) regarding virtualization technologies such as SDN, NFV and SFC. Then an extensive related work is provided in section 3.3. Further Sections describe the problem formulation (Section 3.4), the pre-processing phase and the mathematical model (Section 3.5). Section 3.6 details the evaluation scenario as well as the testing realization and results. Finally Section 3.7 concludes the Chapter by firstly giving a brief report of the Chapter contribution and secondly providing several further improvement suggestions. Chapter 4 concludes the thesis with a schematic resume of the topics that have been addressed and the contributions provided.

Chapter 2

Energy savings applications

This Chapter describes the energy savings applications that have been implemented to provide the home-user with targeted information in order to improve energy consumption awareness and significantly reduce energy waste. The rational use and management of energy is considered a key societal and technological challenge. Home Energy Management Systems (HEMS) have been introduced especially in private home domains to support users in managing and controlling energy consuming devices. Recent studies have shown that informing users about their habits with appliances as well as their usage pattern can help to achieve energy reduction in private households. This requires instruments able to monitor energy consumption at fine grain level and provide this information to consumers. While most existing approaches for load disaggregation and classification requires high-frequency monitoring data, in this work I propose two different approaches that exploits low-frequency monitoring data gathered respectively from smart plugs displaced in the home and from the whole-house electric meter. Firstly I propose a distributed approach where data processing and appliance recognition is performed through an Artificial Neural Network algorithm locally in the home gateway. The approach is based on a distributed load monitoring system made of smart plugs attached to devices and connected to *a home gateway via the ZigBee protocol. The home gateway is based on the OSGi platform, collects data from home devices and hosts both data processing and user interaction logic. Instead the second approach consists in disaggregating the whole-house power consumption profile into each appliance portion through a Factorial Hidden Markov Model that have been accounted with contextual information related to the user presence in the house and the hourly utilization of appliances.* [1] [2]

2.1 Energy Consumption Monitoring

Energy conservation is considered a main challenge to be faced at national and international level. Several factors, such as climate change, the growing resource consumption rate and poor availability of energy resources (raw materials), are making this challenge a priority. Riveiro, Johansson and Karlsson (2011) argued that "the challenge lies in finding technologies that reduce the energy consumption, while guaranteeing or even improving customer comfort levels and economic activity" [12]. Indeed, the technological progress in power efficiency is expected to produce a remarkable reduction in energy consumption, in both industrial and private domains. In this context, the residential sector plays a significant role as it owns a non-negligible percentage of the energy demand. In fact, domestic consumptions represent approximately one third of the whole energy usage in the European Union [13] as well as in the United States [14]. Some studies [15, 16] report that significant results in the energy consumption reduction in private households can be achieved through fine-grained monitoring of energy consumption (i.e. Load Monitoring) and accurate provisioning of this information to consumers. These studies on domestic consumption habits [15, 16], in fact,

[1] This Chapter has been published as: "ANN-based appliance recognition from low-frequency energy monitoring data" in *Proc. of World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013* [9]; "Appliance Recognition in an OSGi-Based Home Energy Management Gateway" in *International Journal of Distributed Sensor Networks, 2015* [10]; "Context-Based Energy Disaggregation in Smart Homes" in *Future Internet (MDPI), 2016* [11]

[2] *Acknowledgments:* this work was partially supported by Telecom Italia, Torino, Italy.

have shown that often users are not aware of how much energy is consumed by the devices they use. It has been recognized [17] that this may impair the understanding and adoption of energy saving behaviors. In other words, if the user were informed about how much a specific device affects total consumption, he might change his behavior in order to save energy as well as money. Thus it is expected that proper use of ICT (e.g. sensing, processing and actuation capabilities) would facilitate the achievement of this objective. Home Energy Management Systems (HEMS) have been introduced especially in private home domains to support users in managing and controlling energy consuming devices. Hence, in this context, the introduction of Load Monitoring can provide more useful information, such as the consumption profile of specific appliances. In this way, could be kept informed of how much total energy consumption is affected by the usage of a specific device and decide whether to replace it with a more efficient one or just postpone its usage to a time with a less expensive fare. Moreover, the analysis of an appliance consumption for detecting anomalies could also help in recognizing possible malfunctions and undertaking actions to prevent additional appliance deterioration.

Load monitoring can be achieved through three main approaches:

1. *Use of smart appliances*: this approach relies on the adoption of household appliances equipped with sensing, processing and communication resources. Common examples are the last generation of so called "white goods" (e.g. refrigerator, washing machine) that can be remotely controlled and configured.

2. *Intrusive Load Monitoring (ILM)*: this method consists in attaching a metering hardware module (e.g. a smart plug) onto each household appliance so that its energy consumption can be easily collected. This implies that a distributed system of low-cost metering devices has to be deployed in the house and data should be collected via the proper (wireless and/or wired) network infrastructure. This approach is also known as Hardware-Based Sub Metering [18].

3. *Non-Intrusive Load Monitoring (NILM)* [19]: NILM refers to a family of techniques that aim at recognizing the power consumption of a specific device from the whole-house consumption profile.

Thus it require a disaggregation algorithm to detect each device consumption according to the integrated knowledge base.

Although the adoption of smart appliances is expected to boost the effective and efficient implementation of energy saving policies, this is not likely to take place in the short term inasmuch only a subset of devices are usually available as “smart appliances”, (i.e. TVs, dishwashers or ovens) and consumers are inclined to consider their prices too high. The Intrusive Load Monitoring approach (i.e. distributed load metering through smart plugs) has the advantage gathering per-device consumption profiles while keeping costs and resource requirements as low as possible [20]. In general smart plugs can be attached to almost any type of device; however this approach can be resource demanding since a fine-grained monitoring would require the use of a relevant number of smart plugs. In addition to the required financial commitment, the physical deployment might not be easy for fixed appliances (i.e. washing machine, dishwasher, refrigerator, etc.) or the user may be bothered by the obligation to constantly attach a smart plug to every portable device (i.e. hair dryer, phone charger, laptop, etc.). On the contrary, Non-Intrusive Load Monitoring (NILM) approaches, can be easily deployed by leveraging existing and widely adopted smart meters. The deployment of a smart plug to each device is no longer needed, but the disaggregation of the whole-house consumption trace is difficult to achieve. Several load disaggregation and classification algorithms have been proposed in literature [21] to extract more meaningful information from the house aggregated load profile (Energy disaggregation). Most of them need medium or high frequency monitoring data (at least 1 Hz frequency) to obtain accurate results. In real world scenarios, this assumption may be resource demanding as more sophisticate smart meters usually have high prices and the management huge amount of data imposes the installation of higher computational and storage resources. Indeed, only few works have experimented the adoption of NILM algorithms in real home settings [22]. In any case, regardless of the chosen technique, both appliance profiling and classification are essential for understanding their consumption characteristics and producing convincing energy saving applications.

This Chapter describes two separated works that refer to the Load Monitoring topic, but applying two of the above mentioned approaches:

- **Appliance profiling and classification**
  This work proposes a method to identify the appliance in use within a home environment analyzing energy consumption data collected by a distributed load monitoring system. This is part of a Home Energy Management System (HEMS) proposed by the Energy@Home Association and deployed in some real households in Italy. The monitoring system is made of smart plugs attached to devices and connected to a home gateway via the ZigBee protocol [23]. The home gateway, based on the OSGi platform [24], collects data from devices connected to the home wireless network and hosts both data processing and user interaction logic. The algorithm that has been implemented aims at simplifying the manual configuration phase of such a system, where the end user usually needs to add a label and a description to each smart plug to associate energy consumption data to a specific monitored device (e.g. washing machine, refrigerator, dish-washing machine, etc.). Automatic appliance classification techniques can be used when a smart plug is moved from one device to another to monitor different appliances in the house: in this case the system could ask the user to provide a description of the monitored device the first time it gets connected to a smart plug and then use data collected from the smart plug to automatically recognize the same device in a second time. The initial training phase where the user needs to add or confirm the label associated to a monitored device can be reduced and sometimes totally avoided if a database with energy consumption data associated to specific device categories or models is provided as an input to the system. While existing works dealing with appliance classification delegate the classification task to a remote central server [20, 25], here I propose a distributed approach where data processing and appliance recognition is performed locally in the home gateway. Although this approach has several advantages such that of minimizing the communication requirements between local home premises and a central server, preserving privacy on user sensitive data and simplifying the user profile management, design and implementation choices should carefully take into account the peculiarities of the Energy@Home HEMS, especially in terms of storage and process-

ing constraints. A major constraint of the application is the ultra low frequency of sampled data (1 sample collected every 2 minutes) which limits both type and number of features that can be extracted for analysis purposes. Thus, the original contribution of this work consists in a supervised classification algorithm based on Artificial Neural Networks (ANN) [26] conceived for processing ultra low-frequency monitoring data to recognize appliances in use and their consumption profile. I also present a Java-based prototype implementation running as an OSGi bundle in the local home gateway.

- **Context-based Non-Intrusive Load Monitoring**

This Chapter is structured as follows. In Section (2.2) a detailed description of the Background is provided. Section (2.3) discusses the related work by firstly giving a panoramic overview of state-of-the-art works and secondly providing separated details according each contribution. In Sections 2.4 and 2.5 the **Appliance profiling and classification algorithm** and the **Context-aware NILM algorithm** that have been implemented are respectively fully described. Each section provides the pre-processing phase, the implementation of the proposed algorithms and evaluation scenarios and metrics. Moreover each testing activity is provided and accurately discussed. Section 2.6 provides a further improvement discussion with final considerations.

2.2 Background

This Section provides background information on appliance profiling, which refers to the observation of an electronic device's consumption behavior in order to extract all the features that could characterize it in detail. Appliance Profiling consists in defining a set of relations between the working states of an appliance and the energy that it consumes [27]. Thanks to the knowledge of these characterizing features, a monitoring system would be able to analyze the output of a meter and recognize the appliance(s) in use.

As suggested by Hart [19] and Zeifman and Roth [28], depending on their power profile, home appliances can be divided into four main categories:

1. *Permanent consumer devices.* Devices that are permanently on and are characterized by an almost constant power trace (e.g. smoke alarms, telephones, etc.).

2. *On-off appliances.* Appliances that can be modeled with on/off states (e.g. lamp, toaster, etc.).

3. *Finite State Machines (FSM) or Multistate devices.* Devices that pass through several switching states. An operation cycle can thus be represented through a Finite State Machine and can be repeated on a daily or weekly basis. Examples are a washing machine, a dishwasher, a clothes dryer, etc.

4. *Continuously variable consumer devices.* Devices that are characterized by a variable non-periodic power trace. Examples of such appliances include notebook and vacuum cleaners.

Furthermore, in order to characterize the behavior of an appliance, a minimal set of three power mode states can be defined [22]:

- *Active*: the appliance is fully operational; the trend of the power consumption trace depends on the specific appliance.

- *Stand by*: the appliance is turned off, but some activities continue to run. The power consumption trace is zero, except for some sporadic low consumption samples.

- *Disconnected*: the device is disconnected from the electric network.

A further classification can be made by considering the type of device load: resistive, inductive or capacitive load. This differentiation is related to the typology of device internal circuits and strongly influences its power consumption profile. The Active Power is the real part of the Apparent Power complex equation; it represents the amount of energy consumed by an appliance during its ON period. Since the Apparent Power is the product between the current and voltage effective values, then a current/voltage shifting causes a variation in the power transferred to the appliance. This variation can be detected through the analysis of the Reactive Power, the imaginary part of the Apparent Power equation, which represents the amount of power absorbed by inductive/capacitive elements and therefore not exploited by the load. As stated in [22] "the larger the current/voltage shift the greater the imaginary component" and, consequently, the lower the active power is transferred to the appliance. Therefore, the types of component that can be found in a device can be distinguished as follows:

- Inductive type: affects the power consumption by shifting the alternate voltage with respect to the alternate current (e.g. washing machine).

- Capacitive type: affects the power consumption by shifting the alternate current with respect to the alternate voltage (e.g. rechargeable battery).

- Resistive type: shows no shift of current and voltage; if the appliance is a pure resistive type, the current and voltage waveforms will always be in phase and the imaginary part (reactive power) of the complex apparent power is zero (e.g. toaster).

An appliance profile, also mentioned as "appliance signature" or "appliance fingerprint", is thus composed by several characteristics which can help to identify that specific device (e.g. real power, maximum power value, waveform shape, ON period duration, etc.).

A refrigerator power trace, for example, presents a periodic pattern whose periods depend on the overcoming of an internal temperature threshold manually or automatically set. This appliance is always connected to the electric network. A washing machine is switched on to perform a washing program and presents a consumption cycle over a specific time interval. Instead, a LCD television, even if it causes occasional consumption peaks due to sequences of very clear pictures, presents an almost uniform power trace; a microwave oven has typically a minute-usage and presents uniform peaks of high consumption. A coffee maker consumes less than the microwave oven but they have a similar behavior: long periods of inactivity interspersed with short duration periods of almost uniform consumption.

2.3 Related Work

In the Smart Home Environment, energy consumption monitoring represents one of the most challenging topics. Basically, the objective is to track individual appliances consumption to gain awareness of actual energy demands of household devices. This knowledge results to be critical for strategic energy management while aiming at improving consumption efficiency.

In literature, two strategies are mainly adopted for pursuing load monitoring as introduced above: intrusive and non-intrusive. **Intrusive Load Monitoring**, or ILM, requires the installation of a meter in association with the appliance to be monitored. An example is the so-called Smart Plug, which is placed in-between the appliance plug and the wall socket. This strategy requires some kind of user involvement because household appliances must be properly equipped with meters. A different approach is **Non-Intrusive Load Monitoring**, or NILM, which is based on a meter that detects the whole household consumption. Because of this, NILM techniques focus on discriminating the individual contributions by leveraging disaggregation strategies.

Non-Intrusive Load Monitoring techniques have been theorized since the late 80s when Hart [19] proposed to measure the total power consumption of a household through the use of an electricity meter and disaggregate the measurement into partial consumptions due to various devices in use. The advantage brought by the non-intrusiveness of NILM techniques methods consists in the installation of a single electric meter with higher capabilities in place of a distribution of low capabilities plugs. However this advantage decreases as the devices to be identified increase in number and type, due to several factors: low-consumption devices cannot be detected correctly [20], the proposed algorithms become increasingly computationally expensive, and devices with similar behavior are difficult to distinguish [21]. On the contrary ILM approaches are able to identify all kind of loads, but it is required the deployment of distributed sensors, such as the Plug [29], Plugwise [30] and SmartMeter.KOM [31] platforms which must necessarily be very low cost to justify any further cost savings. Indeed it doesn't make sense for a consumer to invest money in a whole-house distributed system of electric meters if the monetary savings is not even proportional to the investment. Moreover these distributed sensing platforms support the load monitoring and control (e.g. switch on/off) of the connected devices, but do not offer capabilities for identifying the type of attached device [20]. Both approaches have therefore their strengths and weaknesses and non necessarily one has to be better than the other, but simply more suited to a specific user needs. Thus, I have focused the research activity on both the Intrusive and Non-Intrusive Load Monitoring approaches in order to enhance state-of-the-art techniques in each field, aiming at achieving a compromise between satisfying classification accuracy results, and low installation costs. Due to the similarity of basic characteristics of these two approaches, the related work can often present some similar contributions. In order to avoid unnecessary repetitions and simplify the comprehension, I provide a common related work section for both approaches and detail the differences of our work with the state of the art in the subsections immediately below. The common related work has been subdivided in two main aspects (i.e. Features and Approaches) in order to provide a more detailed description.

Features

The Load Monitoring state of the art presents numerous works that differ in the type of features employed. Technological progress has made possible the refinement of metering hardware and allowed managing bigger quantity of data collected at ever higher frequencies. Nowadays, there are a lot available metering solutions with a configurable sampling rate. With low-frequency rates I refer to sampling rates up to 1 kHz, which allow gathering steady-state features as opposed to those known as high-frequency (up to 100 MHz) at which even the transient-state features can be detected [21]. As it can be deduced, smart meters able to manage high amount of data gathered at high sampling rate, have the advantage to provide huge amount of samples to analyze, but also the drawback of a not negligible hardware cost together with the need for a HEMS with high processing capacity and storage. According to the necessity to maintain a unitary low cost in order to install a metering device for each appliance, the low frequency is more suited for the ILM approaches.

- *Low sampling rate*

  The choice to work with low sampling rates allows analyzing steady-state features and provides several advantages from the economic point of view; the cost of the hardware required to collect these features is in fact relatively low as well as the system capacity needed to the data management. One of the most investigated feature is the Real Power, which has been defined in Section 2.2. ILM approaches usually use this only feature to perform firstly the Appliance Profiling and secondly the Appliance classification and recognition [20,25,32]. From the NILM point of view, instead, this assumption could be relatively challenging according to the disaggregation complexity to which the problem is characterized. Anyway several works [33–36] have tried to use this unique feature to perform disaggregation, especially as regards high-power consuming appliances with distinctive power draw characteristics for which satisfactory accuracy results have been reached. However, in order to distinguish devices with similar consumption traces and handle possible simultaneous state changes, other features should be taken into account [21] too, such as Reactive Power [19,37].

Other research works have investigated if further information could allow NILM systems to reach better accuracy results [19], [22], [38], [39]. Such information can be directly measured (i.e. Voltage, Current) or derived (i.e. power peaks, Power Factor, Root Mean Squared voltage and current, phase differences, etc.) [21]. Furthermore, in several works [40], [41] a Fourier series analysis have been performed to determine current harmonics, although the low sampling constraint allow extracting only the lowest ones. These additional features have helped to identify non-linear loads with a non-sinusoidal current trace and to discriminate between loads with constant power and constant impedance [21]. In most works data were sampled up to 1 kHz [19,38], while in [22] the proposed appliance classification approach was using samples gathered every 1 minute.

- *High sampling rate*

  High frequency sampling measurements have been considered in order to reach a higher detection accuracy, by taking into account also the transient-state. In [33] the power shapes of transient events have been used as features; the authors have observed that the transient behavior of several appliances is different and thus can be used as characterizing feature. In [42] the authors used as a feature the energy calculated during the “turning ON” transient event. High frequency collection also allows performing a deepen Fourier analysis and extracting higher harmonics as it have been experimented in [43]. [28] asserted that a set of harmonics (instead of a single one) can be used as complementary features of active and reactive power. In order to save resources and improve performance, [33] enhanced Hart’s method introducing harmonics analysis using transient signal. In [44] Patel *et al.* have used the high frequency analysis of the voltage noise during the transient events.

Recognition approaches

The Load Monitoring methods implemented so far can also be distinguished for the approach type. There are two ways to conceive the training phase of a learning method: supervised and unsupervised. Both of them have weaknesses and strengths [45].

A *supervised* approach make use of labeled data in the training phase in order to allow the recognition system to detect device contributions coming the different plugs (i.e. ILM approach) or from the aggregate consumption load (i.e. NILM approach) [21]. Consequently an increase in terms of both computational resource investments and human effort for the system startup phase has to be considered; however it generally offers good accuracy results. Starting from Hart’s work, in 1992 [19], which have made use of Finite State Machine (FSM), many other different supervised approaches have been proposed for both ILM or NILM, as those based on k-Nearest Neighbor (k-NN) [25,46] and Support Vector Machine (SVM) [43,47,48]. Kramer *et al.* [49] have recently performed an analysis for comparing disaggregation accuracy results achieved by different classifiers such as SVM, NN and Random Forests. As it has been shown that the temporal transitions information could improve recognition accuracy [28], few algorithms that could manage this combination have been investigated. For instance, Artificial Neural Networks (ANN) have been used in many ILM and NILM works as they offer better extensibility, dynamicity and capability to incorporate device state transition information such as in [22], [42], [38].

In an *unsupervised* approach the system does not have any a priori knowledge about the devices and often requires a manual appliance labeling when the disaggregation or the classification phase has finished.

In [50] the genetic k-means clustering has been used to isolate the Real Power and Reactive Power steady-states and to detect the number of the turned-ON devices. Zia *et al.* [51] propose an appliance behavior modeling approach which uses Hidden Markov Models on Real Power traces. One of the most recent and original unsupervised approaches is the one proposed by Kolter and Jaakkola [52] in 2012. This method consists in fact in modeling each appliance consumption behavior with a Hidden Markov Model and the aggregate consumption with the additive factorial version; the authors also proposed a new inference algorithm, called Additive Factorial Approximate MAP (AFAMAP) to separate appliances traces from the aggregated load data. Egarter *et al.* [53] propose an approach based on additive FHMM that introduces the use of Particle Filtering for estimating the appliance states. Reinhardt *et al.* [20] investigate how J48, Naive Bayes and Bayesian network algorithms are suitable for the Appliance Classification issue.

2.3.1 Appliance Profiling and Classification related work

Reinhardt *et al.* [20] suggest means to identify appliances by considering their electric current consumption. Their distributed load monitoring system is based on embedded current monitoring devices, which collect current readings at a sampling rate of 1.6kHz and extract ten features from them. A machine learning implementation maintains a model which matches these fingerprints to the learned appliance types. Results of the evaluations show that a very high classification accuracy of more than 98% can be achieved when the fingerprint of the inrush current is regarded in addition to the features extracted from the steady state current waveforms. In a contemporary work [54] the same authors evaluate the accuracy of appliance identification based on the characteristic features of traces collected during the 24 hours of a day. They evaluate nine different classifiers using more than 1,000 traces of different electrical appliances' power consumptions achieving up to 95.5% accuracy rate for the Random Committee algorithm. Ridi *et al.* [25] adopted a system based on low-cost Smart Plugs periodically measuring the electricity values and producing low-frequency (10^{-1}Hz) time series of measurements. They propose to use dynamic features based on time derivative and time second derivative features and they compare different classification al-

gorithms including K-Nearest Neighbor and Gaussian Mixture Models. The best combination of features and classifiers shows 93.6% accuracy. In [32] Gisler *et al.* present an open database of appliance consumption signatures populated with the power consumption signatures obtained using plug-based low-end sensors placed between appliance power plugs and electricity sockets. The database has been populated through two acquisition sessions of one hour for 100 different appliances distributed among 10 categories. In addition they propose two test protocols intended to compare the performance of different machine learning algorithms for appliance recognition where researchers can work on common data. In the first protocol, first session instances are used for the training set and the second session instances for the test set. In the second protocol, all instances of both acquisition sessions are successively taken in training and test sets by performing a k-fold crossvalidation.
This research activity aims at proposing a classification algorithm capable of returning high accuracy levels taking advantage of ultra low-frequency metering data (about one Real Power value every 2 minutes). These data have been collected by a distributed metering system of smart plugs which is part of a Home Energy Management System (HEMS) proposed by the Energy@Home association [55] and promoted, among other companies, by Telecom Italia. The low frequency constraint has led me to design a classification system which finds its originality in the minimization of resources required for managing and storing energy data and in the use of low-cost monitoring devices. The work that have been proposed is similar to the one by Reinhardt *et al.* [20] as regards the adoption of a distributed metering system of smart plugs. However my contribution differs in that the implemented algorithm exploits ultra low frequency metering data (a power consumption sample each 2 minutes), while the work in [20] is based on traces with higher temporal resolution (one- and eight-second average real power consumption). The low-frequency constraint has the aim of minimizing the resources needed for managing and storing measurement data in real-world home settings, as described in Section 2.4. Moreover, while the majority of the proposed approaches carry experimentation activities in centralized servers, this work includes a Java-based implementation of the classification approach that runs directly in a HEMS at customer's home.

2.3.2 Context-based Non-intrusive Load monitoring related work

In the last decade, several NILM works based on the use of machine learning approaches have been proposed, such as the ones in [22,42,43]. In 2012 Kolter and Jaakkola [52] proposed a disaggregation technique based on unsupervised learning. This technique combines the use of Additive Factorial Hidden Markov Models for modeling the appliances behavior and the authors proposed a new inference algorithm, called Additive Factorial Approximate MAP (AFAMAP,) to separate appliances traces from the aggregated load data. Few recent projects have remarked the need to provide the system with context information in order to both better characterize the appliance profiles and improve disaggregation performances which usually decrease as the number of monitored appliances increases. In 2011 Kim *et al.* [56] extended the FHMM approach with an unsupervised disaggregation algorithm that uses appliances behavior information (i.e. ON-duration, OFF-duration, dependency between appliances, etc.).

With respect to Kim's work, our original contribution is based on the addition of environmental and statistical features such as respectively the user presence and the daily usage distribution of several appliances. Also Shahriar *et al.* [57] proposed a similar approach which uses temporal and sensing information but with the aim of performing an appliance classification of power traces of single or a combination of two devices. Furthermore a private dataset has been used in both [56] and [57], thus non-comparable results have been produced; conversely our work uses a public dataset [20], which is thus available also to other researchers. Several open data sets are available: high frequency datasets such as BLUED (Building-Level fUlly-labeled dataset for Electricity Disaggregation) [58] or REDD (Reference Energy Disaggregation Dataset) [59]; low frequency data sets such as TRACEBASE [20] or ultra-low frequency as AMPds (Almanac of Minutely Power dataset) [60]. As BLUED and REDD include various features for each analyzed appliance, TRACEBASE, provides simple active power data for each monitored appliance. In [61] a detailed comparison among some of the above-mentioned public datasets has been published. The authors also provide semi-automatic labeling algorithm to help researchers in creating fully labeled energy disaggregation datasets. I chose TRACEBASE since it provides public low frequency power consumption traces of various devices gathered in real houses. Moreover the whole data set is fully labeled and contains temporal information for each power sample.

2.4 Appliance profiling and Classification

In this Section the ILM approach that I have implemented, together with the evaluation metrics and testing results, is described. I propose a classification algorithm that take advantage of ultra low-frequency real power samples (i.e. 1 sample every 2 minutes) to return high accuracy appliances' classification results. The logical architecture is fully described below as well as the Appliance Consumption dataset that have been used and the Home Energy Management System (HEMS), in which the algorithm has been integrated.

2.4.1 The logical classification architecture

The logical architecture of our classification system is shown in Figure 2.1. Raw power data are acquired by Smart Plugs - i.e. metering devices standing in between the appliance power plug and the wall outlet - and collected by the Home Energy Management System, described in the following Section.

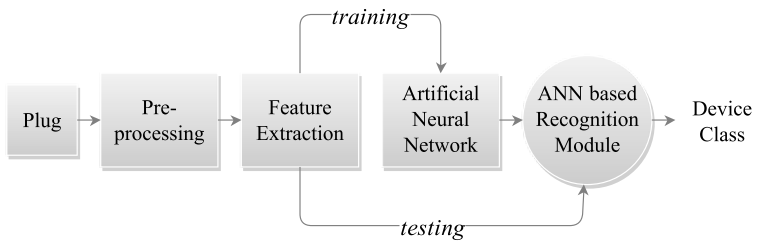


Figure 2.1: Appliance Classification System architecture.

The proposed Appliance Classification System is made of three main blocks: "Pre-processing", "Feature Extraction" and "Artificial Neural Network". An Artificial Neural Network (ANN) is a massively parallel distributed processor that has a natural propensity for storing experimental knowledge [26]. In the context of this work I preferred a Neural

Network approach with respect to other methods because of the low frequency, low numerosity and jagged typology of available data. The algorithm is implemented in the “Artificial Neural Network” block of Figure 2.1. The “Feature Extraction” block is the module where the experimental observation is processed in order to catch its most discriminating features (and labeled during the training phase), while the “Pre-processing” block process raw or incomplete data to feed the Feature Extraction block with a uniform data structure.

The *Pre-processing* block is responsible for collecting power data samples and for extracting significant power measures that will be given as input to the classification system. In order to obtain reliable results, I excluded all the power traces containing measuring faults (i.e., missing samples caused by disconnections). In addition, to fully exploit all the remaining traces, I configured an observation window set to the number of samples of the longest non-periodic trace (100 samples). Consequently, devices characterized by hourly activities such as washing machines or dishwashers will often be represented with a full-length load trace; conversely, devices with a short or highly variable duration such as microwave ovens or coffee machines, will have their power traces padded with zeros as observed in the original daily trace.

The *Feature Extraction* block processes the 100-samples trace provided by the Pre-Processing block to build a vector of features catching the peculiar characteristics of the power trace (e.g., the shape of the consumption profile, maximum peak, ascending or descending consumption steps, duration). Table 2.1 shows the features selected by the empirical analysis of available measurement datasets.

The extracted features are provided in input to the ANN-based classification algorithm, which returns the recognized type of device.

The chosen neural paradigm for this application, is a multilayer perceptron neural network with backpropagation algorithm (commonly called MLBPNN) [26], a well-known supervised model, used because of its simplicity and guaranteed convergence. This type of network is a universal approximator [62] based on the perceptron elementary neuron, i.e. an information-processing unit that takes its origin in the biological counterpart and that can be mathematically described by the following pair of equations:

Table 2.1: Feature Extraction

Extracted Features
1. Maximum power value.
2. Minimum nonzero power value.
3. No. of samples equal to zero.
4. No. of samples less than or equal to 30W.
5. No. of samples between 30 and 400W.
6. No. of samples between 400 and 1000W.
7. No. of samples greater than 1000W.
8. No. of steady state consumption changes greater than 1000W.
9. No. of steady state consumption changes between 10 and 100W.
10. Average value of the nonzero power samples.

$$net_k = \sum_{j=1}^{m} w_{kj} x_j \tag{2.1}$$

$$O_k = \psi(net_k + b_k) \qquad k = 1, ..., N_{out} \tag{2.2}$$

where o_k is the output of the kth neuron (if the used neurons are more than one, in the given number N_{out}); $x_1, ..., x_m$ are the input signals (with dimension given by m); $w_{k1}, ..., w_{km}$ are the synaptic weights of the neuron k; net_k is the weighted sum of the input signals, b_k is an external bias and ψ is the "activation function"; in this application the activation function is the bipolar sigmoid:

$$\psi(net_k) = \frac{2}{1 + e^{-\lambda net_k}} - 1 \tag{2.3}$$

that is chosen because of a couple of main reasons: i) sigmoid is a nonlinear activation function that allows to quickly perform a classification of non-linearly separable problems when used in multi-layer structures; ii) in addition, sigmoid allows to use well known always convergent backpropagation algorithm as steepest descent [63] or Levenberg-Marquardt [64] during the learning process. The multi-layer structure is a three-layer network with a number of neurons 10-30-8, respectively in the input-hidden-output layers; the number of neurons in input and output layers is imposed by the nature of the problem, whereas the number of neurons in the hidden layer (30) is coming from an empirical choice, motivated by the best "trial&grow" result. In other words, the number of neurons starts from a medium low number (around 10) and is increased until no further results improvement is observed. The ANN configuration is represented in Figure 2.2. The adopted error function is the classical mean squared of errors (MSE). As discussed in Section 2.4.5, I also performed some tests using Bayesian regularization, in this case the error function specified above is corrected with the mean squared of the network weights as regularization term. In order to avoid the generalization drawback, an early stopping approach is used [65] during the learning phase. Once the ANN block has been trained with the proper knowledge base, the ANN Recognition Model is ready to classify new power consumption traces. As detailed in the evaluation Section, I exploited to this purpose the measurement data sets collected during the trials in the Telecom customers' sites.

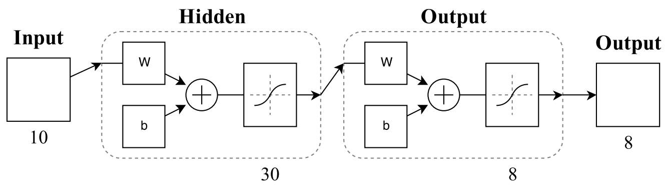


Figure 2.2: The Artificial Neural Network configuration adopted for the experiment.

2.4.2 The HEMS architecture

This work extend the Home Energy Management System (HEMS) proposed and developed by the Energy@home association [66] with device load classification capabilities provided by the ANN-based Appliance Classification Systems. Energy@home is a no-profit association participated by some major Italian companies (i.e. an electric utility company, telecommunications operators, manufacturers) and research bodies that aims at developing and promoting technologies and services for energy efficiency in the home based upon device to device communication.

Energy@Home Home Energy Management System

The Energy@Home HEMS is an OSGi-based software infrastructure for home automation and energy management based on a Home Area Network (HAN) of objects that cooperate and communicate through the the ZigBee protocol [23]. Main cooperating objects are smart plugs, which have both sensing and actuating capabilities, and various types of smart devices and actuators, ranging from simple on-off devices, (e.g. a TV-set), to configurable devices (e.g., air-conditioner) and smart appliances.

The OSGi [67] standard specifications define a Java-based service platform made by software modules (i.e. OSGi bundles) that can be installed, stopped, started, updated, and uninstalled at runtime. The OSGi Service Platform is made of a service execution framework and a set of standard service definitions that support the development of service-oriented applications in networked environments. This motivates its widespread adoption in the development of Home Energy Manamegement Systems, such as PeerEnergyCloud [68] and Ogema [69]. The OSGi framework provides a simple component model, a service registry, and utilities for dynamic service deployment, while the standard service definitions specify the interfaces and semantics for some reusable services (e.g. a logging service and an HTTP service) [70]. Services are implemented as bundles, which are Java archives that contains code, resources and a manifest file with meta-information such as dependencies and activation. When the bundle is active, it can publish its services or discover and bind itself to services provided by other bundles through the service registry. The HEMS OSGi Service Platform hosts a set of web applications providing users with home automation and energy consumption awareness capabilities.

A typical home-based deployment of the Energy@Home system is made of the following hardware components:

1. A *Smart Info* device provides end users with the certified information on electricity consumption managed by the electronic smart meter. It can be plugged in every domestic socket to collect data

from the smart meter leveraging powerline communication. The Smart Info can be provided by the Distribution System Operator (DSO). Published data are a sub-set of those already made available by the home electricity meter, hence the Smart Info acts like a proxy in respect of the meter.

2. *Smart Appliances* are white goods (e.g., dishwasher, washing machine) that have local intelligence and networking capabilities. They can provide information on their energy consumption (e.g. used energy, instant power, etc.), respond to remote commands and interact with the user through a GUI.

3. The *Home Gateway* is the core of the Energy@Home HEMS. It is the centralized management component that connects the Home Area Network with external application services via Wide-Area Network (WAN) connectivity. It is based on a modular and highly configurable OSGi framework and hosts application logic modules. It offers multiple network interfaces, including a HAN interface to communicate with the abovementioned home devices via the ZigBee protocol, a Home Network (HN) interface to interconnect additional local devices, such as PCs, TV via wired and wireless LAN, and a WAN interface used to communicate with remote service providers' systems (through xDSL connection). In addition, it provides local service logic and remote services with high-level APIs for discovering, managing, and communicating with HAN devices. The protocol used for the communication between the HAN devices and the Home Gateway is based on ZigBee, since it is a low-cost, low-power-consumption, two-way, wireless communication standard [71]. ZigBee can be used in different application domains (e.g., home automation, healthcare, energy management and telecom services) and a set of extensions have been designed for the Energy@home system [72] and integrated in the version 1.2 of the Home Automation profile specification, ratified by the ZigBee Alliance in the second half of 2013.

4. *Cloud Services*: the Home Gateway interacts with a remote Service Platform hosted in the telecommunication operator's data centers providing storage and processing capabilities. The Service Platform collects and stores the data sent by the Home Gateway.

It can host applications that performs statistical and analytics processing on historical data for providing users with consumption awareness, predictions and possible suggestions for consumption reduction. The Service Platform can also host downloadable application bundles that can be selected by users and eventually run on a local home service gateway.

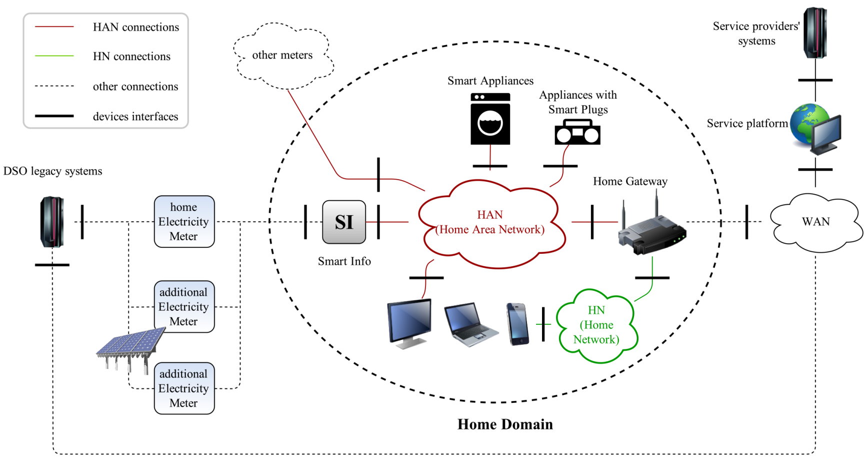


Figure 2.3: Energy@home Home Energy Management System architecture.

This HEMS has been deployed in 20 private homes in Italy and both experimentation and data collection are currently ongoing. Several types of monitoring approaches (whole house monitoring, real-time monitoring of identified devices through smart appliances, low-frequency monitoring of unidentified devices) are in place, thus allowing the experimentation of different data analysis and service provisioning approaches.

Home Gateway Architecture

The Home Gateway is an OSGi based system made by several bundles, whose interaction is realized through service provision and consumption. It is made of a set of key bundles (i.e., the *Java-Gateway Abstraction Layer*, the *Home Automation Core* and the *Local Gateway Service*

Logics bundles) supporting the development and deployment of energy monitoring and management applications. Hereafter I will briefly describe the features provided by these Home Gateway components and subsequently I will focus on the classification algorithm implementation.

The *Java-Gateway Abstraction Layer* (Java-GAL) implements the ZigBee Gateway Device specifications for managing ZigBee Networks and guaranteeing interworking with IP [73]. Java-GAL implements the features required to perform active nodes discovery in order to keep a constantly updated image of the current ZigBee network. It also provides an abstraction layer that allows applications to control and access ZigBee devices hiding low-level implementation details. To this purpose, it translates the ZigBee product-dependant low level APIs into a set of HTTP/REST and local APIs that can be invoked by other bundles as well as by external applications to control ZigBee devices.

The *Home Automation Core* offers a high-level API exposing attributes and commands defined by ZigBee Home Automation 1.2 Service Clusters. It also implements a basic web-based interface allowing the users to configure and test the home automation system (e.g. for adding and configuring new ZigBee devices).

Local Gateway Service Logics: application bundles that exploits the high-level Home Automation Core APIs for providing users with value-added services for energy management and home automation.

The HomeGateway has been implemented by leveraging the OSGi Equinox 3.5.2 Framework. More detailed specifications of the Home Gateway supported ZigBee Service Clusters can be found at [72] and the HomeGateway source code is available online [74].

2.4.3 Classification Algorithm

The ANN-based classification algorithm has been implemented as an OSGi bundle, called Classifier bundle, deployed in the HomeGateway as one of the *Local Gateway Service Logics* OSGi bundles.

The software logic that implements the classification algorithm contains the following main classes:

1. *ILM_Model*: is the class that handles the information used to configure the neural network. This information can be provided as input as an xml file and specifies the number of inputs, hidden

nodes and outputs of the network and, for each input row, the minimum and maximums value of the allowed range values, which are used in the preprocessing phase to normalize data entering the ANN.

2. *ILM_Classifier*: is the class that implements the ANN-based classification capability. The class offers the methods for training and test the ANN, save it as a file, and finally, classify input data. The class has been implemented by relying on a Java-based framework for Neural Networks, Neuroph [75], that have been wrapped in an OSGi bundle and deployed in the Home Gateway.

2.4.4 Evaluation

In this Section I describe the testing activities carried out for evaluating the proposed ANN-based approach for low-frequency distributed load monitoring.

This Section is structured as follows: first, I introduce the dataset used for testing and explain how data have been acquired from an experimentation campaign in real households, subsequently, I explain the performance evaluation procedure of the proposed algorithm.

Data Collection from Home Trials

Data collected from a trial of the Energy@Home system carried out in 2012 have been put together to form the datasets used for the test. The trial included 10 private houses of collaborating italian customers where the Energy@Home HEMS was deployed in, with the following configuration: i) a Smart Info device connected to the home electricity meter; ii) many Smart Plugs to collect consumption information from connected loads, such as washing machine; refrigerator; dishwashing machine; smart TV; iron; microwave oven; lighting stuff; coffee machine.

Energy consumption data extrapolated from the activity of these devices are used by the Home Gateway and the Service Platform to implement use cases with the aim of enhancing customer awareness of energy consumption [76]. In order to meet the trial application requirements, energy and instantaneous power data are collected from HAN devices and stored in the remote Service Platform database so that customers are provided with historical and statistical information on their energy consumption. Stored data include whole in-house consumption from the power meter (Smart Info) and single device energy information coming from Smart Appliances and Smart Plugs.

The HAN devices are connected to the Home Gateway via ZigBee protocol and the Home Gateway uses the reporting strategy defined in the ZigBee Cluster Library specification [77] to receive from each of them the following consumption information: i) the summed value of energy delivered and consumed in the premise (Smart Info) or by a specific device (Smart Plugs and Smart Appliances); ii) the instantaneous real power absorbed by the whole house (Smart Info) or by a specific device (Smart Plugs and Smart Appliances).

The reporting parameters configured on each Smart Plug and Smart Appliance provide real time instantaneous power information: every change in instantaneous power that is greater than or equal to 5 W is notified to the Home Gateway with a maximum configured delay of 2 seconds. Subsequently, the Home Gateway processes these data to provide users with real time information. Please note that all these measurements are not directly stored in the platform database. Indeed, to avoid storage overloads, only a subset of these data filtered by the Home Gateway is retained. Information is stored on the basis of the reporting of summed energy values sent every 2 minutes by each device: for each of these time intervals, the gateway calculates the device's energy consumption (Wh) and stores this value along with the minimum and maximum instantaneous power values (W) pertaining to the same time interval.

Data Analysis and Pre-processing

The test cases have been led using a new dataset created from a subset of the data collected during the Energy@Home trials. Among all the available devices, I selected the ones that were present in all three houses (washing machine; refrigerator; dishwashing machine; smart TV; iron; microwave oven; lighting stuff; coffee machine) and for each type of device I extracted the same number of traces.

This new dataset is made of 528 examples (66 for each of the 8 monitored devices) and contains the first 100 samples of each device load curve of power, measured in Watt. Signatures are entered according to the following criterion: given a device, I consider as a signature the trace starting from the first non-zero sample of the daily consumption curve and ending with the hundredth sample, regardless of the duration of the consumption itself. If the consumption trace exceeds the hundredth sample, it is simply trimmed, while shorter traces are expanded by padding them with zero samples. Finally, these signatures undergo the features extraction process.

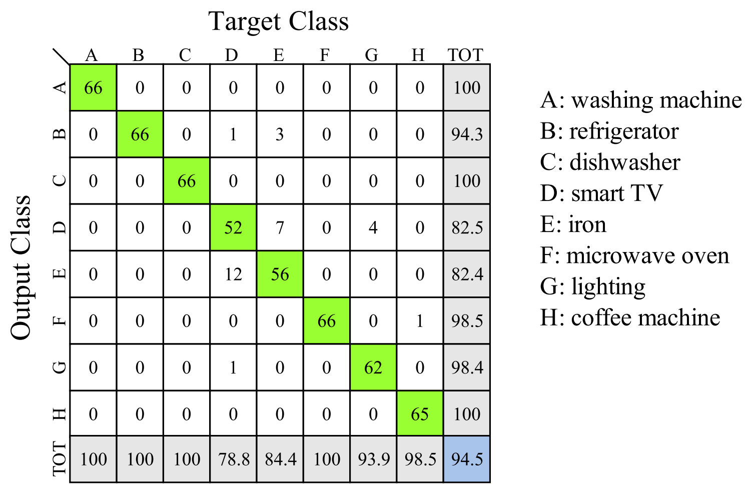
This criterion has been defined in order to preserve the nature of the original daily traces belonging to the dataset collected in the home trials; in fact, the appliances consumption is represented by power samples separated by zeros (no consumption detected). Moreover, I considered that a 100 samples window (the longest non periodic trace duration) corresponding to a 200 minutes time interval would be sufficient for exhaustively represent a device signature of the dataset. Zero-padding has been performed on traces characterized by less than 100 samples for keeping them accordant with the original and isolating them for proper classification. The trimming operation has been basically performed on "always on" devices traces characterized by a periodical consumption (e.g. freezer, refrigerator); for such devices zero-padding the end of a single signal period could cause a significant loss of information while using a properly set trimming frame contributes to maintain the original signature shape.

This dataset was divided in three parts; the 70% of examples has been allocated to train the ANN, the 15% for validation, while the last 15% has been used for testing purposes. In order to validate the effectiveness of the algorithm, the "overall accuracy" index in terms of percentage have been used.

2.4.5 Evaluation and Testing Results

In this Section I present the test cases carried out for the accuracy estimation of our classification algorithm. The software used for testing is Matlab version R2012a running on a machine having an Intel Core2 Duo CPU T7500 at 2.20GHz, 2 GB RAM. Figure 2.4 shows the overall confusion matrix resulted from a single classification test of the above mentioned 528 samples dataset (66 per device). The matrix rows and columns represent the output and the target classes, respectively; in the diagonal the correctly classified samples for each device are reported. Observing the iron (E) as an example, the ANN has correctly classified

56 out of the 66 samples; the remaining 10 samples have been misclassified 7 times with the smart TV (D) and 3 times with the refrigerator (B).



Target Class

Output Class	A	B	C	D	E	F	G	H	TOT
A	66	0	0	0	0	0	0	0	100
B	0	66	0	1	3	0	0	0	94.3
C	0	0	66	0	0	0	0	0	100
D	0	0	0	52	7	0	4	0	82.5
E	0	0	0	12	56	0	0	0	82.4
F	0	0	0	0	0	66	0	1	98.5
G	0	0	0	1	0	0	62	0	98.4
H	0	0	0	0	0	0	0	65	100
TOT	100	100	100	78.8	84.4	100	93.9	98.5	94.5

A:	washing machine
B:	refrigerator
C:	dishwasher
D:	smart TV
E:	iron
F:	microwave oven
G:	lighting
H:	coffee machine

Figure 2.4: A confusion matrix sample resulting from a single classification test

Table 2.2 shows the classification results for each device and for five test iterations, where each iteration includes an independent training phase and, consequently, a different ANN configuration. The final column shows the average accuracy value for each device. The Overall Accuracy row shows the algorithm successful recognition percentage for all the devices.

The relevant fact that emerges from Table 2.2 is the high accuracy achieved for each device mostly as regards the dishwasher, washing machine and coffee machine. Conversely, devices such as smart TV, iron and lighting can be misclassified because of their usage duration variability.

The results presented so far have been achieved while trying to classify traces belonging to devices whose signatures have been used in the training phase. From an application point of view, this would call for a preliminary collection activity where the user is requested to associate a signature with the proper label. Unfortunately, such procedure can not be implemented when electric power consumption data are used in unsupervised approaches. Therefore, I tried to evaluate the accuracy of the ANN in recognizing types of devices whose signatures were not used in the training phase.

Table 2.2: Classification Accuracy for Each Device Type and Test Iteration

	Correct classification percentage (True Positive)					
	test1	test2	test3	test4	test5	**total**
washing machine	100	100	100	100	98.48	99.70
refrigerator	92.42	100	100	100	96.96	97.88
dishwasher	100	100	100	100	100	100
smart tv	69.69	84.84	84.84	90.90	81.81	82.42
iron	90.90	89.39	95.45	90.90	86.36	90.60
microwave oven	100	100	100	100	100	100
lighting	90.90	90.90	93.93	93.93	90.90	92.11
coffee machine	98.48	100	100	98.48	100	99.39
overall accuracy	92.80	95.64	96.78	96.78	94.31	**95.26**

A second test case consisted in testing the ANN using both a Levenberg-Marquardt (LM) and Bayesian regularization (BR) configurations with power traces collected from a new house. To this purpose, I selected the devices monitored in all the houses i.e. washing machine, refrigerator, dishwasher and microwave oven.

Table2.4.5 shows the test results: although these appliances are completely unknown to the ANN, the washing machine and the smart TV have been detected albeit with medium and low accuracy values, which decreases dramatically for both refrigerator and dishwasher. While comparing results obtained with a LM network with the ones coming from a BR network, it is evident how the latter slightly improves the identification of those appliances whose time of use is considerable, while dramatically worsens the identification of the ones whose consumption has typically a short duration and an on/off behavior and thus cannot be easily characterized at such low monitoring frequency (e.g., microwave oven). The result of the second test case have helped improve the features extraction criterion. According to the results in Table 2.4.5, the ten features chosen to characterize the load curve appear to be over-specific. Therefore, I reduced their number to six, by choosing the more general ones (i.e., features 1-2 and 7-10 in Section 2.4). Overly specific features, such as the number of samples between a minimum and a maximum value (features 3-6 in Section 2.3), could in fact over-describe a power consumption signal worsening in generalization. In other words, I selected those features that better describe device classes rather than particular usage patterns: for example, two smart TV traces can differ significantly with respect of the usage time, still preserving typical characteristics. If I include the number of zero samples (feature 3) as a feature I will force the ANN to consider a usage pattern as a possible discriminating characteristic resulting in an unnecessary complication of the classification task.

Table 2.3: Classification Accuracy for Previously Unknown Devices

	Average correct classification %			
	10 features		6 features	
	LM	BR	LM	BR
washing machine	56.93	8.3	98.4	98.32
refrigerator	4.38	11.5	83.73	99.52
dishwasher	6.32	1.5	16.13	24
microwave oven	37.47	11.6	44.15	0

Since the classification OSGi bundle has been designed to be executed on the Home Gateway, some tests have been carried out for estimating its time performance on a typical HEMS deployment. The Home Gateway used for these tests is equipped with an ARM Processor 400MHz, 1 GB DDr 200 MHz RAM and 2 Gbit NAND Flash memory. The operating system is Linux 2.6.35 with Java SE for Embedded 6 (JAVA ARMv5 Linux - Headless) and the Equinox 3.5.2 OSGi Framework installed.

As regards the machine learning part, the ANN has the same configuration as the one detailed above. The tests have been carried out using the measures coming from the smart plugs preprocessed by taking 100 samples, trimming or zero padding the curves as needed. Again, the signatures obtained at this stage undergo a feature extraction process to obtain a vector of 10 discriminating features.

For each type of device (washing machine; refrigerator; dishwashing machine; smart TV; iron; microwave oven; lighting stuff; coffee machine) 10 preprocessed power traces are entered in the system to measure the output time. The average classification time resulted to be 4ms. Moreover, the average time needed to load and run the ANN has been measured to be 269ms. This is meaningful because the ANN could also be changed when a new neural network trained with more examples is available, and this would require a complete reload.

2.5 Context-based Non-Intrusive Load Monitoring

This Section describes the proposed new energy disaggregation algorithm. First, I briefly mention the principles of the state of the art approach that have been adopted and then I describe how this approach has been enhanced by leveraging context information (e.g., timing usage statistics and user presence).

2.5.1 The probabilistic data model

The observation of the devices' consumption traces has underlined that most of them usually switch from a power consumption value to few others during each period of use; every trace can thus be considered as a set of transitions from a consumption level to the subsequent one. Consequently, the mean value of each power level with its associated variance can be regarded as a state whose sequence can be modeled with Hidden Markov Models (HMM). HMM is a probabilistic learning method for time series where the information about the past is transmitted through a single discrete variable, precisely named "hidden state" [78]; in this work the HMM represents the power consumption evolution as a sequence of states. Such Markov processes are labeled starting from the outputs; analyzing the observed state, the algorithm assess what is the most likely Markov model hidden state capable of generating the observed output. Each device, thus, have been modeled through an HMM according with the power states and the transition matrix which determines the probabilities of each state to evolve in another. HMMs have been treated as additive Factorial HMM as described in [78] to consider the Â overlaps of independent utilization of each device; the observed output is thus composed as a state additive function of the different hidden states and, in this case represents the aggregated power consumption.

The single hidden Markov model, with its conditional independencies, is graphically represented in figure 2.5a, where a sequence of observations $\{Y_t\}$ with $t = \{1, ...T\}$, is modeled by a probabilistic relation with a sequence of hidden states $\{S_t\}$, and a Markov transition structure connecting the hidden states [78].

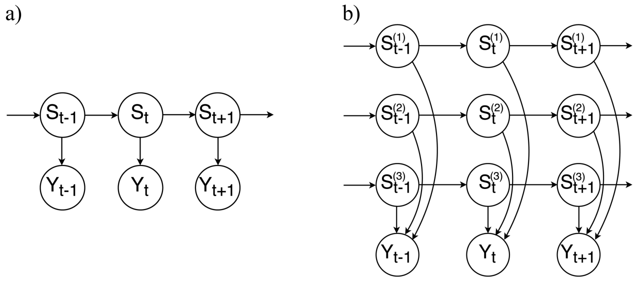


Figure 2.5: a) The single Hidden Markov Model b) The additive Factorial Hidden Markov Model [78]

The state can take one of K discrete values, $S_t \in \{1, ..., K\}$, which have been extracted from the occurrence histogram composed by several power traces for each appliance, through the use of a clustering technique (i.e., Gaussian Mixture Model). The transition matrix has therefore a $K \times K$ dimension and represents the state transition probabilities, $P(S_t|S_t - 1)$.

Figure 2.5b shows the additive FHMM in which each independent HMM for each monitored device evolves in parallel. The sequence of observed output $\{Y_t\}$ represents the aggregate hidden states (i.e. the aggregated power consumption); the algorithm thus estimates which is the most probable sequence of Markov hidden states that could have produced that output.

2.5.2 NILM Inference

As mentioned above, Kolter and Jaakkola [52] in 2012 proposed an approach based on additive Factorial Hidden Markov Model that aimed at improving inference complexity performances and avoiding local optima issues [52]. As the number of devices to disaggregate grows, in fact, the evaluation of all the possible HMM evolutions that could have generated the aggregate output implies an increase in the computational complexity of the disaggregation process. Therefore, the authors proposed an algorithm called Additive Factorial Approximate MAP (AFAMAP) which is able to bypass the unreachable exact inference through the approximation of the Maximum A-posteriori Probability [52]. [52] have released a Matlab version of the AFAMAP algorithm (2012); in their paper they provide some test results and discuss the effectiveness of the algorithm compared to other inference algorithms (i.e. Maximum A-posteriori Probability, Structured Mean Field, etc.) in terms of disaggregation error.

For simplicity, I do not quote the mathematical model as it is available in detail in [52] with some comparative results.

2.5.3 Context-based disaggregation

The contextual conditioning has been realized by adopting and extending the Conditional FHMM [56] solution, which allows integrating context information to the classical FHMM in order to obtain dynamical, rather than static, state transition matrices. Among the various approaches made available in literature, including in particular the Conditional Random Fields [79], our choice fell on Conditional FHMM as it allowed extending the approach by [52], while maintaining the use of the above-mentioned AFAMAP inference algorithm. As the Conditional FHMM works on transition matrices is explained below with a graphic example.

The following types of context information have been selected:

- timing-usage statistics, which has been generated through a statistical analysis over the Tracebase dataset.

- user presence information, which has been synthetically generated for the purpose of this work. In real world cases, these data could be collected through presence sensors located in the private home rooms.

The selection of these features has been performed taking into account cost of real-life deployments. Therefore I preferred to use a very small number of presence sensors, typically one for each room in the house, instead of a huge number of different sensors (e.g., pressure, ignition switches, movement, etc.).

Timing usage statistic conditioning

The Tracebase dataset provides active power measurements and their relative sampling instants. This information has allowed me to evaluate the timing-usage statistics of the different devices. From the available daily measurements, the number of turning-ON events (OFF-ON transitions) of each device in time intervals of 30 minutes has been derived. Then, the occurrence histograms of the turning-ON events (in 24 hours evaluation periods) have been generated. Their analysis has allowed extracting the usage probability distribution of each device and detecting eventual devices inactivity according to which this conditioning has been performed. For example, Figure 2.6a shows the occurrence histogram of a refrigerator. Relevant trends for the conditioning are not visible, due to the "always ON" nature of the device. Vice versa, a washing machine (figure 2.6b) shows a very low turning-ON probability during night hours; this information can thus be employed to modify the state transition probabilities of this device.

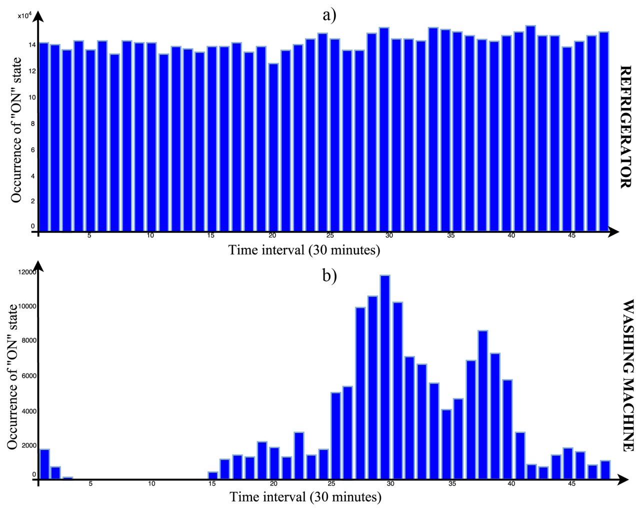


Figure 2.6: Usage statistics distribution for a refrigerator (a) and a washing machine (b)

User presence/absence conditioning

The necessary information to perform this second conditioning derives from a presence sensors appropriately deployed in the house. This type of conditioning consists in modifying the probability associated to the "OFF-ON" transitions of a device trace, according to the presence/absence of a user in a specific time interval. It was assumed that for specific types of appliances, the turning-ON event cannot occur without the presence of a user.

- Presence/absence of users in a single time interval;
- Presence/absence of users in two consecutive time intervals.

The time interval duration is arbitrary set as the longest consumption cycle that have been detected among the analyzed traces (i.e. a washing machine cycle).

In order to clarify this critical claim, it is useful to analyze the operating characteristic of a washing machine through a three-state Markov chain: state 1 "OFF", state 2 "WASHING PROGRAM", state 3 "WATER HEATING". In figure 2.7, the P_{ij} terms indicate the transition probability from the state j to the state i, typical of the single device.

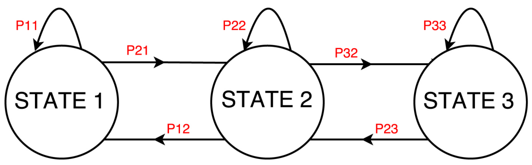


Figure 2.7: Example of a State Machine with associated transition probability modeling a washing machine operation

The example shows that the device in examination does not provide, among the possible transitions, the transition from state 1 to state 3 without passing through state 2 (P_{13} and P_{31} are nil). Table 2.4 shows the probabilities to transit from state i to state j and vice versa in a single step.

User *absence* in the single time interval implies that:

Table 2.4: Example of a transition matrix for a washing machine

j / i	S1	S2	S3
S1	0.9978	0.0022	0
S2	0.0307	0.9690	0.0003
S3	0	0.0012	0.9988

- If the device is ON, it will continue its customary working cycle until the end of the washing cycle;
- If the device is OFF, the transition to state 2 is not possible during the time interval (because there are not users in its neighborhood), thus: $P_{21} = 0$ as it is shown in figure 2.8 and in table 2.5.

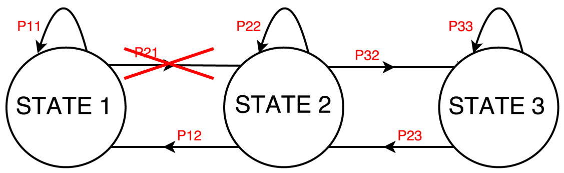


Figure 2.8: Example of a state machine for a washing machine derived by taking into account user presence information

Table 2.5: Example of a transition matrix for a washing machine derived by taking into account user presence information

j / i	S1	S2	S3
S1	1	0	0
S2	0.0307	0.9690	0.0003
S3	0	0.0012	0.9988

The user absence for two consecutive time intervals, instead, implies that:

- If the device was ON before the beginning of the observation (before the first interval), it will have terminated its working cycle within the first interval, therefore it is currently OFF (OFF in the second interval);

- If the device was OFF, it will not have had a new turning-ON, thus it is currently OFF (always in the second interval). The Markov chain for the device in the second time interval collapses in a single state, precisely the OFF state, with, as the sole possible transition, itself; the probability from state 1 to state 1 results thus unitary ($P_{11} = 1$) as shown in figure 2.9 and in table 2.6

.

Hereafter I will refer to these conditioning mechanisms as follows: Usage Statistic Conditioning (USC) and User Presence (UP) single/double Interval Conditioning (IC).

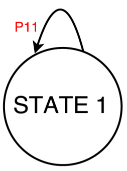


Figure 2.9: State machine of a washing machine taking into account user presence information for two consecutive time intervals

Table 2.6: Transition matrix of a washing machine taking into account user presence information for two consecutive time intervals

j / i	S1	S2	S3
S1	1	0	0
S2	0	0	0
S3	0	0	0

2.5.4 Evaluation

In this Section I describe the experimental activities carried out to validate our approach. First, I describe the dataset [20] that has been used, and how I extracted the HMM models for each considered appliance. I then show and discuss a meaningful disaggregation test for each context-based conditioning mechanisms by providing both the graphical and the numerical disaggregation results at appliance level. Averaged disaggregation results are also discussed for 4 different test cases and compared with the basic algorithm by [52]. Appliance profiling has been performed using Python scripts on a machine equipped with an Intel Core 2 Duo P8400 at 2.26GHz, 3 GB RAM; disaggregation test campaigns have been performed using Matlab version R2012a on a machine equipped with an Intel Core2 Duo CPU T7500 at 2.2 GHz, 2 GB RAM and another with Intel Core 4 i7-3610QM at 2.3 GHz, 8 GB RAM.

In order to provide experimental results which could be compared with those of other works, the precision and recall parameters [80] have been chosen. The parameters are calculated as follows:

$$Precision = \frac{True\,positive}{True\,positive + False\,positive}$$

$$Recall = \frac{True\,positive}{True\,positive + False\,negative}$$

Considering the real and the disaggregated power samples for each device:

- The true positive parameter represents the number of samples that have been correctly classified or more precisely the power quantity correctly assigned to that device.
- The false positive parameter represents the number of samples that have been incorrectly classified or more precisely the power quantity incorrectly assigned to that device.
- The false negative parameter represents the number of samples that should be but have not been classified or more precisely the power quantity that should have been assigned to that device but have been assigned to another or have not been assigned at all.

The *precision* parameter measures the portion of power samples that has been correctly classified among the power samples assigned to a given device. The *recall* parameter measures what power portion of a given device is correctly classified in general, also considering that samples that would belong to that device but have been wrongly assigned to another or not assigned at all.

In order to show a general parameter that could combine the results obtained through the *precision* and *recall* analysis, the *F-Measure* parameter has been considered and calculated as follows.

$$F - Measure = 2\,\frac{Precision\,Recall}{Precision + Recall}$$

Although *F-Measure* represents a statistical combination of *precision* and *recall*, in our experimentation the first parameter has a more pertinent meaning in the single appliance disaggregation results, as it enhances the percentage of the right assigned power samples. For this reason, in our test-cases discussed below, *precision* results are shown at a single appliance level, while at a test and overall level recall and F-Measure are also pointed out.

Data analysis and pre-processing

As a preliminary step, I have evaluated Tracebase [20], which is the dataset that have been adopted in this work, and performed few preprocessing operations on the data. Tracebase, which has been introduced in Section 2.3, is a public, password-protected dataset. it consists of real power consumption traces of a range of electric appliances that have been collected in more than ten households and office spaces. The trace collection script, described by [20] (2012), has been configured to gather one sample per second; furthermore every sample is stored with its timestamp. However, because of the topology of the data collection network and the encountered delays, the authors stated that traces may also show a higher or lower frequency; this physical characteristic forced to perform an accurate data analysis and a pre-processing phase that are described below. Moreover, this dataset is conceived to perform the appliance classification, thus it provides reliable power consumption traces as they all have been detected with a dedicated smart plug. Therefore, it does not include an aggregate consumption signal. In this work, I set up a synthetic aggregate power trace consumption that is composed of a sample-sum of a selected subset of the available traces. Indeed, Tracebase includes up to 1270 monitoring traces of 122 devices of 31 different appliances types, but I used a subset made by 423 traces of 43 devices belonging to 6 types (table 2.7), by selecting those devices that presented a major number of traces and less holes in the monitoring interval.

As stated by [20], Tracebase presents several detection inhomogeneities; a daily power trace can in fact shows more than one sample per second

Table 2.7: Tracebase [20] subset of appliances used in our approach

Device Type	#appliances	#traces
Coffee Maker	5	39
LCD TV	10	94
Microwave Oven	5	48
PC Desktop	9	90
Refrigerator	7	130
Washing Machine	7	22
6	**43**	**423**

or a lack of data for some seconds. Hence, I processed all the daily power trace by normalizing each one with 86400 samples (number of seconds in a day). I have performed the zero-padding or the average operations on the missing/surplus samples and put the obtained values in a normalized trace. These operations have become reasonable after the data analysis. For instance, when a device has resulted disconnected to the electric network (OFF state), the meter has obviously gathered a zero-consuming trace; thus I have zero-padded the missing samples and reduced in an only zero value the surplus samples gathered at the same second. Moreover, when the device is active (ON state) I have evaluated the samples immediately before and after the missing one/ones and performed an average operation of them and then filled the missing value with the obtained result. An analogous operation has been performed in the case when there was more than a sample per second.

Appliance Profiling

In order to extract the power levels that typically characterize an appliance consumption, I analyzed all the available traces for each appliance in our subset. As mentioned above, according to the consumption behavior and the nature of the devices, each device consumption profile can be approximately characterized by just few power states. To identify them, I generated for each type of device a power value occurrence histogram aimed at highlighting the most frequently achieved values ranges.

After this operation, a further sub-sampling operation is performed. The zero power, which corresponds to the disconnected state, could in fact mislead the research of accumulation values as it reasonably represents, except for the "always ON" appliances, the most frequent sample value. Therefore a coherent sub-sampling has been applied by processing each sequence through a sliding window of fixed size (10 samples); in the case where the samples observed in the window result all 0, nine values of these will be barred from the data on which to search the state value. After this pre-processing phase, the problem of determining the intrinsic structure of the data to be grouped, in the case that only the observed values result accessible have been considered; hence the preciseness of the state extraction has been tested through clustering analysis [81].

Clustering analysis organizes the data according to an abstract structure in order to recognize groups or hierarchies of groups. A cluster is composed by a number of similar objects collected or grouped together according to a specific parameter named distance. How the distance is set up and which parameter it represents depends on the chosen algorithm and on the type of data to be processed. In our experimentation case, the objects are represented by the power values; the clustering algorithm has to evaluate and group them together in order to extrapolate few power states that could effectively describe the consumption behavior of each type of appliance. The clustering algorithm identifies few mean values and their associated variances that could represent as accurately as possible each consumption state.

To solve the problem, preliminary tests with some clustering algorithms have been performed; k-means [82] and Gaussian Mixture Model (GMM) [83] reported the most consistent results.

With k-means, given a set of n points defined in a d-dimensional space R^d and an integer k, the problem consists in determining a set of k points, belonging to R^d, called centroids, such that each mean squared distance for each point belonging to the cluster is minimal when compared to the centroid. In our case the centroids represent the states of the descriptive model which is associated with each device observed. This type of technique usually fails in the general categories of clustering that are based on the variance [84]. As mentioned above I have also investigated a probabilistic approach, named Gaussian Mixture Model (GMM). It is assumed that the data are generated by a mixture of latent probability distributions in which each component represents a different group of clusters [81]. It consists in the weighted sum of M components of Gaussian densities as described by the following equation:

$$p(x|\lambda) = \sum_{i=1}^{M} \omega_i g(x|\mu_i, \sum_i)$$

Where x is a D-dimensional data vector (e.g. measured features), ω_i $(i = 1, ..., M)$ represent the mixture weights and $g(x|\mu_i, \sum_i)$ with $(i = 1, \ldots, M)$ are the components of Gaussian densities. Each Gaussian component is represented by the following shape:

$$g(x|\mu_i, \sum_i) = \frac{1}{(2\pi)^{\frac{D}{2}} |\sum_i|^{\frac{1}{2}}} \exp\{-\frac{1}{2}(x - \mu_i)' \sum_i^{-1}(x - \mu_i)\}$$

Where μ_i is the mean vector and $\sum_i$ is the covariance matrix. The mixture weights meet the following condition:

$$\sum_{i=1}^{M} \omega_i = 1$$

There are several variants of the GMM that have just been introduced, depending on the calculation type of the parameters that describe the distribution. The choice of model configuration (number of components, dense or diagonal covariance matrices, link among parameters, etc.) is often determined by the amount of available data to estimate the parameters of GMM and the environment in which the GMM is applied. One of the most important attributes of GMM is its ability to form smooth approximations of arbitrary distribution densities. A GMM acts as a sort of hybrid that uses a discrete set of Gaussian functions, each with its own parameters (mean and covariance matrices), in order to permit a better modeling capability. In this work the data model that have been associated to each device is composed from the following components:

- The data $x = \{x_1, \ldots, x_L\}$ represent the sample data which is in turn a realization of $X = \{X_1, \ldots, X_L\}$
- X_i represents the i^{th} data flow which is described by a d-dimensional feature space $\{F_1, \ldots, F_D\}$.
- X can be divided in data that have been labeled as X_1 but not X_u.

- $K = \{k_1, \ldots, k_N\}$ represents the set of state classes that are associated to each device.

Therefore our clustering problem is reduced to finding the N-states which better represent each monitored device [85].

The GMM have been adopted because of its excellent characteristics of adaptability to the proposed data. This approach allows in fact to more clearly extract the device representative states characterized by an average value and its respective variance. Table 2.8 shows comparative results regarding state extraction obtained with the two algorithms for a refrigerator and a washing machine.

Table 2.8: Comparative results of k-means and GMM clustering algorithm for extracting the power levels (mean and variance) of a refrigerator and a washing machine

Algorithm	**Mean**	**Variance**
Refrigerator		
k-means	66.09	7.42
	302.48	56597.83
	61.09	37.66
GMM	64.13	8.46
	491.01	24501.37
	30.56	921.85
Washing Machine		
k-means	6.72	205.39
	2100.47	9339.21
	167.65	6804.05
GMM	2.00	0.001
	2100.47	9339.21
	105.28	2349.91

In the refrigerator case k-means returns two mean values that are too similar (about 66 and 61) and thus result not useful at the HMM model extraction aim. Instead GMM reported more defined low consumption mean values together with acceptable variance values. In the washing machine case GMM emerges for its smaller variance as the obtained mean values for each algorithm are similar.

As introduced in Section 2.5.1, the Hidden Markov Model includes the definition of a transition matrix. Therefore, I extracted the statistical model associated with each device, or more precisely the state transition function that models the appliance power consumption behavior with its associated probability. For each type of appliance, according to the set of states generated through the GMM, I mapped the sequence of samples in the power traces into a sequence of states. All the transition events (including the self transitions) have then been detected; then their occurrences have been counted to extract the corresponding probability. Figure2.7 and 2.4 show the state machine and transition probability matrix that have been obtained for a washing machine, respectively.

Testing Results

The experimentation has been composed of two phases. Firstly, each context conditioning has been singularly applied to the algorithm and the obtained results have been compared to those obtained in the work by [52]. Secondly, disaggregation results have been evaluated considering both context information items. Each test has been performed by providing the system with the full-knowledge regarding each appliance that could compose the aggregated consumption trace (table 2.7), i.e. including even those turned off. As mentioned above the aggregate consumption trace has been composed synthetically by summing the daily traces of each single appliance. In order to create the test set, I combined each daily trace of a given appliance for a given day with all the daily traces of the other appliances. First, I describe how each single context-based conditioning approach operates.

1. **Usage statistic conditioning** In figure 2.10 an aggregate power consumption trace is shown; as it can be noticed in this temporal portion, a PC-Desktop is always ON just like the Refrigerator, a LCD-TV is turned ON a little after 8:00 am and left ON until the end of the examined temporal portion. Moreover a Coffee Maker is used in other two daily moments.

   Figure 2.11 graphically shows the results obtained by applying disaggregation algorithm by [52].

   Figure 2.12 shows the graphical disaggregation results obtained by applying our NILM algorithm with *Usage statistics conditioning (USC).*

   An improvement can be observed; this is plausible, especially for devices that are typically switched ON and OFF in a portion of

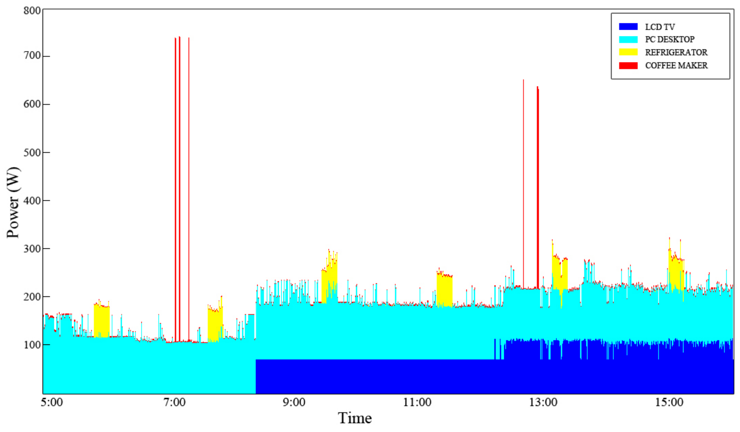


Figure 2.10: Aggregate power consumption trace - test case 1

a specific time such as the coffee maker. As expected a typical "Always ON" device such as the Refrigerator does not benefit of the effects of this type of conditioning. Table 2.9 shows the

Table 2.9: Precision results obtained with [52]'s AFAMAP algorithm in basic and with *Usage Statistics Conditioning* version

	AFAMAP [52]	**USC**
Refrigerator	27.88%	30.77%
LCD-TV	99.28%	100.00%
PC-Desktop	50.99%	74.07%
Coffee Maker	36.14%	77.21%

precision obtained with the timing usage statistics conditioning compared with those obtained by using the AFAMAP algorithm [52].

2. **User Presence conditioning**

   The second conditioning is analyzed below. Figure 2.13 shows the real aggregate consumption trace of a Washing Machine with the same LCD-TV trace that has been analyzed above; the graphic shows a portion of washing cycle with a high consumption phase (corresponding to the water heating phase) in the middle. In this case the LCD-TV disaggregation is a little worse (demonstrating

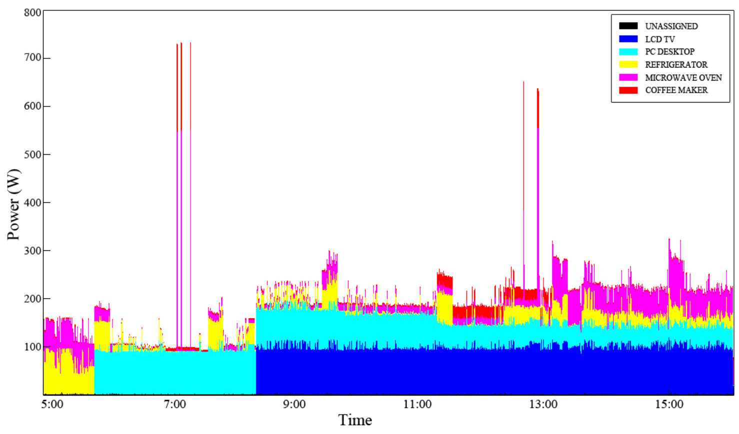


Figure 2.11: Power consumption trace disaggregated with the basic [52]'s AFAMAP algorithm - test case 1

that depending on the device traces combination, disaggregation precision can change) and this kind of conditioning, both in the single (fig. 2.15) and double (fig. 2.16) interval version, does not introduce relevant improvements with respect to the algorithm by [52]. This is due to the fact that in this case the TV usage lasts for a very long period, probably longer than the user presence observation interval. Figure 2.15 and 2.16) shows an improvement in the Washing Machine disaggregation, obtained through the *Single interval conditioning* and the *Double interval conditioning*, respectively. The washing phase, which is characterized by low power consumption, is difficultly distinguishable; the basic algorithm, in fact, confuse it for a PC-Desktop execution (fig. 2.14). Although even with the *Single Interval conditioning* few errors are encountered, a portion of washing machine consumption is well-assigned (fig. 2.15). The graphical results are confirmed by the precision percentage shown in table 2.10 .

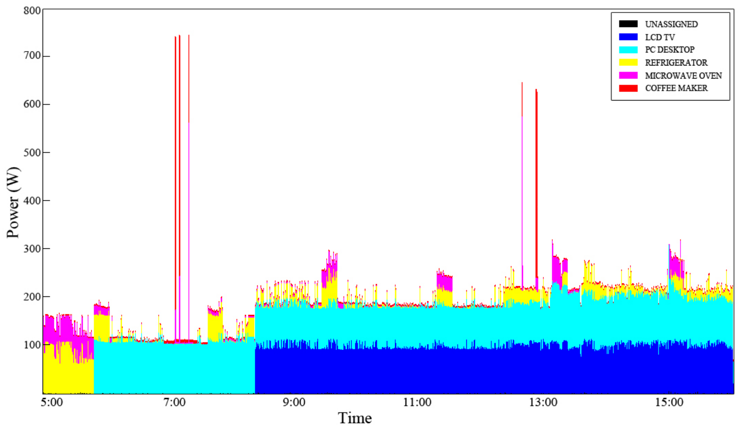


Figure 2.12: Power consumption trace disaggregated with the *Usage statistics conditioning (USC)* - test case 1

Table 2.10: Precision percentages comparison between [52]'s AFAMAP algorithm and the User Presence single and double Interval Conditioning

	AFAMAP [52]	**UP single IC**	**UP double IC**
Refrigerator	70.73%	78.50%	94.78%
LCD-TV	93.86%	90.54%	94.04%

Discussion

Table 2.11 compares average results of 4 tests using the basic Kolter and Jaakkola's algorithm [52], the *User Presence single interval conditioning*, the *User Presence double interval conditioning*, the *Usage Statistics conditioning* and a combination of the last two conditioning mechanisms executed together. In almost all cases a disaggregation precision average improvement is observed with respect to the basic algorithm. Even if the combination of the usage statistics conditioning with the double interval conditioning is better in most cases, percentually the most effective is the *User Presence double interval conditioning*. As regards the *Recall* parameter, the average results are a little worse than the *precision* ones. This is caused by the nature of this parameter that, by definition, also considers the wrongly assigned or unassigned samples of a given device. However the *recall* improvement over the basic al-

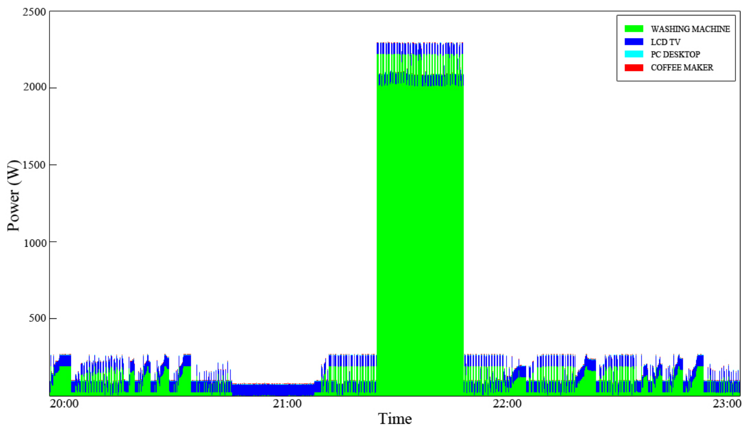


Figure 2.13: Aggregate power consumption trace - test case 2

gorithm tightly depends on the analyzed test case, as, for example, a greater quantity of not assigned power samples can worsen this value.

Table 2.11: Average *precision* / *recall* percentage results comparison among each tested algorithm for 4 test cases

	AFAMAP [52]	UP IC	UP double IC	USC	2UP IC + USC
Test 1	**44,72** / 73,88	**61,23** / 81,11	**61,44** / 74,12	**48,68** / 72,98	**61,36** / 73,84
Test 2	**49,63** / 80,49	**43,06** / 69,39	**57,48** / 70,46	**57,40** / 73,45	**58,18** / 70,68
Test 3	**44,37** / 81,73	**54,83** / 86,97	**69,11** / 71,59	**54,18** / 80,74	**68,94** / 71,43
Test 4	**50,76** / 59,51	**41,52** / 70,74	**53,27** / 55,26	**53,43** / 55,01	**52,92** / 54,38

The experimentation campaign, carried out with the complete test set over the basic algorithm for each conditioning, has highlighted the average improvements that have been reported in table 2.12.

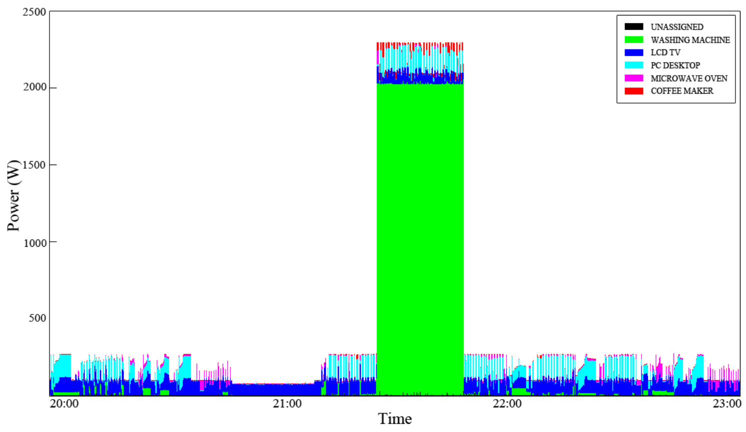


Figure 2.14: Power consumption disaggregation obtained with [52]'s AFAMAP algorithm - test case 2

Table 2.12: Average *Precision* and *F-Measure* improvements over the basic AFAMAP algorithm [52]

	Context-based conditionings	**Precision**	**F-Measure**
1	User Presence single interval conditioning	$\approx 3\%$	$\approx 2\%$
2	User Presence double interval conditioning	$\approx 12\%$	$\approx 14\%$
3	Usage Statistics conditioning	$\approx 6\%$	$\approx 3\%$
4	The combination of 2) and 3)	$\approx 13\%$	$\approx 14\%$

As it can be observed, even though the *recall* parameter apparently worsened the disaggregation results at test level, the *F-Measure* evaluation parameter, through the whole test set, reports a significant improvement as well as the *precision* parameter.

2.6 Conclusions and Further Improvements

In this work two different Load Monitoring approaches have been presented.

The first project has regarded a Neural Network based system for

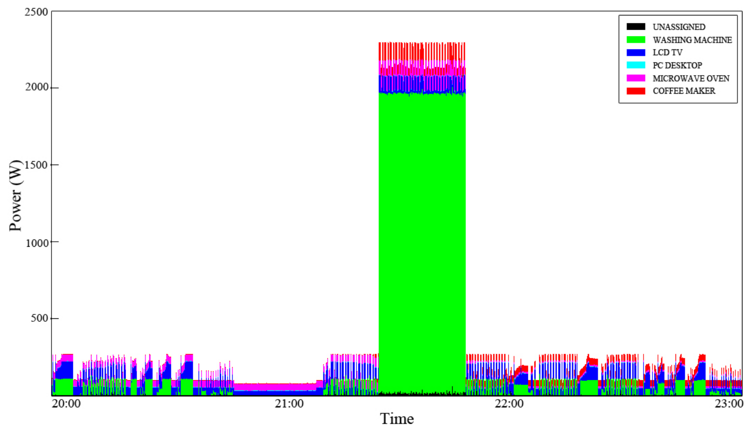


Figure 2.15: Power consumption disaggregation obtained with the *UP single Interval Conditioning* - test case 2

appliance classification. With respect to the state of the art, our contribution is focused on 1) the analysis of low frequency metering data (i.e. 1 active power sample every 2 minutes) and 2) the implementation and testing of the implemented algorithm in a home gateway. Adopting low frequency data has the major advantages of allowing the exploitation of cheap smart plugs and saving resources for storing, processing and transmitting data. The latter is even more appealing if we consider to run the algorithm in the home gateway instead of a central server. The algorithm was tested using data coming from real households with encouraging results. I experimented good classification accuracy for the ANN trained with examples coming from similar devices (i.e. from same device, producer, model). From the user perspective, this approach can cause a significant discomfort due to the mandatory manual labeling of household appliances. To overcome this problem can be useful a remote repository able to store devices load profiles for future uses. In this way, a knowledge base is incrementally built and maintained thanks to users contributions. A possible use of the knowledge base is at configuration time: when a new system is deployed in a house, the knowledge base is queried to find the models of appliances present in the house. Therefore, the ANN training can be personalized selecting the best power trace examples for that case. In the long run, when the knowledge base contains

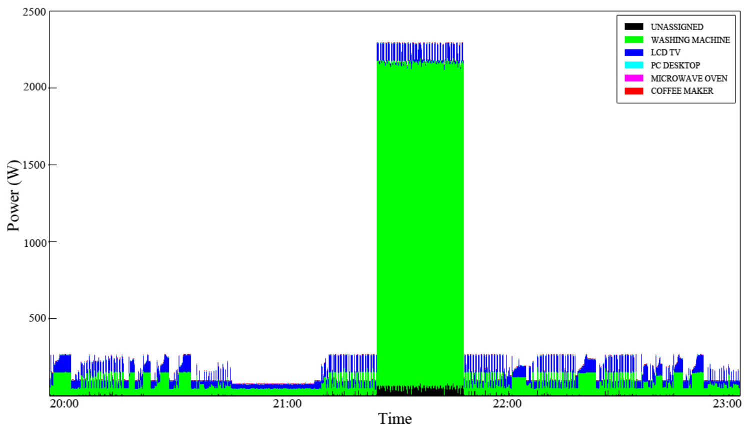


Figure 2.16: Power consumption disaggregation obtained with the *UP double Interval Conditioning*

a significant number of different traces, this solution would require only a minimum configuration effort for the user (who should list just the brand and model of the appliances to be monitored). In the future, it would be useful to evaluate the use of different neural models characterized by dynamic retraining mechanisms in order to improve the balance between efficiency and complexity [86].

In the second work I proposed a new energy disaggregation algorithm that takes into account context-related information, that can be gathered from low-cost sensors, and statistical analysis of energy consumption data. With respect to most existing works, which are based on the analysis of data collected at a high sampling frequency [33,42,43], our contribution consisted in investigating a disaggregation approach on energy monitoring data collected at low frequency. This choice has the following advantages: it is possible to use low-cost and widely available smart meters; data storage and transfer tasks are less resource demanding. Context features (e.g. user presence and device usage consumption patterns) have been exploited to improve the statistical model of each appliance. Results of testing activities and their comparison with a state of the art solution are encouraging. In the future, it would be useful to extend the proposed approach to include the use of additional context information (e.g. profile of users, weather information, etc.) in order to improve the disaggregation algorithm as well as to enhance the proposed approach with optimization algorithms and suggestion mechanisms to help consumers in saving energy costs. Moreover, tests described in this work are based on the use of data available from a publicly accessible dataset, i.e. Tracebase [20]. I believe that the adoption of open data sets in this field may speed up research and innovation processes by favoring repeatable research and easing the comparison of different approaches.

Given the energy saving as a common purpose of these two works, it can also be envisage that some future developments may be related. Indeed, as mentioned above, a user can choose between these two approaches according to its needs and not only focusing to the weaknesses and strengths of these algorithms. For instance a user can be more suited to the ILM approach as he would plan to upgrade his house; this kind of change could favor a massive sensor deployment to obtain more targeted information. On the contrary, a user who does not foresee a remarkable change in his living environment could choose the NILM approach in order to obtain energy consumption information without an intrusive installation phase. Furthermore, in a future version of a Smart Home not necessarily these approaches have to be mutually exclusive, therefore, I expect that ILM and NILM, together with Smart Appliances (introduced in Section 2.1) could collaborate to fill the shortcomings of each approach. In a real life scenario it is plausible to assume that a home can be equipped with few smart appliances and several smart plugs to gather accessible devices' consumptions. In this context, in a possible further work, a Home Energy Management System can firstly record energy consumption contributions coming from smart plugs or smart appliances and then provide the NILM system with this information in order to improve its knowledge and to help the disaggregation phase by directly remove, from the whole-house power consumption trace, the already recognized contributions.

Chapter 3

Multi-user VNF Placement optimization in a SDN-based multi-stakeholder NFV architecture

Recently, communication services are becoming more diverse and dynamic with an increasing number of users who are connecting to communication networks. Network Function Virtualization (NFV) is an effective technique to deal with this situation; it enables the operator to change configurations of network functions dynamically. In the research field of NFV, many researchers have tackled the VNF placement problem as a facility location problem which decides the placements of virtual network functions (VNF) according to a specific objective. This work addresses the issue of optimal VNF placement in a multi-stakeholder network infrastructure by considering the framework of a NFV Management and Orchestration architecture that leverages the Software Defined Networking paradigm. Given a set of service requests and considering a set of constraints (e.g., maximum end-to-end delay, monetary cost, allowed server utilization level), a mathematical model has been formulated to maximize the profit that can be obtained by both tenants (i.e.

infrastructure providers, cloud providers) and renters (i.e. service providers/users). In order to favor generalization and to ease the treatment of parameters that in literature have not accounted (e.g. multiple users), the choice of the actual forwarding path of incoming traffic flows is deferred to a later step (optimal routing), to be performed by the SDN Controller. Moreover, the work provides a detailed formalization of service requests and Data Centers and considers two types of users with different privileges (i.e. Premium and Best Effort). The energy efficiency and sustainability goals have been also taken into account.

3.1 Introduction

The last few years have witnessed a relevant increase in the number of users who ask for connection to communication networks. Indeed, the increasingly frequent releases of more and more computationally fast and capacious devices (e.g. smartphone, tablet, etc.) without a significant price growth, have favored their diffusion among users belonging to different social abstractions and age groups. Furthermore, this has caused an increase and a differentiation in terms of service requests which have become dynamic and user-centric. This trend is also reinforced by the future vision of Internet of Things (IoT) which will contribute to a new generation of service requests. In this scenario, service providers have to fulfill user needs with the dynamic provision of heterogeneous services across multiple infrastructures and platforms while aiming at a satisfactory operating margin. However, to achieve this evolution, additional capabilities and features need to be integrated in their current basic cloud infrastructures and services (i.e. memory, compute, storage). Network Function Virtualization (NFV) [87], [88], and Software Defined Networking (SDN) [6] are emerging technology paradigms that aim at enabling flexible and dynamic network control and management approaches. Traditionally an operator network consists of Network Functions (NF) (e.g., Deep Packet Inspection, Network Address Translator, Load Balancer, Firewall, etc.) which are implemented on physical middle-boxes with the aim of processing the network traffic for some purposes (e.g. energy performance, security, availability). The

NFV concept consists in the decoupling of network functions from the hardware they run on, by leveraging the virtualization abstraction [87]. As a consequence, the adoption of virtualization technologies brings the advantage of enabling the on-demand and flexible provision of network services, while also providing mechanisms for containing Capital Expenditure (CAPEX) and Operational Expenditure (OPEX). CAPEX can be reduced as network operators are no longer dependent from specialized hardware; on the contrary the reduction of OPEX can be achieved by involving automatic orchestration for deployment and maintenance of Virtual Network Functions (VNFs) instead of specially trained personnel. This kind of technology also helps to support the energy efficiency policies. To accomplish a network service, traffic flows typically traverse several network functions in a specific order; we refer to this concept as Network Function Chaining or Service Function Chaining [89]. As a result, with the adoption of the VNF paradigm, a network service can be specified as an ordered chain of VNFs which are deployed as software running on virtualization platforms (e.g., Virtual Machines or containers) on physical servers in Data Centers distributed in different network locations. The Software Defined Networking concept works complementarily to NFV as it separates the control plane from the data plane, in order to use higher level entities (i.e. SDN Controllers, Orchestrators) to control/manage the network. Hence, SDN offers the capability to programmatically enforce traffic forwarding rules across network nodes, thus steering traffic through the dynamically established sequence of network service endpoints, thereby effectively supporting the dynamic provision of a network service chain [90]. The European Telecommunication Standard Institute (ETSI) has begun the NFV standardization process in 2012 [87]. ETSI specifies the NFV architecture as a three-layer structure composed by the VNFs, the NFV Infrastructure (NFVI) and the MANagement and Orchestration (MANO) [8]. A technical report recently published by ETSI extends the MANO specification by discussing the options for integrating SDN in the NFV Architectural Framework [91]. By taking into account these specifications, I consider a reference architecture whose main components are:

- a **VNF Orchestrator** which is in charge of managing the provision and lifecycle of network services, by orchestrating virtual resources, potentially across different organization domains. This includes also the task of placing VNF instances in the cloud infrastructure, and governing the dynamic allocation of resources to these instances according to VNF service requests workloads and energy efficiency targets (VNF Placement) [92]

- an **SDN controller**, which is in charge of dynamically defining the optimal network forwarding path for incoming traffic flows that have to traverse the chained VNFs.

By leveraging such definition of functional roles for this architectural framework, this work provides a contribution in the field of VNF Placement. The Placement of Virtual Network Function (VNF) addresses the issue of choosing the set of optimal locations for chained VNF instances according to the current characteristics of available computing resources (nodes) and network links. State-of-the-art literature reports many works which differ in several aspects (i.e. conceptual approach, objective functions and other considered features). In [93–95], the placement and routing optimization are considered as joint problems. In particular the work in [95] aims at minimizing link utilization and allocated computational resources, while [96] performs the minimization of E2E delay and bandwidth consumption deploying chained VNFs in a multi-domain network environment. Single-domain focused works are instead [97,98]. However, they consider a more complex problem involving orchestration-level tasks, such as respectively the evaluation of network operational costs and horizontal scaling of VNFs. Moreover [96–98] provide detailed formalization of service requests and mathematical models aiming, among other objectives, to promote energy efficiency.

In this thesis, aiming to model a realistic scenario linked to the market *law of supply and demand* regarding network Service Provisioning, both *tenants* (i.e. infrastructure providers, cloud providers) and *renters* (i.e. service providers/users) have been taken into account. Tenants will aim to increase service provision to obtain higher profits, while renters will aim to receive the best possible service quality at the lowest possible price. These goals have been jointly and hierarchically accounted with the definition of a mathematical model that aims at maximizing the overall obtained profit in terms of accomplished service requests (for tenants) and VNF preferences satisfaction as regards DCs to be deployed on (for renters). In the optimization model, users generating service requests have been differentiated into privileged and standard users (i.e. Premium and Best Effort) while service requests have been formalized and characterized by several parameters (i.e. latency, bandwidth, required resources, etc.). The implemented model chooses the set of optimal locations for chained VNF instances according to the current characteristics of available computing resources (nodes) and network links, as well as several constraints that, among other goals, aim at favoring energy efficiency and reducing CO_2 emissions. Furthermore, in order to favor generalization and to address issues that are not discussed in the literature, the choice of the actual forwarding path of incoming traffic flows is likely deferred to a later step (optimal routing), to be performed by the SDN Controller; this assumption also eases the optimization phase and encourage tasks separation among management layers (i.e. Control, Orchestration).

This work is structured as follows. In Section 3.2 I provide a focused background in which the main involved technologies are introduced. In Section 3.3 I discuss the related work in detail and motivate this work contribution. Section 3.4 provides a detailed problem formulation, mathematical model and design assumptions; moreover, it describes the proposed VNF Placement algorithm and its implementation. In Section 3.6 I firstly describe the realistic evaluation scenario and secondly present the testing activities carried out to evaluate the performance of the implemented algorithm with related result discussion. Section 3.7 concludes this work with final considerations and provides further improvements.

3.2 Background

In recent years, significant interest has been shown in Network Virtualization, thus encouraging the emergence and the development of new technologies such as Cloud Computing [99], Software Defined Networking (SDN) [6], Network Function Virtualization (NFV) [87], [88] etc. The new paradigm proposed by Cloud Computing is coming out for on-demand provisioning of IT resources as well as infrastructure externalization, favoring the increasing exigences of *tenants* and the needful cooperation of *cloud providers*. User needs are thus growing beyond the simple provisioning of virtual machines, to the requirement of flexible and complex virtual resources and services. Emerging complementary technologies such as SDN and NFV could encourage the transition to a revolutionary new way in which IT services are delivered and managed.

3.2.1 Software Defined Networking

Software Defined Networking (SDN) [5] [6] is an emerging network architecture where control plane is decoupled from data plane and directly programmable. As originally defined, *"SDN refers to a network architecture where the forwarding state in the data plane is managed by a remotely controlled plane decoupled from the former."* [100]. In the traditional network view, the control and data planes are coupled and, for this reason, difficult to modify. These planes are physically embedded in network elements, hence adding new functionalities results difficult and expensive as it would imply both software and hardware upgrades of all network devices. Middleboxes, (e.g. load balancer, intrusion detection system, firewall, etc.) represent the specialized hardware needed to upgrade the network with new features; hence they are expensive, hard to configure and, once positioned in the network, difficult to modify [101].

SDN, with its programmable control plane, seems to be a suited solution to overcome the innovation difficulties typical of the traditional network.

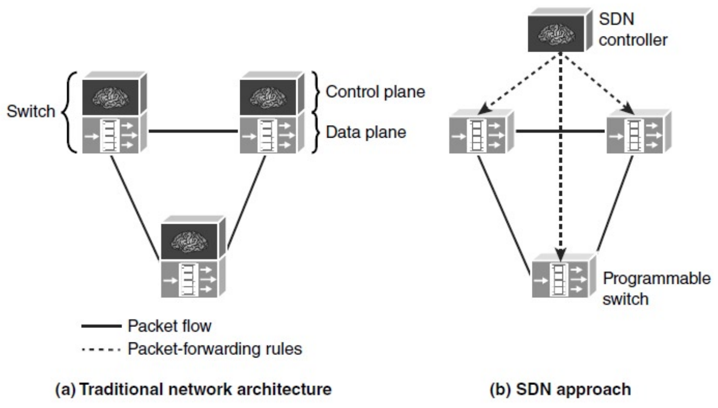


Figure 3.1: Traditional Networking versus Software Defined Networking [102]

Figure 3.1 compares, respectively, the traditional networking (a) and the SDN (b); the concept of coupled/decoupled planes results evident. Indeed, in Figure (a) the networking element (i.e. Switch) is composed by the Control Plane and the Data Plane; differently in Figure (b) the Control plane separation is evident through the presence of the SDN Controller. The Programmable Switches in this approach are simply responsible for packets forwarding while the SDN Controller designs routes, sets priority and routing policy parameters to respect QoS and to cope with the shifting traffic patterns [102]. Although in recent years the original definition of SDN provided above has undergone various changes, a schematic description given by Kreutz *et al.* [101] is summarized below. SDN can be defined by accounting these four main features:

- **Separation**: The Control Plane is separated from the data plane. Network devices become simple forwarding elements without any control functionality.

- **Flow-orientation**: Forwarding decisions are flow based, instead of destination based. In the context of SDN, flows can be defined as a sequence of packets between a source and a destination where all packets of a flow receive identical service policies at the forwarding devices. This abstraction gives flexibility by unifying the behavior of different types of network devices (i.e. routers, switches, firewalls, middleboxes).

- **External Control**: The so-called *SDN controller* is hosted by an external entity and, having an abstract network view, acts as a coordinator by programming the forwarding devices.

- **Programmability**: The applications, running on top of the SDN controller, program the network through the intermediation of the SDN controller.

The **Controller** in the SDN architecture represents a relevant aspect as it manages the centralized state of the network, or, at least, a portion of it. This last statement refers to the future vision of the Internet in which the control plane would be composed by a network of SDN controllers intercommunicating to maintain the full knowledge of the whole network state. This external control (logical centralization) of the network can produce several significant benefits such as:

- **Ease of modification**: network policies can be easily modified through high-level languages and software components.
- **Automation**: network state changes can be automatically detected and monitored in order to maintain unaltered service quality and policies.
- **Ease of development**: The global knowledge of the SDN controller over the network, simplifies the development of more sophisticated networking functions, services, and applications [101].

In the last few years several SDN controllers have been implemented and made available; although each of them may differ in some characteristics (e.g. programming languages, etc.), their aim is common. Among others, the most popular SDN Controllers are: Floodlight [103], POX [104], Ryu [105], OpenDayLight [106] and ONOS [107]; they all provide low-level control over switch flow tables and are typically object-oriented. Some of them such as OpenDayLight, Ryu and ONOS provide also integration capabilities with Cloud Management System, e.g. Openstack [108].

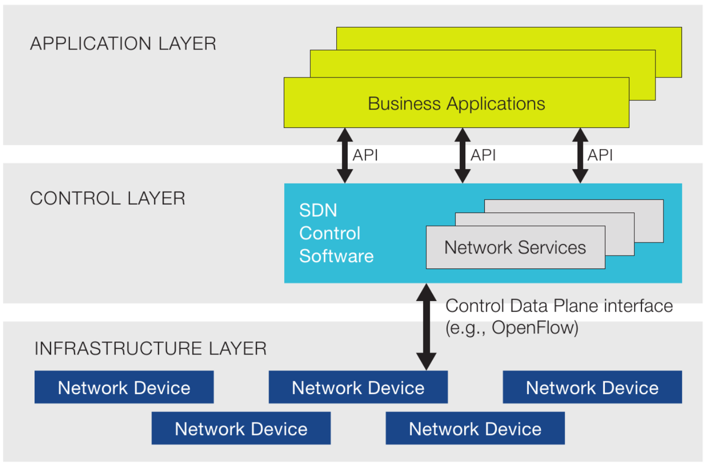


Figure 3.2: Software Defined Network architecture [109]

OpenFlow [110], developed by the Open Network Foundation (ONF) [111], is the communication protocol defined between the control and forwarding layers of an SDN architecture; it is considered the de-facto standard for SDN. Switches and routers, the network devices in the data plane, can be directly accessed and manipulated through Openflow. Figure 3.2 shows the three SDN *stacks* (i.e. Application Layer, Control Layer and Infrastructure Layer) with the architecture APIs that define the communication between layers. The APIs that provide the connection between applications and the controller are often referred to as Northbound Interface while the Southbound Interface represents the connection between the controller and the networking hardware. Figure 3.2 shows the Northbound Interface has not a reference standard protocol; on the contrary the Southbound has Openflow as reference protocol.

A Software Defined Network can also be defined by the three abstractions detailed below [110]:

- **Forwarding**: this abstraction allows the application over the SDN controller that requests a specific traffic forwarding, to get any desired forwarding behavior without even know hardware details.

- **Distribution**: the abstraction is realized with the centralized logical control, which represents a common distribution layer (SDN controller) responsible of installing the control commands on the forwarding devices and of collecting status information about the forwarding layer. This view let applications having a global network knowledge without facing the problem of distributed states.

- **Specification**: toward a future vision of service provisioning, and for the scope of this thesis, this abstraction has a key role. Indeed, with this abstraction, a network application is able to express the desired network behavior without responsibilities on the implementation phase. This can be achieved through virtualization solutions, as well as network programming languages [5].

Since a detailed SDN description is out of the scope of this thesis, I refer the interested reader to the comprehensive survey provided from Kreutz *et al.* [101].

3.2.2 Network Function Virtualization

Network Function Virtualization (NFV) is an emerging approach to network service provisioning that mainly involves the implementation of Network Functions (NF) in an open and standardized IT virtualization environment. Indeed, in the traditional network system, network functions are composed by a combination of software and vendor-specific hardware, while NFV proposes to decouple the software (i.e. Network Functions) from the hardware and, consequently, to remove vendor-dependency. The Network Functions (e.g. routers, firewalls, load balancers, NAT, DNS, etc.) are therefore decoupled from their dedicated hardware, virtualized as Virtual Network Functions (VNF) and implemented as software components on fully-virtualized network infrastructures. Mijumbi *et al.* in [7] state *"This allows for the consolidation of many network equipment types onto high volume servers, switches and storage, which could be located in data centers, distributed network nodes and at end user premises"*. This approach could favor, on the one hand, innovation and optimization for network service provisioning and, on the other hand, the reduction of Capital Expenditure (CAPEX) and Operational Expenditure (OPEX) for service providers and network operators. Given that VNFs are implemented as software, CAPEX could be reduced because the Service Providers are no longer burdened by dedicated hardware devices cost. On the contrary, personnel costs reduction contributes to OPEX decrease due to networking automation and simplification produced by NFV. Therefore NFV applies to Network services a new design, deployment and management approach.

The conceptual difference between traditional and virtualized appliances is shown in Figure 3.3; the network service provisioning in NFV is characterized by several differences in comparison to current practice. According to [7] these differences are the followings:

1. **Decoupling software from hardware:** the benefits obtained by the separation between NF software and hardware derives from the independent evolution and management of each other.

2. **Flexible network function deployment:** infrastructure resources could be reassigned and shared among network services.

3. **Dynamic scaling:** the possibility to dynamically instantiate soft-

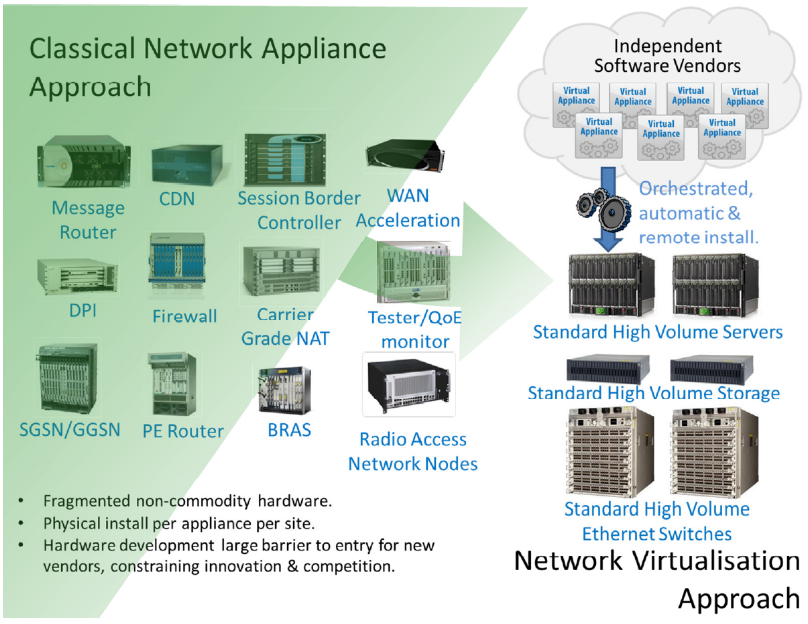

Classical Network Appliance Approach
Message Router
CDN
Session Border Controller
WAN Acceleration
DPI
Firewall
Carrier Grade NAT
Tester/QoE monitor
SGSN/GGSN
PE Router
BRAS
Radio Access Network Nodes
• Fragmented non-commodity hardware.
• Physical install per appliance per site.
• Hardware development large barrier to entry for new vendors, constraining innovation & competition.
Independent Software Vendors
Virtual Appliance
Orchestrated, automatic & remote install.
Standard High Volume Servers
Standard High Volume Storage
Standard High Volume Ethernet Switches
Network Virtualisation Approach

Figure 3.3: Classical Network appliance approach versus Network Virtualization approach [87]

ware components, according to upcoming events or user/application needs, provides the scalability feature.

Thus in NFV, Network Functions are virtualized into Virtual Network Functions (VNFs) that provide the equivalent functional behavior and interfaces. The European Telecommunication Standard Institute (ETSI) has begun the NFV standardization process in 2012 with initially seven operators (AT&T, BT, Deutsche Telekom, Orange, Telecom Italia, Telefonica and Verizon) which, over time, have grown in number. The Industry Specification Group for NFV (ETSI ISG NFV) [112] is responsible to develop the required standards for NFV [7]. After few years (2015) ETSI has published 11 Group Specifications [113] which, among other provided information, specify the NFV architecture as a three-layer structure composed by the VNFs, the NFV Infrastructure (NFVI) and the MANagement and Orchestration (MANO) [8]. ETSI

NFV architecture is shown in Figure 3.4.

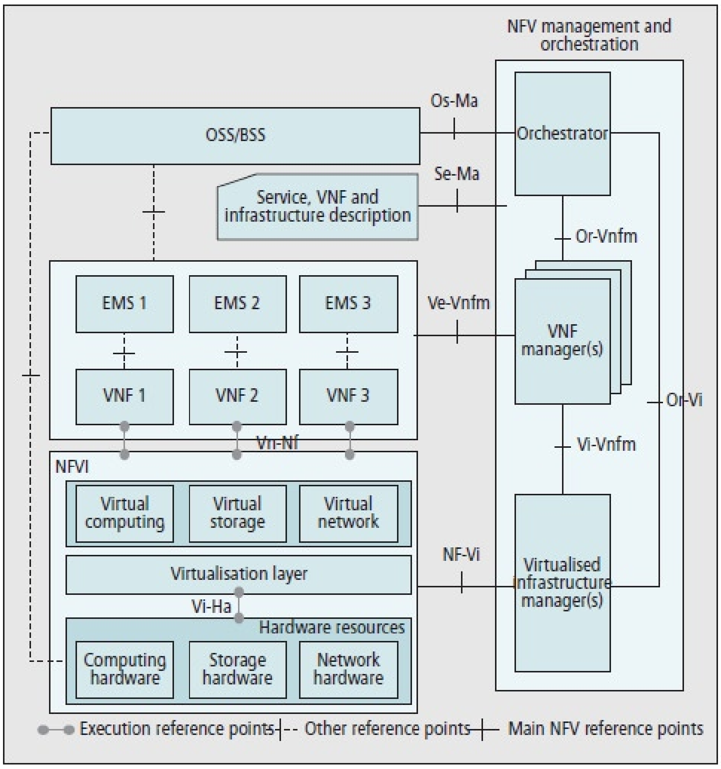


Figure 3.4: The ETSI NFV three-layer infrastructure [114]

The **NFVI** is mainly composed by virtual and physical resources with the virtualization layer between them aimed at providing an abstraction layer. The virtual resources are thus abstractions of the physical resources (Computing, Storage and Network hardware) which provide processing, storage and connectivity to VNFs.

Virtual Network Functions, key elements of Network Function Virtualization, are instantiated in Virtual Machines or containers (e.g. Docker) into servers of a Data Center. They reflect both the external interfaces and functional behavior of the associated Network Functions. In other words VNFs represent the virtualized version of NFs, but, as virtual, they can be deployed, scaled, migrated where it is more convenient.

Management and Orchestration (MANO) in Figure 3.4 is the vertical block on the right composed by the Orchestrator, the VNF Manager and the Virtualized Infrastructure Manager (e.g. Openstack); it focuses on all virtualization-specific management tasks necessary in the NFV framework [7]. For instance MANO is responsible of the VNFs lifecycle as well as the functionality required for their provisioning, configuration and maintenance.

To deepen the NFV topic, which in this thesis work is only synthesized to produce a comprehensive background, these two State-of-the-art papers [7, 114] are suggested. Indeed, the first one provides a detailed survey and identifies promising research directions on this topic, while the second one, after a focused introduction on topic, discusses challenges and requirements of using NFV in mobile networks.

3.2.3 Service Function Chaining

A Network Service, provided by an operator, is a complete end-to-end functionality delivered making use of one or more service functions in a specific order; each of them is respectively responsible for specific treatment of received packets. The Virtual Network Functions that have been introduced above (Section 3.2.2), represent the virtual realization of service functions. To accomplish the network service, traffic flows typically traverse several VNFs that could be located in different network nodes; this concept is referred as Service Function Chaining or Network Service Chaining [89] and could be realized by both SDN and NFV. Although these technologies have been independently developed and can be implemented individually, their high complementarity induces their joint use. Indeed, both SDN and NFV provide resource flexibility, respectively working at higher levels (L4-L7) and lower levels (L2-L4), aiming at exploiting automation and virtualization and at avoiding vendor-addictions by making use of open software and standard network hardware.

Thus, service provider networks may achieve value improvement by the logical combination of SDN and NFV as these concepts are complementarily beneficial and can ease each other implementation and deployment. Cui et al. in [115] provide an interesting and clear example, summarized in the following sentences, that highlights how SDN and

NFV can benefit from each other. An SDN controller, running for instance on a Virtual Machine, may be implemented as part of a service chain. SDN can thus benefit from NFV's reliability and elasticity features by realizing its centralized control and management applications (i.e. load balancing, monitoring and traffic analysis) as VNFs. In the same way NFV deployment can be accelerated through SDN, able to offer several advantages such as a flexible and automated way of chaining functions, provisioning and configuration of network connectivity, automation of operations, security and policy control, etc. [7, 115].

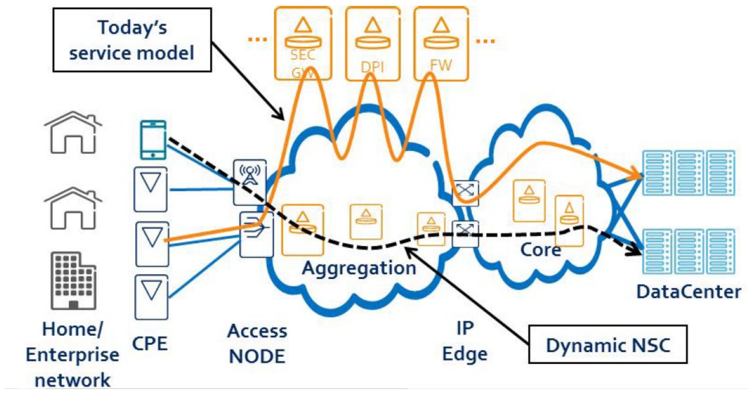


Figure 3.5: Traditional service composition models versus the dynamic NSC (also referred as SFC) [89]

Figure 3.5 depicts the conceptual difference between the traditional service creation models and the dynamic Service Function Chaining (or NSC) emphasizing the benefits accomplished by SFC principles. In the picture the orange bold line follows a predefined order of monolithic service elements instead of the dashed black line which passes physical and virtual service functions embedded into different network domains [89]. The main components of NFV and SDN i.e. respectively the Orchestrator and the Controller, have an alike fundamental task in the provision of an optimized dynamic Service Function Chain. The *NFV Orchestrator* is responsible for the VNFs that compose the chain; firstly it will have to assess whether the network nodes (e.g. Data Centers geographically located in different sites) are already provided with these specific virtual functions, and successively, if necessary, it will have to deploy the missing ones. The decision of where to place the Virtual Functions has to be taken according to several constraints and optimization policies. This topic, referred as *VNF Placement*, is accurately deepen in Sections 3.3 and 3.4 as it represents the second main focus of this thesis work. The *SDN Controller*, instead, is the network management element responsible for the traffic flows that traverse the network paths. Once the VNFs have been placed in the network nodes, the SDN controller instructs forwarding devices (Section 3.2.1) on how to route traffic across the Virtual Functions to respect both their order and the various constraints (e.g. maximum end-to-end delay) that have been specifically requested to correctly provide the service.

In December 2015 a technical report published by ETSI extends the MANO [8] specification by discussing the options for integrating SDN in the NFV Architectural Framework together with some insights for Service Function Chaining and Proof of Concepts [91], in the same way the Internet Engineering Task Force (IETF) has created the Service Function Chaining Working Group (IETF SFC WG) [116] with the aim to produce an SFC architecture. The SFC architectural model has thus to be flexible enough to satisfy the dynamic user requests and service policies. Furthermore, allocation of resources and placement of virtual functions have to be done in a dynamic and automatic way according to the specific service request. However automation must be governed by optimization strategies in order to choose optimal parameter values such as cost, end-to-end latency, network bandwidth, overall resources required etc. [117].

3.3 Related Work

The placement of VNFs has recently become a popular research topic as a result of the increasing interest in the emerging technologies that have been introduced in Section 3.2. As mentioned above a number of standardization activities in the NFV/SDN/SFC area have been carried out by ETSI, resulting in the technical reports that have been referred respectively in Section 3.2.2, 3.2.1, 3.2.3. Several draft documents like *Problem statements in NFV* [118,119], *Use Cases* [120] and *Frameworks for network function chaining* [121] have also been released by Internet Engineering Task Force (IETF) in order to provide researchers with common guidelines. The reference architectural concept as well as the VNF placement model presented in this work have been implemented by taking into account these specifications with the aim of proposing a compatible extension. The Placement of chained virtual functions is a complex problem that involves all the technologies that have been introduced above. For instance when a service request is received, the NFV Orchestrator has to check if the VNFs (needed to accomplish the service) are already instantiated and exploitable into the network nodes (i.e. Data Centers), otherwise it must arrange for their deployment. Afterwards the SDN Controller can manage the network by programming the forwarding devices to route the traffic through the ordered chain of VNFs. Each aspect of this high-level example generates several issues to be explored; state-of-the-art reports a lot of works that have provided a contribution on these aspects. Xin Li and Chen Qian [122] provide a survey on some existing VNF placement approaches. They highlighted that some works propose approaches that consider *VNF Placement and Routing optimization* (also known as VNF-PR) as joint problems [93–96, 123, 124]. In most cases, this assumption is not motivated by a reference to an architectural model and leads to the specification of complex mathematical formulations often more coherent to ideal than real scenarios. Then heuristics have to be proposed to make the problem computationally tractable. An introductory and mainly theoretical work has been proposed in 2014 by Mehraghdam et al. [96]; they provide a detailed insight of VNF chain placement problem by formalizing service requests and proposing a mathematical model that considers the requirements of individual requests as well as the overall requirements of all network applications and the combination of requests. In order to reduce the run-time of the Placement decision algorithm, a heuristic has been proposed. Mijumbi *et al.* in [93] propose an online NFV framework to orchestrate VNFs; they consider VNFs orchestration as a scheduling problem and resort to a heuristic to solve the mathematical problem. Also Xia *et al.* in [94] propose a heuristic algorithm to solve the VNF-PR problem which has been modeled as a binary integer programming problem. They focus on the minimization of the expensive optical/electrical/optical (O/E/O) conversions between the optical steering domain and the packet domain. Bouet *et al.* in [123] instead, reduce the problem complexity by addressing the placement of only a

virtual function, without therefore any chaining constraint; anyway, despite a formal definition of the problem is provided, a heuristic greedy algorithm is proposed to solve it. Even Addis *et al.* in [95] make use of a math-heuristic to solve the mathematical model, but also propose a more specific approach to the VNF-PR optimization problem by introducing compression constraints and different forwarding latency regimes. On the contrary, as a consequence to the separation between Placement and Routing, the works presented in [125–127] may be considered complementary to this one. In fact, each of these papers focused on the traffic routing through Network Functions in an SDN architecture; in particular Cao *et al.* [127] proposed an online algorithm to perform an optimal traffic steering.
The works referenced immediately below, address a more complex problem in comparison to those in [93–96] in which the VNF Placement is only a portion of the whole orchestration problem that they account. Indeed, Bari *et al.* [97] focus on the *VNF Orchestration Problem* (VNF-OP) and perform VNF Placement by optimizing network operational costs and utilization, taking into account Service Level Agreement (SLA). The model is formulated as an Integer Linear Program (ILP), implemented and tested to find optimal solutions for small scale networks but, in order to extend its evaluation to real world topologies, a heuristic to find a near-optimal solution has been used even in this case. Also Riera *et al.* [124] model the problem as an ILP by minimizing both the cost of assigning VNFs to Points of Presence (PoPs) and network parameters (i.e. end-to-end delay and overall resource link usage) and by maximizing the number of accepted requests. A similar VNF-OP work, without however taking into account operating cost and network performance, is implemented in [98]; Luizelli *et al.* propose an architecture for orchestrating VNFs in a remote Data Center by addressing several issues such as horizontal scaling of VNFs.
The behavior and functionality of the chained VNFs to be placed have been treated in [96,97,128,129] while Joseph and Stoica in [130] deepen the middleboxes modeling topic by presenting a model for describing the functionality of a small number of Network Functions, without however master some aspects that are fundamental for VNFs such as computational resource requirements.
A deep characterization of different service requests has been provided

in [128,129] while Bari *et al.* [97] use only a chain request type composed by Firewall, IDS and Proxy. Bouet *et al.* [123] perform the VNF placement of the Deep Packet Inspection (DPI) only, therefore avoiding the SFC complexity, while Addis *et al.* [95], even though they consider a multi-VNF service request, do not preserve the order of the chained VNFs.
Another aspect that differentiates the contribution from the literature consists in the possibility to deploy VNFs in single [97,98,128,131,132] or multi domains [94,96,124,129,133]; this in fact implies a totally different approach to the problem. In particular Bellavista *et al.* [132] deal with network-aware optimal VNF embedding in a single Cloud environment, but considering multiple Virtual DCs (VDCs) with multiple Virtual Machines. Energy efficiency is also taken into account in several works especially by minimizing the OPEX [97] or the number of used nodes in the network [96] by favoring the so-called *Server Consolidation* (in order to favor the usage of already working servers/DCs). Luizelli *et al.* [98] minimize the number of instantiated VNFs to encourage sharing in order to promote energy usage reduction, while Addis *et al.* [95] minimize the required resources quantity in terms of CPU cores as they assume that the greater portion of energy consumption is attributable to processors.

3.3.1 Motivation of this work

With respect to the state-of-the-art, this thesis addresses the VNF Placement problem by searching the optimal locations, in a multi-stakeholders environment (i.e. distributed Data Centers). By considering a realistic scenario driven by the market *law of supply and demand* regarding network Service Provisioning, both *tenants* (i.e. infrastructure providers, cloud providers) and *renters* (i.e. service providers/users) have been taken into account. Tenants will aim to increase service provision to obtain higher profits, while renters will aim to receive the best possible service quality at the lowest possible price. These goals have been hierarchically accounted with the definition of a mathematical model that aims to *maximize the overall obtained profit* in terms of accomplished service requests (for tenants) and VNF preferences satisfaction as regards DCs to be deployed on (for renters). The hierarchy of the objective function confers to tenants' needs a greater weight compared to renters, by according them the right to decide the market price (as in a real scenario). In the optimization model, users generating service requests have been differentiated into privileged and standard (i.e. *Premium* and *Best Effort*) to let tenants favoring those renters, so they (renters themselves) can provide a greater profit. On the contrary, the user's profit is to obtain the best possible service (each VNF of the chain is positioned in the preferred DC) depending on the price that he is willing to pay. Therefore service requests have been accurately formalized and characterized by several parameters that refers to network status (i.e. latency, bandwidth), service-specific requirements (i.e. resource required, request to avoid setup-time) or user's preferences (i.e. avoid DCs with higher CO_2 emissions). In order to contribute to the energy efficiency control, in the mathematical model a resource utilization constraint has been imposed, for instance, to turn off servers or DCs whose usage percentage does not reach a minimum threshold. Among other considerations, a Carbon footprint Server/DC characterization has been done to encourage reduction of CO2 emissions. Furthermore, in order to favor generalization and to address issues that are not discussed in the literature (e.g. multi-users, joint hierarchical objectives, service requests and network status constraints, etc.) the choice of the actual forwarding path of incoming traffic flows is likely deferred to a later step (optimal routing), to be performed by the SDN Controller; this assumption also eases the optimization phase and encourage tasks separation among management layers (i.e. Control, Orchestration). Among the state-of-the-art related works that have been evaluated, a few of them were major inspiration for this work and deserve a deeper analysis; to put in evidence the differences as well as to emphasize the novelty of this work, a schematic and detailed description of the related works in [96, 97, 124, 128] is provided in Tables 3.3.1 and 3.2.

Table 3.1: VNF Placement related works by respectively Mehraghdam *et al.* [96] and Bari *et al.* [97]

Papers / Features	Mehraghdam et al. (2014) [96]	Bari et al. (2015) [97]
VNF Placement Objectives	- Max the remaining data rate on network links - Min the number of used nodes in the network - Min the latency of the created paths	- Min OPEX considering: - cost of deployment of new VNF - energy cost for running a VNF - the cost of forwarding traffic to and from a VNF - penalty for SLO violation - Min fragmentation of the physical resource pool to increase the possibility to accommodate more traffic on the same physical resources
Request Parameters	- Chaining request (the order of the VNFs) - % of incoming data rate produced by a VNF - Start and End points set for each flow - Start and End points locations in the network - Initial data rate entering the chained functions - Max tolerated delay between Start and End points	- Ingress switch - Egress switch - Bandwidth demand - Expected propagation delay - Ordered sequence of VNF - SLO policy (to determine the penalties)
Essential Parameters	- DC resources - Switch node resources - Link Bandwidth capacity - Link Propagation delay - Number of possible instances of each VNF - Number of requests an instance of a VNF can handle	- Server resources - Link Bandwidth capacity - Link Propagation delay - Different types of VNFs with specific characterization parameters - VNF Deployment cost - VNF Processing capacity and delay - SLO violation policy
Testing Objectives and Model Symplification	- Observing the behavior of the placement process and the heuristic for reducing the runtime of the process when there are several ordering possibilities in deployment requests - Pareto analysis for showing the possible trade-offs between the metrics of interest	- Modeling a multi-stage directed graph with associated costs - Finding a near-optimal VNF placement from the multi-stage graph by running the Viterbi algorithm - Three-length middlebox sequence - Deployment cost have not been considered - SLO violation penalty is null
Environment	- Small evaluation scenario with manually created deployment requests - Machines with Intel Xeon X5650 CPU 2.67 GHz - Optimizer: Heuristic + Gurobi Optimizer to solve the MIQCP problem.	- Time varying traffic for different topologies from real and synthetic traffic traces - Each service request composed by a chain of 3 middleboxes - Machine with 10x16-Core 2.40 GHz Intel Xeon E7-8870 CPU - Optimizer: CPLEX + heuristic
Topologies	- Abilene (12 nodes, 42 links)	- Internet2 (12 nodes, 15 links) - Data Center network (23 nodes, 43 links) - ISP network - AS-3967 (79 nodes, 147 links)
Metrics	- Number of network nodes - Total remaining data rate (links) [Gb/s] - Sum of latencies (links) [s]	- OPEX - Execution Time - System Utilization (Fraction of used CPU per server) - Topological properties of middleboxes locations
Metodologies	- One grafical result for a sample request set withoptimal solution - One grafical result for a sample request set including a Pareto set	Several graphics for each of the followings evaluation tests: - Hardware vs VNF - Heuristic vs Optimal solution - Comparison with previous works - Scalability of heuristic - Effect of high traffic volume

Table 3.2: VNF Placement related works by respectively Riera *et al.* [124] and Liu *et al.* [128]

Papers / Features	Riera et al. (2016) [124]	Liu et al. (2015) [128]
VNF Placement Objectives	- Min the cost of assigning VNFs to PoPs - Min the overall delay - Min the overall resource link usage - Max of accepted Network Service requests	- Min end-to-end delay - Delay from ingress to the first mbox - Delay form the last mbox to the egress switch - Delay between adjacent mbox pairs - Min bandwidth consumption - Bandwidth consumption from ingress switch to the first mbox - Between mbox pairs - From the last to the egress
Request Parameters	Network Service Request modeled as an estention of ETSI Network Service Descriptor (NSD): - Ordered sequence of VNF - Set of paths connecting all the pairs of VNFs - Maximum allowed delay associated to each path	- Ordered sequence of middlebox (NF o VNF) - Middlebox number of the request - Ingress node - Egress node - Traffic features (e.g. packet info) - Bandwidth demand
Essential Parameters	- PoPs resources - PoPs connections between each other - Link resources - Request resource requirements - Network Infrastructure availability	- Set of switches - Switches resources - Switches delay between each other - Request resource requirements - Total bandwidth consumption of all requests
Testing Objectives and Model Symplification	- For simplicity, only one resource type is considered in the definition of the load (CPU resource) - NS allocation requests were modelled as a Poisson process of average inter-arrival time - the duration of each allocation request is assumed to follow a Gaussian distribution with mean and standard deviation	- Experiment combination of: - Testbed: to evaluate real performance of VNF Placement of chains in a small-scale network - Numeric simulation: to evaluate different placement algorithms in larger scale networks -Multiple requests with random: - sequence of middleboxes for each chain - bandwidth demand (normal distribution) - link delay (uniform distribution)
Environment	Representative scenarios based on the dataset in [134], including 210 instances of NSs and infrastructure graphs - Optimizer: GPLK (open) and CPLEX (commercial)	- ESCAPE - Mininet (to emulate network nodes and links) - Click (to implement network functions) - POX (to control packets forwarding) - iperf (to generate traffic) - ping (to evaluate packet tx and rx)
Topologies	- 20, 30, 50 and 100 PoPS referenced in [134]	- Abilene (11 nodes, 14 links) - FatTree4 (20 nodes, 32 links) - FatTree16 (320 nodes, 2432 links) - Campus (415 nodes, 531 links)
Metrics	- Acceptance rate of service requests - Computing time - Scalability of the algorithm	- Cumulative Density Function of performance of different algorithms - Average bandwidth consumption for each topology - Average end-to-end delay
Metodologies	- Comparation between CPLEX and GPLK optimization. Graphical evaluations: - Acceptance rate vs Networks size - Computing time vs Networks size - Acceptance rate vs. increasing load	- Cumulative Density Function of performance of different algorithms over optimal (end-to-end delay and bandwidth) - Average end-to-end delay and bandwidth consumption for each topology with increasing number of middleboxes in each policy and ports on a switch. With the simulation based evaluation there is a reduction of 22% of E2E delay and 38% of bandwidth consumption savings.

Table 3.3.1 and 3.2 evaluate four different VNF Placement works

by performing a comparison among the main aspects that characterize them, the notations were reported as in related papers to not lose differences. In most cases the minimization/maximization regards objective functions related to network parameters (i.e. End-to-end delay and Bandwidth consumption) as in [96, 124, 128] while Bari *et al.* [97] concentrate their approach on the minimization of an objective function that regards deployment cost and energy consumption. On the contrary in this work the objective function regards the hierarchical maximization of profits obtained with the accomplishment of service requests and the satisfaction of VNF Preferences among DCs to be deployed on. Although all the deepened papers provide a more or less detailed formalization of the service request, none of them reported a formal difference among users as it has been theorized in this work. Moreover, the objective function in this thesis, has been formulated to perform a "targeted-matching" among the profit needs of:

1. **Infrastructure providers**: they aim at accomplishing as many service requests as possible giving priority to users who are willing to pay higher prices (i.e. Premium users);

2. **Users**: they aim at achieving a satisfactory service according to the price that they are willing to pay and to its quality. The satisfaction can be expressed in terms of VNF Placement as the service is the composition of a chain of VNFs: each VNF is intended to be placed in the DC that provides the more suitable combination of resources and features to make it able to operate best (Section 3.5.1).

The service requests that have been formalized in these related works reports several common parameters e.g. Ingress node, Egress node, Ordered sequence of VNF. Riera *et al.* [124] consider a maximum tolerated propagation delay for each path (i.e. from a VNF of the chain to the subsequent) while in this thesis service requests provide the maximum tolerated E2E delay (as well as in [96]) to favor service provision flexibility. Besides, service request formalization provided in this work reports other parameters to improve user satisfaction such as the possibility to avoid Virtual Machines setup time or avoid compute nodes whose CO_2 emissions exceed a prefixed threshold. As mentioned above related works in [93–96, 123, 124] consider VNF-PR as joint problems having to resort to the use of heuristics to treat the problem. On the contrary, the formulated mathematical model proposed in this thesis has been solved without applying any heuristic algorithm; indeed, in contrast to the works that have been referenced so far, VNF placement and routing optimization have not been considered as joint problems. The optimization model chooses the set of optimal locations for VNF instances according to the current characteristics of available computing resources (nodes) and network status, but the choice of the actual forwarding path of incoming traffic flows is deferred to a later step, to be performed by the SDN Controller. Also this assumption has been made by considering a realistic scenario in which users or service providers ask for virtual resources to provide a specific requested service. Traffic flows may not have to traverse the network in the exact instant in which VNFs are deployed and, consequently, once VNFs have been placed, the SDN controller could instruct the forwarding devices when the service request have actually to be accomplished. Due to the substantial difference among considered approaches, constraints and parameters and to a lack of common guidelines to test the algorithms, a comparison among evaluation methods and results is quite difficult to set up. Indeed Mehraghdam *et al.* [96], even though they provide a detailed formalization of the problem, only discuss the differences between the optimal and the heuristic-based solution by considering their three objectives. Bari *et al.* [97] perform a deep evaluation phase by minimizing OPEX and resource usage fragmentation in order to highlight the difference between physical middleboxes and VNFs. Although these metrics are appropriate to this kind of problem, they are not suited to the approach and objectives of this thesis work. One of the model objective functions of Riera *et al.* [124] is similar to the one of this thesis; they aim to maximize the number of accepted requests and reported, as in this thesis, the acceptance rate metrics to evaluate their algorithm performance. Anyway the specification of different user types and the further users/VNFs satisfaction objective, make the approach evaluation incomparable. Works in [128, 135], instead, perform a deep evaluation campaign to test the effectiveness of the VNF Placement method that they propose. The former by aiming to minimize E2E delay and bandwidth consumption and the latter by proposing VNF-sharing in order to favor VNF-consolidation. Despite the substantial difference in approach with this thesis model, these papers report a careful analysis of realistic cases of VNF service chains from which my work took inspiration to compose the request sets. This work contributions are summarized below:

1. **Realistic scenario**: the problem has been formulated by considering a realistic scenario with both *tenants* and *renters* as actors in a market-transaction;
2. **Multi-stakeholder** (multi-commodity) network infrastructure; each Data Center exposes its features to be preferred by VNFs;
3. **Multi-user**: two types of users (i.e. Premium and Best Effort users) have been considered by accounting different privileges on service provision;
4. **Multi-request**: the system has to manage a set of service requests instead a single one. The optimal VNF Placement to provide both tenants and renters with the maximum profit, is performed for the whole request set.
5. **Multi-constraint**: the model has several constraints that impose the respect of end-to-end delay, bandwidth consumption, DC resource occupation.
6. **New parameters**: each service request is characterized specific requirements such as the avoidance of DCs that are not sustainable or that not allow to avoid the VM setup-time by providing a Container.

3.4 Problem Formulation

This work investigates if the VNF Placement problem can be addressed by considering the routing optimization as a separated further step. This assumption is supported to the separated architectural layers that deal with the two phases. In fact the VNF Orchestrator performs the VNF placement by managing each instance lifecycle while the routing phase is operated by the SDN controller which chooses the best path among VNF installed in different compute nodes, according to a routing algorithm. This approach allows formulating a VNF Placement optimization with a reduced problem treatment complexity and an openness to the introduction of original new specifications that favor a greater adherence to a real case. Although the total separation of these two steps may result in sub-optimal solutions and in a generalization loss, the proposed approach encounters these potential issues by considering the network parameters as constraints. Hence the approach proposed in this work performs the VNF Placement by considering Placement and Routing as *partially* disjoint problems accounting network status (i.e. Bandwidth consumption and latency) to make an approximate selection of the end-to-end paths with respect to the network constraints expressed in each service request of the set. The routing optimization is demanded to a second step (e.g. when actually a traffic flow has to be processed by the VNFs) that may refine the path choice that has been approximately made during the first phase. This Section describes the formulation of the Virtual Network Function placement problem for a set of requests to be instantiated across a set of distributed DCs.

In this paragraph a conceptual model of all the VNF Placement problem is provided. It describes the various entities, their attributes, roles, and relationships, plus the constraints that govern the problem domain.

Firstly the following entities are introduced:

- **Virtual Network Function (VNF)**: in a NFV architecture, a VNF, is responsible for handling specific network functions (e.g. DPI, LB, NAT, etc.) that run in one or more virtual machines on top of the hardware networking infrastructure.

- **Service Function Chain (SFC)**: a set of functional capabilities (i.e. VNF) properly chained for assuring the end-to-end delivery of a given service.

- **Compute Node**: entity within the network that provides VNF capabilities. Compute Nodes are Data Centers of different sizes and capabilities, positioned more or less close to users, which provide the general infrastructure able to host on demand deployments.

- **Link**: entity within the network that provides connectivity between two Compute Nodes; it is characterized by two parameters: bandwidth and latency.

By taking into account a realistic network configuration which is composed from nodes (e.g. forwarding elements or storage and compute elements such as Data Centers) and links that connect nodes, a network transformation has been performed in order to obtain a simplified version. Hence, the considered simplified network is composed by a set of Compute Nodes (Data Centers) and represented as a graph $G = (D, E)$, where D is the set of Data Centers and E represents the edges, i.e., Links, between those compute nodes. All edges e belonging to E are bidirectional. Data Centers are characterized by a set of resources (i.e. CPU, Memory and Storage) that allow them to run VNFs. The amount of these computational resources is defined as *Resource Capacity.*

It is assumed that the links that connect Compute Nodes are characterized by:

- **Latency** L [ms] - i.e. propagation delay which separates two nodes, thus directly dependent on the physical distance. In particular in this problem is considered the end-to-end delay between the Ingress and Egress node which represents the time taken by traversing the intermediate nodes from the source to the destination. The network latency is also given by the transmission delay which is the time taken to transmit a packet over a link which depends on the packet size and on data rate available on the link itself. Keeping to the purpose of the VNF placement problem without accounting an actual VNF workload, this latency component is considered in this work as negligible.

- **Bandwidth** B [Gbps] - i.e. a link data rate capacity. It is shared among all paths which respectively occupy a portion.

E_{ij} is the set of network links between a compute node i and a compute node j and e_{ij} is the link between compute node i and j with minimum latency. Correspondingly, L_{ij} is the minimum latency between compute node i and j, by supposing that exist more paths between i and j.

The Orchestrator has to manage a *set* of service requests which are composed by VNF chains for which it is assumed that no VNF has already been placed.
Service requests may arrive from two types of users:

- **Premium user**: against a higher amount of money, this type of user benefits from some privileges than other users, to take advantage of network resources. In the mathematical model that is detailed in Section 3.5.2, these privileges are given by the assignment of a greater weight (w_p) than that to Best Effort user (w_b).
- **Best Effort user**: this user requests the use of resources for a service, but is not willing to pay more for Premium privileges and thus, in the mathematical model, will be assigned with a lower weight.

A service request r, belonging to a request set R, is characterized as follows:

- **User Identifier** (Request typology): this is a binary parameter which indicates if the service request comes from a Premium (1) or a Best Effort (0) user;
- **Ingress node** o^r: this parameter defines the network node which is the source of the traffic flow;
- **Egress node** d^r: this parameter defines the network node which is the destination of the flow;
- **Maximum End-to-end delay** l^r [ms]: this parameter represents the maximum tolerated end-to-end delay of the single request;
- **Bandwidth Consumption** b^r [Gbps]: this parameter represents the bandwidth consumption that is the minimum accepted data rate capacity for the service chain execution;
- **Setup-time** s^r; this parameter is associated with a specific requirement of the service request that concerns the necessity to avoid the Setup Time typically required to instantiate a VNF on a Virtual Machine that is not already booted. This requirement can be accounted by compute nodes that provide the VNF instantiation on *Containers* such as Docker [136]. This is a binary parameter whose values mean:
  - value 0: the setup-time avoidance does not represent a mandatory requirement for the provision of the service.

  – value 1: it requires that the setup-time has the least possible value; in my scenario means that all the VNF of the chain must be placed on a container (if available). In fact, the time to instantiate a VNF in a container is typically less with respect to that needed to instantiate the VNF on a Virtual Machine.

- **Service Cost** c^r [euro/dollars per capacity unit]: this parameter represents the total amount of money that the user is willing to pay to obtain the requested service;
- **Carbon footprint** f^r [CO_2 emission per capacity unit]: this parameter represents the intention to take into account the emissions of a server/DC by expressing server/DC tolerated CO_2 emissions, hence this is the maximum tolerated CO_2 emission.
- **Service chain** H^r: this is the ordered sequence of VNFs

$$[V_0^r, V_1^r, ..., V_n^r], \qquad n = 1, ..., H$$

  that compose the service request. Each VNF is in turn characterized by Resource Required $u_{V_h^r}$ [quantity of capacity units]: this parameter represents the resource capacity required by each VNF in terms of CPU, memory, storage, etc. In this work, this parameter has been considered as per capacity unit without making explicit reference to the actual components.

The *User Identifier* has been introduced to allow the system to recognize the user typology in order to properly grant the priority privileges. A Premium user can achieve these privileges for various motivation such as a promotion, a custom offer or a fixed subscription. The higher monetary amount that a Premium User may pay for a fixed subscription, refers to an added fixed amount that could be proportional to the requested resources, but does not change the parameter *Service Cost* whose intent is common to all users. Indeed a possible variation of the *Service Cost* parameter among different users (also belonging to the same category) is however taken into account to propose a heterogeneous request set. This parameter is also deprived from the *Operating Margin* that, for instance, a Service Provider could extract from the whole business with the infrastructure provider. Also this assumption has been done to preserve the service cost proportionality that allows comparing the service requests.

Compute Nodes, or rather Data Centers, are characterized by the following parameters:

- **Resource Capacity** U_i [quantity of capacity units]: this parameter represents the whole resource capacity of a Data Center in terms of CPU, memory, storage, etc. In this work, this parameter has been considered as capacity units quantity without making explicit reference to the actual components.
- **Container** S_i: this parameter expresses if the DC is able to instantiate VNFs in a Container (as mentioned above) in order to allow a quicker service provision by avoiding Virtual Machine instantiation Setup-Time. This parameter is binary and its value is 0 if the DC is not able to provide Container instantiation, 1 otherwise;
- **Service Price** C_i: this is the price for capacity unit that a DC exposes to provide its resources;
- **Carbon footprint** F_i: this parameter represents DC emissions in terms of CO_2 per capacity unit.

As mentioned above, this work has been designed by avoiding idealistic scenarios that deviate from real cases; in further support of this assumption, it has been considered that the system does not guarantee that all received requests could be served. By accounting the above-mentioned and this last assumption, the following hierarchical joint objectives are defined:

1. **Maximization of served requests**: the first objective is represented by the maximization of accomplished requests belonging to Premium and Best Effort users by according priority to Premium. This objective can be considered *Infrastructure perspective* as the Infrastructure Provider is interested in maximizing its profit through the weighted sum of accomplished requests.
2. **Maximization of granted VNF preferences**: this second objective takes into account the *User perspective* as it aims at serve the VNFs of each request according to their specific preferences.

In other words each VNF of a request expresses a *Preference ranking* of all the available DCs by considering their exposed features; the objective is to maximize the overall preferences.

The objective function that models these joint hierarchical goals, is defined in Section 3.5.2. For the definition of the model, the following constraints are specified:

- **End-to-end delay**: for each request, the maximum tolerated end-to-end delay l^r must be respected to provide the user with the required QoS; hence this parameter should be less than the maximum propagation delay associated to the chosen forwarding path.
- **Compute Node efficiency**: a resource capacity occupation interval is considered in order to take into account energy efficiency and avoid workload overhead. It has been assumed that to be considered for VNF deployment, a Compute Node load must be within $\overline{U}$ and $\underline{U}$. This constraint has been included to favor the so-called "server consolidation" by progressively reducing low occupations of capacious resources in order to fill those not busy enough; this is precisely what underpins this virtualization-based new paradigm and the opposite of what happens in the non-virtual world. Moreover with an eye on the energy saving context that has been explained in Section 2, by imposing a constraint that consists in turning OFF a rarely/low used compute node, the energy efficiency aim is taken into account and this is another contribution of the thesis.
- **Cost/Price meeting**: the service price, which is given the sum of the prices exposed by the Data Centers in which the VNFs would be instantiated, must not exceed the service cost c^r.
- **Bandwidth**: each link is characterized by a total data rate capacity (i.e. Bandwidth capacity) that can be shared by many paths which use a bandwidth portion. In other words several traffic flows can simultaneously share the same link through different paths. The sum of bandwidth portions occupied by each path on a specific link must be less or equal to the data rate capacity of that link. If this constraint is respected, for a new request the

unused bandwidth portion must be greater than or equal to the request bandwidth consumption b^r.

3.5 Pre-Processing phase and Optimization Model

The system that has been designed to solve the VNF Placement problem addressed in this thesis is graphically represented in Figure 3.6.

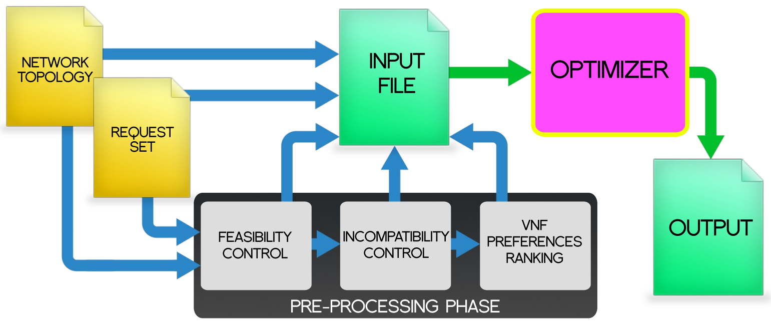


Figure 3.6: High-level representation of the implemented system to address the VNF Placement optimization problem

It is composed by four main components:

- **The Feasibility Control block**: this block performs the Feasibility control over the request set to be served. This algorithm determines whether each request is consistent with the system i.e. it excludes those requests that are characterized by some conflicting parameters in comparison with availability of resources or other exposed parameters at the DCs or with the network status (3.5.1).
- **The Incompatibility Control block**: this block is responsible of a comparison between the features exposed by each DC and those that characterize the chained VNF of each request in order to detect VNFs incompatibilities with DCs.
- **The VNF Preferences Ranking block**: this block provides the Optimizer with an ordered ranking of the VNF Preferences as

regards the Compute Nodes able to host them.

- **The Optimizer block**: this block performs the optimization of the VNF Placement problem aiming at achieving the hierarchical objectives by taking into account both infrastructure and users requirements and by respecting defined constraints.

By referring to the considered network that has been introduced in Section 3.4, a simplified network version which reports only the minimum latency links between compute nodes (DCs), previously referred as e_{ij}, is introduced. The *Network Topology file*, (see 3.6), contains three matrices that describe the simplified network introduced above.

- ***latency_matrix* [ms]:** this is a squared matrix that reports in both rows and columns all the DCs of the network; each cell contains the minimum latency L_{ij} between selected DCs;
- ***bandwidth_matrix* [Gbps]:** this is a squared matrix that reports in both rows and columns all the DCs of the network; each cell contains the minimum bandwidth value detected in the minimum latency link e_{ij} between selected DCs;
- ***DC_matrix*:** this matrix contains the specifications that characterize each DC of the network i.e. resource capacity, service price, Container, DC carbon footprint.

The *Request Set file* (see 3.6) contains all the information, provided in Section 3.4, that characterize each service request whose chained VNFs have to be placed.

3.5.1 Pre-processing phase model

As shown in Figure 3.6 this macro block processes the request set and the network topology information in order to produce data of interest for the *input-file* to be processed by the optimizer.

Feasibility control

The Feasibility Control block evaluates if a service request that belongs to a request set can be defined as "Feasible" or "Rejected". This block performs a preliminary analysis on the request set in input to the system in order to exclude those requests that present one or more conflicting elements with the systems (e.g. a service request asks for a free resources rental). This control is done in order to avoid processing requests that surely would be excluded. Unaccomplished requests represent a meaningful information to actually evaluate the goodness of the optimization, but accounting requests that a-priori can be classified as "infeasible", would cause invalidate results.

In order to decide whether a request can be defined as "Feasible" the algorithm perform a comparison between some request parameters and some other features belonging to both the network topology (i.e. bandwidth and propagation delay) and DCs features (i.e. Cost per Capacity Unit and Service Price). By taking in input the two above-described files, the *Feasibility Control* algorithm performs the following Limit-Case Controls (LCC):

1. **Latency LCC**: this control performs a comparison between the maximum tolerated end-to-end delay for each request (l^r) and the minimum possible achievable propagation latency ($\underline{l^r}$). This last parameter is obtained by the consideration of the limit case for which all the VNFs of the chain are placed in the same DC, thus with a direct forwarding path between Ingress o^r and Egress d^r nodes; the $\underline{l^r}$ can thus be defined as $latency\_matrix(o^r, d^r)$. Then the following Feasibility Control is performed:

$$latency\_matrix(o^r, d^r) \leq l^r \tag{3.1}$$

   If this comparison is true the request can continue the Feasibility Control, otherwise it is classified as "Rejected".

2. The Feasibility algorithm continues with the following which is composed with both the same procedure and assumptions that have been described at point 1 but referring to the Bandwidth parameter.

$$bandwidth\_matrix(o^r, d^r) \geq b^r \tag{3.2}$$

   The maximum possible bandwidth that can be assigned, in this case, should be greater than or equal to the bandwidth consumption of the service request b^r. Again if the comparison is true, the request undergoes the next check, otherwise is excluded.

3. Finally the last control is done by considering the capacity unit cost c_n^r (where n represents the capacity unit) and the Service Price C_i per capacity unit of each DC as expressed as follows:

$$c_n^r \geq C_i \tag{3.3}$$

   The c_n^r value is calculated through the ratio between the Service Cost of a request c^r (i.e. how much the user is willing to pay for that request) and the total capacity units requested u^r (i.e. the sum of requested capacity units by each chained VNF above referred as $u_{V_h^r}$). In this way a unitary medium cost value is obtained and can be compared with the unitary capacity price exposed by each DC. This control refers to the balance between supply and demand; if a request asks to pay a service price that is lower than those exposed from DCs, this request will not be surely accepted by any DC; thus is infeasible.
   If the request r also passes this control can be labeled as "Feasible", otherwise it will be rejected.

These three controls must all be satisfied to allow the request to access to the Placement process; this does not mean that it will definitely be served, but only that it is compatible with the system supply.

Incompatibility control

The Incompatibility Control block (Figure 3.6) performs a comparison between the specifications of the chained VNFs (of the feasible request set) and those exposed by each DC of the network. This comparison produces a response that informs firstly the next block (i.e. VNF Preferences Ranking) and secondly the Optimizer about the DCs that are forbidden for specific VNFs. As described above (3.4) each Data Center and each request are characterized by several components; those features used in this block are listed and matched in Table 3.3.

The Incompatibility control block performs a comparison between the matched components of these two entities as listed below:

1. $U_i \geq u_{V_h^r}$; Resource Capacity of the i_{th} DC must be greater than or equal to the Resource Required by each VNF. If this comparison gives a negative result, the i_{th} DC is incompatible with VNF h of

Table 3.3: Comparison between matched components of DCs and Requests/VNFs

Data Centers	**Requests**
Resource Capacity U_i	Resource Required $u_{V_h^r}$
Service Price C_i	Service Cost c^r
Container S_i	Setup-Time s^r
DC Carbon Footprint F_i	VNF Carbon Footprint f^r

request r and therefore it can not be instantiated in that Compute Node.

2. $C_i * u^r \leq c^r$; the exposed Service Price of the i_{th} DC for the whole service must be less than or equal to the Service Cost c^r. If this comparison gives a negative result, the i_{th} DC is incompatible with r and therefore its VNFs cannot be instantiated in that Compute Node.

3. S_i compared to s^r; in this case the control is true/false as these parameters are binary. Hence if the VNF requires to be instantiated on a container to avoid Setup-time, this comparison must be true $(1:1)$, moreover if the VNF does not need to avoid setup time $(1:0)$ the comparison is true again as the i_{th} DC Capability to provide a container does not prevent to serve VNFs without this requirement. On the contrary if the i_{th} DC does not have this skill but the VNF has this requirement $(0:1)$, the i_{th} DC is incompatible with VNF h of request r and therefore it can not be instantiated in that Compute Node. The $(0:0)$ refers to the case in which v_h^r does not need to avoid setup-time and the i_{th} DC does not have the container.

4. $F_i \leq f^r$; the CO_2 emissions of the i_{th} DC must be less than or equal to the VNF tolerated CO_2 emissions. Again if this comparison gives a negative result, the i_{th} DC is incompatible with request r and therefore its VNFs cannot be instantiated in that Compute Node.

If all the above controls have been satisfied, the i_{th} DC is compatible for r and the process will proceed with the Ranking algorithm. On the contrary, if at least one of the above checks fails, the VNF is incompatible for DC$_i$ and the corresponding placement is inhibited. At the end of this step the Incompatibility block will return the **Incompatibility Matrix** I which will indicate the number of DCs where that specific virtual function can not be placed and, in detail, which they are. It will contribute to the composition of the *input-file* to the Optimizer block.

VNF Preference Ranking

The VNF Preferences Ranking block (Figure 3.6) deals with the definition of *Preference Ranking* among available DCs of each h VNF of a request r. In other words by means of this algorithm each VNF will propose its DC ranking by assigning to each DC of the network a preference rating. Before express the ranking, a DC analysis step has to be done: each DC is, in fact, evaluated by 4 parameters (the votes V_i) that allow each VNF to express their preferences. These values (which are calculated below) are standardized according to a scale of values ranging from 0 to 10 and then, in a second step, summed together according to the actual preferences of each VNF. Each DC vote is calculated as follows:

1. **Data Center Resource Capacity**: at this stage a rating is assigned to the DC depending on its Resource Capacity. The vote is made as follows:
$$V_{U_i} = \frac{U_i * 10}{U_{max}} \tag{3.4}$$
The V_{U_i} variable is a numerical value between 0 and 10, as it can be observed from the above formula 3.4. U_i is the parameter that represents the Resource capacity of the i_{th} DC and U_{max} is the maximum Resource Capacity that belongs to a DC of the network.

2. **Data Center Service Price**: in this case a rating is assigned to the DC depending on the Service Price that it asks to accomplish a service (calculate per capacity unit). This vote is made as follows:
$$V_{C_i} = \frac{C_{min} * 10}{C_i} \tag{3.5}$$
To allow a further ratings sum, even in this case the vote is between 0 and 10. C_i represents the price per capacity unit of the i_{th} DC

while C_{min} represents the minimum price that belongs to a DC of the network.

3. **Data Center Container availability**: in this case the vote is calculated in a different way. The parameter V_{S_i} can obtain the following rating values:

    - $V_{S_i} = 10$ if the VNF asks for a container to avoid setup time and the i_{th} DC can provide it;
    - $V_{S_i} = 7,5$ if the VNF does not ask for a container but the i_{th} DC can provide it anyway; this rating is greater than the following ($V_{S_i} = 5$) as it is considered as an added value to the DC features.
    - $V_{S_i} = 5$ if the VNF does not ask for a container and the i_{th} DC cannot provide it;

4. **Carbon footprint emissions**: at this other stage a rating is assigned to the DC depending on its CO_2 emissions. The vote is assigned as follows:

$$V_{F_i} = \frac{F_{min} * 10}{F_i} \tag{3.6}$$

    Where F_i indicates the CO_2 emissions for the i_{th} DC and F_{min} the minimum CO_2 emissions that belong to a DC of the network. Also this rating can assume a value between 0 and 10.

Once all the DC votes have been calculated, each VNF preference for each DC, that had not already been excluded in the preceding block, can be calculated as follows:

$$p^r_{V^r_h i} = \frac{w_1 * V_{U_i} + w_2 * V_{C_i} + w_3 * V_{S_i} + w_4 * V_{F_i}}{\sum_{i=1}^{4} w_i} \tag{3.7}$$

where w_i represents the binary weight that each VNF associates to the rating procedure. w_i can assume a binary value that is 1 if it is associated to an essential component for the deployment of that specific VNF, 0 otherwise. For instance let us consider the case in which a service request to which that specific VNF belongs, does not need to avoid a Virtual Machine setup time by asking for a Container installation. The VNF, which does not need this specific feature, prefers that this not required service would not affect the definition of its preference. Hence it asks the system to set $w_3 = 0$. By eliminating one of the weights, the sum in the denominator will be achieved by the three remaining weights. Therefore, as regards the VNF placement phase (Section 3.5.2), the consideration of the matching between all the requested features and those offered by DCs is essential, but to generate the VNF ranking only those parameters that effectively are of interest for single VNF of a request will be considered. At the end of this step the VNF Preference Ranking block will return the **VNF Preference Matrix** which will report in its columns all the chained VNFs for each request of the set and in its rows all the DCs. Thus each cell of this matrix will contains the vote (between 0 and 10) accorded by the single VNF to the i_{th} DC.

Even this matrix will contribute to the composition of the *input-file* to the Optimizer block that has been shown in Figure 3.6.

3.5.2 Optimization model

This Section describes the mathematical model that addresses the optimal VNF Placement in a multi-domain NFV infrastructure and in igure 3.6 is represented as the Optimizer block which receives the abovementioned *input-file*.

For each request set R consisting in a group of service requests indexed by r each one composed by an ordered sequence of h virtual functions V_h^r, an auxiliary graph $G^R(D^R, A^R)$ is built. Specifically, G^R is a layered graph with a level for each of the h virtual functions appearing in each request r. The greater length of a chain of R (i.e. the greater number of VNFs present in a chain) is considered to define the number of level h. This work does not lose of generality if, for the sake of clarity, it is assumed that all the service chains have the same length. Each level h is composed by all those DCs D_i with $i = 1, .., n$ able to host the h virtual function V_h^r and shared among requests.

Figure 3.7 shows an example of a multi layer graph G^R; it is composed by H levels. Each level h corresponds to the h-VNF in each chain; the number of levels will be hence equal to the number of the considered VNFs. Each level is composed by the whole DCs that belong to the network which in turn can host one or more VNF of each service chain. Two extra layers are considered:

- **Level 0** that contains the source node o^r

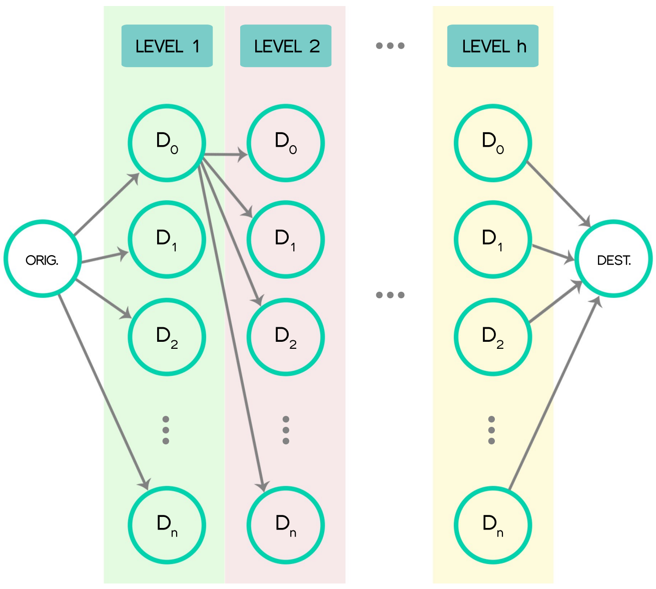


Figure 3.7: Auxiliary multi-layer graph for a request set R [137]

- **Level h+1** that contains the destination node d^r

A^R represents the set of arcs which can be grouped in three groups:

1. The first group contains arcs from the source node o^r to each node in level 1.

2. The second group contains arcs from each DC in layer $h+1$ to the destination node d^r.

3. The third group is composed by all the arcs that link each DC_i in level h with each DC$_j$ in level $h+1$ for each intermediate level.

Each arc corresponding to the link (or path) between DC_i and DC_j in A^R is characterized by the propagation latency l_{ij} between D$_i$ and D$_j$ and the bandwidth b_{ij} as an indicator of the arc data rate capacity. The layered structure of the graph ensures that the order of VNFs specified in the request is preserved. A request can be represented as a path through the layered graph; since in this work the system is expected to manage a request set, each request can be represented as a separate path through the layered graph, thus each request shares available resources. By construction such a path visits exactly a node in each level: at each intermediate level h the visited node identifies the Data Center D_i that will host the VNF h of the chain in the request r.

In order to model the DC instantiation in which a VNF will be deployed, the following decision variables are defined:

$$x^r_{ihjh+1} = \begin{cases} 1 & \text{if the arc linking node } i \text{ in level } h \text{ and node } j \text{ in level} \\ & \quad (h+1) \text{ belongs to the path relative to } r \in R \\ 0 & \text{otherwise} \end{cases}$$

$r \in R$, $i \in D \cup \{o^r\}$, $j \in D \cup \{d^r\}$, $h \in H^r$
if the V^r_h is placed on the DC i and the V^r_{h+1} is placed on the DC j

$$y_i = \begin{cases} 1 & \text{If a VF is placed on DC } i \\ 0 & \text{otherwise} \end{cases} \qquad i \in D$$

$$z^r = \begin{cases} 1 & \text{If the request } r \text{ has been accepted} \\ 0 & \text{otherwise} \end{cases} \qquad r \in R.$$

By using these variables, and the above-defined notation, formally the problem can be stated as follows.

The hierarchical Objective Function is defined in (3.8) and it consists in the maximization of the weighted sum of the two objectives that have been introduced in Section (3.4). The hierarchy between objectives (i.e. the weighted profit received by the maximization respectively of served requests and of accorded VNF Preferences) is obtained through weight W which gives more relevance to the first. Premium user requirements, with respect to Best Effort user, acquire a greater value by means of assigned weights (i.e. $w_p \gg w_b$). (3.9), (3.10) e (3.11) are for each $r \in R$ of traffic flow conservation constraints. Specifically, constraint (3.9) assures that exactly one unit of flow leave the source node o^r when request r is accepted; since, by definition of the decision variables, the flows are unsplittable exactly one of the arc outgoing from the source o^r will be selected. The ending node of such an arc belongs to L_1 and it identifies the DC that hosts the first virtual function in the request r. Symmetrically, for each $r \in R$ exactly one unit of flow enters the destination node d^r as imposed by constraint (3.10) when the request is accepted. Constraint (3.11) assures that for each request $r \in R$ for each intermediate node j other than the source o^r or destination d^r, the quantity of flow entering node j is exactly the same as the one leaving node j. Constraints (3.12) and (3.13) guarantee that the sum of required resources to execute the whole request set R is less than the available resources of the i_{th} Data Center. If this condition is respected, the algorithm proceeds with the Placement operations. In particular these two conditions specify the expected upper and lower range to consider the compute node as turned ON or OFF. As mentioned above, each compute node is characterized by an upper and lower usage threshold in order to respectively avoid workload overhead and improve energy efficiency. For this purpose constraints (3.12) prevents the instantiation of new VNFs on a node whose hosted capacity would exceed upper threshold as well as constraints (3.13) discourage the proliferation of lightly loaded nodes. Constraints (3.14) guarantees that for each request, the total sum of the service prices incurred to host each chained VNF in the selected DCs must be less than or equal to the request cost c^r that a user (Best Effort or Premium) is willing to pay to get the requested service.

$$\max \quad W[\sum_{r\in R_p} w_p \cdot z^r + \sum_{r\in R_b} w_b \cdot z^r] + \sum_{r\in R}\sum_{h=0}^{|H^r|-1}\sum_{i\in L_h}\sum_{j\in L_{h+1}} p^r_{V^r_{h+1}i} \cdot x_{jhih+1} \tag{3.8}$$

$$\sum_{j\in L_1} x^r_{o^r0j1} = z^r, \quad \forall r \in R \tag{3.9}$$

$$\sum_{j\in L_{|H^r|}} x^r_{j|H^r|d^r|H|+1} = z^r, \quad \forall r \in R \tag{3.10}$$

$$\sum_{j\in L_{h-1}} x^r_{jh-1ih} - \sum_{j\in L_{h+1}} x^r_{ihjh+1} = 0, \forall r \in R, \forall h \in \{1,\dots,|H^r|\}, \forall i \in L_h \tag{3.11}$$

$$\sum_{r\in R}\sum_{h=0}^{|H^r|-1}\sum_{j\in L_h} u_{V^r_{h+1}} x^r_{jhih+1} \leq \overline{p} \cdot U_i \cdot y_i, \quad \forall i \in D \tag{3.12}$$

$$\sum_{r\in R}\sum_{h=0}^{|H^r|-1}\sum_{j\in L_h} u_{V^r_{h+1}} x^r_{jhih+1} \geq \underline{p} \cdot U_i \cdot y_i, \quad \forall i \in D \tag{3.13}$$

$$\sum_{h=0}^{|H^r|-1}\sum_{i\in L_h}\sum_{j\in L_{h+1}} c_j x^r_{ihjh+1} \leq c^r, \quad \forall r \in R \tag{3.14}$$

$$\sum_{h=0}^{|H^r|}\sum_{i\in L_h}\sum_{j\in L_{h+1}} l_{ij} x^r_{ihjh+1} \leq l^r, \quad \forall r \in R \tag{3.15}$$

$$\sum_{r\in R}\sum_{h=0}^{|H^r|} b^r x^r_{ihjh+1} \leq b_{ij}, \quad \forall i,j \in D \tag{3.16}$$

$$\sum_{j\in L_{h-1}} x^r_{jh-1ih} = 0, \quad \forall r \in R, \forall (V^r_h, i) \in I \tag{3.17}$$

$$x^r_{ihjh+1} \in \{0,1\}, \forall r \in R, \forall i \in D \cup \{o^r\}, \forall j \in D \cup \{d^r\}, \forall h \in \{0,\dots,|H^r|\} \tag{3.18}$$

$$y_i \in \{0,1\}, \quad \forall i \in D \tag{3.19}$$

$$z^r \in \{0,1\}, \quad \forall r \in R \tag{3.20}$$

Constraints (3.15) guarantees that the end-to-end delay to accomplish the service request must be less than or equal to the maximum tolerated latency of the request l^r. Constraints (3.16) guarantee that the data rate capacity of link (i, j) must be greater than or equal to the bandwidth consumption of all the service requests. In other words it ensures that the bandwidth required to accomplish all of the service requests must be less than or equal to the maximum available bandwidth on the path from D_i to D_j. Constraints (3.17) refer to the Incompatibility matrix I (see Section 3.5.1) and guarantee that in each service request for each VNF of the chain, if a VNF is incompatible with a specific DC, the arc that connects the previous VNF of the chain (or rather the Ingress node o^r) with the incompatible DC will be closed or equivalently the corresponding x variable set to 0. Finally constraints (3.18), (3.19) and (3.20) define the binary variables domain. This optimization model has been defined taking a cue from an already existing work, described in [137], that however, addresses the composition of computing and networking Virtual Functions to select the nodes that already contains instantiated VNFs over the path that minimizes the overall latency; the VNF Placement, in fact, is not considered in this work as it concerns only VNF selection.

3.6 Evaluation

This Section provides the evaluation scenario and the testing activities carried out to examine the proposed VNF Placement optimization model.

3.6.1 Evaluation scenario

The testing activity has been carried out by considering a realistic scenario as regards both the network topology (i.e. nodes, links and general parameters) and the sets of requests considered.

Reference Network

A telecommunication network is typically organized over three levels: access, aggregation and core. The access level represents the most peripheral network portion; also called Edge network for its closeness to the users, it is typically provided with low-capacity compute nodes but also characterized by low latencies. The aggregation level is the network portion in which traffic flows are aggregated to be transmitted to other access nodes or to the core network. This last level, the core, can be described as the principal part of the network as it consists of high-speed interconnections devoted to carrier-grade packet delivery services across wide areas (e.g. national context) where are typically positioned high-capacity Data Centers [137]. Proportionally, access and aggregation networks are associated to smaller areas than core network, such as respectively metropolitan or regional.

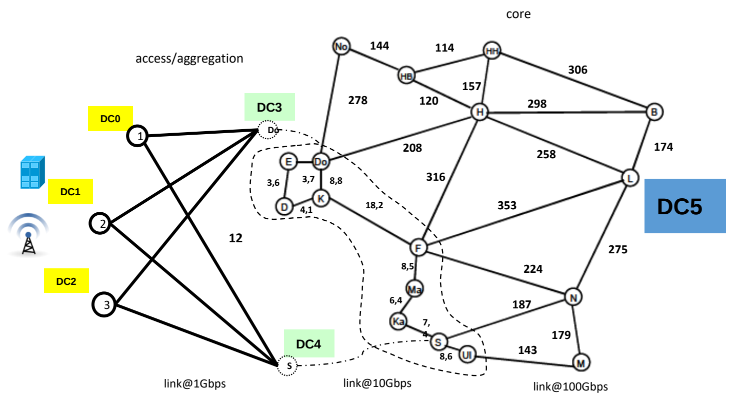


Figure 3.8: Reference network infrastructure and DCs distribution throughout the Access, Aggregation and Core network

By considering these definitions of a *telco* network and the evaluation scenario proposed by Martini *et al.* [137], the network scenario that has been taken into account is represented in Figure 3.8. The image shows, on the left side, the access network that is composed by three nodes, namely 1, 2 and 3, in which as many Data Centers (i.e. DC_0, DC_1, DC_2) have been positioned. The edge network interconnections are inspired to a typical tree-like topology with redundant links among small and medium DCs. These small DCs, characterized by low processing capability (3GB/s), are connected with the aggregation nodes through links with a data rate capacity (bandwidth) of 1Gbps and with an approximate distance of 12 km, represented in the Figure by thick black lines. In the aggregation network, that in Figure 3.8 is located approximately in the middle, there are two medium DCs (i.e. DC_3, DC_4) with a higher processing capability (15GB/s) and links characterized by a bandwidth of 10 Gbps. Finally in the right side of the Figure is shown the core network that benefits from high capacity links performing 100 Gbps; in there a large DC (i.e. DC_5) with high processing capability (30GB/s) has been positioned. The core network interconnections are inspired to a real telco network topology in Germany [138] with 17 nodes and 26 links. In Figure 3.8 specific links lengths are reported in km above the connection lines for both edge and core networks. Each link is characterized in terms of propagation delay and bandwidth capacity.

Table 3.4 provides the specification of each DC of the network with the features that have been considered to test the proposed model.

Table 3.4: Data Centers specifications

Data Center	Resource Capacity	Service Price	Container	Carbon Footprint
DC0-small	3	0	3	2
DC1-small	3	1	3.5	3
DC2-small	3	1	3.5	4
DC3-medium	15	0	4	3
DC4-medium	15	1	5	5
DC5-large	30	1	6	3

The Resource Capacity values of the three DC typologies have been set as in [137]. The service price of each DC has been proportioned to its skills; for instance the large DC_5 exposes the higher price as it provides the highest processing capability and the possibility to use a Container instead of a Virtual Machine to instantiate VNFs. As it can be observed the opposite is true for DC_0. The Carbon Footprint has been arbitrarily assigned by assuming that DC_2 and DC_4 were of older generation and causing more substantial emissions.

Request Set

In this scenario it is assumed that six VNF types are available to compose the service chains that characterize the requests of the request sets. Table 3.5 lists all the VNF types and their identification numbers.

As mentioned above (Section 3.3) the service requests have been composed by considering the papers in [128, 135] which provide Virtual Functions listed in Table 3.6.

For instance a Web Service is composed by the following service chain: NAT-FW-TM-WOC-IDPS. In the *input-file* to be processed by the algorithm, this chain will be specified with identification number

Table 3.5: Typology of VNF used to perform the test

VNF Type	VNF ID Number
NAT (Network Address Translation)	0
FW (FireWall)	1
TM (Traffic Monitor)	2
WOC (WAN Optimization Control)	3
IDPS (Intrusion Detection and Prevention Systems)	4
VOC (Video Optimization Control)	5

Table 3.6: Service chains that have been considered to compose each request set [128,135]

Service	Chain	Latency	Bandwidth
Web Service (WS)	NAT-FW-TM-WOC-IDPS	500 ms	100 kbit/s
VoIP	NAT-FW-TM-FW-NAT	100 ms	64 kbit/s
Video Streaming (VC)	NAT-FW-TM-VOC-IDPS	80 ms	4 Mbit/s
Cloud Gaming (CG)	NAT-FW-VOC-WOC-IDPS	60 ms	50 kbit/s
5G Service (5GS)	NAT-FW-TM-WOC-VOC	20 ms	2Mbit/s

of each VNF which, in this case, are 0, 1, 2, 3, 4. Its important to remark that the typology expresses only an identification as regards the functionality of a VNF. This assumption is relevant for two reasons: firstly it allows distinguishing among different VNF types with the same specifications (i.e. resource required, setup-time, etc.) and secondly it considers that two VNFs of the same type can be specified with different features (e.g. two VNFs of the same type may have to process different workloads and therefore require more or less resources).

Table 3.7: Request Set sample

	User Type ID BE=0, P=1	**Ingress node**	**Egress node**	**E2E Delay [ms]**	**Container [0/1]**	**Service Cost [euro]**	**Bandwidth [Gbps]**	**Carbon Ftp [emis param]**
r_1	1	0	5	500	0	88	0.0001	4
r_2	0	2	4	100	0	35	0.000064	4
r_3	0	0	0	60	0	60	0.000050	5
r_4	1	1	5	20	1	62	0.002	5

	Required resources for each chained VNF					**Chain**
	VNF$_1$	**VNF$_2$**	**VNF$_3$**	**VNF$_4$**	**VNF$_5$**	**composition**
r_1	2	1.8	1	2.5	2.1	0, 1, 2, 3, 4
r_2	1	2	0.5	2	1	0, 1, 2, 1, 0
r_3	1	1	1.5	2.5	2	0, 1, 3, 4, 2
r_4	2	3	2.5	2	1.8	0, 1, 4, 3, 2

Test set and environment

By considering Table 3.6, each request set have been composed and specified as shown in Table 3.7. The first Table reports the four request with their general service specifications while the second details each request by explaining the composition of the service chain. Premium/Best Effort user weights (introduced in Section 3.5.2 in the Objective Function) have been set to respectively $w_p = 3$ and $w_p = 1$. This model enhances generalization bringing as a special case $w_p = w_b = 1$; in this case the maximization of the objective function would regard, on the tenant perspective, the whole request set acceptance without any discrimination between users. Moreover, for this preliminary testing phase, the weight that, in the objective function (3.8) grants profit privileges to Infrastructure providers, is set to $W = 100$. In order to evaluate the effectiveness of the model that have been formulated, both requests acceptances and rejections are relevant to be analyze. Hence each request set have been composed by "borderline" requests namely characterized by not too favorable features with respect to the DCs offers. For example if a generic DC in the network asks for a certain price to rent its resources, the cost that a user is willing to pay has been set approximately (more or less) as that of DC. This has been done for two main reasons: firstly to keep the perspective of *demand/offer* balance typical of a realistic scenario and secondly to detect in the motivated rejections a proper behavior of the model.

Table 3.8: Heterogeneous test set composed by combination of Best Effort (BE) and Premium (P) request in each request set

Request Sets	R_6	R_5	R_4	R_3	R_2
Tests					
T_1	5P - 1BE	4P - 1BE	3P - 1BE	2P - 1BE	2P - 0BE
T_2	5P - 1BE	4P - 1BE	3P - 1BE	2P - 1BE	2P - 0BE
T_3	5P - 1BE	4P - 1BE	3P - 1BE	2P - 1BE	2P - 0BE
T_4	3P - 3BE	3P - 2BE	2P - 2BE	2P - 1BE	1P - 1BE
T_5	3P - 3BE	3P - 2BE	2P - 2BE	2P - 1BE	1P - 1BE
T_6	3P - 3BE	3P - 2BE	2P - 2BE	1P - 2BE	1P - 1BE
T_7	1P - 5 BE	1P - 4BE	1P - 3BE	1P - 2BE	0P - 2BE
T_8	1P - 5 BE	1P - 4BE	1P - 3BE	1P - 2BE	0P - 2BE
T_9	1P - 5 BE	1P - 4BE	1P - 3BE	1P - 2BE	0P - 2BE

The test set has been created by combining heterogeneous Premium and Best Effort Requests in each Request set that have been composed of 2 to 6 requests. Each test composition is shown in Table 3.8.

The optimization as well as the Pre-Processing Phase algorithms (i.e. Feasibility, Incompatibility and Ranking) have been implemented on a machine with Intel Core i7-4710HQ (up to 3.5GHz), 8 GB DD3 RAM, 1 TB HDD. The mathematical model presented in Section 3.5.2 has been implemented by means of the IBM ILOG CPLEX Optimization v12.6.3. running on a Virtual Machine with Ubuntu 16.04 Operating System with 4 GB RAM, 30 GB HDD. The *input-file* received by the Optimizer block is generated through the Pre-Processing Core algorithms 3.5.1 that have been realized with MATLAB academic version R2016A.

3.6.2 Evaluation metrics

In order to verify the correctness and the effectiveness of the implemented system, three different evaluation metrics have been considered:

1. **Acceptance Rate**: this metric has been used to show how many feasible requests of the request set have been positioned on the available and VNF-compatible DCs. In other words this metric assesses how much and how the objective of maximizing the Infrastructure provider profit, by accomplishing as many requests as possible, is reached. Moreover, it also provides an insight of this evaluation by making explicit the number of Premium and the Best Effort accepted requests. This is an essential evaluation to assess if the goal of granting a higher weight to Premium users is achieved.

2. **VNF preference satisfaction**: this metric evaluates the level of satisfaction reached by the single VNFs by considering their expressed preferences. To perform this analysis the VNF Preference matrix is compared to the output matrix that contains the DC-position of each VNF that belongs to an accepted request. Then, if the VNF is positioned in its first choice DC, it will receive vote 10; if in its second choice the vote will be 5, otherwise the vote will be null. The vote assignment has been accorded to a real satisfaction situation. When a person expresses a preference among four of five choices, it considers the first and the second respectively as a full or a partial satisfaction. The others can be considered as dissatisfaction, thus the 0 vote as well as the others

are realistic. The single request and the request set satisfaction will be calculated as the average vote respectively among VNFs and among requests.

3. **Execution time**: this last metric regards the optimizer code and the time it takes to provide the optimal solution.

The output produced by the Optimizer consists in a series of information used to firstly evaluate the proper functioning and then arrange the testing results (Section 3.6.3). It provides the total number of accepted requests, differentiating between Premium and Best Effort. Then, it is specified which request has been executed and in which DC the chained VNFs have been placed. The output file also reports the VNF preference degree by, even in this case, specifying Premium and Best Effort users. At the end it is also reported the list of turned on/off DCs within the network and the time required by the algorithm takes to perform the VNF Placement optimization.

3.6.3 Testing Results

All of the tests that are explained below were performed after a preliminary testing phase on the correctness of the model. In fact, after checking that actually the system excluded *unfeasible* demands, a verification on the objectives achievement was made. By composing *ad-hoc* requests, it has been verified that the model returned results according to expectations. For each test that is presented below, every request set has been composed by accounting the contents in Table 3.7 and the realistic service chains proposed in Table 3.6. The combinations of P and BE requests for the whole test set are shown in Table 3.8.

Acceptance Rate test

The acceptance rate test has been performed to evaluate the quality of the optimization in terms of *accepted requests* by also making explicit Premium and Best Effort users' requests.

Figure 3.9 reports the Average Overall Acceptance rate percentage that has been obtained by considering the increasing number of requests in the request set. Each reported value concerns an average evaluation of the results of accepted requests by considering the whole request set.

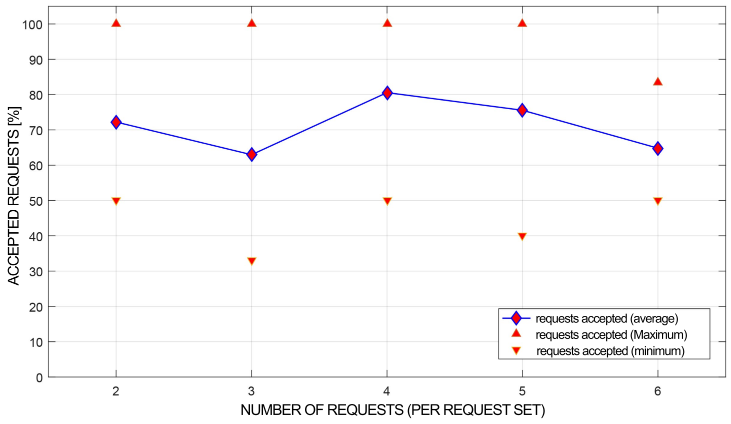


Figure 3.9: Average Overall Acceptance Rate percentage over the performed tests, in quantity-varying request sets

The trend denotes that for each request set typology the average number of accepted request among those arrived is always greater than 60% with maximum peaks of 100% and no 0% negative peaks. The request set composed of 4 requests, for instance, reports the highest average acceptance rate of about 80%; this can be observed also in Table 3.9 which details the Overall Acceptance Rate (both Premium and Best Effort users' requests are included) for each request set of tests T_n.

Table 3.9: Overall Acceptance Rate percentage and explicit accepted requests for each performed test T_n in quantity-varying request sets

Request Sets / **Tests**	R_6	R_5	R_4	R_3	R_2
T_1	4	4	3	2	2
T_2	4	5	4	2	2
T_3	4	2	3	2	1
T_4	4	4	3	1	1
T_5	4	4	4	3	2
T_6	4	3	2	1	1
T_7	3	3	3	1	1
T_8	5	5	4	3	2
T_9	3	4	3	2	1
Average Acceptance (%)	**64,81**	**75,56**	**80,56**	**62,96**	**72,22**

In this Table 3.9 average results that have been shown in figure 3.9 have been calculated with each single test results for each request set. Each cell reports the number of accepted requests over the total of the request set but does not explicit if they came from Premium or Best Effort users. This evaluation is strictly related to the combination of P/BE requests of each request set. As it can be seen from Table 3.8 in fact, green cells report a greater number of Premium requests with respect to Best Effort, on the contrary blue cells. The yellow ones are composed by an equal number of requests coming from Premium or Best Effort users. If, for instance, the average acceptance percentage of Premium and Best Effort for R_6 in test 1 (5P - 1BE) was compared to test 9 (1P - 5BE), the obtained results would be me meaningless. Appendix B reports each test Table with detailed results; thus the following evaluation examples can be deduced. This level of detail has been added to give more emphasis to the achieved results because, by considering only the average value among quantity-varying request sets, some key information would be lost. T_1R_4 (3P - 1BE) reports, for instance, 3 Premium accepted requests over the feasible 4 as well as T_2R_3 (2P - 1 BE) reports 2 Premium accepted requests over the feasible 3. Instead, for example, T_9R_5 (1P - 4BE) reports 1 Premium and 3 Best Effort accepted requests over the feasible 5; T_1R_4 (1P - 2BE) reports 1 Premium and 1 Best Effort requests over the feasible 3. Thus it can be deduced that the optimizer in the first cases accomplishes all the P by excluding the only BE and in the second cases prefers to serve the only P by again excluding a BE.

VNF Preference satisfaction test

Even the VNF Preference Satisfaction test has been performed on the same complete test set that has been used to evaluate the Acceptance Rate 3.8. Anyway this kind of evaluation is more complicated as it takes to explicit each request in its chained VNFs and then to analyze their satisfaction as regards position in the DCs and the associated vote (as detailed in Section 3.5.2).

Figure 3.10 shows the average VNFs preference satisfaction as the number of request per request set increases. The graphical trend has been obtained, as explained above for the Acceptance test, by considering the complete test set (Table 3.8). Figure 3.10, in addition to the average satisfaction trend (which also includes unaccomplished requests), indicates the maximum and the minimum percentage satisfaction value that have been obtained in one of the performed test.

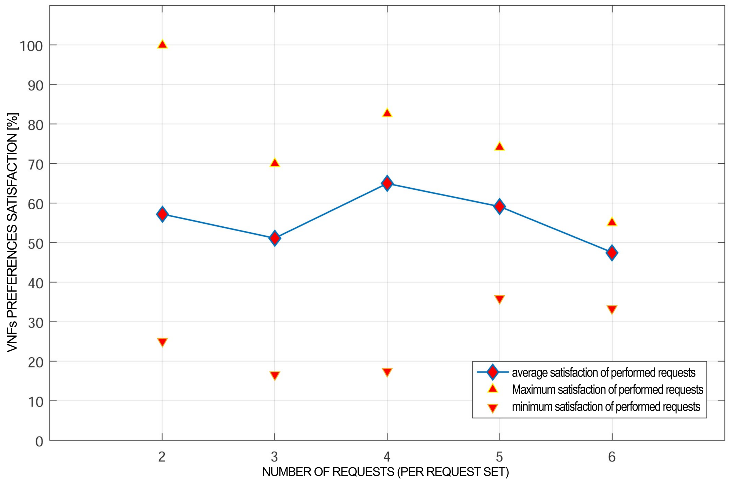


Figure 3.10: Average percentage of VNF Preference satisfaction with increasing number of requests per request set

As it can be observed the trend, even though the unaccomplished requests votes contribute to decrease the average satisfaction, results always greater than 47%.

Table 3.10 shows the VNF Preference Satisfaction vote in tenths (10 is the maximum assigned vote), averaged over VNFs for each request set; moreover it also provide the Overall VNF Preference satisfaction percentage.

As each request of a request set is composed itself by the chained VNFs, each cell in Table 3.10 reports an averaged value. Moreover, as each of these values is also affected by the unaccomplished requests, a meaningful evaluation on the optimization quality as regards VNFs preferences satisfaction cannot be performed only considering this data. Hence, Figures 3.11,3.12 and 3.13 are provided to show the average VNFs preference satisfaction for each performed test by excluding unaccepted requests. Each Figure shows the VNF Preference satisfaction percentage trend over the test set that includes respectively 6, 4 and 2

Table 3.10: VNF Preference satisfaction of the complete test set for each quantity-varying request set

Request Sets / **Tests**	R_6	R_5	R_4	R_3	R_2
T_1	5,36	6,60	7,00	6,67	10,00
T_2	5,50	7,20	8,25	6,67	6,50
T_3	3,58	3,66	7,00	6,67	5,00
T_4	5,83	6,80	6,50	1,67	8,33
T_5	5,50	6,60	8,50	8,33	7,50
T_6	3,33	3,80	1,75	2,67	2,50
T_7	3,83	4,80	7,00	1,67	2,50
T_8	5,50	7,40	6,20	7,00	8,50
T_9	4,33	6,40	6,25	4,67	4,00
Average VNF Pref. satisf.(%)	**47,42**	**59,11**	**64,94**	**51,12**	**57,22**

requests per request set. For each test is also reported the maximum, minimum and median obtained values. Moreover for each shown test, is also remarked the Load Factor (LF) percentage which is evaluated by means of the formula 3.21.

$$LF = \frac{\sum_{r \in R} u^a_{V^r_h} z^r}{\sum_{i \in D} U_i} \tag{3.21}$$

The LF parameter shows how the request set affects the whole network load as it is calculated by the ratio between the sum of all the capacity units that have been accepted and the sum of all the capacity units of all the available DCs in the network. In this analysis this is a relevant indicator as it underlines the influence of available/employed resources on according preferences. As it can be observed, in fact, in Figure 3.11 the minimum VNF Preference satisfaction is associated to a high LF in comparison to other tests. This is not the only factor that influences VNF preference satisfaction, but can help to understand some results.

Execution Time Test

This test has been done in order to evaluate the execution time of the VNF Placement optimization. Figure 3.14 shows the average overall execution time over the test set in quantity-varying request sets.

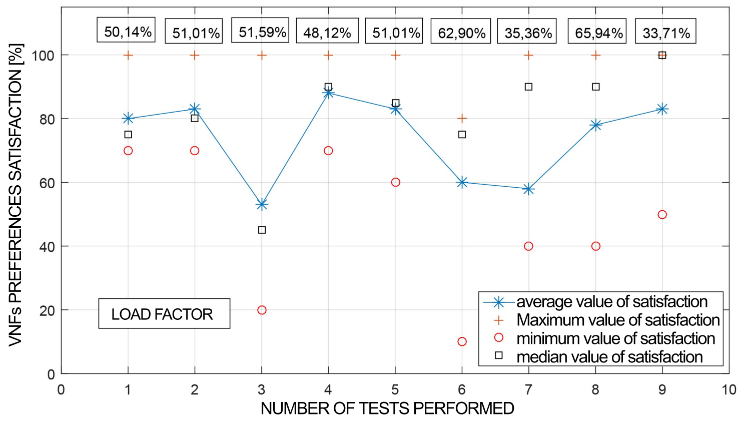


Figure 3.11: Average percentage of VNF Preference satisfaction for each performed test over a request set composed of 6 requests

In particular Table 3.11 reports the average execution time values (in seconds) per request set by specifying each performed test results.

Table 3.11: Average execution time values [s] per Request set for the whole test set

Request Sets / **Tests**	R_6	R_5	R_4	R_3	R_2
T_1	2,279	2,648	0,261	0,098	0,078
T_2	1,378	1,040	0,710	0,114	0,144
T_3	1,151	0,142	0,161	0,110	0,069
T_4	0,191	0,189	0,259	0,179	0,100
T_5	1,473	0,405	0,340	0,229	0,491
T_6	1,212	2,456	0,484	0,142	0,149
T_7	0,196	0,327	0,243	0,091	0,069
T_8	1,168	1,056	0,246	0,180	0,076
T_9	0,260	0,278	0,241	0,120	0,115
Average Exec. Time (%)	**1,034**	**0,949**	**0,327**	**0,140**	**0,143**

The graphic in Fig.3.14 shows that for request sets composed by up to 4 requests, the execution time remains far below 1s, but with the increase of requests per request set it is about 1 second. This analysis let envisage the need of an expanded computational campaign in order to know whether dedicated resolution techniques based on model have

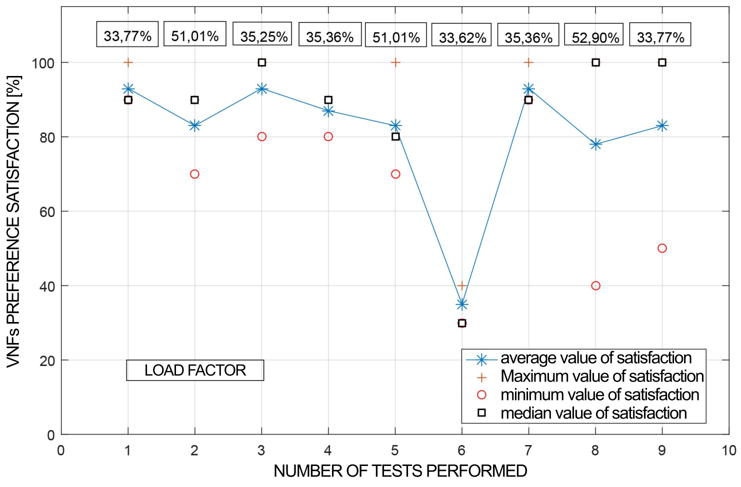


Figure 3.12: Average percentage of VNF Preference satisfaction for each performed test over a request set composed of 4 requests

to be included.

3.7 Conclusions and Further Improvements

In this research, a VNF Placement model has been formalized and evaluated in order to propose an optimized solution that considers both *tenants* and *rental* in a realistic scenario. By referring to a reference architecture that includes a VNF orchestrator (to manage VNFs lifecycle) and an SDN Controller (to deal with the traffic forwarding) a VNF Placement optimization model has been formulated. In order to favor the realistic vision, two kinds of users (i.e. Premium and Best Effort) with different granted priority, have been considered. The multi-objective mathematical model hierarchically performs profits maximization of both tenants and renters perspective: the former refers to service requests accomplishment and the latter to VNF deployment preferences satisfaction. The optimization model chooses the set of optimal locations for chained VNF instances according to the current characteristics of available computing resources (nodes) and network links, as well as several constraints that, among other goals, aim at favoring energy ef-

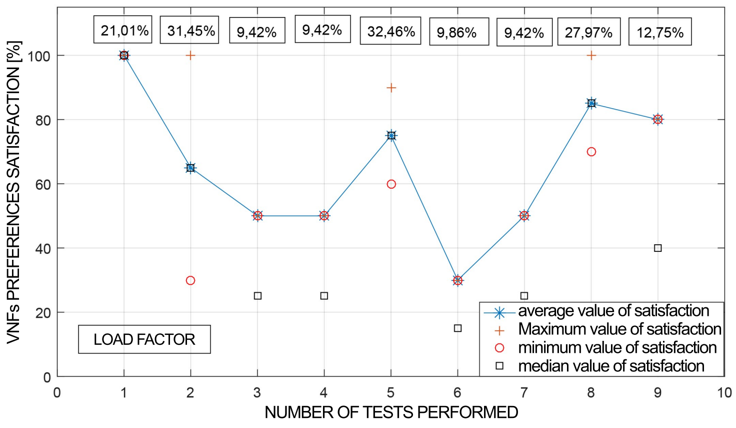


Figure 3.13: Average percentage of VNF Preference satisfaction for each performed test over a request set composed of 2 requests

ficiency and avoiding compute nodes that are guilty of high CO_2 emissions. Several tests have been performed in order to evaluate firstly the model correctness and then the performance in terms of Acceptance rate, VNF Preference satisfaction and Execution time. As reported in Section 3.6.3, the Acceptance Rate test reports an overall average acceptance rate percentage always greater than 60%. Moreover it highlights that, if convenient, the model choose to favor Premium users. The Placement optimization model also satisfies VNF preferences as it always reports an average preference satisfaction over the arrived request sets greater than 45% and over the accepted requests of request sets grater than 65%. The execution time over the whole test set, reports a value lower than the second but, with the increase of requests in the request set, this value tends to increase itself. This analysis is indicative of a computational campaign to be expanded and, whether this trend is confirmed, model-based dedicated resolution techniques could be developed.

However, the encouraging results obtained gave birth to new ongoing works on this topic. This work is still in its preliminary stage and urges to continue the experiments by considering, for instance, a greater number of randomly-generated request sets, to include a network simulator and/or to integrate the model in the reference architecture. The number of tests that has been performed to firstly check this model

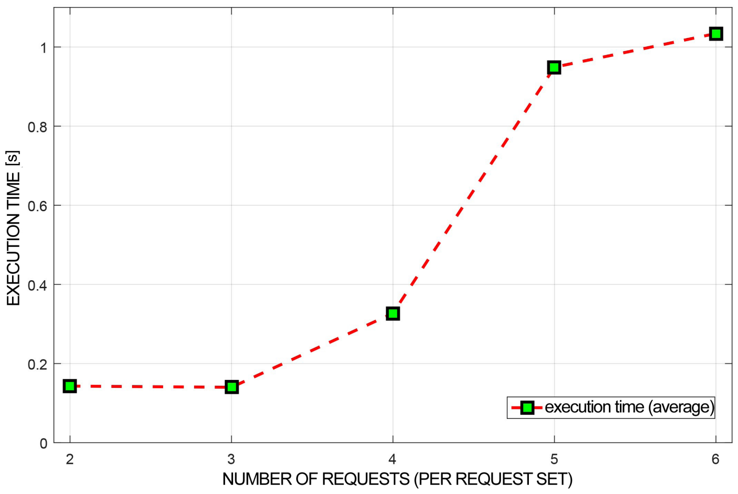


Figure 3.14: Average execution time [s] per request set

is tailored to a targeted evaluation (i.e. to analyze the correctness and the effectiveness of the model). A further consideration of a request set consisting of e.g. hundreds of requests, would allow emphasizing the effective maximization of the objective function. In the same way the network status variation could provoke some interesting changes by comparing VNF placement results among DCs. The system integration in the reference architecture could be one of the next steps as it would provide a real experience of model usage in order to realize its reliability even in the real "virtual" world.

Chapter 4

Conclusion

The emerging technologies that rely on virtualization (i.e. SDN, NFV, Cloud Computing) together with the increasing rise of new smart devices and services, let envisage a huge revolution in the way a Smart Home environment is currently conceived. It is expected a transition that, on the one hand, will allow users to exploit households objects/information to control/manage the house even from outside and, on the other hand, to provide high capacity services by means of external resources instantiated, for instance, in edge Data Centers. This scenario has been described extensively in the introductory Section (1.1) where a deep contextualization work has been addressed.

This thesis work has provided two contributions in the Smart Home field:

1. On the application perspective, this thesis has addressed the crucial issue of improving user awareness on household devices power consumptions in order to encourage a usage behavioral change that could promote energy savings. Sections 2.2 and 2.3 provides deep Background and Related Work analysis by pointing out differences among both addressed approaches and state of the art. Firstly a supervised classification algorithm for detecting and identifying consuming appliances has been investigated by using *ultra-low power* consumption data. The experimentation campaign has produced an average overall accuracy recognition of about 95%. Secondly a Non-Intrusive Load Monitoring (NILM) approach to

reduce the cost of attaching a single meter (i.e. smart plug) to each device has been proposed; the algorithm aims at recognizing the power consumption of a specific device by exploiting the low frequency whole-house consumption profile and *context information* (i.e. the user presence in the house and the hourly utilization of appliances). In comparison to basic inference algorithm (AFAMAP) [52] applied to the tracebase dataset [20], evaluation tests have underlined an average percentage increase of F-Measure parameter equal to 14% with peaks up to 26%.

2. On the technological infrastructure perspective, this work discusses one of the fundamental aspects in NFV orchestration that is VNF Placement. After a concise but accurate background (Section 2.2), an extensive and detailed Related Work (Section 3.3) has been provided in order to underline the novelty that the approach has proposed. This work has been accorded to a reference architecture which is mainly composed by a VNF-Orchestrator and an SDN controller which are in charge of managing respectively VNFs lifecycle and network forwarding for the service provision across multiple DCs. In this work the VNF placement problem has been solved by the formulation of a mathematical model that considers the maximization of the profit obtained hierarchically by the infrastructure providers (which aim to maximize profits related to request accomplishments) and the users (which aim to obtained the maximum placement satisfaction at minimum expense). In particular a novel differentiation between users (i.e. Premium, with greater privileges, and Best Effort, with standard rights) has been theorized. Preliminary tests have firstly proved the proper operation of the model and secondly shown encouraging results from both the points of view. Indeed, the testing campaign has shown that the average overall Acceptance Rate percentage is always greater than 60% and that, weather convenient, the model choose to favor Premium users. The Placement optimization model also satisfies VNF preferences as it always reports an average preference satisfaction over the arrived request sets greater than 45% and over the accepted requests of request sets greater than 65%.

Beyond the key contributions of this thesis I have foreseen a number of additional investigations and perspectives for future work that have already been detailed respectively in Sections 2.6 and 3.7, and thus here are only synthesized:

1. Both ILM and NILM approaches could be improved; for example, the first by extending the recognition to other devices typology with the aim of encourage generalization. The second could be provided with other different contextual information that, for example could come from a real sensor network. Moreover by starting from the common purpose of the above-mentioned Load Monitoring applications, it can be assumed that ILM and NILM, together with Smart Appliances (introduced in Section 2.1) could collaborate to fill the shortcomings of each approach. Thus the mutual integration of implemented approaches could represent a main further improvement.

2. The work that addresses the VNF Placement Optimization problem is still in its preliminary stage and, for this reason, is open to several improvements/modifications. Some of them are already on-going such as testing with a greater number of randomly-generated request sets, including a network simulator, integrating the model in the reference architecture.

Appendix A

Appendix

This Appendix is related to the VNF Placement optimization problem, previously presented in Chapter (3). Here I provide a list of the fundamental parameters that have been used to formally present the mathematical model (see Section 3.4).

R	Total request set
$R_p \subseteq R$	Premium request set
$R_b \subseteq R$	Best Effort request set
w_p	Weight assigned to Premium requests
w_b	Weight assigned to Best Effort requests
$p^r_{V^r_h i}$	Expressed preference of request r to deploy its h-VF on DC$_i$
W	Weight used to define the hierarchical objective function
D	DC set
H	Level set, i.e. maximum chain lenght
$H^r \subseteq H$	Level set for each request r
I	Incompatibility set between V^r_h and DC
L_h	Node set in levels h (for this problem it coincide with D)
o^r	Ingress node (source) for request r
d^r	Egress node (destination) for request r
l^r	Maximum latency for request r
c^r	Maximum cost for request r
b^r	Bandwidth consumption for request r
s^r	Container required for request r
f^r	Maximum CO_2 emissions for request r

$u_{V_h^r}$	Required resources for VF h of request r
$c_{V_h^r}$	Maximum cost for VF h of request r
$s_{V_h^r} = s^r$	Required container for VF h of request r
$f_{V_h^r} = f^r$	Maximum CO_2 for VF h of request r
U_i	Resource capacity for DC i
$\overline{U}$	DC maximum usage threshold
$\underline{U}$	DC minimum usage threshold
S_i	Container Availability for DC i
C_i	Cost per capacity unit for DC i
F_i	CO_2 emissions per capacity unit for DC i
b_{ij}	Maximum bandwidth on the path from DCi to DCj
l_{ij}	Latency on the path from DCi to DCj
E_{ij}	Existing edges from DCi to DCj in the real network
e_{ij}	Edge with minimum latency from DCi to DCj in the real network
L_{ij}	Minimum latency from DCi to DCj in the real network

Appendix B

Appendix

This Appendix is related to the VNF Placement optimization problem, previously presented in Chapter 3. Here detailed tables of obtained results are made explicit to favor the reproduction of test (see Section 3.6.3).

Table B.1: Number of accepted Premium requests for each test

Request Sets	R_6	R_5	R_4	R_3	R_2
Tests					
T_1	3	3	3	2	2
T_2	3	4	3	2	2
T_3	3	1	2	1	1
T_4	2	2	1	0	0
T_5	2	2	2	2	1
T_6	3	2	1	0	0
T_7	0	1	0	0	0
T_8	1	1	1	1	0
T_9	0	1	1	1	0
Average Acceptance (%)	31	37	39	33	33

Table B.1 shows the number of accepted Premium requests for each test; table B.2 shows the number of accepted Best Effort request for each test. By summing each cell of these two tables, Table 3.9 is obtained.

Table B.2: Number of accepted Best Effort requests for each test

Request Sets / **Tests**	R_6	R_5	R_4	R_3	R_2
T_1	1	1	0	0	0
T_2	1	1	1	0	0
T_3	1	1	1	1	0
T_4	2	2	2	1	1
T_5	2	2	2	1	1
T_6	1	1	1	1	1
T_7	3	2	3	1	1
T_8	4	4	3	2	2
T_9	3	3	2	1	1
Average Acceptance (%)	33	38	42	30	39

Appendix C

Publications

This research activity has led to several publications in international journals and conferences. These are summarized below.[1]

International Journals

1. F. Paganelli, **Francesca Paradiso**, S. Turchi, A. Luchetta, P. Castrogiovanni, D. Giuli. "Appliance Recognition in an OSGi-Based Home Energy Management Gateway", *International Journal of Distributed Sensor Networks*, vol. 11, no. 2 in press, 2015.
   [DOI: 10.1155/2015/937356] *3 citations*

2. **Francesca Paradiso**, F. Paganelli, D. Giuli, S. Capobianco. "Context-Based Energy Disaggregation in Smart Homes", *Future Internet*, vol. 8, no. 1, in press, 2016. (Special Issue: Ecosystemic Evolution Feeded by Smart Systems)
   [DOI:10.3390/8010004] *1 citation*

International Conferences and Workshops

1. **Francesca Paradiso**, F. Paganelli, A. Luchetta, D. Giuli, P. Castrogiovanni. "ANN-based Appliance Recognition from Low-frequency Energy Monitoring Data", in *IEEE 14th International Symposium and Workshops on World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Madrid (Spain), 2013, p.1-6. *10 citations*

[1]The author's bibliometric indices are the following: H-index = 2, total number of citations = 14 (source: Google Scholar on January, 2017).

Technical Reports

1. **Francesca Paradiso**, F. Paganelli. “State of the Art Analysis on Non-Intrusive Load Monitoring and Appliance Recognition techniques and NILM experimentation report”, TELECOM Italia S.p.A., CNIT, Technical Report, 2013.
2. **Francesca Paradiso**, F. Paganelli. “Analisi di tecniche di classificazione dei dispositivi e sperimentazione di una tecnica basata su ANN”, TELECOM Italia S.p.A., CNIT, Technical Report, 2013.

Bibliography

[1] M. K. Weldon, *The Future X Network: A Bell Labs Perspective.* CRC Press - Taylor & Francis Group, 2016.

[2] V. Pankakoski, *Experimental design for a next generation residential gateway.* PhD thesis, Aalto University, 2010.

[3] R. H. Weber and R. Weber, *Internet of Things*, vol. 12. Springer, 2010.

[4] C. Dixon, R. Mahajan, S. Agarwal, A. Brush, B. Lee, S. Saroiu, and P. Bahl, "An operating system for the home," in *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, pp. 337–352, 2012.

[5] N. McKeown, "Software-defined networking," *INFOCOM keynote talk*, vol. 17, no. 2, pp. 30–32, 2009.

[6] K. Kirkpatrick, "Software-defined networking," *Communications of the ACM*, vol. 56, no. 9, pp. 16–19, 2013.

[7] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236–262, 2015.

[8] ETSI, "Network function virtualization - management and orchestration gs nfv-man 001 v1.1.1," 2014. http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf [Online; accessed 10-January-2017].

[9] F. Paradiso, F. Paganelli, A. Luchetta, D. Giuli, and P. Castrogiovanni, "Ann-based appliance recognition from low-frequency energy monitoring data," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a*, pp. 1–6, IEEE, 2013.

[10] F. Paganelli, F. Paradiso, S. Turchi, A. Luchetta, P. Castrogiovanni, and D. Giuli, "Appliance recognition in an osgi-based home energy management gateway," *International Journal of Distributed Sensor Networks*, vol. 11, no. 2, 2015.

[11] F. Paradiso, F. Paganelli, D. Giuli, and S. Capobianco, "Context-based energy disaggregation in smart homes," *Future Internet*, vol. 8, no. 1, p. 4, 2016.

[12] M. Riveiro, R. Johansson, and A. Karlsson, "Modeling and analysis of energy data: state-of-the-art and practical results from an application scenario," no. HS-IKI-TR-11-002, p. 9, 2011.

[13] H. Bertoldi and Labanca, "Energy efficiency status report," 2012. https://ec.europa.eu/jrc/sites/default/files/energy-efficiency-status-report-2012.pdf [Online; accessed 10-January-2017].

[14] U. E. I. Administration, "Electric power annual 2008." http://www.eia.doe.gov/cneaf/electricity/epa/epaxlfilees1.pdf [Online; accessed 10-January-2017].

[15] S. Darby *et al.*, "The effectiveness of feedback on energy consumption," *A Review for DEFRA of the Literature on Metering, Billing and direct Displays*, vol. 486, p. 2006, 2006.

[16] K. Ehrhardt-Martinez, K. Donnelly, and J. Laitner, "Advanced metering initiatives and residential feedback programs," *Washington, DC*, 2010.

[17] B. Mills and J. Schleich, "Residential energy-efficient technology adoption, energy conservation, knowledge, and attitudes: An analysis of european countries," *Energy Policy*, vol. 49, pp. 616–628, 2012.

[18] M. Berges, E. Goldman, H. S. Matthews, and L. Soibelman, "Training load monitoring algorithms on highly sub-metered home electricity consumption data," *Tsinghua Science & Technology*, vol. 13, pp. 406–411, 2008.

[19] G. W. Hart, "Non-intrusive appliance load monitoring," *Proceedings of the IEEE*, vol. 80, no. 12, pp. 1870–1891, 1992.

[20] A. Reinhardt, D. Burkhardt, M. Zaheer, and R. Steinmetz, "Electric appliance classification based on distributed high resolution current sensing," in *Local Computer Networks Workshops (LCN Workshops), 2012 IEEE 37th Conference on*, pp. 999–1005, IEEE, 2012.

[21] A. Zoha, A. Gluhak, M. A. Imran, and S. Rajasegarar, "Non-intrusive load monitoring approaches for disaggregated energy sensing: A survey," *Sensors*, vol. 12, no. 12, pp. 16838–16866, 2012.

[22] A. G. Ruzzelli, C. Nicolas, A. Schoofs, and G. M. O'Hare, "Real-time recognition and profiling of appliances through a single electricity sensor," in *Sensor Mesh and Ad Hoc Communications and Networks (SECON), 2010 7th Annual IEEE Communications Society Conference on*, pp. 1–9, IEEE, 2010.

[23] "Zigbee alliance protocol," *IEEE Standard 802.15.4*, 2006.

[24] "Osgi alliance: Open services gateway initiative - https://www.osgi.org/," 1999-. https://www.osgi.org/ [Online; accessed 10-January-2017].

[25] A. Ridi, C. Gisler, and J. Hennebert, "Automatic identification of electrical appliances using smart plugs," in *Systems, Signal Processing and their Applications (WoSSPA), 2013 8th International Workshop on*, pp. 301–305, IEEE, 2013.

[26] S. Haykin, *Neural networks: a comprehensive foundation.* Prentice Hall PTR, 1994.

[27] A. Foglar and S. Plosz, "Appliance profile specification," 2008.

[28] M. Zeifman and K. Roth, "Nonintrusive appliance load monitoring: Review and outlook," *IEEE Transactions on Consumer Electronics*, pp. 76–84, 2011.

[29] J. Lifton, M. Feldmeier, Y. Ono, C. Lewis, and J. A. Paradiso, "A platform for ubiquitous sensor deployment in occupational and domestic environments," in *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*, pp. 119–127, IEEE, 2007.

[30] B. Plugwise, "Plugwise circle," *Online: http://www. plugwise. com/idplugtype-f/circle*, 2010.

[31] A. Reinhardt, D. Burkhardt, P. S. Mogre, M. Zaheer, and R. Steinmetz, "Smartmeter. kom: A low-cost wireless sensor for distributed power metering," in *Local Computer Networks (LCN), 2011 IEEE 36th Conference on*, pp. 1032–1039, IEEE, 2011.

[32] C. Gisler, A. Ridi, D. Zujferey, O. A. Khaled, and J. Hennebert, "Appliance consumption signature database and recognition test protocols," in *Systems, Signal Processing and their Applications (WoSSPA), 2013 8th International Workshop on*, pp. 336–341, IEEE, 2013.

[33] L. K. Norford and S. B. Leeb, "Non-intrusive electrical load monitoring in commercial buildings based on steady-state and transient load-detection algorithms," *Energy and Buildings*, vol. 24, no. 1, pp. 51–64, 1996.

[34] L. Farinaccio and R. Zmeureanu, "Using a pattern recognition approach to disaggregate the total electricity consumption in a house into the major end-uses," *Energy and Buildings*, vol. 30, no. 3, pp. 245–259, 1999.

[35] M. Baranski and J. Voss, "Nonintrusive appliance load monitoring based on an optical sensor," in *Power Tech Conference Proceedings, 2003 IEEE Bologna*, vol. 4, pp. 8–pp, IEEE, 2003.

[36] M. Baranski and J. Voss, "Genetic algorithm for pattern detection in nialm systems," in *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, vol. 4, pp. 3462–3468, IEEE, 2004.

[37] S. Drenker and A. Kader, "Nonintrusive monitoring of electric loads," *Computer Applications in Power, IEEE*, vol. 12, no. 4, pp. 47–51, 1999.

[38] H.-H. Chang, P.-C. Chien, L.-S. Lin, and N. Chen, "Feature extraction of non-intrusive load-monitoring system using genetic algorithm in smart meters," in *e-Business Engineering (ICEBE), 2011 IEEE 8th International Conference on*, pp. 299–304, IEEE, 2011.

[39] M. B. Figueiredo, A. De Almeida, and B. Ribeiro, "An experimental study on electrical signature identification of non-intrusive load monitoring (nilm) systems," in *Adaptive and Natural Computing Algorithms*, pp. 31–40, Springer, 2011.

[40] C. Laughman, K. Lee, R. Cox, S. Shaw, S. Leeb, L. Norford, and P. Armstrong, "Power signature analysis," *Power and Energy Magazine, IEEE*, vol. 1, no. 2, pp. 56–63, 2003.

[41] J. Li, S. West, and G. Platt, "Power decomposition based on svm regression," in *Modelling, Identification & Control (ICMIC), 2012 Proceedings of International Conference on*, pp. 1195–1199, IEEE, 2012.

[42] H.-H. Chang, C.-L. Lin, and J.-K. Lee, "Load identification in non intrusive load monitoring using steady-state and turn-on transient energy algorithms," in *Computer supported cooperative work in design (cscwd), 2010 14th international conference on*, pp. 27–32, IEEE, 2010.

[43] T. Onoda, G. Rätsch, and K.-R. Müller, "Applying support vector machines and boosting to a non-intrusive monitoring system for household electric appliances with inverters," 2000.

[44] S. N. Patel, T. Robertson, J. A. Kientz, M. S. Reynolds, and G. D. Abowd, *At the flick of a switch: Detecting and classifying unique electrical events on the residential power line (nominated for the best paper award).* Springer, 2007.

[45] C. M. Bishop *et al.*, *Pattern recognition and machine learning*, vol. 4. springer New York, 2006.

[46] T. B. Fomby, "K-nearest neighbors algorithm: Prediction and classification," 2008.

[47] D. Srinivasan, W. Ng, and A. Liew, "Neural-network-based signature recognition for harmonic source identification," *Power Delivery, IEEE Transactions on*, vol. 21, no. 1, pp. 398–405, 2006.

[48] F. Chen, J. Dai, B. Wang, S. Sahu, M. Naphade, and C.-T. Lu, "Activity analysis based on low sample rate smart meters," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 240–248, ACM, 2011.

[49] O. Kramer, T. Klingenberg, M. Sonnenschein, and O. Wilken, "Non-intrusive appliance load monitoring with bagging classifiers," *Logic Journal of IGPL*, p. jzv016, 2015.

[50] H. Goncalves, A. Ocneanu, M. Berges, and R. Fan, "Unsupervised disaggregation of appliances using aggregated consumption data," in *The 1st KDD Workshop on Data Mining Applications in Sustainability (SustKDD)*, 2011.

[51] T. Zia, D. Bruckner, and A. Zaidi, "A hidden markov model based procedure for identifying household electric loads," in *IECON 2011-37th Annual Conference on IEEE Industrial Electronics Society*, pp. 3218–3223, IEEE, 2011.

[52] J. Z. Kolter and T. Jaakkola, "Approximate inference in additive factorial hmms with application to energy disaggregation," in *International conference on artificial intelligence and statistics*, pp. 1472–1482, 2012.

[53] D. Egarter, V. P. Bhuvana, and W. Elmenreich, "Paldi: Online load disaggregation via particle filtering," *Instrumentation and Measurement, IEEE Transactions on*, vol. 64, no. 2, pp. 467–477, 2015.

[54] A. Reinhardt, P. Baumann, D. Burgstahler, M. Hollick, H. Chonov, M. Werner, and R. Steinmetz, "On the accuracy of appliance identification based on distributed load metering data," in *Sustainable Internet and ICT for Sustainability (SustainIT), 2012*, pp. 1–9, IEEE, 2012.

[55] "The energy@home technical team - energy@home: a user-centric energy management system," 2010. http://www.energy-home.it/ [Online; accessed 10-January-2017].

[56] H. Kim, M. Marwah, M. F. Arlitt, G. Lyon, and J. Han, "Unsupervised disaggregation of low frequency power measurements.," in *SDM*, vol. 11, pp. 747–758, SIAM, 2011.

[57] M. Shahriar, A. Rahman, D. Smith, *et al.*, "Applying context in appliance load identification," in *Natural Computation (ICNC), 2013 Ninth International Conference on*, pp. 900–905, IEEE, 2013.

[58] K. Anderson, A. Ocneanu, D. Benitez, D. Carlson, A. Rowe, and M. Berges, "Blued: A fully labeled public dataset for event-based non-intrusive load monitoring research," in *Proceedings of the 2nd KDD workshop on data mining applications in sustainability (SustKDD)*, pp. 1–5, 2012.

[59] J. Z. Kolter and M. J. Johnson, "Redd: A public data set for energy disaggregation research," in *Workshop on Data Mining Applications in Sustainability (SIGKDD), San Diego, CA*, vol. 25, pp. 59–62, Citeseer, 2011.

[60] S. Makonin, F. Popowich, L. Bartram, B. Gill, and I. V. Bajic, "Ampds: A public dataset for load disaggregation and eco-feedback research," in *Electrical Power & Energy Conference (EPEC), 2013 IEEE*, pp. 1–6, IEEE, 2013.

[61] L. Pereira and N. J. Nunes, "Semi-automatic labeling for public non-intrusive load monitoring datasets," in *Proceedings of the 4th IFIP/IEEE Conference on Sustainable Internet and ICT for Sustainability*, 2015.

[62] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[63] D. E. Rumelhart and J. L. McClelland, "Parallel distributed processing: Explorations in the microstructure of cognition, volume 1: Foundations," 1986.

[64] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the marquardt algorithm," *Neural Networks, IEEE Transactions on*, vol. 5, no. 6, pp. 989–993, 1994.

[65] R. Gençay and M. Qi, "Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging," *Neural Networks, IEEE Transactions on*, vol. 12, no. 4, pp. 726–734, 2001.

[66] "The energy@home technical team - http://www.energy-home.it/."

[67] "Osgi alliance - osgi alliance specifications," *Web pages at http://www. osgi. org/Specifications*, 2013.

[68] C. Busemann, V. Gazis, R. Gold, P. Kikiras, A. Leonardi, J. Mirkovic, M. Walther, and H. Ziekow, "Integrating sensor networks for energy

monitoring with service-oriented architectures," *International Journal of Distributed Sensor Networks*, vol. 2013, 2013.

[69] "Ogema - open gateway energy management alliance." http://www.ogema.org/ogema/index.html [Online; accessed 10-January-2017].

[70] N. Series, "Challenges in building service-oriented applications for osgi," *IEEE Communications Magazine*, p. 145, 2004.

[71] A. Ranalli and C. Borean, "Energy@ home leveraging zigbee to enable smart grid in residential environment," in *Smart Grid Security*, pp. 132–149, Springer, 2013.

[72] "The energy@home technical team - energy@home technical specification v.0.95," 2012. http://www.energy-home.it/Documents/Forms/AllItems.aspx [Online; accessed 10-January-2017].

[73] Z. Alliance, "Zigbee network device gateway specification," 2011. https://www.zigbee.org/Standards/ZigBeeNetworkDevices/download.aspx [Online; accessed 10-January-2017].

[74] "Jemma - java energy management application framework." http://ismb.github.io/jemma/ [Online; accessed 10-January-2017].

[75] "Neuroph - java neural network framework neuroph." http://neuroph.sourceforge.net/index.html [Online; accessed 10-January-2017].

[76] "The energy@home technical team - energy@home use case v.1.2," 2010. http://www.energy-home.it/Documents/Forms/AllItems.aspx [Online; accessed 10-January-2017].

[77] Z. Alliance, "Zigbee cluster library specification," *ZigBee Document 075123r01*, 2007.

[78] Z. Ghahramani and M. I. Jordan, "Factorial hidden markov models," *Machine learning*, vol. 29, no. 2-3, pp. 245–273, 1997.

[79] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," 2001.

[80] I. D. Melamed, R. Green, and J. P. Turian, "Precision and recall of machine translation," in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: companion volume of the Proceedings of HLT-NAACL 2003–short papers-Volume 2*, pp. 61–63, Association for Computational Linguistics, 2003.

[81] C. Fraley and A. E. Raftery, "How many clusters? which clustering method? answers via model-based cluster analysis," *The computer journal*, vol. 41, no. 8, pp. 578–588, 1998.

[82] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, pp. 281–297, Oakland, CA, USA., 1967.

[83] D. Reynolds, "Gaussian mixture models," in *Encyclopedia of Biometrics*, pp. 659–663, Springer, 2009.

[84] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 7, pp. 881–892, 2002.

[85] F. Qian, G.-m. Hu, and X.-m. Yao, "Semi-supervised internet network traffic classification using a gaussian mixture model," *AEU-International Journal of Electronics and Communications*, vol. 62, no. 7, pp. 557–564, 2008.

[86] A. Doulamis, N. Doulamis, and S. D. Kollias, "On-line retrainable neural networks: improving the performance of neural networks in image analysis problems," *Neural Networks, IEEE Transactions on*, vol. 11, no. 1, pp. 137–155, 2000.

[87] ETSI, "Network function virtualization - introductory white paper," 2012. https://portal.etsi.org/nfv/nfv_white_paper.pdf [Online; accessed 10-January-2017].

[88] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 90–97, 2015.

[89] W. John, K. Pentikousis, G. Agapiou, E. Jacob, M. Kind, A. Manzalini, F. Risso, D. Staessens, R. Steinert, and C. Meirosu, "Research directions in network service chaining," in *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*, pp. 1–7, IEEE, 2013.

[90] F. Paganelli, M. Ulema, and B. Martini, "Context-aware service composition and delivery in ngsons over sdn," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 97–105, 2014.

[91] ETSI, "Network function virtualization; ecosystrem; report on sdn usage in nfv architectural framework v1.1.1," 2015. http://www.etsi.org/deliver/etsi_gs/NFV-EVE/001_099/005/01.01.01_60/gs_NFV-EVE005v010101p.pdf [Online; accessed 10-January-2017].

[92] M. Ghaznavi, A. Khan, N. Shahriar, K. Alsubhi, R. Ahmed, and R. Boutaba, "Elastic virtual network function placement," in *Cloud Networking (CloudNet), 2015 IEEE 4th International Conference on*, pp. 255–260, IEEE, 2015.

[93] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and S. Davy, "Design and evaluation of algorithms for mapping and scheduling of virtual network functions," in *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*, pp. 1–9, IEEE, 2015.

[94] M. Xia, M. Shirazipour, Y. Zhang, H. Green, and A. Takacs, "Network function placement for nfv chaining in packet/optical datacenters," *Journal of Lightwave Technology*, vol. 33, no. 8, pp. 1565–1570, 2015.

[95] B. Addis, D. Belabed, M. Bouet, and S. Secci, "Virtual network functions placement and routing optimization," in *Cloud Networking (CloudNet), 2015 IEEE 4th International Conference on*, pp. 171–177, IEEE, 2015.

[96] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*, pp. 7–13, IEEE, 2014.

[97] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions," in *Network and Service Management (CNSM), 2015 11th International Conference on*, pp. 50–56, IEEE, 2015.

[98] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "Piecing together the nfv provisioning puzzle: Efficient placement and chaining of virtual network functions," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 98–106, IEEE, 2015.

[99] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[100] K. Greene, "10 breakthrough technologies: Software-defined networking mit technol.," 2009. http://www2.technologyreview.com/article/412194/tr10-software-defined-networking/ [Online; accessed 10-January-2017].

[101] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[102] W. Stalling, *Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud*. Addison-Wesley Professional, 2015.

[103] "Floodlight." http://www.projectfloodlight.org/floodlight/ [Online; accessed 10-January-2017].

[104] "Pox." http://www.noxrepo.org/pox/about-pox/ [Online; accessed 10-January-2017].

[105] "Ryu." https://osrg.github.io/ryu/ [Online; accessed 10-January-2017].

[106] "Opendaylight." https://www.opendaylight.org/ [Online; accessed 10-January-2017].

[107] "Onos." http://onosproject.org/ [Online; accessed 10-January-2017].

[108] "Openstack: Open source cloud computing software." https://www.openstack.org/ [Online; accessed 10-January-2017].

[109] "Sdx central." https://www.sdxcentral.com/sdn/definitions/inside-sdn-architecture/ [Online; accessed 10-January-2017].

[110] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[111] "Open network foundation (onf) - https://www.opennetworking.org/," 2011. https://www.opennetworking.org/ [Online; accessed 10-January-2017].

[112] ETSI, "European telecommunications standards institute, industry specification groups, network function virtualization (etsi isg nfv)," 2012. http://www.etsi.org/technologies-clusters/technologies/nfv [Online; accessed 10-January-2017].

[113] ETSI, "Etsi group specifications on network function virtualization. 1st phase documents," 2015. http://docbox.etsi.org/ISG/NFV/Open/Published/ [Online; accessed 10-January-2017].

[114] H. Hawilo, A. Shami, M. Mirahmadi, and R. Asal, "Nfv: state of the art, challenges, and implementation in next generation mobile networks (vepc)," *IEEE Network*, vol. 28, no. 6, pp. 18–26, 2014.

[115] C. Cui *et al.*, "Network functions virtualisation: Network operator perspectives on industry progress. white paper no. 3, issue 1," in *SDN and OpenFlow World Congress, Dusseldorf-Germany*, 2014.

[116] IETF, "Internet engineering task force - service function chaining - working group," 2016. https://datatracker.ietf.org/wg/sfc/documents/ [Online; accessed 10-January-2017].

[117] D. Bhamare, R. Jain, M. Samaka, and A. Erbad, "A survey on service function chaining," *Journal of Network and Computer Applications*, vol. 75, pp. 138–155, 2016.

[118] IETF, "Problem statement of network functions virtualization model," 2013. https://tools.ietf.org/html/draft-xjz-nfv-model-problem-statement-00 [Online; accessed 10-January-2017].

[119] IETF, "Service function chaining problem statement," 2014. https://tools.ietf.org/html/draft-ietf-sfc-problem-statement-05 [Online; accessed 10-January-2017].

[120] IETF, "Service function chaining (sfc) use cases," 2014. https://tools.ietf.org/html/draft-liu-sfc-use-cases-00 [Online; accessed 10-January-2017].

[121] IETF, "Service function chaining: Framework and architecture," 2014. https://datatracker.ietf.org/doc/draft-boucadair-sfc-framework/ [Online; accessed 10-January-2017].

[122] X. Li and C. Qian, "A survey of network function placement," in *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pp. 948–953, IEEE, 2016.

[123] M. Bouet, J. Leguay, T. Combe, and V. Conan, "Cost-based placement of vdpi functions in nfv infrastructures," *International Journal of Network Management*, vol. 25, no. 6, pp. 490–506, 2015.

[124] J. F. Riera, J. Batallŷ, J. Bonnet, M. Dŷas, M. McGrath, G. Petralia, F. Liberati, A. Giuseppi, A. Pietrabissa, A. Ceselli, *et al.*, "Tenor: Steps towards an orchestration platform for multi-pop nfv deployment," in *NetSoft Conference and Workshops (NetSoft), 2016 IEEE*, pp. 243–250, IEEE, 2016.

[125] Y. Zhang, N. Beheshti, L. Beliveau, G. Lefebvre, R. Manghirmalani, R. Mishra, R. Patneyt, M. Shirazipour, R. Subrahmaniam, C. Truchan, *et al.*, "Steering: A software-defined networking for inline service chaining," in *2013 21st IEEE International Conference on Network Protocols (ICNP)*, pp. 1–10, IEEE, 2013.

[126] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "Simple-fying middlebox policy enforcement using sdn," *ACM SIGCOMM computer communication review*, vol. 43, no. 4, pp. 27–38, 2013.

[127] Z. Cao, M. Kodialam, and T. Lakshman, "Traffic steering in software defined networks: planning and online routing," in *ACM SIGCOMM Computer Communication Review*, vol. 44, pp. 65–70, ACM, 2014.

[128] J. Liu, Y. Li, Y. Zhang, L. Su, and D. Jin, "Improve service chaining performance with optimized middlebox placement," *IEEE Transactions on Services Computing*, vol. PP, Issue:99, 2015.

[129] D. Dietrich, A. Abujoda, and P. Papadimitriou, "Network service embedding across multiple providers with nestor," in *IFIP Networking Conference (IFIP Networking), 2015*, pp. 1–9, IEEE, 2015.

[130] D. Joseph and I. Stoica, "Modeling middleboxes," *IEEE network*, vol. 22, no. 5, pp. 20–25, 2008.

[131] A. Gember, A. Krishnamurthy, S. S. John, R. Grandl, X. Gao, A. Anand, T. Benson, A. Akella, and V. Sekar, "Stratos: A network-aware orchestration layer for middleboxes in the cloud," tech. rep., Technical Report, 2013.

[132] P. Bellavista, F. Callegati, W. Cerroni, C. Contoli, A. Corradi, L. Foschini, A. Pernafini, and G. Santandrea, "Virtual network function embedding in real cloud environments," *Computer Networks*, vol. 93, pp. 506–517, 2015.

[133] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 1346–1354, IEEE, 2015.

[134] J. Zhu, "Benchmarking virtual network mapping algorithms," *Masters Theses 1896 - February2014*, 2012. http://scholarworks.umass.edu/theses/970/ [Online; accessed 10-January-2017].

[135] M. Savi, M. Tornatore, and G. Verticale, "Impact of processing costs on service chain placement in network functions virtualization," in *Network Function Virtualization and Software Defined Network (NFV-SDN), 2015 IEEE Conference on*, pp. 191–197, IEEE, 2015.

[136] "Docker: Build, ship, run." https://www.docker.com/ [Online; accessed 10-January-2017].

[137] B. Martini, F. Paganelli, P. Cappanera, S. Turchi, and P. Castoldi, "Latency-aware composition of virtual functions in 5g," in *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*, pp. 1–6, IEEE, 2015.

[138] A. Betker, C. Gerlach, R. Hülsermann, M. Jäger, M. Barry, S. Bodamer, J. Späth, C. Gauger, and M. Köhn, "Reference transport network scenarios," *MultiTeraNet Report, July*, 2003.