UQAC
Université du Québec
à Chicoutimi

Architectural Model for Collaboration in The Internet of Things; A Fog Computing Based Approach

THÈSE PRÉSENTÉE À

L'UNIVERSITÉ DU QUÉBEC À CHICOUTIMI

COMME EXIGENCE PARTIELLE DE LA THÈSE DU

DOCTORAT EN SCIENCE ET TECHNOLOGIES DE L'INFORMATION

Par

Jabril ABDELAZIZ

2018

RÉSUMÉ

Dans les dernières années, les avantages du Cloud Computing l'ont mis au cœur des architectures proposées pour l'Internet des Objets (IoT). L'infrastructure homogène, prédictible et performante a fait du Cloud une solution adéquate pour le traitement et l'analyse des données en provenance des objets de l'IoT. Cependant, les avantages de l'utilisation du Cloud se révèlent problématiques pour les systèmes IoT sensibles au temps de latence, et qui exigent la distribution géographique, la prise en compte de l'environnement local ainsi que la mobilité des objets. Le Fog Computing est un nouveau concept visant l'extension du Cloud vers la périphérie de l'IoT. Ainsi, il envisage une couche de nœuds (Fogs) permettant de fournir aux objets connectés un support à la gestion de la communication, à la persistance des données et à la gestion d'accès.

Ce projet de recherche est motivé par les opportunités prometteuses du concept du Fog computing. Il anticipe ces opportunités et vise à proposer une architecture fédératrice, jusqu'à présent inexistante, pour la collaboration dans le Fog.

De ce fait, dans cette thèse, nous tirons parti de l'idée derrière ce nouveau concept afin de proposer une architecture à cette fin. Cette architecture consiste en un modèle référentiel qui promeut à la fois une grande abstraction dans la conception des applications, ainsi que la facilité et l'efficacité dans le développement et le déploiement au niveau des nœuds de la couche du Fog. En effet, pour renforcer ces nœuds avec des services dynamiques, nous proposons des moyens formels pour la génération dynamique de nouveaux services à travers des opérations d'agrégations, de compositions ou de transformations. En conséquence, les nœuds du Fog deviennent un nid où les objets connectés peuvent interagir et collaborer à travers des mécanismes expressifs de définition et d'abstraction d'objets, des analyses de données et des services.

ABSTRACT

Through sensors, actuators and other Internet-connected devices, applications and services are becoming able to perceive and react on the real world. Seamlessly integrating people, and devices is no longer a futuristic idea. Converging the physical world with the human-made realm into one network is rather a present and promising approach called The Internet of Things (IoT).

A closer look at the phenomenon of IoT reveals many problems. The current trends are focusing on Cloud-centric approaches to deal with the heterogeneity and the scale of this network. The blessing of the Cloud computing becomes, however, a burden on latency-sensitive applications, which require processing and storage mechanisms in their proximity to meet low-latency, location and better context-awareness requirements. In addition to mobility support and high geographical distribution requirements. Fog computing is a new concept that focuses on extending the Cloud paradigm to the edge of the Internet of Things, via providing communication, computing, and access management support.

This research project foresees and is driven by the promising opportunities of the concept behind Fog computing. In this thesis, we leverage this new concept by delivering a Collaboration Architecture for the Fog computing. This architecture constitutes a referential model to better design and to implement Fog platforms. It powers the freedom of abstraction to make development and deployment at the Fog nodes easier and more efficient. Moreover, it provides a nest where IoT-connected objects can interact and collaborate. To this end, we introduce expressive mechanisms to define and abstract objects, data analytics, and services. To leverage Fog nodes with dynamic services and service-based collaboration, we propose the concept of Operation: a formal way to dynamically generate new services through mechanisms such as aggregation, composition, and transformation. Finally, we deliver a comprehensive study and a collaboration-oriented access control model for the proposed architecture.

ACKNOWLEDGMENTS

I would like to acknowledge the support of the people who made this thesis possible. I would like to express my gratitude to my advisors Professor Hamid Mcheick whose maddening attention to detail drove me to finally learn to punctuate prose, and to Professor Mehdi Adda whose care and insights were sometimes all that kept me going. I would also like to thank my thesis committee members for providing insightful feedback and constructive suggestions to improve the quality of this thesis.

Many thanks go to my friends who have supported me through the ups and downs of my journey. To Tarik for being behind the publication of my first article; To Hicham for sharing the many nights of work and for the countless lectures on the mechanics of ice. To Amira, Anouck, Mahdiyeh, Mustapha, Redouane and Zahira for the support, for periodically disturbing the loneliness of my office with words of encouragement, and for the tasty homemade food.

Finally, I am eternally grateful to my parents and my family for their unconditional love and full support, despite the long years of my absence.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter I
INTRODUCTION

The pyramids of Egypt, the Panama Canal or sending a man to space are great achievements that could only be accomplished through collaboration. In modern life, problems are getting more and more complex. Like ancient achievements, these problems require knowledge and expertise from a wide range of disciplines and domains. In addition, the amount of data generated and needed for such activities is immense, and might not be managed by any individual organizations. All of these factors have made global collaborations become increasingly important in modern scientific, industrial and daily life activities. The rise of the Internet led us to change our perspective on ways of approaching our activities and means of interacting with both the digital and the real world. Indeed, dealing with distributed activities on a large scale has given rise to many modes of collaboration.

Since the nineties, there were two levels to approaching distributed systems [112]. The first level addresses the human-oriented level principally via the Web. Whereas the second level focuses on the low-level interoperability between systems through distributed objects and middleware-powered technologies - e.g., CORBA [85] and DCOM [79]. By the dawn of the millennium, there was an explosion of platforms and middleware that exploited the emerging Peer-to-Peer and Grid technologies [112]. These technologies aim at supporting collaboration between heterogonous and distributed applications, and at enabling users to contribute in more active ways. Hence, Collaborative computing can be defined as "*a fertile mélange of technologies and techniques which facilitate people working together via computer-assisted means*" [97]. It arose from the early *groupware* [23] systems that were intended to bridge geographic distances between people engaged in a common task and that provide an interface to a shared environment. Collaborative computing aims not only at bridging distances, it adds capabilities that enhance and assist in the work process [41] (e.g., smart boards).

Contemporary systems are moving from static desktops to dynamic, mobile and ubiquitous models; from discrete nodes (i.e., stand-alone machines) to embedded architectures (e.g., embedded sensors); and from autonomous nodes to pools of interacting nodes that provide services (i.e., Fogs, see chapter II). This work fosters collaboration between Internet-connected objects in the Internet of Things (IoT). The term "Collaboration" in this work transcends facilitating and assisting cooperation between people to provide a framework where Internet-connected objects can identify, retrieve, and exploit the capabilities of each other. Next section emphases on the reason behind our interest in bringing collaboration into the IoT and the main challenges this work is dealing with.

1. Motivation and Problem Description

Humanity is passing through an age with almost limitless potential. In minutes or even seconds, information and ideas can reach and can be reached by almost any person across the globe. Likewise, in the Internet of Things, devices are used to collect data from their environment. Nonetheless, the real value of such data comes only through processing and analysis. As shown in the next Figure 1, the process of exploiting the data begins with inferring information.

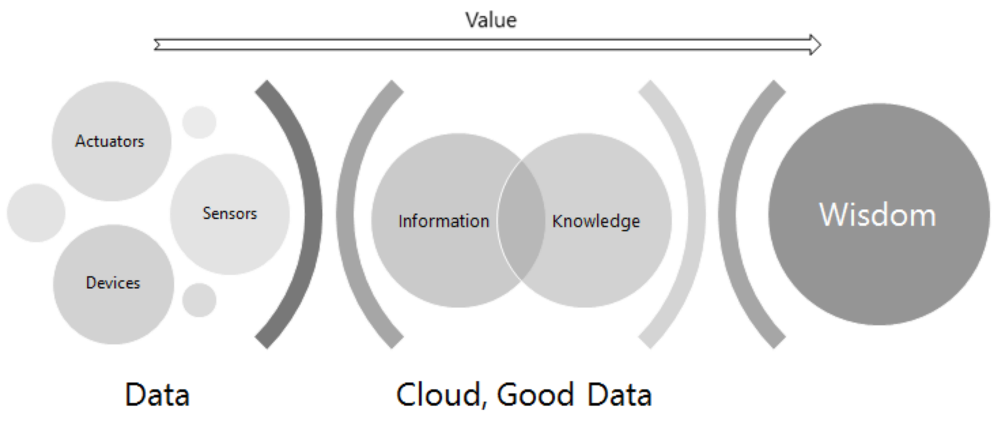

Figure 1. The process of exploiting data and generating wisdom

The structuration of information into knowledge will lead to more optimized systems with higher performance, better user experiences and more efficient energy consumption [35]. The IoT will provide us with new insights into solving many problems, wise ways to exploit our environment, and better solutions toward generating a timeless knowledge, that is wisdom.

Although we tend to think of IoT as a way of connecting singular devices, the most interesting applications are not coming out from individual devices, but rather from how they work with each other in a collaborative manner. Hence, prior to using data for decision-making, the main challenge in this regard is to provide Internet connected-objects with suitable mechanisms to discover the functionalities of each other according to their capabilities, their location and the information and services that they can provide. In addition to developing technologies and protocols to allow the use of such resources efficiently, securely, and with minimal human intervention [63].

The current architectural model and the trends in IoT are toward Cloud-centred architectures. Processing and analyzing the data coming from IoT-connected objects occur solely in the Cloud, therefore, raising challenges related to the network bandwidth, the communication latency, and to the ability to access local information. We foresee the aggregation of sensing activities and the distribution of collaborative interaction between connected-objects at the edge of the IoT network as opportunities to tackle the aforementioned challenges. Therefore, we propose to adopt and extend the idea of Fog computing to embrace a distributed and collaborative computing model for the IoT. Such model will help using resources (i.e., network and the device resources) more efficiently, in addition to supporting more sophisticated application scenarios. Bringing such idea to life requires widespread distribution, high mobility support, low latency and real-time services. In addition to taking into account the constraint nature of edge devices – i.e., in term of processing, storage, memory, and other resources. Atop of this challenges, research into Fog computing concept is still in its early stages. There is no standard or precise definition and an architectural model is yet to be provided. Hence the need to providing an architectural framework to ease the development and deployment of IoT solution at the level of Fogs. While such framework ought to be domain agnostic, the full or partial instantiation of its components must be easy.

The reader will find more details on key concepts used in this document in the second chapter. In addition to further emphases on the challenges that we are tackling in our research project regarding collaboration in a Fog-based architecture.

2. CONTRIBUTIONS

The homogeneity, the efficiency, and the many other advantages offered by the Cloud computing infrastructure have made it a reliable solution at the core of the Internet of Things. Relying on the Cloud to deal with the growing IoT applications and services has been a valid choice, however, this cloud-centric approach proved to be limited in its application domains. This thesis proposes an extended vision of the Fog computing concept. Indeed, since its inception, the Fog computing has been perceived as a simple extension of the Cloud, capable of offering computational, storage and networking capabilities between the Cloud and end devices [18].

In this work, we made a step forward in the context of Fog computing paradigm toward embracing collaboration between IoT's connected-objects. The first major contribution of this work consists in an IoT architecture model called *CoFog*. We focused on delivering an architecture that, on the one hand, leverages IoT objects with services, and on the other hand augments data representation and consumption with local analytics at the Fog nodes level. Up to the date of writing this document, this work is being the first proposition of its kind in the domain of Fog computing (the second being the OpenFog [87] reference architecture proposed lately in 2017). In addition, to being the first to propose an object-based collaborative model for the Fog, which is the second major contribution of this thesis. Details on this architecture are presented in Chapter IV.

We foresee the aggregation of services at the Fog level as an engine toward more sophisticated IoT applications. Hence, the *CoFog* architecture provides a service layer that provides means to define and dynamically create services based on predefined templates. These services can be aggregated through formal mechanisms called *operations*. The introduction of the *operation* concept constitutes the third contribution of this thesis. An operation represents a relation between a given collaboration request and services that may be used to answer this request. Through mathematical formulas, a service (or more) can be composed, transformed or aggregated to dynamically create new services. In the scope of this thesis, there are two types of operations: conservative and non-conservative operations. The second type results in new kind of data –e.g., the result of dividing two integers may be a real number, we say that division is a non-conservative operation. The first type of operations, however, conserves the same kind of data –e.g., the addition of two integers always results in an integer. This way, depending on the use case, a conservative operation can be applied recursively to obtain the desired results. Further details on services aggregation are the subject of the fifth chapter.

Although providing and aggregating services are important mechanisms, the *CoFog* architecture could not be complete without the ability to discover and retrieve such services. Therefore, the fourth contribution of this thesis resides in its data sharing model. This model provides IoT-connected objects and applications with the capacity to discover services that are being offered in other Fog nodes. In case a Fog node does not and cannot provide the requested services, it forwards the request to the neighbouring nodes that are listed in its *whitelist*. This way, any Fog node in the vicinity can be used to satisfy the request.

One of the most critical aspects of collaboration is security, especially access control. Our fifth contribution includes a comprehensive study of access control for Fog computing and in particular for the collaboration in the *CoFog* architecture. Two of the major access control models has been studied and expanded to incorporate the collaboration aspects –i.e., the Role-based access control and the Attribute-based access control models. Via the evaluation of both models, we demonstrated that the extended attribute-based model is more suitable for collaboration, mainly due to its fine-grained access rules and its support for context information representation.

After this overall presentation of the major contributions of this thesis, the next section presents the research methodology that we followed to achieve these goals.

3. RESEARCH METHODOLOGY

The context of our research project is constantly evolving with rapid research inputs. To keep pace with the changing research domain, we have adopted a learning through the act of building methodology. Such research process iterates between phases rather than flowing in a waterfall fashion from one phase into the next Figure 2.

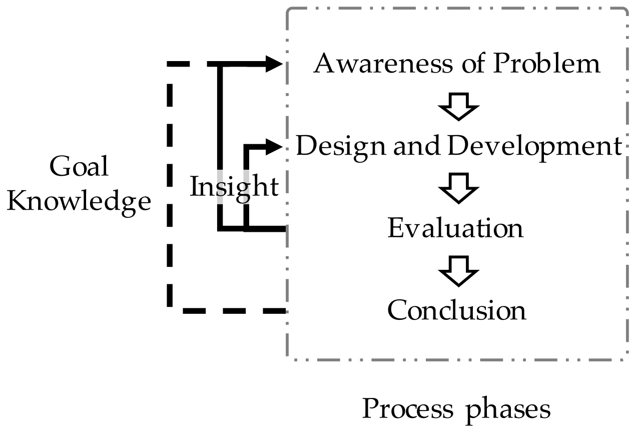

Figure 2. Research process phases (adapted from [67])

The main advantage of such methodology is that it provides an insight into the development and the outcomes at early stages of the research process. Hence, issues and flaws can be corrected as soon as they arise.

The first phase of this research project aimed at gaining deeper awareness on the Internet of Things and its related problems introduced in previous sections. The first stage, in this two stages phase, allowed us to focus on potential collaboration-enabling technologies coming from related domains. It spanned the use of Cloud computing, Semantic Web Services and the use of Peer-to-Peer architecture in the IoT [30,62,63,92,112,113]. In addition to projects directly addressing different perspectives of the IoT vision such as the Coordination and Support Action for Global RFID-related Activities and Standardization [54], Internet of Things Architecture (IoT-A) [14], and The Internet Connected Objects for Reconfigurable Ecosystems (iCore) [77]. At the end of this first stage, we were fully aware of how such traditional technics and technologies have been adopted and adapted mainly to deal with data storage and data processing in the IoT. The second stage took us deep into current research propositions focusing on the Edge-computing and the Fog computing principles, with emphases on propositions such as the Edge-Centric Computing [39] and the Cloudlets [102]. This phase sharpened our perspective on the key problems toward the real collaboration we envision in the Fog Computing. The literature review has been subject to a scientific publication [1] that appeared on the *International Workshop on Healthcare systems and Internet of Things for Humanity (eHealthForHumanity'2015)* held in Conjunction with the *6th International Conference on E-Technologies MCETECH'2015* published by *Springer*.

The second phase was dedicated to providing the referential architecture model. Findings from the first phase have been stripped down to a list of requirements, to which a Fog architecture must adhere to. As a proof-of-concept, a subset of the extracted components has been the subject of a first platform prototype called *CCE* (see Appendix A). We have implemented this prototype targeting mobile devices and task sharing collaboration. The results of this second phase have been published on the *First International Francophone Conference on Collaborative Systems (SYSCO'2012)* [4]. Lessons learned from the development and evaluation of this prototype helped to extend, improve, and refine the architecture.

Indeed, in the third phase, the verdict of Fog computing requirements has been translated to a set of components and modules organized in layers within the architecture. This architecture is called *CoFog* and it defines three levels –i.e., the middleware (Mdl), the operational (Opl) and the dependencies (Vrl) level. The first level is divided into two layers -i.e., the abstraction and the data transformation and unification layers. Similarly, the second level (Opl) is organized into two layers -i.e., the operation and the service layer. The third and last level (Vrl) constitutes the level of non-functional or technical dependencies (e.g., security, persistence, management). The operational level - being the core of the proposed architecture - has been the subject of a scientific publication [2] in the *13th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2016)* held in conjunction with the *11th International Conference on Future Networks and Communications (FNC 2016)*.

The fourth phase was fully dedicated to tackling the architecture security aspect. We have focused on the access control dimension in order to provide an access control model for collaboration in the proposed CoFog architecture. We began by analyzing access control requirements for thing-based collaboration. We have selected, studied and amended both role and attribute-based access control models according to the architecture requirements. This comprehensive study provided us with insight into access control mechanisms for our formal data sharing model (Dsm). The Dsm provides mechanisms to discover and select thing-based services. The results of this phase have been published [6] in the *6th International Conference on Ambient Systems, Networks and Technologies (ANT-2015),* held in conjunction with *the 5th International Conference on Sustainable Energy Information Technology (SEIT-2015).* It is worth noting that the final reference architecture has been accepted for publication [6] in the *Special issue of the International Journal of Ubiquitous Systems and Pervasive Networks (JUSPN).*

The fifth phase of this project has been committed to instantiating the proposed architecture. In order to accelerate the realization of the prototype, we have invested less time on some aspects, but more on the performance and extensibility. The resulting prototype has been subjected to validation, in addition to verifying its impact on the host resources.

4. Thesis Organization

In this chapter, we have positioned the work in its general context, motivated and stated the outcome of this research project.

The next chapter, Chapter II, presents a study and an analysis of the issues related to the current trends in the Internet of Things. It brings to light the problems of data processing and data analysis centralization in a Cloud-based architecture. Such architecture implies the existence of a large distance (network links) between IoT devices and the Cloud. In addition, this chapter introduces a set of key concepts relevant to this work.

Chapter III is a review of research literature on collaborative technologies. It first reviews the characteristics of technologies that support or enable collaboration in the IoT. Then the review focuses on the current research state of the art on existing approaches for collaboration in Fog infrastructure. In addition, the chapter discusses the limits of the trending approaches and gives further clarification on the problem statements.

As opportunities arise with the Fog computing concept, Chapter IV presents the main aspects and perspectives to consider in a Fog computing platform. Furthermore, we introduce in this chapter the principles and the rationales behind the proposed architectural model, named *CoFog*. Principal functional components, operational behaviour, and the flow of information within the architecture's middleware level (Mdl) have been described.

Following the general descriptions and the detailed depiction of the *CoFog* architecture, Chapter V continues the description of the reference architecture. It presents service presentation, management and transformation component at the operational level (Opl). This level aims at delivering a set of different, yet homogenous and complementary services. Such services can be leveraged through the application of a set of operations.

Chapter VI raises questions about the principal requirements of Fog computing, in general, and especially requirement of the collaboration in the *CoFog* architecture regarding access control mechanisms. The security problem in Fog computing is a complex and a multidimensional one. Access control models are a highly important dimension of this problem. The answers to these questions have shaped the design of the proposed access control mechanism.

Chapter VII discusses a case study and a proof-of-concept instantiation of the *CoFog* architecture. This instantiation aims at designing and implementing a Fog platform to tackle a real-world use-case –i.e., a Smart parking use case. The platform has been implemented following the architectural model to show the role and importance of the different components.

The last chapter of this thesis is dedicated to discussing the contributions alongside with the limitations and the prospects of this work.

Chapter II
FUNDAMENTALS AND PROBLEM STATEMENT

1. THE INTERNET OF THINGS

1.1. THE RISE OF THE THINGS

The eighties were the years of microprocessors when we built our computers. By the dawn of the next decade, the technological power received a boost with the inception of networking and communication. For the first time, computers were connected together leading to the phenomenon of the Internet and the World Wide Web. Due to the arrival of Mobile-Internet and Social Networks with the new millennium, users started to become constantly connected together over the Internet. Today, the computer is everywhere, connected to everything and embedded in almost every object. That is, machines first learned to do, then they learned to think; nowadays, they are learning to perceive, sense, react and interact within a global network called the Internet of Things.

The concept of the Internet of Things (IoT) has first appeared as the title of a presentation by K. Ashton of the MIT Auto-ID Center back in 1999 [3]. The presentation promoted not just how the Radio Frequency Identification (RFID) tags might be used to enable computers to observe, identify, and understand the world; but to envision and to develop a network connecting everything needed to create an Internet of connected things. In the first papers of general interest on the IoT, this concept was considered as the mere extension of Radio Frequency Identification, uniquely identifiable objects and their virtual representations in an Internet-like structure [11]. Nevertheless, the idea was mentioned before several times in Billy Joy's speeches and lectures [29] as the sixth Web. In fact, the sixth Web or the Device-to-Device Internet (D2D) describes the Internet of sensors that embed machine intelligence in our daily life activities.

Since its first appearance, the hype surrounding the concept of IoT grew to substantial proportions. The IoT came broadly into public view in 2005 with the International Telecommunications Union (ITU) publishing the first report on the subject [94]. According to statistics data reported in [110], it can be estimated that IoT has come into existence, at least physically, sometime near 2009 (Figure 3). At that point, while the world's human population increased to 6.8 billion, the number of Internet-connected devices has known an explosive growth reaching nearly 12 billion devices. As a result, the ratio of connected objects per capita raised to 1.84 for the first time in human history.

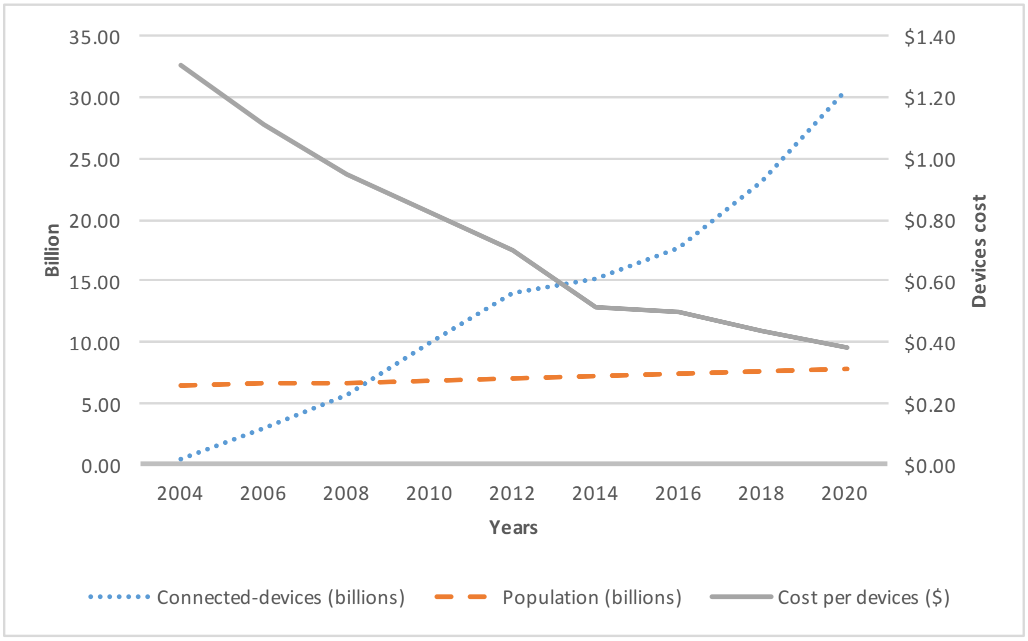

Figure 3. The growth of the Internet-connected devices vs. the world population

Due to its capabilities, the IoT gained significant attention in academia as well as in the industry. It promises a "*Smart World*" [114] where all the smart objects around us might be connected to the Internet using minimal or no direct human interaction. Ultimately, the goal is to create a world where our surrounding objects know what we like, what we want, what we need, and act accordingly with no explicit instructions [114]. Hence, the IoT inherently share a significant amount of concepts with other computer fields. It packs different technologies and concepts ranging from sensors, actuators, data modelling, and storing, to Cloud and other various communication technologies [31]. That is, researchers are using existing and well-known technologies in different ways to satisfy the characteristics and the demands of IoT; we are still shaping our future vision of this global network.

With the rise of the Internet of Things, the issue of defining and setting boundaries to such paradigm arises too. Hence, the following sections emphasize on the definitions that have been proposed to capture the various facets of the IoT. In Addition, special care has been given to describe the status quo on IoT and its relation to current technologies and trends.

1.2. Definition and vision of IoT

Internet of Things research is only at its early stages. A standard definition is yet to be provided [29]. To capture the different aspects and meanings given to the concept of IoT, many definitions have been proposed. Back in 1999, Ashton [91] stated that passive RFID transponder, as a very simple and low-cost computer, can connect to the Internet through a reader. Then computers can see, smell and hear the world without the human-introduced data. Nonetheless, some experts say that the act of reading an RFID tag, capturing information about the location and status of an object, and then sharing the data over the Internet is not part of the Internet of Things.

Syntactically, the expression is a two concepts combination: "Internet" and "Thing". While the word "Thing" refers to a non-precisely identifiable object, "Internet" is the worldwide network of interconnected computer networks, based on the standard communication protocol TCP/IP. Therefore, semantically, "Internet of Things" rise up as "*a worldwide network of interconnected objects uniquely addressable, based on standard communication protocols*" [11]. Otherwise, from a data-centric perspective [74], "*the Internet of Things refers to uniquely addressable objects and their virtual representations in an Internet-like structure*". This vision of the Internet of Things implies that the uniquely addressable and Internet-connected objects use the same protocols already used to connect our computers to the Internet.

In the recent years, many organizations have been leading efforts toward the standardization of the IoT definition. For instance, the IEEE is leading an ongoing project in this direction. The current draft of the P2413 standard provides an overview of an architectural framework and describes the IoT as *"a network of items each embedded with sensors which are connected to the internet"* [8]. In a similar vein, the European Telecommunications Standards Institute (ETSI) discusses the concept under the Machine-to-Machine (M2M) umbrella. ETSI defines M2M communication as "*the communication between two or more entities that do not necessarily need any direct human intervention. M2M services intend to automate decision and communication processes.*" [55]. The Internet Engineering Task Force (IETF) has also stated that "*the basic idea is that IoT will connect objects around us to provide seamless communication and contextual services provided by them … to make the service better and accessible anytime, from anywhere*" [34].

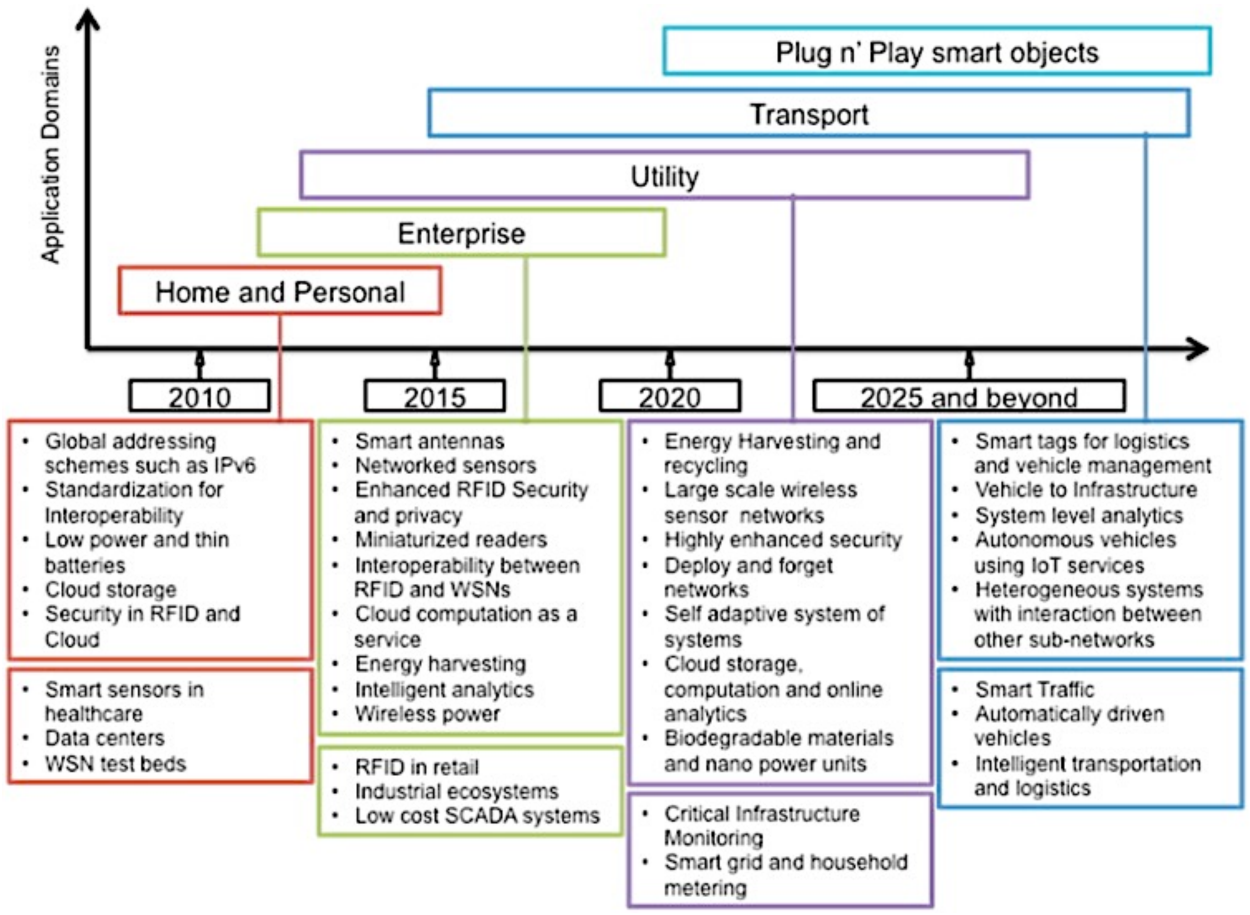
Application Domains
Plug n' Play smart objects
Transport
Utility
Enterprise
Home and Personal
2010
2015
2020
2025 and beyond
Global addressing schemes such as IPv6
Standardization for Interoperability
Low power and thin batteries
Cloud storage
Security in RFID and Cloud
Smart sensors in healthcare
Data centers
WSN test beds
Smart antennas
Networked sensors
Enhanced RFID Security and privacy
Miniaturized readers
Interoperability between RFID and WSNs
Cloud computation as a service
Energy harvesting
Intelligent analytics
Wireless power
RFID in retail
Industrial ecosystems
Low cost SCADA systems
Energy Harvesting and recycling
Large scale wireless sensor networks
Highly enhanced security
Deploy and forget networks
Self adaptive system of systems
Cloud storage, computation and online analytics
Biodegradable materials and nano power units
Critical Infrastructure Monitoring
Smart grid and household metering
Smart tags for logistics and vehicle management
Vehicle to Infrastructure
System level analytics
Autonomous vehicles using IoT services
Heterogeneous systems with interaction between other sub-networks
Smart Traffic
Automatically driven vehicles
Intelligent transportation and logistics

Figure 4. Envisioned technological developments in the Internet of Things [42]

Along with the aforementioned standardization bodies, many project and initiatives are providing definitions of the IoT. Such definitions vary depending on the envisioned implementation technologies [42] (Figure 4), and are mainly general and descriptive rather than being formal. That being said, the common aspect is that IoT describes the next generation of the Internet, where the physical things could be accessed and identified through the Internet. In addition, it provides things with the ability to exchange and process data according to predefined schemes.

Beyond the definition of the IoT, the future is to move from objects with identifiers toward networks of objects with abilities to collaborate and interact with their environment [105]. Hence, we could not discuss the IoT paradigm without considering the definition and characteristics of these objects – i.e., things. Things are a building block of this infrastructure, and they are an active participant in the business process [105] (e.g., RFID tags to track product in supply chain management). From the previous section, a thing can be defined as an entity with a unique identifier, that may carry an embedded application logic (system), and that have the ability to transfer data over the network. The IoT already comprises a panoply of different things (tags, sensors, actuators, and other devices) that augment physical objects (thermostat, lamp, fridge, etc.) with sensing, processing, networking, and reacting capabilities [105]. From a functional perspective, these augmented functionalities transform everyday physical objects into Smart Objects. A smart object is a physical object in its association with a Digital Entity. The latter is the thing that acts as a digital proxy providing a unique and synchronized representation of the object on the IoT [63]. Nevertheless, a given digital entity can be deployed as an autonomous agent with no bounds to the physical world, thus providing processing capabilities as a set of services on the network. In this document, "thing" and "object" are used interchangeably unless stated otherwise.

1.3. THE IOT TODAY

In Santander, northern Spain, the city has spread sensors across its landscape [105]. The purpose of this ambitious project is to transform insensate physical objects into little Internet-connected things. The project has deployed sensors at the city's main entry points gauging traffic flows and volumes. Another set of sensors have been deployed in parks and gardens to measure moisture and rainfall in an attempt to achieve more efficient irrigation systems. Here in Canada, a Toronto firm named Sensebridge produces simple pieces of jewelry that vibrate every time the customer faces north [100].

These are simple yet perfect examples of how IoT has come into our life. The world has begun to receive real working IoT applications that greatly benefit a number of sectors. Public and private sector organizations are moving to smarter governance systems. In fact, it can be clearly stated that the Internet of Things has reached and gained further recognition of many actors in academia and industrial domains. The boundaries and gap between the physical world and the virtual world are slowly being dissolved.

1.4. IoT AS SERVICE-ORIENTED ARCHITECTURE

To exploit the functionalities and the capabilities of IoT-connected Things, such smart Things have to be accessed from the Internet through in a way or another. Since the research in this domain still in its infancy, researchers and experts in both academia and industrial world are using existing and well-known technologies to this end. It is no doubt that the Cloud computing paradigm, through the service-oriented architectural model, has been the "go-to" solution to implement some of the features of IoT.

Indeed, the term "Cloud" was first used by Amazon and was associated with elastic infrastructures that deliver computing resources as a service over the network [104]. This model and the new technology enablers have progressively allowed the support of various paradigms known as "*Applications as a Service*", "*Platforms as a Service*" and "*Infrastructure as a Service*". Such trends help to reduce the cost of ownership and management of virtualized resources enabling provisioning of new services.

Therefore, one potential and obvious trend in Cloud computing area is "*Things as a Service*". The virtualization of connected-objects and the convergence of the Internet of Things and the Cloud computing foster an unprecedented area of use. This is far beyond virtualizing sensors' data; it demands the ability to virtualize Internet-connected objects and their ability to be composed and orchestrated. Based on such architecture, thing-based services are offered on demand in a Cloud environment fashion [76].

The next section deals with the drawbacks of this tight relationship between IoT and its current architecture centred around the Cloud.

2. IoT and the Cloud problem

It is clear that the Internet of Things has arrived. The recent research trends to tackle the many challenges and issues that arise with it are mostly toward centralized Cloud architectures. As shown in Figure 5, the network infrastructure (i.e., the Internet) is used to transmit aggregated data from the sensing infrastructure toward the decision-making layer at the top of the architecture (i.e., the Cloud). Such architecture uses the efficiency and the high computational and storage power of the Cloud to process and store data.

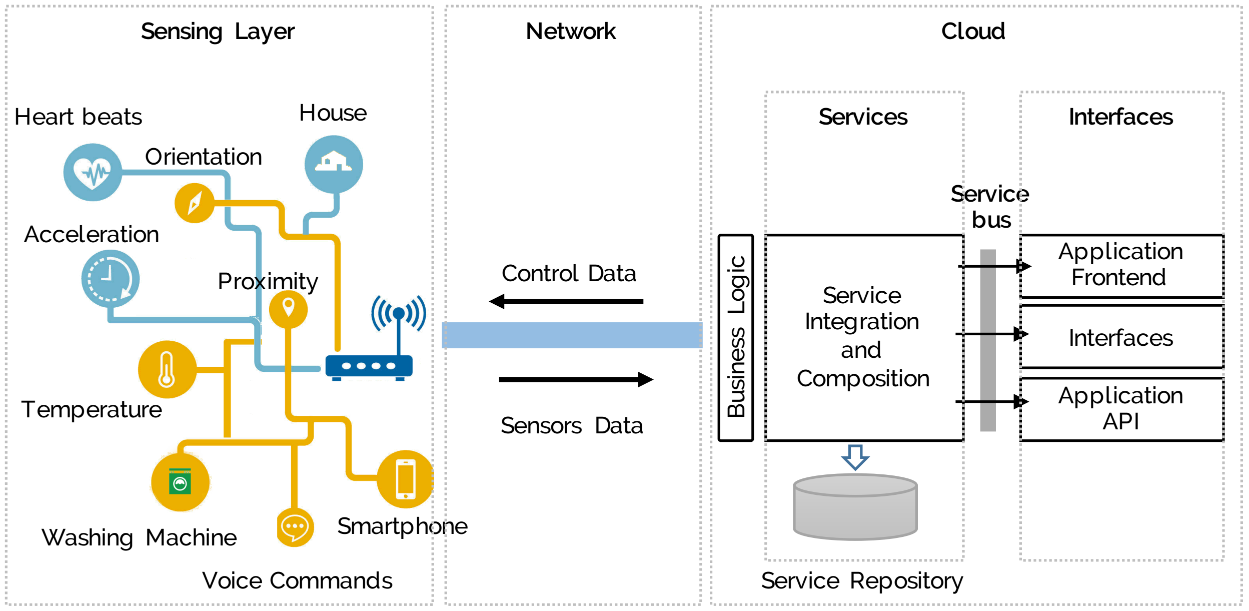

Figure 5. The Cloud at the heart of IoT (adapted from [71])

Indeed, the Cloud has the commodity to better serve a huge number of users and to process the enormous quantity of data coming from the various IoT devices (sensors and other devices). Nonetheless, since most of data processing and analysis occur at the top level of the architecture, the distance between IoT objects (i.e., sensing layer) and the Cloud raises problems related to network bandwidth, communication latency, and to accessing local context and mobility information.

In its latest Global Cloud Index report (GCI 2018 [71]), Cisco Systems estimates that by 2021 more than 850 Zettabytes (ZB) of data will be generated on the Internet, mostly by Things. The GCI report reveals that only 10% of the generated data will be useful (85 ZB), which will exceed the data centres traffic (21 ZB by 2021) by a factor of four. Certainly, the continuous torrent of heterogeneous and potentially irrelevant data will have a huge impact on the network bandwidth, leading it to become a bottleneck for the Cloud. In fact, Cortés *et al*. [25] conducted a study on the challenges facing real-time processing of tracking data generated by a healthcare sport-oriented application called Endomondo. This study concluded that for such a medium sized application, there is an average data flow of 25000 GPS tuples per second (≈1Gb/s). Such an application, and many other examples, will challenge the capacity of the Cloud to maintain a reasonable and predictable communication latency and response time; for many use cases do require very short to real-time response. For instance, in a vehicular network (VANETs [28]), lives may depend on how fast the decision to applying the brakes is made.

In addition, such Cloud-centric approach comes with many drawbacks related to the easiness to access local context information. The sophistication of IoT applications relies mainly on the analysis of data coming from the connected devices. The analysis uses data related to users' and devices' context –e.g., precise user location, local network condition, users' mobility, devices' resources and capabilities (CPU, memory…), and so forth. Unfortunately, the physical distance between the Cloud and the end devices makes Cloud services not capable to directly accessing such local contextual information. Even if such information could be sent, in a way or another, to the top level of the architecture, many use cases do not require decisions to be made in the Cloud. For instance, in a Smart Home, the decision to change the intensity of the lights, depending on whether a person is working, reading, or watching TV, does not require the intervention of Cloud services. In addition, given that some decisions have to be made in the Cloud, it is not efficient to send the entirety of the sensed data, since not all the data are relevant to the decision making.

Furthermore, there is a growing concern among users about transferring local and personal information to the Cloud. That is, products and devices we use in our daily activities are constantly leaking data. We can argue that encrypting such sensitive data might lighten few of these concerns, however, the encryption makes processing and analyzing the data extremely difficult or even impossible [116]. Hence, restraining the full expansion of IoT applications. On a more personal level, we share the same view as Albrecht *et al*. [119] from the Consumers Against Supermarket Privacy Invasion and Numbering group (CASPIEN) as they stated "… *but let's not fool ourselves. The information is not ours. It belongs to Google, and IBM, and Cisco Systems and the global Mega-Corp that owns your local super- market. If you don't believe us, just try removing 'your' data from their databases*".

IoT applications require context-awareness, low latency and more interestingly real-time data processing. Thus, a new kind of "Cloud" flourishes at the edge of the network leading to "Micro-Clouds" to manage, analyze and extends the Cloud computing paradigm [10]. Security is a crucial aspect in such environment; as extending existing security mechanisms will not be sufficient to satisfy the features of IoT [98]. In such infrastructure, many security threats come from the interactions between the digital and physical world. Things have a limited and a cost-ineffective support of security. In addition, they operate in unprotected and vulnerable environments (cars, medical devices, wearables). Cloud-based and Cloud-like security solutions are needed to protect things beyond enterprises' networks [22].

The aforementioned problems have motivated the introduction of various novel concepts aiming at providing Cloud-like capabilities in the vicinity of users. In this work, we believe that the solution resides on providing such Cloud-like features at the edge of the network. Either by relying entirely on the edge capabilities or via a collaboration between the edge and the central Cloud. This confidence led us to focus on the Fog computing paradigm. Therefore, the next section gives an overview of this paradigm.

3. THE FOG COMPUTING

3.1. THE IDEA AND THE PARADIGM

The need to deploy a computational infrastructure at the edge of the network is mainly the result of the convergence of Mobile and Cloud computing. For instance, we have seen the application of such approach as Cloudlets [20], Mobile Edge computing [102] and Edge-Centric computing [36]. Further details and analysis on the concepts related to computing at the edge of the network are provided in the next chapter. In this section, we introduce a broader paradigm called Fog computing.

The idea of Fog computing has been presented by Bonomi *et al.* from Cisco Systems in 2012 [40]. First, this new concept was considered merely as an extension of the Cloud computing paradigm. As such, Fog computing would use edge devices near users to provide storage, computation, and some basic networking services [19,20]. In the meantime, the Cloud infrastructure takes care of the global coordination of underlying infrastructures and the analysis of data. As the research into this domain gained more interest, Fog Computing has become a paradigm on its own. Its definition has been extended to embrace features of ubiquity, improved networking capabilities and better support of cooperation between devices [119]. Although it shares many similarities with the Mobile Edge computing paradigm, the Fog computing paradigm is broadly intended to deal with applications in the context of IoT.

Indeed, since it was initially proposed, the idea of the Fog computing have been intellectually and technically seductive. The first definition of this paradigm was proposed by Bonomi *et al.* and it states that "*Fog computing is a highly virtualized platform that provides compute, storage and networking services between end devices and traditional Cloud computing Data Centers, typically, but not exclusively located at the edge of the network*" [5]. Hence the name "*Fog*" that comes from the analogy that its infrastructure deployment locations are closer to the end devices than they are to the Clouds. In this context, end or edge devices are referred to as "Things" which include a wide variety of sensors, actuators, mobile devices, embedded systems, and so forth. Another similar definition has been proposed in [21] "*Fog computing is a scenario where a huge number of heterogeneous (wireless and sometimes autonomous) ubiquitous and decentralised devices communicate and potentially cooperate among them and with the network to perform storage and processing tasks without the intervention of third-parties. These tasks can be for supporting basic network functions or new services and applications that run in a sandboxed environment.*" [119]. As recently as 2016, Cisco Systems, ARM Holdings, Dell, Intel, Microsoft and Princeton University founded the *OpenFog Consortium* to promote development and interests in Fog computing. The efforts of this consortium have led to the publication of the *OpenFog*

Reference Architecture for Fog Computing [119] that defines Fog computing as *"a horizontal, system-level architecture that distributes computing, storage, control and networking functions closer to the users along a cloud-to-thing continuum"*.

The aforementioned definitions bring into light three main characteristics of Fog computing: extending the Cloud, edge of the network, and edge devices. The first concept comes from the necessity to preserve the benefits of Cloud computing such as orchestration, efficiency and manageability. As an extension to the traditional Cloud-only model, the implementation of the Fog architecture requires an additional layer (or layers) of Fog nodes that can be located at various points of the network's topology. This layer of Fog nodes is what represents the edge of the network in the context of Fog computing. As such, the distinction between Cloud "nodes" and Fog nodes could be problematic. We believe that the difference resides in the proximity and the capabilities of the nodes; for Cloud "nodes" by definition have more storage and processing power. On the contrary, Fog nodes have more constrained capabilities and they are, usually, closer to the edge devices. Furthermore, regarding the application domain, Fog nodes are more intended for local environments with real-time, latency-sensitive, and geo-distributed applications. Considering all the characteristics of Fog computing, one can clearly see that a Fog Layer is formed between the IoT Things and the Cloud to deal with communication, computing and access management.

3.2. Fog Nodes

As previously described, Fog nodes (a.k.a. Fogs) are to support mobility, real-time data analysis, and decision-making processes. They have more importance in use-cases where data needs to be collected, filtered and analysed locally at the edge level.

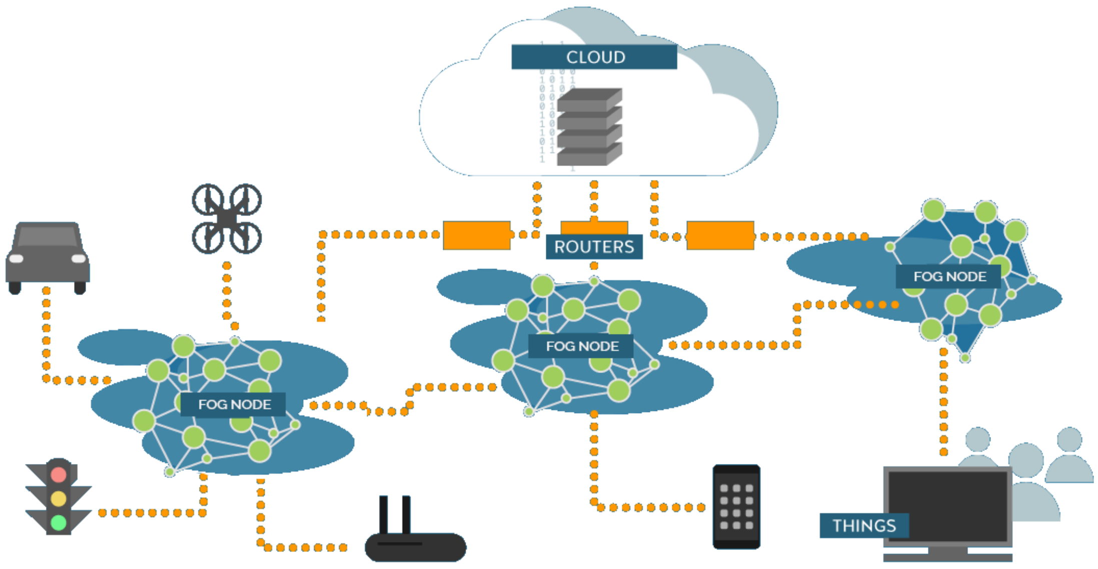

Figure 6. The Fog nodes between the Cloud and IoT objects

Hence, depending on the use-case and the deployment strategy, Fogs can be deployed on low-level elements of the network such as routers, gateways, and access points up to higher levels of the hierarchy including the Cloud (Figure 6). Before the deployment of a Fog node, many aspects must be addressed including compute and storage capabilities; connectivity and networking capabilities; and the node security and management aspects [88]. Indeed, in order to provide analysis, filtering, autonomous learning, etc. Fogs need to have general purpose compute and data storage functionalities. This will leads to higher level of interoperability between Fogs. In addition, it is possible that sensors and other edge devices may not be able to interface directly with such nodes. Consequently, an abstraction layer might be needed to connect and exploit such devices. Furthermore, Fogs nodes will acts as gateways between the sensing infrastructure and the IoT upper levels. This aspect bring the role of security gateways as important task for such nodes.

In light of this, our interest lies in extending the new paradigm of Fog computing to embrace a Thing-collaboration computing model. In such model, Things could be enabled to collaborate and exchange date with each other to achieve common or distinct goals. However, collaboration at such low level, will increase even more the complexity of interactions in this model [88]. This increasing complexity is due to the huge number, the heterogeneous and the dynamic nature of the Things involved. The heterogeneity between the technics and the technologies used to both offer and consume Fog services will add to this complexity [111].

Chapter III
RELATED WORK

The first step in developing a body of knowledge begins essentially by searching previous achievements to understand the status quo in our field of interest. This chapter is a background review of the state of the art on technologies supporting collaboration in the Internet of Things. The literature review includes works that have been already used in the IoT as well as potential collaboration-enabling technologies coming from related domains. This study of the state of the art allowed us to focus, first, on the importance that collaboration between Things has gained recently and, second, on key problems toward the real collaboration within a Fog Computing environment.

1. STATE OF THE ART IN THE INTERNET OF THINGS

1.1. OVERVIEW OF THE IOT CHALLENGES

The arrival of the Internet of Things have stressed the need for more clarification about the requirements and the setup of global standards for this new era. In fact, the

Internet of Things promises to connect even the smallest device and sensor to the Internet. Hence, the research community will have to address the challenges of common reference architectures for the future networks, communication technologies, global identification, and the challenges of naming and global discovery. In addition to the tasks of integrating legacy systems and networks.

Needless to say that in spite of following the same standard, two different devices might not be interoperable if they fail to grasp the semantic of the exchanged data. Hence, standardization is necessary but it may not be enough due to the complex and the diverse nature of the new network. Advanced interoperability between heterogeneous environments and between heterogeneous devices through different communication technologies is and will always be a hot topic that requires continuous research. Nevertheless, research in wireless sensor networks has already resulted in promising solutions, tools and operating systems that can run on very small and resource-constrained devices [50]. These solutions need to be evaluated in the real world and in large-scale applications in order to illustrate different use-cases. Such use cases will help in defining new solutions to effectively sustain the mobility nature of smart things, which may be equipped with multiple and heterogeneous network resources. These connected devices are characterized by low resources in terms of both computation and energy capacity. Thus, the development in this area will require research for hardware adaptation and parallel processing in ultra-low-power and probably multi-processor systems. Furthermore, energy storage will also become a serious and real challenge and even an obstacle in the road toward the miniaturization of devices. There is a need to deepen the research in areas like Nano-electronics, semiconductors, high-capacity energy storage, sensing technologies, and new ways to harvest energy from the devices' environment [127].

The Internet of Things is born from the vision that things will constitute an integral part of the network infrastructure that wire our world. Thus, this network needs to be built on top of a structure that integrates seamlessly wired and wireless technologies in transparent ways. The low-power devices will need links in a multi-hops fashion to cover wide distances, in addition to power-aware protocols that could turn on or off the links in response to the traffic load and demand. Such a network must provide some kind of adaptability to the heterogeneous environment, the various and mixed contexts, and to the content and application needs. This picture would not be complete without mentioning two of the main building blocks of the Internet of Things: security and intelligence [58]. Capabilities such as self-configuration, self-awareness, context-awareness and intelligent inter-machine communication are considered of high priority for the IoT. Self-x (self-configured, self-organized, self-aware, self-protection…) and intelligent things will be in a constant connection with other objects resulting in new security and privacy problems [80]. Moreover, huge amounts of data will be mapped across billions of things that are updating in real-time; a transaction for instance may need to make change across thousands of objects with different security policies. In order to prevent the unauthorized use of private information, research is needed in the area of dynamic trust, security, and privacy management.

1.2. Selected IoT related Projects

1.2.1. CASAGRAS

The CASAGRAS [81] project, stands for "*Coordination and support action for global RFID-related activities and standardization*," was a project financed by the European Union in 2008. The project focused on shaping the foundational studies about RFID in support of the Internet of Things. In its final report, the project provides an abstract architectural model for IoT (Figure 7). This model consists of three layers:

- The physical layer: this layer comprises identified things (physical objects) that connect through object-connected data carrier technologies such as RFID.
- Interrogators or gateways layer: this layer offers the interfaces between the object-connected devices and the information management systems.
- Application and Information Management Layer: this layer provides the functional platform for supporting applications and services.

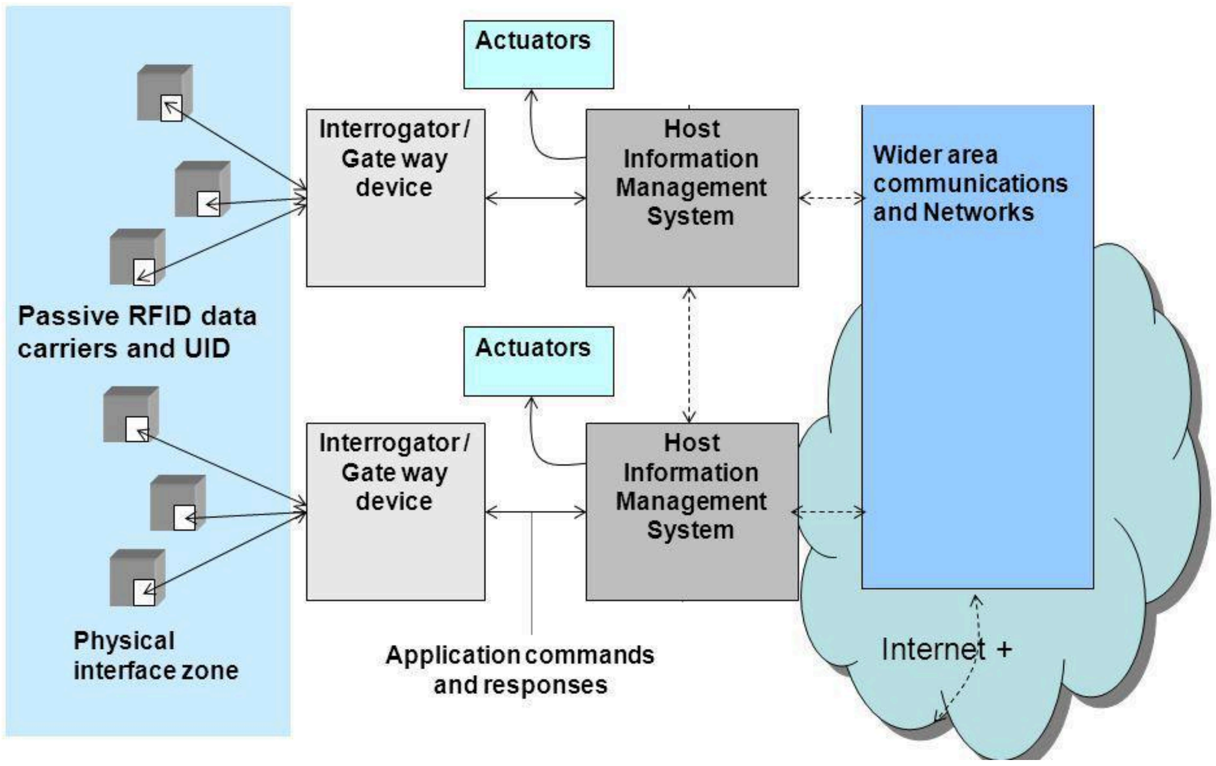
Actuators
Interrogator / Gate way device
Host Information Management System
Wider area communications and Networks
Passive RFID data carriers and UID
Actuators
Interrogator / Gate way device
Host Information Management System
Physical interface zone
Application commands and responses
Internet +

Figure 7. CASAGRAS IoT model architecture

1.2.2. Cyber-Physical Systems

The Center for Hybrid and Embedded Software Systems at the Berkeley University is pursuing research in the abstractions and analytical techniques of Cyber-Physical Systems (CPS) [70]. This project mainly focuses on the integration of embedded computation and networking to monitor and control the physical processes, with feedbacks in between. The concept of CPS is similar to the IoT concept with difference in the application. A CPS is concerned about the collaborative activity between sensors and/or actuators to achieve a certain goal, whether in an intranet or extranet. To achieve the goal, CPS may use an IoT system.

1.2.3. The Internet of Things Reference Model

The IoT Architecture project (IoT-A) [70] proposed an architectural model for the IoT, along with an initial set of key building blocks. The project focused on developing an architectural reference model by tackling security, addressing and protocol interaction of the various components of the architecture [14].

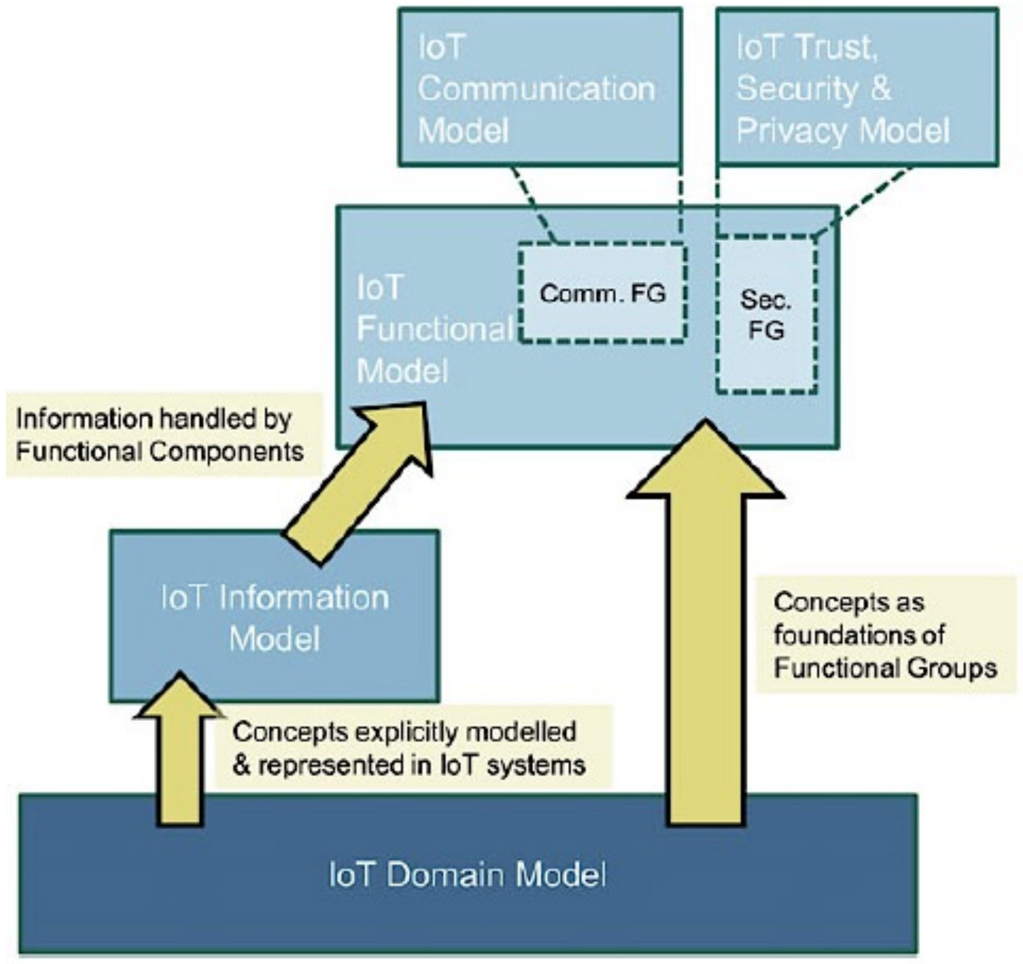
IoT Communication Model
IoT Trust, Security & Privacy Model
IoT Functional Model
Comm. FG
Sec. FG
Information handled by Functional Components
IoT Information Model
Concepts as foundations of Functional Groups
Concepts explicitly modelled & represented in IoT systems
IoT Domain Model

Figure 8. Interaction of sub-models in the IoT-A reference model (adapted from [15])

The Architectural Reference Model (ARM) proposed by IoT-A has five sub-models (Figure 8). The IoT Domain Model includes the main concepts of devices, IoT services, Virtual Entities (VE), and the relations between them (Figure 9.).

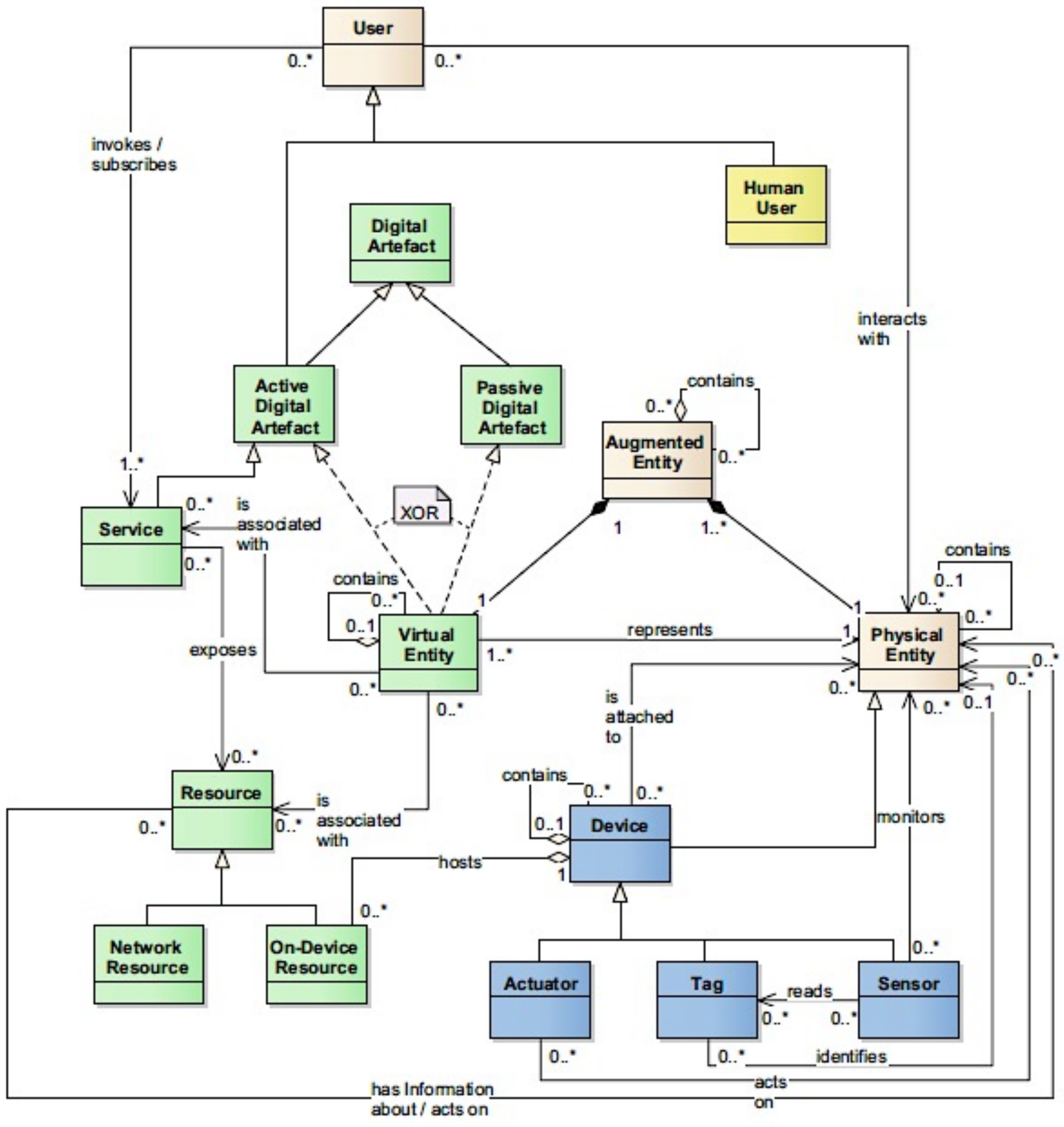

Figure 9. UML representation of the IoT Domain Model (adapted from [16])

Based on this model, the Information Model defines the structure of IoT related information (e.g. information about devices, services, virtual entities). The Functional Model identifies Functionality Groups for interacting with and managing information about the IoT main concepts. The functionalities of the FGs that manage information use the IoT Information Model as the basis for structuring their information.

The IoT Communication Model introduces many concepts that are in charge of handling the complexity of communication in heterogeneous IoT environments. The Trust, Security and Privacy Model introduces these relevant functionalities and their interdependencies and interaction. Both last models are Functionality Groups in the Functional Model.

1.2.4. The Internet Connected Objects for Reconfigurable Ecosystems

The Internet Connected Objects for Reconfigurable Ecosystems (iCore) [16] aims to abstract the technological heterogeneity of the vast amounts of heterogeneous objects and provide high-level reusability for application through virtual objects and cognitive technologies [77]. iCore defines three levels of virtualization that top-level applications can use to control real world objects (Figure 10) : the Service level, the Composite Virtual Objects level and the Virtual Objects levels.

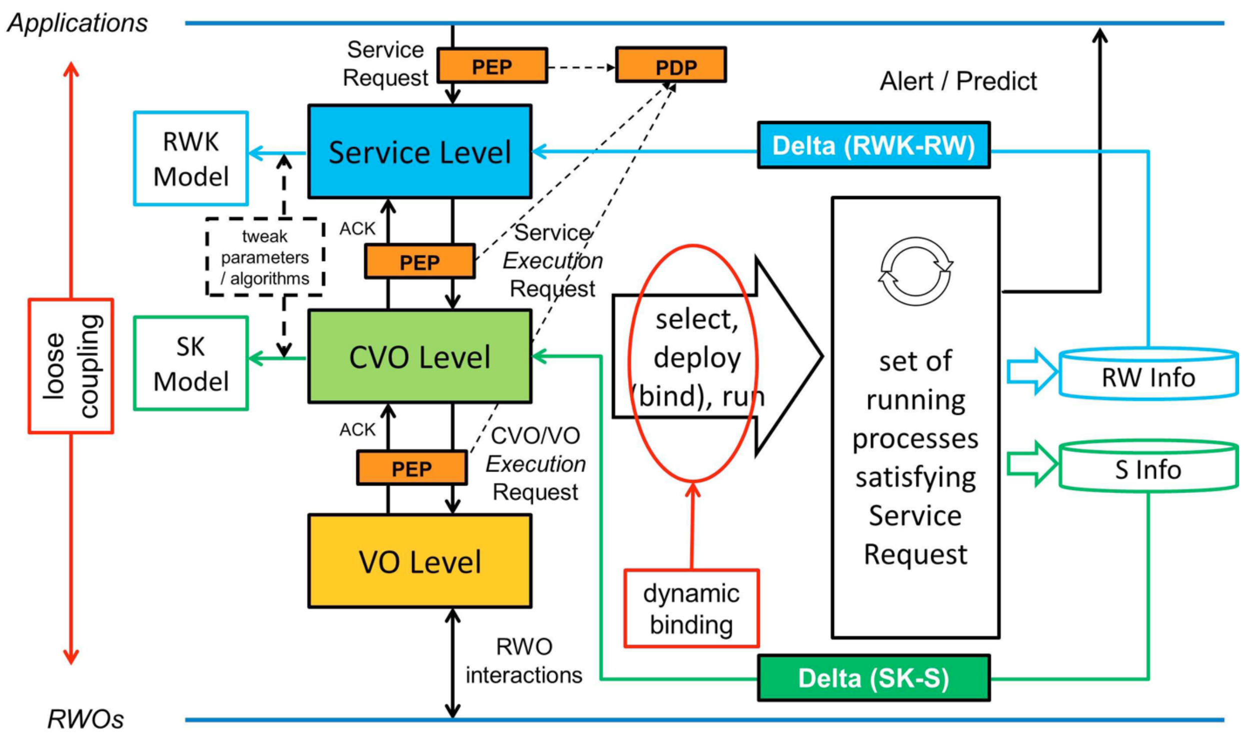

Figure 10. iCore Architecture Model (adapted from [78])

The Service Level provides functionalities for planning and understanding what services are needed in order to achieve a goal and by means of which Composite Virtual Object (CVO) or Virtual Object (VO). The Composite Virtual Objects Level contains a run-time management and execution environment that efficiently manages and executes the requested pool of service instances as a composition of so-called CVOs, connected to the abstraction of Real World Object data (via sensors and actuators) with functional enrichment, though the VOs in the Virtual Objects Level.

At the Virtual Objects Level, URLs are used for both naming and addressing VOs as Web Resources. Yet, the architecture provides no specified naming scheme.

1.2.5. IoT at Work Architecture

The IoT at Work project (IoT@Work) [78] focused on IoT technologies to provide a plug-and-work concept for industrial and automation environments. In its final report [99], IoT@Work is described as a three layers architecture with three planes to structure cross-cutting concerns Figure 11.

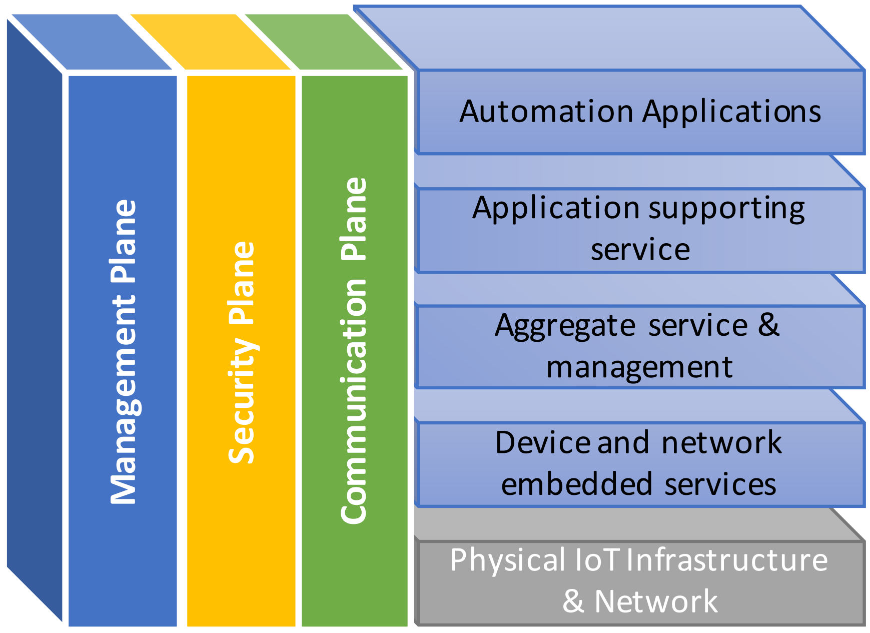
Management Plane
Security Plane
Communication Plane
Automation Applications
Application supporting service
Aggregate service & management
Device and network embedded services
Physical IoT Infrastructure & Network

Figure 11. IoT@Work architecture and main functionalities [109].

The layers include management and orchestration functionalities that deal with the configuration and the execution of applications using resources and services offered in the IoT infrastructure. The three layers are, from bottom to top:

- The Device and Embedded Services layer, which includes identifiers assignment, devices context collecting, communication management physical security.

- The Device Resource Creation and Management Services layer, which abstracts and hides the details about single IoT devices. The functions here include service directories, network abstractions, and low-level system monitoring and security management.
- The third layer of abstraction supports directly the application through specific middleware services. Indeed, the Application Middleware Services layer include a messaging bus, application resource descriptions and other application supporting functions.

The crosscutting orthogonal planes that the architecture focuses on are:

- Communication Plane: managing networks and communication, while delivering guarantees for the applications that need high Quality-of-Service (QoS).
- Security Plane: managing system security to make sure that different management functions at each layer include some security mechanisms.
- Management Plane: supporting service management and orchestration and linking devices to applications and services

2. COMPUTING AT THE EDGE OF THE NETWORK

2.1. MOBILE CLOUD COMPUTING (MCC)

The idea behind computing at the edge of the network is not new. Rather, it is a convergence of experiences with both Mobile and Cloud computing [109]. Indeed, the main features desired in mobile devices are small size, lightweight, ease of use, and long battery life. Due to such requirements, mobile devices are inheritably resource-constrained. Nevertheless, mobile devices can overcome such constraints via remote execution by exploiting remote infrastructure that offers more computational resources. The emergence of the Cloud computing pushed forward the adoption of this remote-execution model toward what is called Mobile Cloud Computing (MCC). MCC represents the convergence between Mobile and Cloud computing.

Unfortunately, Cloud computing encourages a centralized infrastructure that implies a large separation between mobile devices and the Cloud. As it is the case for IoT today, mobile-to-cloud communication involves many network hops and results in high latency and high consumption of the network bandwidth. For these reasons, the problem of proximity between mobile devices and the Cloud has become a crucial issue and a burden on MCC solutions. Many novel paradigms have been proposed to deal with this issue [72], each of which shares the same common goal of deploying Cloud-like capabilities at the edge of the network. As the next sections reveals, the differences arise when considering the deployment, the use, and the ownership of such edge infrastructures.

2.2. CLOUDLETS, PROXIMITY MATTERS

The Cloudlet concept [13,33,39,102] was proposed mainly to promote mobile offload (or delegation) under what can be seen as a cyber foraging -i.e., "*The idea is to dynamically augment the computing resources of a wireless mobile computer by exploiting wired hardware infrastructure*" [13,101,103]. This vision was originally limited to delegating the storage of voluminous data and the execution of intensively computational tasks to the Cloud under the MCC paradigm. In recent years, this paradigm have seen an expansion to include delegation to offload instances at the edge of the network [101]. Indeed, the research community have been proposing various solutions to allow mobile devices to delegate tasks to remote resources, either by migrating parts of the code (selected in advance) or by cloning the entire execution environment of applications. Moreover, other approaches propose the use of mobile agents that handle the processing of information on behalf of mobile devices [101].

The Cloudlets addresses mobile delegation through the implementation of small immobile computing instances at the vicinity of mobile users [98]. As shown in Figure 12, the paradigm of Cloudlets relies on a three-tier architecture (i.e., Clients, Cloudlets, Cloud) with two levels. The first level represents the Cloud infrastructure. The second level consists of small data centers dispersed at the edge of the Internet – i.e., Cloudlets. These small infrastructures have a soft state that is generated locally or cached from the Cloud. That is, they use persistent caching of data and code which means that such information times out unless refreshed. Therefore, Cloudlets can be deployed at the user vicinity (at Wi-Fi access points or LTE base stations) and allow devices to load small virtual machine (VM) instances over pre-existing more complete VM images [38].

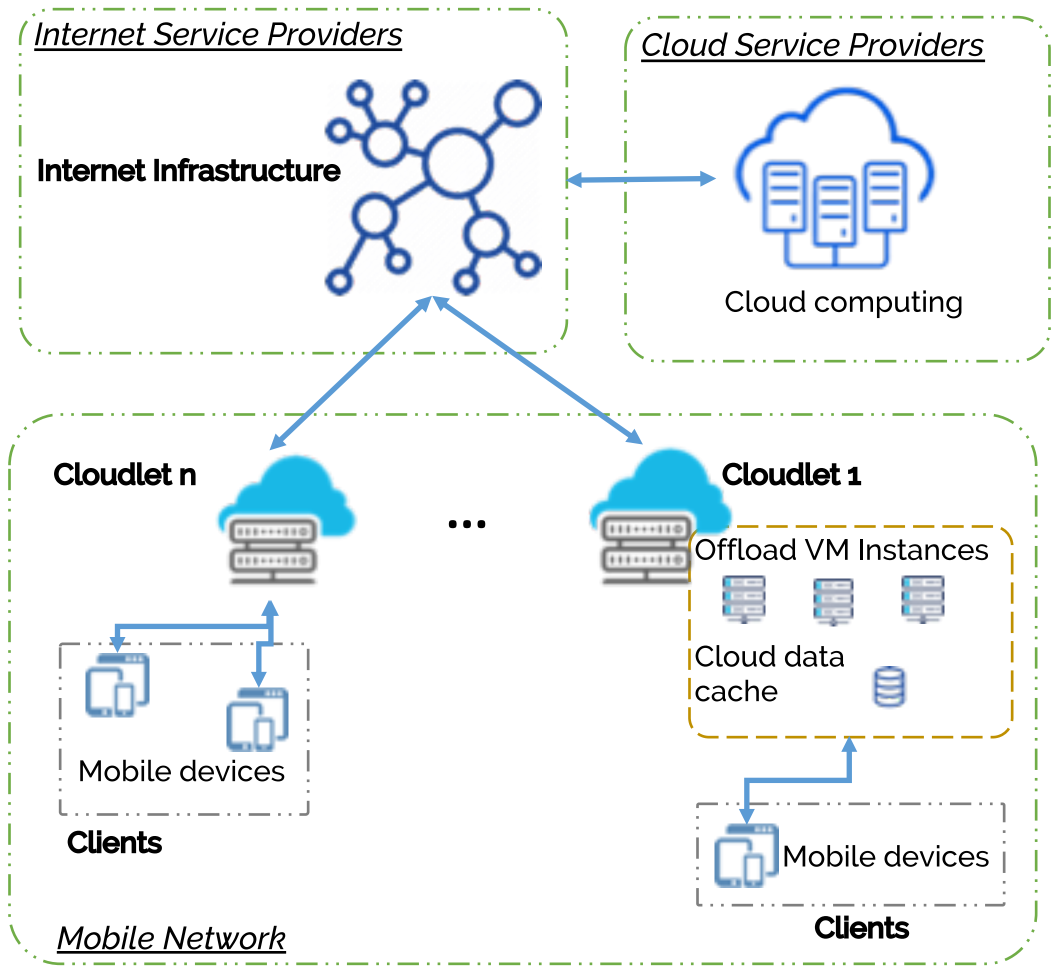

Figure 12. Cloudlet two-levels architecture

2.3. MOBILE EDGE COMPUTING

Mobile Edge Computing (MEC) [33] is another edge computing paradigm that has been drawing much attention in both academia and industrial worlds. In early 2015, the European Telecommunications Standards Institute (ETSI) launched the Industry Specification Group (ISG) for Mobile-Edge Computing in an attempt to standardize MEC [33]. The reference architecture (Figure 13) that has been presented lately shows the functional elements and the reference points between them.

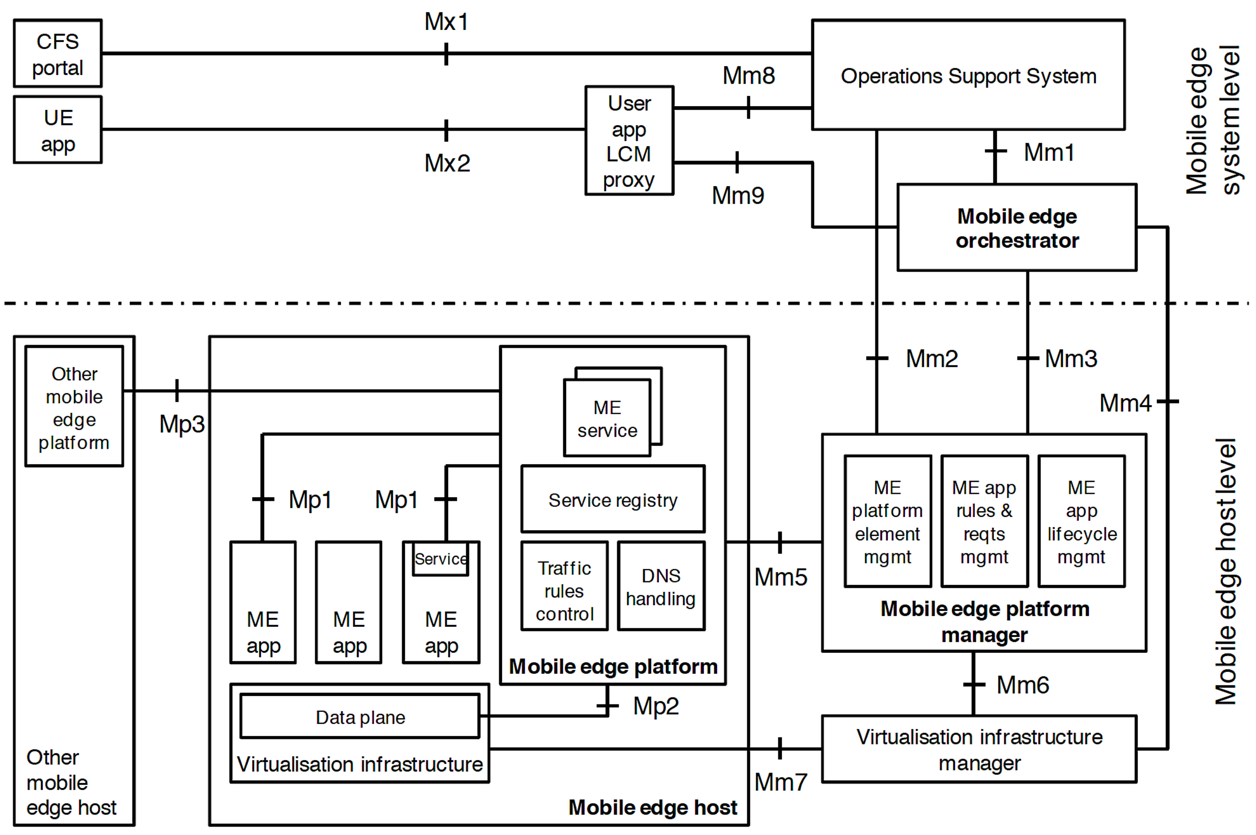

Figure 13. Mobile Edge Computing system reference architecture

The MEC consists of the mobile edge hosts that provide facilities for mobile edge application execution, to radio networks information access, and to location awareness services. In addition to the mobile edge management necessary to manage both the system and its hosts. Under this specification, MEC will provide an ecosystem of Cloud-like capabilities within mobile base stations at the edge of the mobile network. Deployment locations include but are not limited to LTE/5G base stations (eNodeB), 3G Radio Network Controllers (RNC), and other cells of multi-Radio Access Technology (3G/LTE/WLAN) aggregation.

Thanks to this ecosystem, mobile networks will benefit from low latency, high bandwidth, location awareness, and access to radio network information. Furthermore, if such ecosystem is opened not only to mobile network providers but also to other service providers, such openness could bring more contributions and more application scenarios from third-party companies – e.g., augmented and virtual reality.

2.4. Edge-Centric computing

From a human-centred perspective, Edge-Centric computing paradigm [40] shares the common interest of providing Cloud-like services and resources near users with the aforementioned paradigms. This work envisions that the solution to the growing user concerns about trust, privacy, autonomy, and security comes mainly from the integration of humans in the decision making loop. It proposes the deployment of edge computing systems that collaborate with each other in a peer-to-peer fashion. Hence, Cloud services take an auxiliary role in providing stable resources at need. The human-centred perspective arise with the ability to create user-centred ecosystems at the edge of the network. Ecosystems that allow the creation of personal spaces –i.e., spaces where the user can manage personal information, access control and trust mechanisms, social spaces –i.e., spaces where the user can control social activities, and public spaces –i.e., spaces with collaborative information flows where multiple actors, either humans or services, can interact with each other.

2.5. Edge Computing, Analysis

The core idea of edge computing is to bring network functions and Cloud resources to the vicinity of users and end devices, including computing, storage (and caching), and communication resources. This approach evolved from the early years of Mobile Cloud computing which moved computing power and data storage away from mobile devices. Alas, it became quickly obvious that proximity do matter, hence the introduction of recent edge computing paradigms.

In addition to the Fog computing introduced in the previous chapter, the paradigms presented in the previous sections represent the major paradigms in the edge computing approach. In general, they share the common goal of deploying, in a way or another, various forms of edge data centers with computing and storage services. Furthermore, while such edge data centers can generally exist and work autonomously, they are still connected to the Cloud, which can potentially play the role of management and coordination.

A closer look at these paradigms reveals, however, many differences. Mobile Edge computing focuses on infrastructures provided by mobile network operators, whereas Cloudlets and the Edge-Centric computing focuses on private and user-owned ones. By consequence, ownership, deployment location and underlying protocols and interfaces differ from one paradigm to another. MEC considers only radio bases and controllers as deployment locations. Hence, only telecommunication companies can own and operate MEC infrastructures. Edge-centric computing focuses on local servers managed and owned by users. In contrary, Cloudlets focuses on more distributed deployment locations, as even devices themselves can be part of the service provisioning infrastructure. Such ease of deployment encourage a variety of companies to create their own MCC nodes.

As we introduced in the previous chapter, Fog computing paradigm is a broader and more general edge-computing concept that aims to accommodate IoT applications. Fog nodes can be deployed at different levels of the Internet architecture -i.e., either vertically near or far from the Cloud data centers, or horizontally on many locations such as on user-managed servers, access points, routers, gateways, etc.

Table 1 summarizes the main properties of the aforementioned edge paradigms discussed above. The Mobile Cloud computing, being a centralized approach, is mentioned in this table for the sake of comparison only.

Table 1. Comparison of different edge computing paradigms

	Mobile Cloud Computing	**Cloudlets**	**Mobile Edge Computing**	**Edge-Centric Computing**	**Fog Computing**
Architecture hierarchy	Two tiers (centralized)	Three tiers	Three tiers	> Three tiers (distributed)	> Three tiers (distributed)
Location	Large Data Center	Between mobile devices and Data Centers, mobile devices	Radio access network	Edge servers	Any location in the hierarchy (near-cloud, near-edge and edge)
Ownership	Cloud services providers (Amazon, Microsoft, et.)	Local business (private)	Mobile network operators	Private entities, Individuals	Private entities (Fog nodes owners), Individuals
Cooperation between nodes	No	No	No	Yes	Yes
Latency	High	Low	Low	Low	Low
Context awareness	No	Could be equipped	Yes	Yes	Yes

3. SERVICE AGGREGATION IN FOG COMPUTING

The main goal of our work is the collaboration between IoT objects. In the one hand, it aims at bringing service aggregation and composition to the edge of the network using Fog Computing. On the other hand, it focuses on providing a middleware to abstract the undelaying heterogeneity. In this same vein, *Mobile Fog* [51] presents a high-level programming model for the Internet of Things. This model is intended for latency-sensitive and on-demand scaling applications, but a more general approach is needed to deal with resources mobility.

Similarly, by assuming that every Thing provides its functionality as a standard service, the presented composition model [51] uses artificial potential fields to deliver a decentralized service composition. In an attempt to tackle decentralized service composition, Rain4Service [95] models the behaviour of rain drops to achieve service composition. However, this framework is not intended for deployment at the edge of the network. In such environment, filtering and unifying data are the main issues in order to be able to implement the middleware layer [9] since the use and the presentation of data should be adapted depending on the context of the service to provide. To this end, the system in [90] uses a goal-driven and context-aware filtering method. Though, in case of an aggregated or time-dependent sensing activity, issues like mobility support may rise.

Sharing resources between devices at the edge of the network was the focus of Mobile Cloud [82]. The work proposed a framework to share resources in a local cloud; the different measurements of resources are mapped into time.

4. OBJECT AND SERVICE DISCOVERY

Finding entities and services is an essential aspect of an Internet of Things systems. Unlike in small-scale application, IoT applications and services cannot be configured with respect to a fixed set of services. Instead, there is a need to setup resolution, look-up and discovery for IoT services and objects with the adequate level of abstraction. Jacquet *et al.* [84] proposed a routing protocol to support routing in heterogeneous Mobile ad-hoc networks (MANET), where each node can have many interfaces. In the Optimized Link State Routing Protocol (OLSR), *"a flat mechanism is employed, whereby a node sends control messages through all interfaces without regards to the link capacities of the other network"*. OLSR does not scale and does not support the heterogeneous nature of MANET. Hence, the

Hierarchical OLSR [59] came as an extension to the former OLSR. It is aimed at reducing the overhead caused by sending messages regardless of the link capacities, and to make the routing algorithm more scalable. In spite of its ability to improve the scalability of the MANET, the HOLSR affects the network scalability. Indeed, in order to reach the destination node, data travels in normal ways up to the topological level where the destination node is located. Shepherd *et al.* [121] suggested the use of parallel processing across handheld devices to enhance robot sense capabilities. A message passing system, called DynaMP, was developed to allow communication in the "scatternet" network using Ad-Hoc On-Demand Distance Vector-based routing to reduce energy consumption. Based on the Java class loading mechanism, this environment may be deployed on any device with a Java virtual machine. In 2005, Harihar and Kurkovsky [106] attempted to pave the road to Jini [49] in the world of pervasive mobile computing. The work discussed the use of this platform's networking capabilities to develop pervasive computing environments. As claimed by the authors, this framework has the ability to satisfy the demands of ubiquitous systems, namely context awareness, intelligent behavior, interaction, reliability and safety. Perich *et al.* [123] developed a collaborative query processing protocol. This protocol, the CQP protocol, is based on the Contract Net Protocol [92]principles, and it is designed to reduce the computational and energy consumption of the devices implicated in collaboration [96]. The features of the protocol enable any device, irrespective of its limited computing, memory, and battery resources, to locate and obtain data source streams on other peer-devices in order to answer its queries.

Many attempts have been made to connect physical objects to networks. *Diya et al.* [96] proposed an infrastructure framework for Mobile Collaborative Environments. The MCE is based on socket communication that allows any device to connect easily with the other devices on the network. Yet, this approach is still a server-centric one. The server IP and listening port must be known to the client in order to allow transmission of code files between both ends. In 2007, Jeong *et al.* [30] presented a distributed health-care environment, based on a distributed object group framework (DOGF). It provides functions of object group management, real-time object exchange and security services for distributed applications. TMO scheme and TMOSM have been used for the interactions between distributed components. In 2008, Silva *et al.* [60] introduced a grid-based framework to support distributed task execution. Indeed, in order to speed up the execution of common computing tasks, SPADE allows mobile devices to takes advantages of idle remote computer in a Grid way. This tool requires that the application, subject of collaboration, be pre-installed and registered in the server. Hence, the user must manually give the location, the parameters and the appellation of this application. Furthermore, the user provides input files that have to be uploaded to the server.

In order to avoid connecting physical objects directly to the Internet, some approaches suggested abstracting those objects as services by adopting the Service Oriented Paradigm [108]. For instance, the work presented by Guinard *et al.* [43,44,89] describes the architecture of the Web of Things (WoT) based on the principles of the traditional Web such as scalability and modularity. They promote the reuse and the adaptation of existing Web technologies such as REST architectural style [44] to interact with IoT objects. An information sharing architecture for collaborative IoT is presented in [37]. The authors suggested the concept of a user-centric architecture to the IoT that seamlessly integrates IoT objects, Web protocols, Web applications, and Social platforms, etc. Adda and Saad [118] presented a data sharing framework for the collaborative IoT. The framework introduced a formal theoretical model, the IOTCollab domain specific language, and an IDE that implements this model.

The Web of Things has been the focus of many other research projects. For instance, the Constrained Application Protocol (CoAP) [7] allows the connection of resource-constrained devices to the World Web. Using a publish/subscribe mechanism, a CoAP client can receive the last update of resources in URI path representation. Moreover, since the protocol is based on UDP, it supports group communication using IP multicast. In addition, this project included a study that have been conducted among academics, professionals, and hobbyists to show the needs and the correctness of the development road of the CoAP protocol [64]. As a result, a prototype of the full CoAP experience has been released as an add-on for the Firefox browser. Similarly, the Xively project [65] proposes and constitutes a Platform as a Service (SaaP) intended to simplify the connection of applications, objects, devices, and users to the Web. The ThingSpeak project [124] proposes an open source application platform and API that aims to facilitate data storage and retrieval from IoT devices on the Web. Finally, the IoTivity framework [115] is a promising open source framework for a collaborative WoT, that allow smart things to discover, expose their capabilities and work together. In spite of the limited set of their supported protocols, one of the advantages of the aforementioned platforms is their openness to different hardware profiles

5. ACCESS CONTROL FOR COLLABORATION

The Internet of Things promotes a widespread adoption of smart devices. Thus, more data are being collected on people than ever before. The repercussion of any gap in security will have huge effects on personal security and privacy. Authorization and access control are a highly important dimension of the security problem.

Therefore, Kerschbaum [56] proposed an attributes-based access control model for mobile physical objects. This later extends the attributes to include information about the objects' trajectory in a supply chains. In addition to a trajectory-based policy that has been integrated to provide a mutual access authorization and control.

Shi e*t al.* [61] extended the attribute-based access control model to prevent unauthorized access to the search engine of an EPCglobal network. In fact, the Secure Discovery Service (SecDS) system provides a variety of fine-grained access control policy implementations to protect the sharing of product information in RFID supply-chain networks. From a service-oriented perspective, Zhang and Liu [107] proposed a workflow-oriented and attribute-based access control model to treat access control issues. Attributes related to the subject, the resources, the environment, and the task to have authorization for, all these parameters have been taken into consideration to grant permissions to subjects.

Similarly, extending the role-based access control model (RBAC) was claimed by Zhang *and* Tian [126] to enhance the security in a service-based IoT infrastructure. The paper introduced the incorporation of contextual information in RBAC as a way to produce more efficient mechanism for access control for web service application.

Following the same vision, Liu *et al.* [45] proposed a authentication and an access control model for the IoT. The adopted access policy inherits from the RBAC mechanism, while the authentication process was based on Elliptic Curve Cryptography keys.

Mahalle *et al.* [73] based their suggested access control model on devices capability and identity. The Identity Authentication and Capability-based Access Control (IACAC) scheme creates the capability based on the identity to grant access on the local network. This scheme still not fully suitable for small devices within the IoT.

Following the same vein, [75] promoted the use of capability-based security approach to managing access control in the Internet of Things. Indeed, a capability defines the resources, the subject and the granted rights and authorisations. Key features supported by the Capability Based Access Control (CapBAC) include delegation and revocation of capability, as well as information granularity and standard capability representation through XML-based languages.

In Lee *et al.* [46,47], authors propose a model that combines location and time with security level to control access to the information within the IoT. The model is named Location-Temporal Access Control Model (LTAC). LTAC is meant to give access to requested operations on a defined node only if the requesting node is located in an appropriate location within the appropriate time interval regarding the object.

Oh and Kim [69], in addition to the context of the thing subject to the access demand, they included the identity and the internet address of the requester to the process of access control. Considering the web of things and REST-compliant resource-oriented web characteristics, they provide a decentralized access permission control structure. By exploiting smartphone built-in sensors, the Context-Aware Platform using Integrated user Mobile sensors platform (CAPIM)[86] is a user authentication and session management based on Public Key Infrastructure (PKI). This platform has been used to manage access to secure area within a building.

6. CHAPTER CONCLUSION

Current solutions for the Collaborative Internet of Things stand on a set of inappropriate models and do not provide the appropriate interoperability, privacy and security handling. Each middleware solution focuses on different aspects in the IoT, such as device management, interoperability, platform portability, context-awareness, security, and privacy, and many more. Even though some solutions address multiple aspects, an ideal solution that addresses most of the required aspects is yet to be designed.

An Internet of Things collaboration model must be designed to provide service and object connectivity structures to transport data from one entity to another. These may be in the form of Service-Oriented Architecture (SOA) and universal data appliance protocols that can be a basis for developing federated networks and services. This would allow people to design "plug-and-play" applications.

Today's IoT-intended approaches do not emphasize the provision of security and authentication at the entities and devices level. Authenticated access to naming and identification data should be deployed as part of the look-up and resolution processes. Such authentication ensures granting access to identification data only to applications that have the rights to do so. In addition to preventing risks associated with naming assignments, such as forging identifiers.

In this chapter we have shown the state-of-the-art of current approaches in the area of IoT. In addition, we have highlighted important research directions toward solving IoT problems. Hence, in the next chapter, we discuss the limitations of existing collaborative Internet of Things approaches and technologies and will describe specifically the problem statements..

Chapter IV
CoFog, an Architectural Model for Collaborative Fog Computing

The Internet of Things is paving the road to a future where autonomous objects sense, actuate, interact and react with each other. That is, the human part in machine communication is blurring into a more sophisticated device-to-device communication model. In this model, connected objects will be able to reach other objects in order to provide aggregated and collaboration-based services. Thus, leading to more sophisticated applications with added value.

As we have introduced in the previous chapters, Fog computing has gained more interest lately. As opportunities arise with Fog computing, it is crucial to come up with an architectural model that suits the application scenarios intended for this paradigm. Hence, in this chapter we present the main aspects and perspectives to consider in Fog computing. Beside the principles and the rationales behind the conception of the architectural model, the chapter provides an overview of the architecture and its main supported features and benefits.

1. General Concepts of the Fog Architecture

Fog computing platforms tie together connected devices and other Internet and web-based services. They contribute to defining a reference architecture for the IoT, whilst taking into consideration diverse technologies and a wide range of standards. The Fog infrastructure must allow devices, users and applications to connect to its services. It should be able to coordinate and manage connectivity issues, in addition to ensuring the security and the privacy of exchanged data. The Fog infrastructure must comply to these requirements while overcoming the interoperability issues between the enormous number of connected devices.

Additionally, Fog platforms needs to reduce the complexity of collecting and processing massive amounts of data. This requires considering issues such as openness and scalability while offering features such as self-governance, self-management and context-awareness. We highlight here the openness since it guarantees and encourages building solutions upon open-source technologies. Hence, reducing the cost and opening the doors to more innovative ideas and creative solutions. The following list summarizes the fundamentals that we believe the Fog infrastructure ought to incorporate:

- Abstraction of physical objects to enable uniform access to heterogeneous resources via multiple communication protocols such as CoAP, MQTT, REST,etc.
- Virtualization that provides services, such as look-up mechanisms, that bridge physical network edges and offer a set of consumable services.

- Data management primitives that enable data definition, storage, cashing, interrogation, in addition to functionalities of data aggregation and event-based management.
- Semantic representation for modelling knowledge about devices, data and services.
- Security and policy framework that implements access control mechanisms and identity management for authentication and authorizations policies.
- Networking communication both internal and across platforms leveraging means for self-management, self-configuration, self-healing and optimization.
- Open APIs to support platform extensibility, quick development of Fog applications and tools upon the top of the platform.
- Data analysis to provide real-time processing based on user-defined rules for simple or more complexes capabilities such as decision-making, data visualization and reporting.
- Development toolkits for fast and comprehensive development and integration of devices, services and applications.

Fog computing is intended to provide an intelligence layer composed of many Fog nodes (a.k.a. *Fogs*). This layer will bring some of the Cloud computing capabilities to the edge of the Internet in a distributed and decentralized fashion. This layer can behave as a tier in a multitier-hierarchical architecture, where the Cloud plays the top role of coordination and analysis. Or, the elements of this layer can behave in a decentralized way: The *Fogs* can provide services, take decisions, grow and scale in-demand, and provide collaborative means even without the need of a central tier (i.e., the Cloud).

The Fog nodes will be available in large numbers and widely spread across large geographical areas. However, we foresee that a given node will essentially make use of local devices, and serves local users (user applications, mobile devices …). Still, it can use neighbouring nodes or remotely use distant (geographically) ones, in addition to nodes at a higher level in the network. Indeed, *Fogs* can be created at a local (low) level (e.g. routers, network switching hub, local servers) or deployed in a higher level as on Internet service providers (ISP) infrastructures (e.g. gateways). Thus, gaining the ability to better adjust to their locations functionalities and to the needs of their users and applications. On the one hand, Fogs will gain the ability to access local and nearby resources such as mobile devices, sensors, actuators, user-managed servers, and access to local information such as network-related data and real world-related data. On the other hand, it is crucial for *Fogs* to deal with the mobility nature of resources and the scalability of the entire ecosystem.

Figure 14 is intended to help understand the requirements and the rationales of *Fogs*. It depicts an abstraction of the composition of a typical Fog Computing architecture.

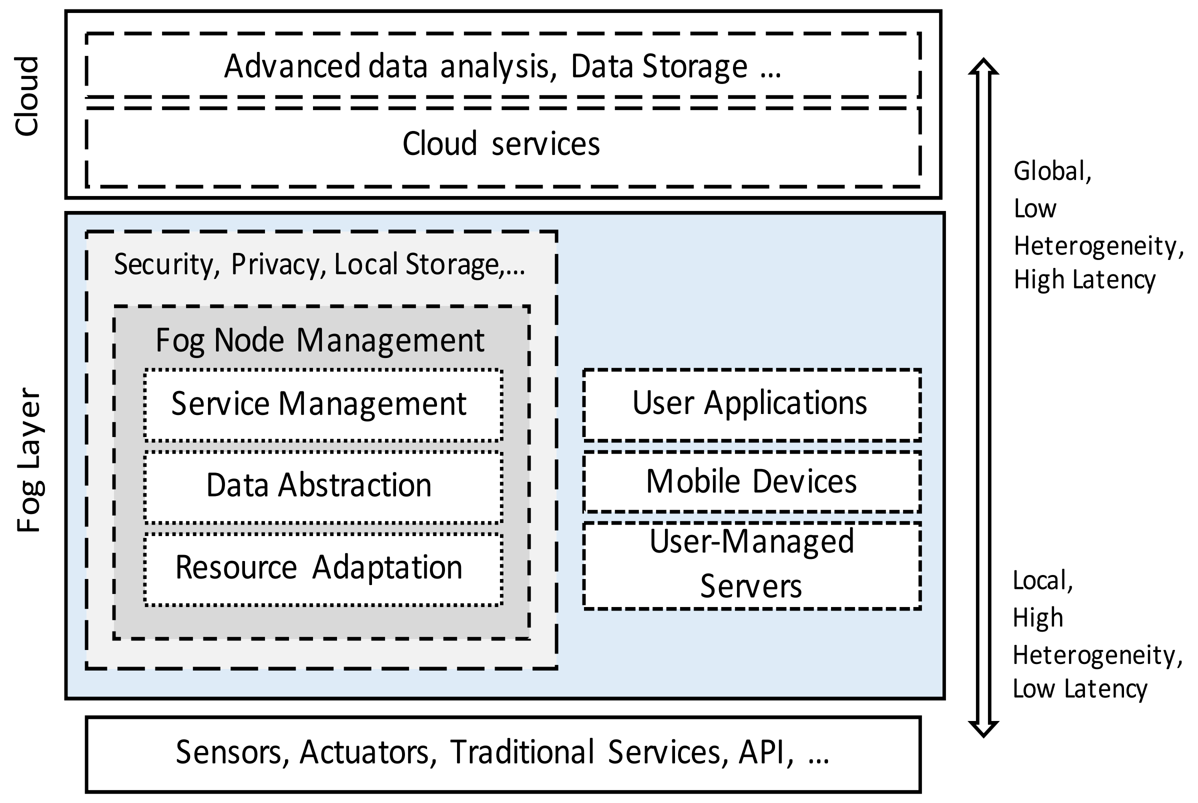
Cloud
Advanced data analysis, Data Storage ...
Cloud services
Fog Layer
Security, Privacy, Local Storage,...
Fog Node Management
Service Management
Data Abstraction
Resource Adaptation
User Applications
Mobile Devices
User-Managed Servers
Sensors, Actuators, Traditional Services, API, ...
Global, Low Heterogeneity, High Latency
Local, High Heterogeneity, Low Latency

Figure 14. Perspectives of a the Fog computing layer architecture

Fog-based applications will benefit from low latency and predictable delays as they are using their surroundings capabilities: computational, sensing, etc. Furthermore, with their ability to access physical aspects of the environment, *Fogs* promote more context-aware applications and use-cases, in addition to a better quality of service (QoS) and more availability since services are hosted locally by the network infrastructure.

On a final note, Table 2 summarizes the overall functionalities and features of Fog Computing compared with those of the Cloud.

Table 2. Comparison of features between Fog and Cloud Computing

	Fogs	**Cloud**
Application	Context-aware, simple analysis, augmented reality, connected vehicles	Advanced analysis, global coordination, centralized control
Latency	Low	Average to high
Storage	Transient, short duration	Long term
Availability	Higher (local services)	High
Scalability	High	Average
Mobility support	High	-
Architecture	Decentralized, distributed, n-tier	Centralized
Hardware	Heterogeneous user devices, sensors, tags, actuator, user-managed servers, edge network	Servers, data centers
Local awareness	High	-
Geographic span	Local	Global

This section presented the core principles and intentions that guided the definition of the Fog architecture. In light of this, next section covers the main layers and features of our *CoFog* architecture.

2. Overview of the Architecture

Although motivated by the issues of Cloud-centric vision of IoT, Fog Computing has many different characteristics. It presents many new challenges, such security and privacy, programming abstractions and models, computing and storage constraints, resource provisioning and management, and distributed Fog management. The proposed architectural model for Fog Computing aims at allowing flexible design choices and user-specific schemes.

Figure 15 depicts the logical separation of the architectural components and the main functional aspects of the architecture. The architecture defines four layers that facilitate the use of real-world resources, existing services and APIs, and the internal functionalities. Many Fog application scenarios need strong requirements of low-latency and dynamic adjustment to changing contexts. Such scenarios can benefit from the instantiation of the architecture capabilities in order to execute and achieve their tasks. From bottom to top, we propose the following levels:

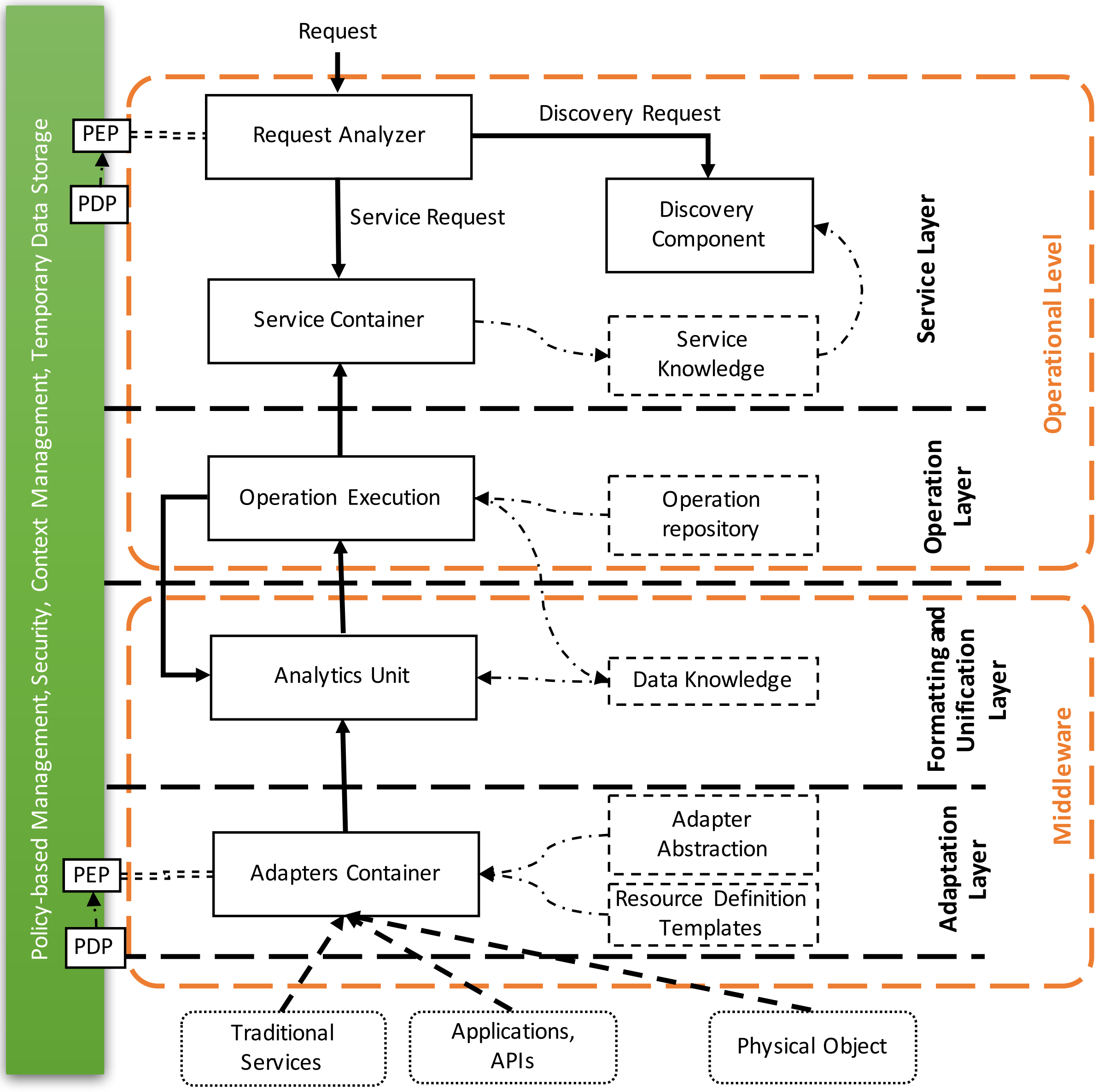

Figure 15. The CoFog architectural structure of a Fog node

- **The Middleware Level** ensures the abstraction of the physical objects, in addition to functional leverage through resource Adapters and data Unification and Formatting. More details on the purpose and functionalities of the Middleware and its two composing layers are presented in the next section.

- **The Operation and the Service Layers** constitute the Operational Level. This is the brain of the whole architecture. It provides runtime management and execution environment for the pool of requested services. In addition to dynamic creation of services using Service Knowledge and defined Operations. The next chapter presents more details on the internal functioning of the Service Discovery, Service delivery, Operation definition and execution, etc.In addition to these two levels, Security constitutes the third plane of this architecture.
- The Security Level crosscuts all the architecture layers. The Policy Enforcement Point (PEP) is meant to monitor resource-data links and intercept service requests. Intercepted events are evaluated by the Policy Decision Point (PDP) against access policies and rules. The result of a policy evaluation may allow or deny the execution requests. Chapter VI details the access control model designed for this architecture. It is noteworthy that due to its vertical arrangement in this architecture, security requirements may be enforced across the different levels.

3. THE MIDDLEWARE LEVEL (MDL)

The Internet of Things is a nest for a huge number of heterogeneous devices and a source of huge amounts of data. The underlying swarm of data sources comprises huge heterogeneity of networked devices that range from simple physical sensors and actuators to virtual objects and classical web services. Abstraction is needed to make data and data sources uniformly usable across divers set of application domains without requiring prior knowledge about embedded systems. That makes the Middleware Level a very important part of the architecture. As depicted in the figure above (Fig. 10), the Middleware Level comprises two layers: The Adaptation, and the Filtering and Unification layers.

3.1. THE ADAPTATION LAYER

The Adaptation layer grantees an abstract interfacing with the underlying resource infrastructure. It provides generic means to define sensors (devices) and virtual objects. In addition, the layer hosts a set of sensors' Adapters and offers mechanisms to manage and hold this set of adapters' definitions and configurations. The general functionalities of the Adaptation Layer are presented in Figure 16.

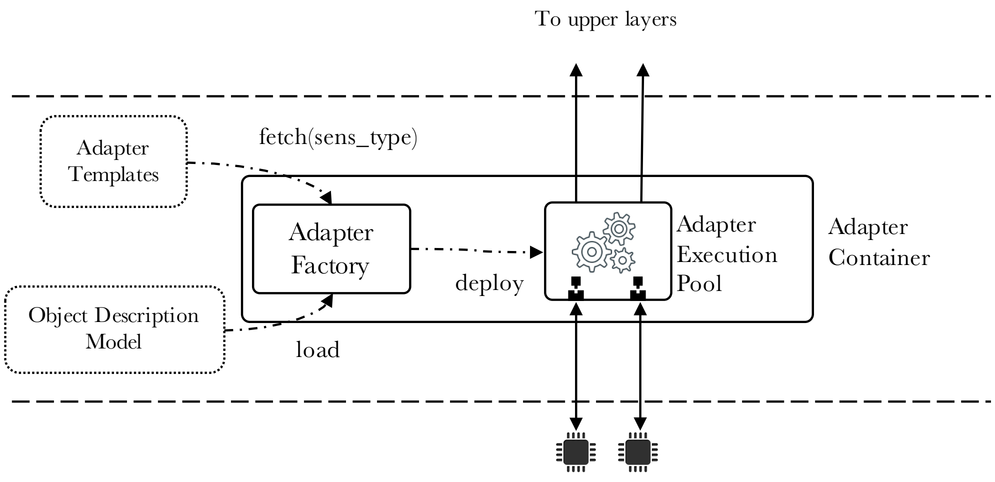

Figure 16. Adaptation layer functional architecture

This figure shows the core components of the level, which are reflected and instantiated in the use case of Chapter VII. At the heart of the Adaptation layer we find the ***Adapter Container***. This component hosts the execution environment of the deployed ***Adapters***. In addition to the Adapter Factory that is responsible of instantiating the appropriate adapters for each connected data source (sensor, API, etc.). Moreover, the level contains the ***Adapter Templates*** that constitutes the core of the informational model describing the set of sensors. The Adapter Container also hosts various default and optional functionalities. In the following sections, we present core concerns of the Adaptation level in more detail and show how the key components have to be reflected in prospective use cases.

3.1.1. Sensor description model

The huge amount of sensors that are and will be deployed in the IoT imposes the need for an abstract information model to describe the heterogeneity of these devices. Consequently, the information model was developed as a generic model. As such, it can be used to describe a wide range of IoT devices, either within simple infrastructures such as sensors and actuators or within more complex technological infrastructures like smartphones or traditional web services. The definition of appropriate metadata into ontologies gives the ability to create semantically enriched representations, which reflects in the virtual world the specification, the capabilities and the commands of heterogeneous IoT objects [32,48]. In addition, the need to describe virtual data-sources (non-physical object) necessitates the definition of relevant metadata that will describe the features of such objects. Thus, the Object Description Model includes a set of metadata used to describe properties and associations of both physical and virtual objects, and that in one common data structure. The next figure (Figure 17) presents this designed informational model.

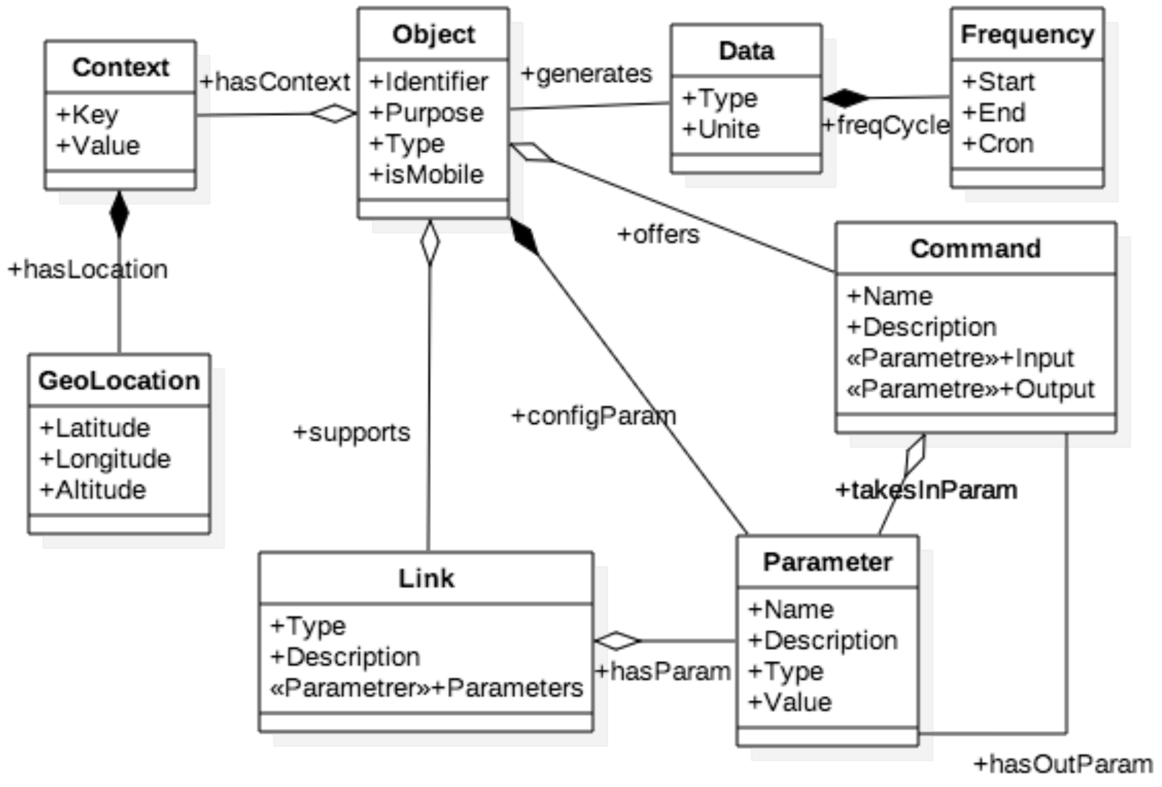

Figure 17. Object Description Model

As illustrated in Figure 18, the above-mentioned model can be instantiated as YAML [17] description file. This file encapsulates information about the hardware sensors, its generated data, and the protocol that can be used for communication purposes.

```yaml
Identifier: c7d6f5a1-2910-436a-a939-d6fdeedceae
Type: simple
Purpose: >
    This is a simple
    sensor for temperature
Data:
  Purpose: Temperature in Celsius
  Type: Double
  Frequency:
    Start: 2017-10-01 21:59:43.10
    End:  2017-11-30 01:59:43.10
    Rate: 30
Context:
  Location:
    Latitude: 46.804334
    Longitude:  -71.980912
    Altitude: ~
Link:
  Type: SerialPort
  Port Name: "/dev/tty.usbserial-A9007UX1"
  Time Out: 2000
```

Figure 18. Temperature sensor YAML description

3.1.2. Adapter Container

The Adapter Container is the core component acting, in a way, as the abstraction component between the heterogeneous physical world and the homogeneous Operational Level. The key back-end interfaces of the Adapter Container with the underlying heterogeneity are the *Adapters*.

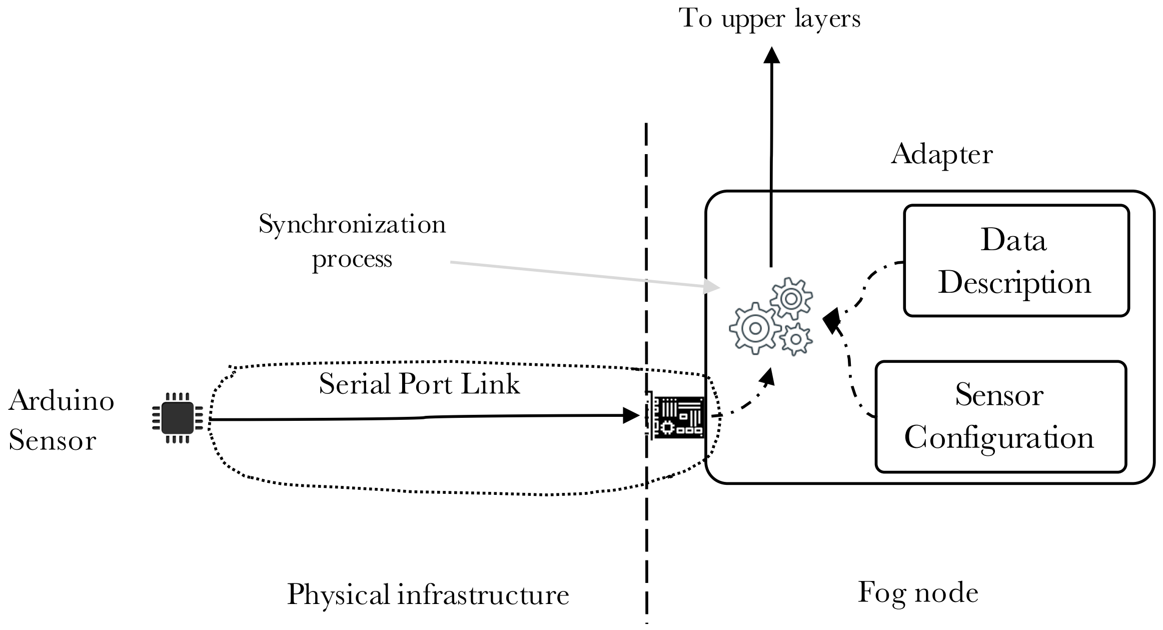

Figure 19. SerialPort Adapter for Arduino sensor: upstream

Adapters are instantiated by the *Adapter Factory* using the *Object Description Model* as input. This description provides the *Adapter* with required information about the object it represents, the data it generates and the type of *Link* needed to ensure communication with that object. *Adapters* may also incorporate object-management functionalities for updating information such as geo-positioning data, data rate, etc. Upon its creation, *Adapter* instances are deployed and run by the *Adapter Execution Pool*. Those instances are responsible for delivering data to the upper services to be formalized and analyzed (Figure 19). Furthermore, they are in charge of using and updating the represented objects.

The *Adapter Container* manages the execution and the life-cycle of the deployed *Adapters* within its run-time environment. Such duty includes the identification, allocation and the destruction of *Adapters*. The Fog ecosystems interact with a large number of physical and virtual data sources. Which implies that each object instance has to be uniquely distinguishable. Thus, the system behaves as an identification authority for the entities it contains. Object identifiers have to be unique and give an informative description of the referred Object. The architecture provides an umbrella under which object are stored, in addition to a naming schema that defines the rules for naming the resources.

3.2. FORMATTING AND UNIFICATION

The *Formatting and Unification Layer* is responsible for delivering information description methods and data filtering mechanisms. It offers a unified and homogeneous view aiming the standardization of the filtered data. The resulting data are consumed through services. Thus, inheritably loose coupled and discoverable.

Indeed, the main difference between data analytics at the Cloud level and at the edge level of the network is the quantity of data. That is, while data analytics at the edge of the network is performed continually on flowing streams of data, analytics at the Cloud level is dedicated to large amounts of data at rest. Hence, we consider the analytics at the Fog node as a successive processing channels [18] of real-time flows of data.

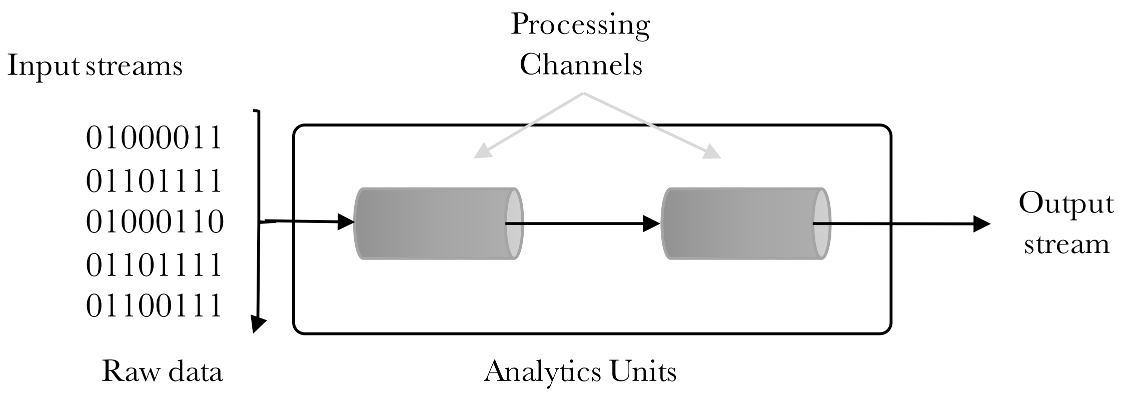

Figure 20. Fog analytics channels

The *Formatting & Unification* layer handles the heterogeneity of the infrastructure from a data semantic perspective. Indeed, various aggregation algorithms can be implemented at the very edges of the network in order to provide enriched data. The Fog data-stream analytics can be broken into three simple stages, illustrated in Figure 20.

- Raw data input: the raw data coming directly from the object data-source (i.e. sensors) through the associated Adapter into the analytics unit.
- Analytics Unit (AU): the AU acts on the raw data by filtering them, combine or separate them as needed. For instance, it may organize them by time windows or execute divers analytical functions.
- Output data streams: the data that is organized, well formatted and ready for delivery to the top layer of the system.

Each *Processing Channel* within the *Analytics Unit* can perform a real-time analysis function such as:

- Filtering: Objects in the IoT are likely to generate an enormous quantity of data. However, most of these data can potentially be irrelevant. For example, a temperature sensor can be configured to send data on a regular basis, simply to confirm its reachability but not upon temperature changes. Hence, most of this data is not really relevant and can be ignored. That is, the filtering function is in charge of identifying important data.
- Time windowing: Time context is a crucial aspect in real-time data streaming. Such operation can be used to correlate average data values from a sensor's real-time data on a time-window basis. Figure 21 illustrates a *Processing Chanel* that reports every half-hour, the input data from a temperature sensor stream in a one-hour window.

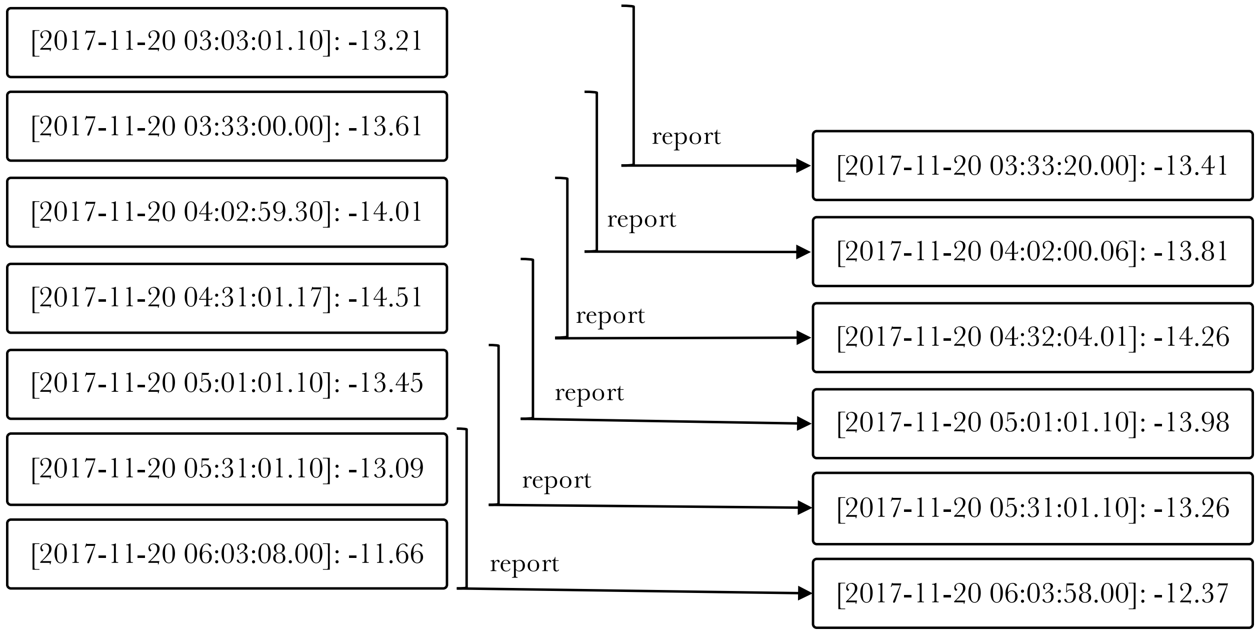

Figure 21. Time window, an analysis function example

- Formatting: Similar to advanced data analytics in the Cloud (data integration, data warehousing), Fog nodes must implement some simplified variation of data transformation. Such function is used to convert filtered data from one format or structure into a form that can be used for other purposes. Such operation can be as simple as converting temperature data from Celsius to Fahrenheit.

4. Chapter Conclusion

This chapter has given an overview of the proposed architecture. This architecture constitutes an approach to solving interoperability issues close to the physical level. It offers an abstraction from any domain-specific scenarios to concentrate on domain-agnostic perspectives that Fog Computing based solution may have in common. Yet, the adoption of such an architectural model could be achieved in a strait straightforward fashion, as shown in Chapter VI. We have covered the functional building blocks of the first level, the Middleware, in addition to the description of the operational behaviour and the flow of information within both the *Adaptation* and the *Formatting & Unification* layers.

In the next chapter, we continue the presentation of the upper level of the proposed architecture. The *Operational Level* is intended for service presentation, management and transformation. An environment embracing such model will provide means for early data analysis, hence low latency and real-time responses. In addition, to providing an ecosystem for direct collaboration between services leading to more sophisticated applications.

Chapter V
THE COFOG OPERATIONAL LEVEL: SERVICE AND OPERATION LAYERS

Cloud Computing takes advantage of a fairly predictable environment of homogenous computing, storage, and networking components to offer higher service aggregation without degrading performance. In other words, the Cloud offers an efficient alternative to owning data and processing centres. Thus, it liberates the end users from the specification of many details. However, Cloud Computing fails to meet the requirements of IoT in term of latency-sensitive applications, mobility support, wide geo-distribution and high location awareness. On the contrary, while Fog Computing might compliment the Cloud at the edge of the Internet, it bestows new breed of services and applications meeting the previously cited requirements. Therefore, Fog applications do not ultimately fit the Cloud Computing paradigm, and they include:

- Applications that require very low and predictable latency: shop-floor monitoring, gaming, video conference.

- Applications with high geo-distribution nature: wind farms, pipeline monitoring, environmental-sensing networks.
- Services for fast and mobile participant: smart connected vehicle, connected rail.
- Large-scale distributed control systems: smart grid, connected rail, smart traffic systems, pollution monitoring.

The application area of the Fog paradigm is large and crosscuts multiple application fields. Therefore, it needs a common platform that supports a wide range of application domains, rather than single solutions for each domain. Hence, analogous to the Cloud, the Fog architecture must provide a service layer that leverages resource virtualization with dynamic-service orchestration [7]. Such ability enhances the scalability and the automation of service management. In addition, the service layer must offer a highly abstract and generic APIs in order to accelerate and ease the deployment of Fog service-based systems.

In the previous chapter, we have introduced our proposed architectural framework. In this chapter, we present the architecture's *Operational Level*. We provide an overview of its principles, components, and the main supported features. In addition, this chapter comprises sections dedicated to more details on each component of this level, along with tools for Service and Operation definition and management.

1. The Operational Level, an overview

Fog nodes provide a large number of services with a wide range of capabilities. Orchestrating such services, on a large number of nodes, requires dynamic and policy-based life-cycle management. This orchestration is achieved in the *Service Layer* via the following components:

- A *Service Template Repository* that facilitates the introduction of new types of services.
- A *Service Factory* in charge of the process of service instantiation that satisfies a given *Service Request*.
- A *Service Container* capable of bearing the management functionalities and the performance requirements of edge devices.

Furthermore, the *Operational Level* augments device-based static services with more complex dynamic services. That is, static services leverage virtualized devices by presenting their data and capabilities as usable services (Figure 22).

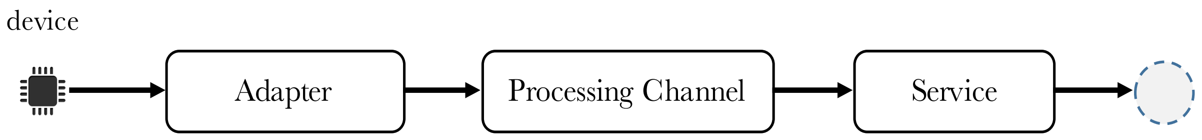

Figure 22. Graphical representation of a static service

Whereas, dynamic services augment the existing devices virtualization and/or the provided static services. The definition of such mechanism is provided by the *Operation Layer* through a set of *Operations*, and based on the service request and contextual information (Figure 23).

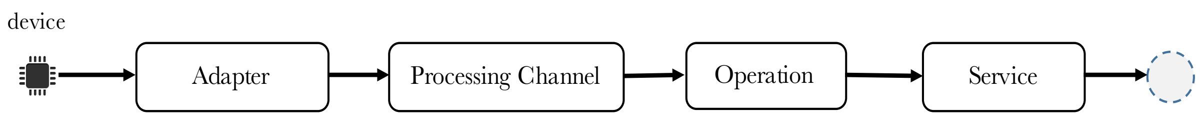

Figure 23. Graphical representation of a dynamic service

Services are accessed via *Service Requests*. The *Service Request Analyzer* function matches such incoming requests with the corresponding service template. Thus, constructing contextual constraints as expressed by the request, and eventually handing on the resulting service specification to the *Service Container* for execution.

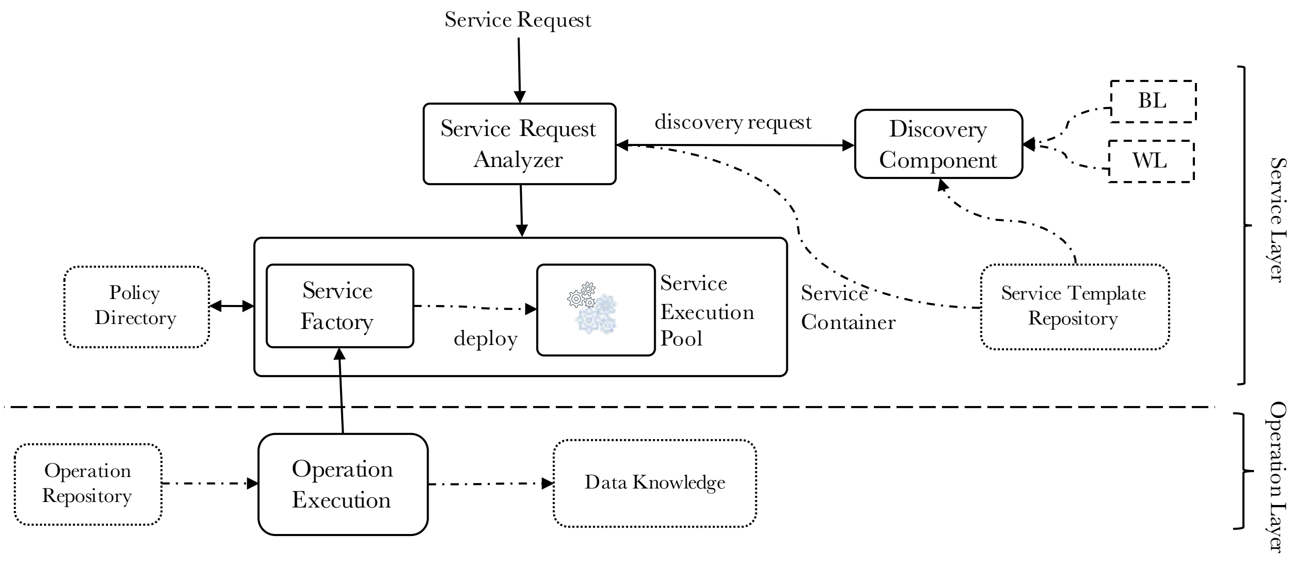

Figure 24. Components in the Operational Level

Finally, the discovery of services is based on a *Data Sharing Model*. This model relays on a propagation query-response process. To summarize, Figure 24 illustrates an overview of the main functions of the *Operational Level* and the architecture entities they interface with. Next sections present more details on the various components of this level, alongside their operational behaviours.

2. SERVICE MODELING AND MANAGEMENT

2.1. SERVICE TEMPLATE

A Fog node, in its approach of functioning, supports and offers an arbitrary number of service-types along with their instances, and this, in one or more application domains. Therefore, to introduce a new type of services, a *Service Template* is added to the *Service Template Repository*. This way enables the platform to support a hypothetically wide range of *Service Requests*. Such requests, when issued, they express the goals and the needs that users and applications ask the platform to fulfil. Hence, the reception of a service request result in the selection and instantiation of a service template according to the request provided parameters.

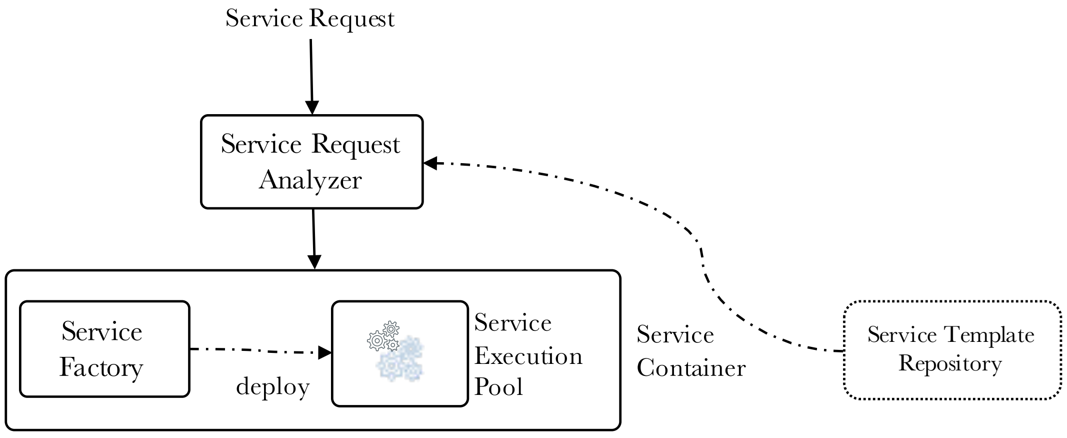

Figure 25. Service container: template instantiation

As shown in Figure 25, the *Service Container* component is responsible for performing the template instantiation (Service Factory), and produces a service execution order (Service Execution Pool). In this perspective, a service is considered as a faithful representation of one or many virtualized data sources (sensors, API, web services) and/or data consumers (actuators, controllers). Thus, a service template must comprise mainly a set of this service type's identifiers, parameters, capabilities and commands; and it is defined as follows:

A service S is couple **<data, context>**, where:

- $data = < \mathrm{d}, \mathrm{frq}, \{\mathrm{opt}\} >$ such as:
- **d = {t, u},** where **t** is the data type and **u** is the unit of this data type,
- **frq = <start, end, cron>**, where **start** represents the start date, **end** the end date, and **cron** is a Unix Crontab-like expression that defines the frequency at which the data is collected,
- **{opt}** might be used to specify other options in the form of a set of couples **<attribute, value>.**
- $context = < \mathrm{lat}, \mathrm{lon}, \{\mathrm{opt}\} >$ where:
- **lat** and **long** represent, respectively, the latitude and the longitude of the geographical location of the IoT object,
- **{opt}** might be used for including other context-related information in the form of a set of couples **<attribute, value>**.

For instance, Service Templates may be stored in a YAML [17] format and queried by the Service Request Analyzer component. As illustrated in Figure 26, this description includes service's parameters and features that need to be met.

```
Service Template:
  Name: Smart Temp
  Description: >
      Smart temperature service
  Data:
    - Type: Temperature
    - Unite: Celsius
    - Frequency:
      - Start: 2017-10-01 21:59:43.10
      - End:  2017-11-30 01:59:43.10
      - Cron: 30 * * * *
  Context:
    - Location:
      -
        Latitude: 46.804334
        Longitude:  -71.980912
        Altitude: ~
  Command:
   - name: getTemperatureC
   - parameters: ~
```

Figure 26. YAML service template for a Smart Temperature Service

2.2. POLICY-BASED MANAGEMENT

The *Service Container* provides a policy-based management framework. This framework is convenient to administer business policies, manage, and monitor the Fog platform. For instance, in a concrete scenario, administrators can interact with such orchestration framework via an intuitive user interface.

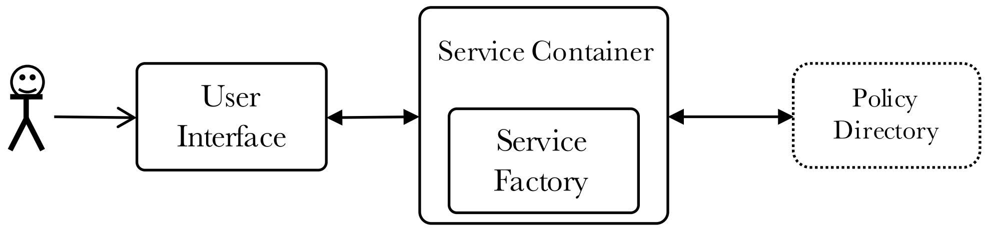

Figure 27. Policy-based management framework

Furthermore, by defining *Generic Policies*, the framework can be extended to support a wide variety of policies. The following are few examples of the functions that policies may include:

- Policies to specify thresholds for load balancing such as the maximum number of users, connections, CPU load, etc.
- Policies to specify QoS requirements (network, storage and computing) associated with services such as minimum delay, maximum rate, etc.
- Policies to configure service instance in a specific setting.
- Policies to associate power management capabilities of the Fog node.
- Policies that specify how and what services must be chained before delivery – e.g., healthcare services before gaming.

Business policies specified via the use of the framework are pushed to a *Policy Directory* (Figure 27). The *Service Container's* policy management may be triggered by an incoming service request, service instantiation, etc. Hence, relevant policies are gathered from the policy repository - i.e., those which are related to the service. In addition to retrieving meta-data about currently active service instances. Both these two sets of data provision the life-cycle management of services on a Fog node. The *Service Container* may also reach out to the policy repository to identify the Fog node and network configuration policies while provisioning the new instance. Such management functionality provides better resiliency, scalability, and faster orchestration for geographically distributed deployments.

2.3. Service generation and execution

Based on the logical description of the requested service, the *Request Analyzer* instructs the *Service Container* on which service instances to construct, deploy and execute. Therefore, the analyzer should obtain all information and context constraints from the *Service Template Repository*, before issuing service execution instructions. The advantage of this architectural resides not only in enhancing the modularity and reusability qualities, but also in the fact that service-related dependencies are analyzed prior to spending valuable real-time execution resources.

As mentioned in the previous section, internal management of the *Service Container* is based on a set of policies. Hence, the instantiation of a service is subject to the satisfaction of the adequate policies. Upon the satisfaction of such policies, the *Service Container* send an instantiation order to the *Service Factory* with the *Service Template*. The enriched data provided by the instantiated service may be presented for example via a RESTful HTTP Request-Response or an MQTT Publish-Subscribe interface.

That is, the “normal” service delivery is based on the presence of the requested service among predefined services. However, the smartness of a Fog node resides on its capability to provide dynamically constructed services. Therefore, triggered by the absence of a service, the *Operation Manager* tries to fulfil the service request by using a set of *Operations*. This process is the topic of the next section.

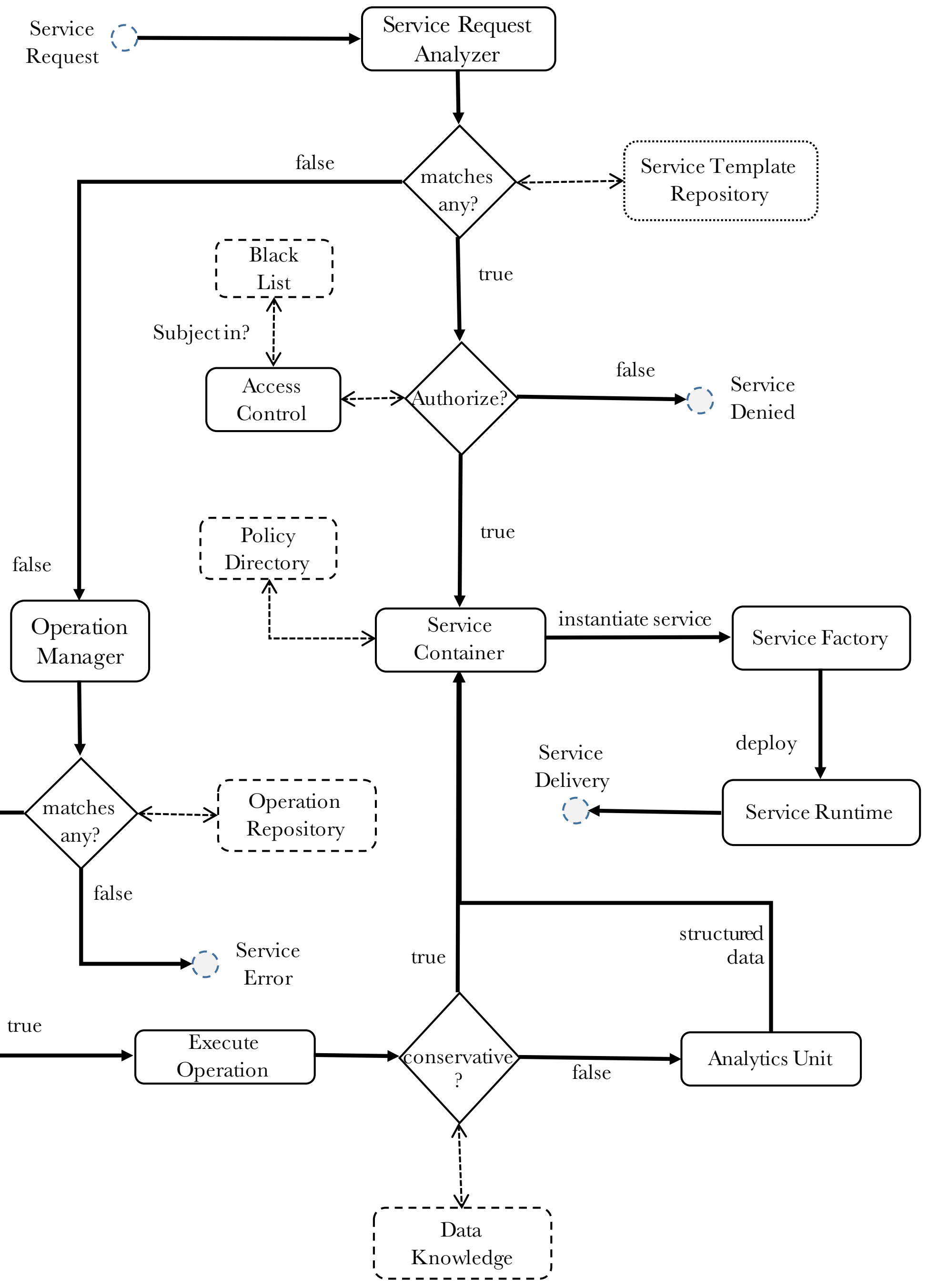

Figure 28. Service execution and delivery process

It is worth mentioning that security, as a non-functional requirement, can be plugged in a given point of the service creation process. For instance, as illustrated in Figure 28, the access control model is deployed at the level of the *Request Analyzer*.

Thus, the Policy Enforcement Point (PEP) can intercept service execution requests. The interception of these request generate events that are signalled to a Policy Decision Point (PDP) component. This late component evaluates security policies and return enforcement actions to the PEP. The result of the policy evaluation may allow, deny, modify, or delay the execution requests in case policies controlling the respective request are already deployed. More information on the access control model are presented in the next chapter.

3. Leveraging service with Operations

3.1. Operation definition

As mentioned earlier in this chapter, the *Operation Layer* provides mechanisms to leverage static services. Indeed, given a set of available web services, a service request and contextual information, the main problem of the *Service Layer* is to automatically find a web service satisfying the request. However, is it possible that the requested service do not exist. In such case, the data from one or many sources (sensor date, classical web services, etc.) are subjected to more treatments, thus ensuring the creation of the desired service. This is achieved by means of a set of operations that are applied to existing services. Those operations may be any transformation, aggregation, or composition primitive. The key element for an automatic Operation execution is through a semantic representation of such applications. This machine-readable representation allows the operation execution-engine to find a correct, consistent and optimal response to the request.

First, to formalize the notion of *Operation* and its composition, let $\mathcal{I}_1, \mathcal{I}_2, \cdots \mathcal{I}_n \mid n \in \mathbb{N}$ be the sets of input parameters, $\mathcal{O}_1, \mathcal{O}_2, \cdots \mathcal{O}_m \mid m \in \mathbb{N}$ the sets of output parameters, and $\mathcal{R}$ a *Relation* of degree $n+m$. An *Operation (op)* defines a set of inputs $\mathcal{I} \subseteq \mathcal{I}_1 \times \mathcal{I}_2 \times \cdots \mathcal{I}_n$, a set of outputs $\mathcal{O} \subseteq \mathcal{O}_1 \times \mathcal{O}_2 \times \cdots \mathcal{O}_m$, and a formula that maps the relation $\mathcal{R}$ to the $(n+m)$-ary relation of all $(n+m)$-tuples from $\mathcal{R}$.

$$op := \mathscr{R} \mid \mathscr{I}_1 \times \mathscr{I}_2 \times \cdots \mathscr{I}_n \rightarrow \mathcal{O}_1 \times \mathcal{O}_2 \times \cdots \mathcal{O}_m$$

Equation 1. The operation's formal definition

We assume that for every Operation (op) invocation with input parameters such as for every parameter $\alpha \in \mathcal{I}_i$ | i: 1 → n, the relation $\mathcal{R}$ returns all the output parameters where for every output parameter $\beta \in \mathcal{O}_j$ | j: 1 → m.

$$\begin{pmatrix} \mathrm{i}_1^1 \\ \mathrm{i}_2^1 \\ \vdots \\ \mathrm{i}_k^1 \end{pmatrix} \times \begin{pmatrix} \mathrm{i}_1^2 \\ \mathrm{i}_2^2 \\ \vdots \\ \mathrm{i}_k^2 \end{pmatrix} \times \cdots \begin{pmatrix} \mathrm{i}_1^n \\ \mathrm{i}_2^n \\ \vdots \\ \mathrm{i}_k^n \end{pmatrix} \rightarrow \begin{pmatrix} o_1^1 \\ o_2^1 \\ \vdots \\ o_k^1 \end{pmatrix} \times \begin{pmatrix} o_1^2 \\ o_2^2 \\ \vdots \\ o_k^2 \end{pmatrix} \times \cdots \begin{pmatrix} o_1^m \\ o_2^m \\ \vdots \\ o_k^m \end{pmatrix}$$

Where $\mathcal{I}_1 \times \mathcal{I}_2 \times \cdots \mathcal{I}_n$ is the domains of input parameters, and $\mathcal{O}_1 \times \mathcal{O}_2 \times \cdots \mathcal{O}_m$ is the range of output parameters.

For instance, let R be the operation that convert temperature from Celsius to Fahrenheit (cTof). This relation is a binary relation over $C \times F$, that maps °C to °F. Such relation is the function $f: C \rightarrow F \mid f(x) = 1.8x + 32$, and would be stored for example in YAML format as follows (Figure 29):

```
Operation:
  Name: cTof
  Summary: >
    Simple Celsius to F conversion
```

```
Input:
  - x: {Type: Temperature, Unit: Celsius}
Output:
  - y: {Type: Temperature, Unit: Fahrenheit}
Formula: $y = 1.8 * $x + 32
```

Figure 29 . An example of a simple temperature conversion operation

We distinguish between two kinds of operations: conservative and non-conservative. A conservative operation (e.g. the cTof operation) is simply any operation for which the result data type belongs to the set of already defined data format. In contrast, a non-conservative operation results in a new data format.

3.2. Request-Operation matching

The elegance of a Fog node resides on its capability to dynamically construct services. As illustrated in Figure 28, the absence of a service that fulfils a given request, activates the *Operation Manager* in order to match the request with the corresponding *Operation*. Matching between an Operation and a Service Request consists essentially of matching all the output parameters of the Operation and the parameters of the request. Hence, request parameters are matched against all the Operations stored in the *Operation Repository* at the level of the Fog node.

The next figure illustrates in details the pseudo-code of the matching algorithm :

```
procedure MatchRequestOperation (sRequest, operationList)
   matchList ← empty list
   for ∀Op_i ∈ operationList do
      if isMatch(sRequest, Op_i) then
         matchList ← matchList + {Op_i}
      end if
   end for
end procedure

procedure isMatch(sRequest, Op)
  reqParam ← fetchParam(sRequest)
  opOutput ← fetchOutput(Op)
  matchDegree ← 0
  for ∀Pr_i ∈ reqParam do
     for ∀OpOut_i in opOutput do
        if Pr_i equals OpOut_i then
            matchDegree ← matchDegree + 1
        end if
     end for
  end for
  if matchDegree / length(reqParam) equals 1 then
      return true
  end if
  return false
end procedure
```

Figure 30. Algorithm for Request-Operation matching

That is, a match is recognized if and only if for each parameter of the request, there is a matching output in the Operation. Thus, the degree of success depends solely on the degree of matching: if one of the request output is not matched by any of the Operation outputs, the match fails.

4. SERVICE DISCOVERY AND DATA SHARING MODEL

Fog services discovery mechanisms enable the search and the discovery of the available services across the Fog nodes. A Fog node uses the Discovery Component to either send discovery request or perform discovery request processing. This section gives further details on such processes.

4.1. DATA SHARING MODEL

The service discovery mechanism is based on a data sharing model (Mds) [117]. Service-discovery process in such model is composed of three phases:

- Service Discovery: it uses whitelists and blacklists to enforce a propagation-like query-response mechanism. Such mechanism allows a decentralized discovery of services.
- Service Selection: the selection of a service passes through 1) a preselection step where both the service request and the service response are compared against each other to determine the rate of correspondence between them, 2) and a selection step where a global rating value is associated to each preselected service. The first service with the highest rating value is then selected.
- Service Consumption: this is the final phase and it refers to the delivery of the service. The consumption of a given service is subjected to the attributes defining such service. For instance, consuming a temperature service must

start and finish following the $start$ and end values defined in the frq attribute provided by the service definition.

4.2. SERVICE DISCOVERY

The service discovery in the data sharing model relays on a propagation-like query-response model. To make the discovery process faster and more accurate, discovery requests are structured by specifying the set of nodes it crosses. In addition, the service template is also added as a search constraint that will be taken into account by the *Discovery Component*. Next to search constraints, the discovery mechanisms consider the access rights regarding the client that performs the discovery request. Specifically, the selection process uses a Whitelist-Blacklist mechanism to enforce a simple and a kind of “friends of my friends are my friends” selection policy. While the whitelist contains actors that a given *actor* trusts, the blacklist contains actors that are to be avoided in the collaboration process. The propagation strategy is depicted in Figure 31.

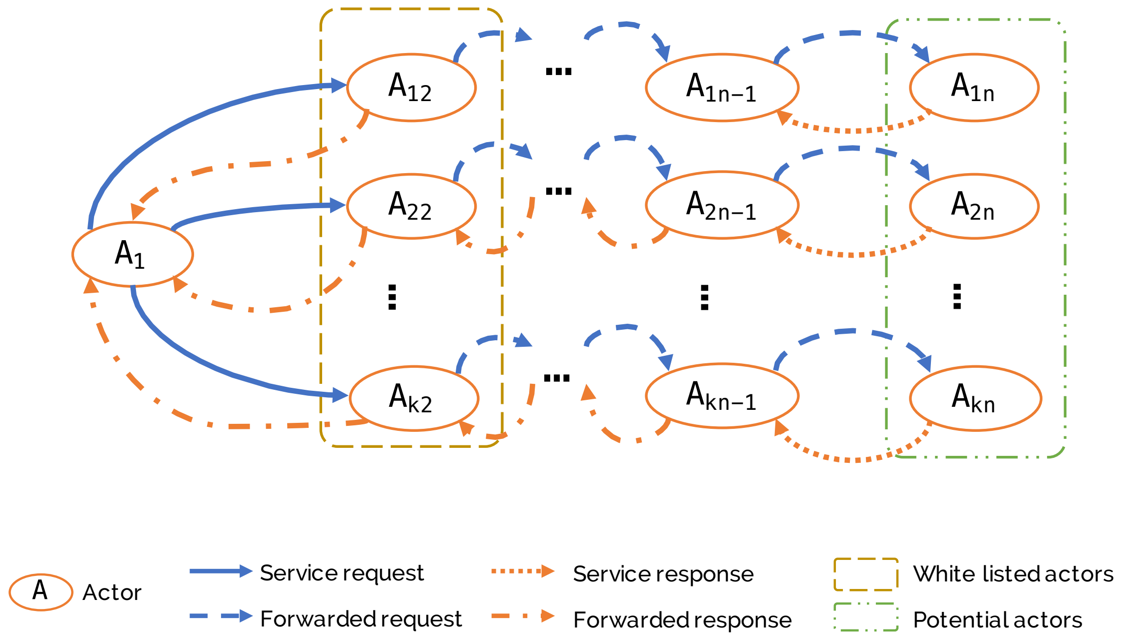

Figure 31. Data sharing model for IoT: Propagation-based service discovery

In this context, it is worth mentioning that an actor represents any object, service or application that might invoke a service discovery request. Actors are defined as follow:

An actor A is represented by a set $\{< O_1, S_1, r_1 >, < O_2, S_2, r_2 >, \dots < O_n, S_n, r_n >\}$, where:

- i $\in [1, \mathrm{n}]$, n is the number of objects an actor represents.
- $O_i \in U_{O,} U_O$ the universe of all IoT objects
- $S_i \in U_{S;} U_S$ the universe of all IoT services

The processing of a discovery request is depicted in Figure 32. First, an actor formulates a service request that describes the needed service. Using its Discovery Component, this request is forwarded to all actors present in the Whitelist.

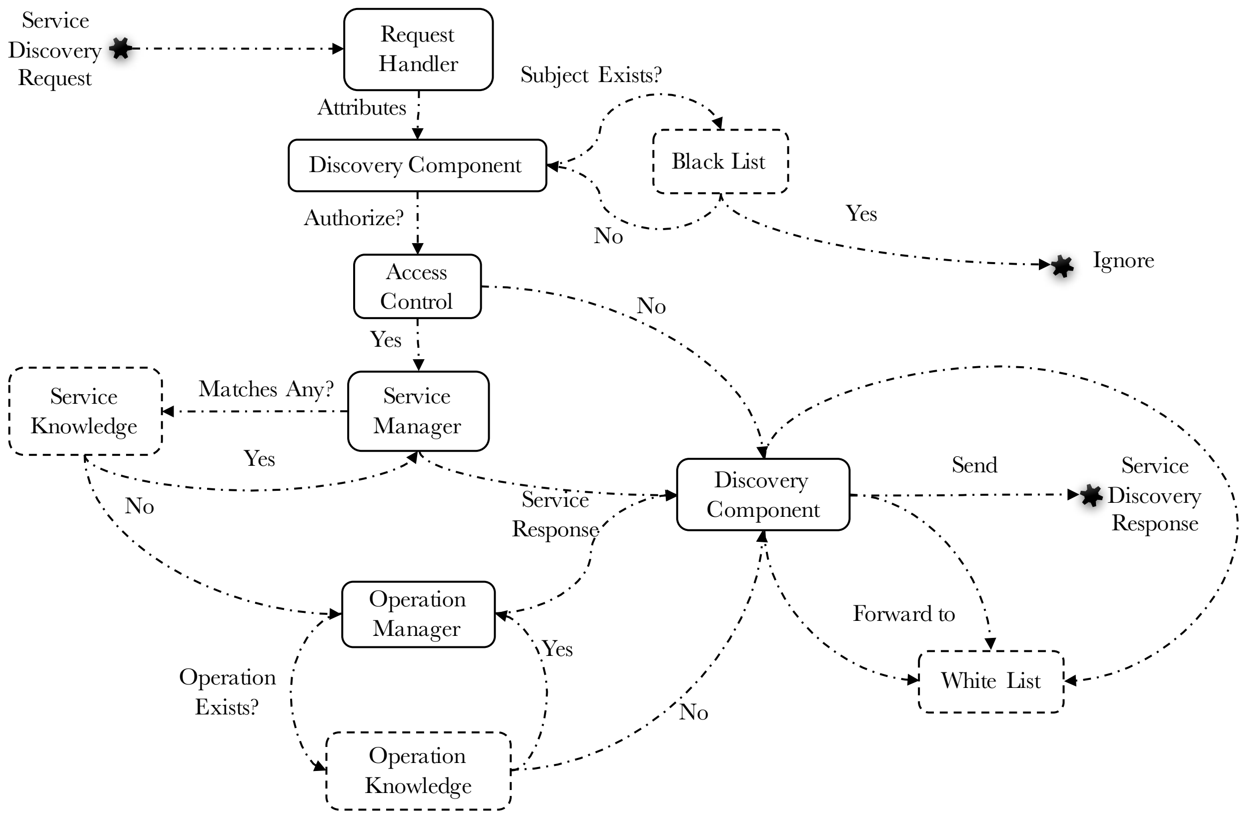

Figure 32. Service discovery request process

Upon the reception of such request, the Request Analyzer forward the request to the local Discovery Component. At this level, the request may be ignored if the request's Actor is listed in the Blacklist. From this point, the Service Container handles the discovery request as a service request, in the same fashion depicted in section 2. In addition, it forwards the request to all actors in his whitelist, along with sending back a discovery response describing the service that matches the requested service.

5. Chapter conclusion

This chapter presented service definition, management and transformation components at the *Operational Level*.. Services that can be leveraged through the application of a set of Operations. The next chapter, raises questions about the principal requirements of Fog Computing regarding security in general and access control mechanisms in particular. The answers to these questions have shaped the design of the proposed access control mechanism, which is presented in the next chaptre.

Chapter VI
A COLLABORATIVE ACCESS CONTROL FOR THE COFOG ARCHITECTURE

Security issues are at the core of collaborative Fog Computing. An infrastructure intended to enable collaboration among devices must target primarily easiness and transparency. However, the security aspect of that same system seeks privacy, authenticity and data integrity. That is, there is a compromise between openness necessary for the collaboration, and the restriction required by a secure system. This compromise gives security problems a multidimensional nature [24]. Among the many dimensions of the security problem, Access Control (AC) is highly important and one of the most critical aspects. Similar to conventional infrastructures, the main function of AC mechanisms is to guaranty right rights to the right subject (user) on the right object. In contrast with the conventional infrastructures, the Internet of Things has its own set of inherited and specific issues and challenges. In this chapter, we present the comprehensive study that helped us designing an access control mechanism for the proposed Fog Computing architecture. We raise questions about the main considerations regarding the requirements of Fog Computing, and the criteria that access control models must meet to be suitable for collaboration in such an environment. The answers to these questions have shaped the design of the proposed access control model

1. Access Control Requirements for Fog Computing

The case here transcends the authentication of subjects and their roles. The Internet of Things is a demanding environment; an access control mechanism must address the requirements of collaboration in this environment. We can summarize those requirements as follows.

- The first thing to come to mind when dealing with the Internet of Things is the scale. The huge number of objects involved makes the coordination and the performance of tasks difficult, it must not affect the scalability of the access control model.
- The actors in such network are highly dynamic. The Internet of Things is by nature dynamic: new devices are continually deployed and the already connected ones are probably physically on the move. The access control model should support changing policies at runtime according to the actors' dynamics.
- The actors are also resource-restrained. Hence, the access control model should perform with a reasonable resource cost.
- The access control has to be suitable for groups and fine-grained access. That is, the level of granularity should not be a difficulty in defining security policies.

- High abstraction of access policies is a requirement. A generic access control model supports more expressive policy definitions. This is a core requirement since the access authorizations are based on a variety of information: data type, frequency, location, service and so forth.
- In addition, high-level definition of authorizations provides better handling of the environmental complexity.
- Although, depending on the design decisions derived from the aforementioned requirements, the access control must provide a suitable and easy to use interfaces for both consumers and devices.

The aforementioned list of requirements is recommended for a Fog environment in the Internet of Things. However, one must admit that listing all the requirements for a practical collaboration is simply pointless, for it is hard to predict all the possibilities and variations within Fog Computing. Instead, we can generalize the criteria in designing an access control model for Fog Computing systems. Such criteria have been deducted from the above requirements and are listed as follows.

- Scalability: To ensure the scalability of the system, the access model should support extensible polices specification and definition mechanisms.
- Dynamism support: The access control model needs to be active in its handling of the management of actors, the assignment of access rights and authorizations.
- Contextual information: Context-awareness is a building block of applications powered by the IoT. Contextual information plays a significant role in any collaboration and in the process of authorization. Moreover, it is important to

know to which extent the access control model is utilizing such information to better secure the whole system.

- Granularity: Often, in the scenario of collaboration, subjects need specific permissions on an object, over a specific period of time, and at a particular step of the collaboration procedure. In such cases, it is not sufficient to have a set of rules for a set of subjects. Thus, while preserving an adequate level of complexity, a fine-grained capable access control model is needed.
- Least authority principle: This well-known principal of security is still valid in the context of Fog Computing. It helps reduce the risk of breaches and the complicity of the model by eliminating unnecessary subject privileges.
- Separation of duties: In this context, the access control model must ensure that a subject has been given only the responsibilities for the current request function.

2. THE COLLABORATIVE ACCESS CONTROL MODELS

We have evaluated a set of access control models that have been proposed and used for the Internet of Things (see Chapter III). This section presents the adaptation of the Role-based and the Attribute-based access control models to fulfil the criteria illustrated in the previous section. The access control model is designed following the service-based data sharing model (Dsm) approach presented in the previous chapter.

2.1. COLLABORATIVE ROLE-BASED ACCESS CONTROL

The Role-Based Access Control model (a.k.a. RBAC) was formalized by *Ferraiolo and Kuhn* [24] in 1992. It was designed to overcome the burden of traditional Access

Control Lists [24] by reducing the cost of access management. Nowadays, the RBAC is still predominant and constitutes the base model upon which many advanced access control systems are proposed. The Collaborative role-based access control (CollRBAC) is an adaptation of the RBAC model to support collaborative Fog computing environment in the IoT. The following are concept definitions and redefinitions required for such adaptation.

2.1.1. ***Definition: Permission***

Given U_O the universe of all IoT objects, U_S the universe of all services, and U_{OP} the universe of all operations, a permission is defined as a triplet $< O_i, S_i, Op_i >$ such that:

- $O_i \in U_O$
- $S_i \in U_S$
- $Op_i \in U_{OP}$

2.1.2. ***Definition: Operation***

An Operation $Op_i \in U_{OP}$ is essentially any access with read or write to the data provided by a given service or the metadata governing the generation of such data.

2.1.3. ***Definition: Role***

Given U_{Perm} the universe of all permissions, a role R is defined as a finite set of permissions such as: $R = \{Perm_i | Perm_i \in U_{Perm}\}$

2.1.4. ***Definition: Role Assignment***

Given U_{Role} the universe of all Roles, and U_{User} the universe of all users, the user-role assignment application $\mathbb{A}_u$ is a non-injective and non-surjective application $\mathbb{A}_u: U_{User} \rightarrow U_{Role}$.

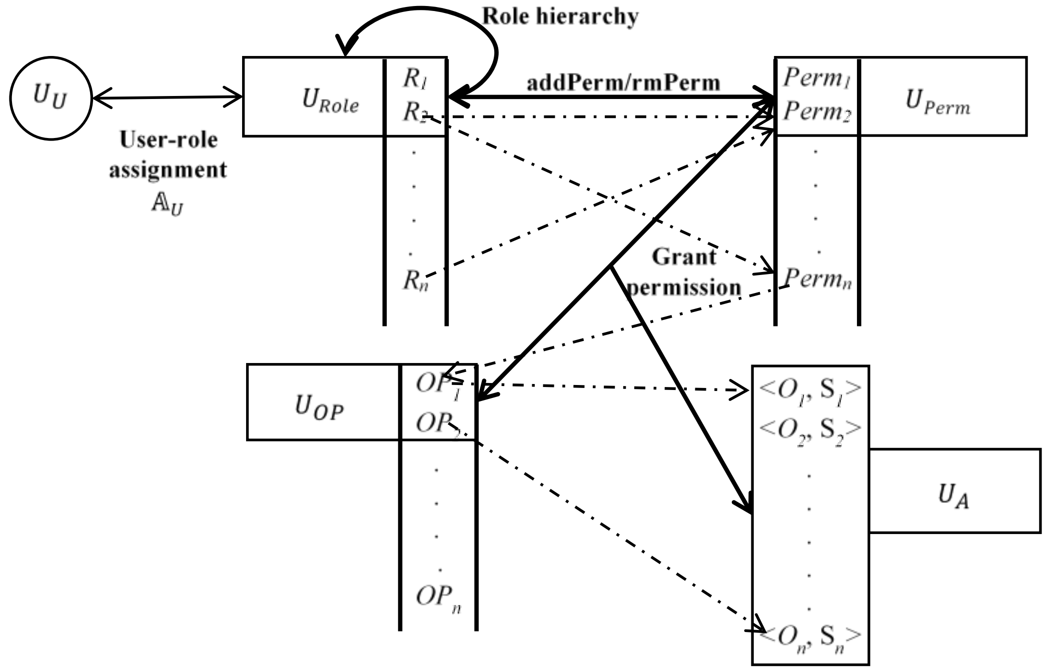

Figure 33. CollRBAC authorization assignment mechanism.

The essence of CollRBAC is that instead of assigning them directly to individual users, permissions are assigned to roles. A permission grants access to a role R for a unique operation Op on a unique service S of an object O (Figure 33.). Hence, roles are created for various task functions, and users are assigned to roles based on their qualifications and responsibilities.

The procedure of specifying user authorizations is divided into two logically independent phases. The first phase, which assigns users to roles: the user-role application assigns a set of roles $\{R\}$ to the appropriate user U_i such as:

$$assignRole(R, U_i) : U_i \in U_{User} \wedge R \in U_{Role} : U_i = U_i \cup \{ R \}$$

The second phase which assigns access rights for operations on objects to roles such as:

$$grantPerm(R_i, \{Perm\}) \wedge \forall\ Perm_i \in \{Perm\}, Perm_i \in U_{Perm} \rightarrow R = R \cup \{R_i, \{Perm\}\}$$

- $grantPerm$ associates a set of permissions $\{Perm\}$ to the corresponding role R within the framework.

$$revokePerm(R_i, Perm_i) \wedge \forall\ Perm_i \in U_{Perm} \rightarrow R = R - \{R_i, Perm_i\}$$

- $revokePerm$ detaches a permission $Perm_i$ from a given role.

2.2. Collaborative Attribute-Based Access Control

The Attribute Based Access Control model [66] (ABAC) uses proprieties associated to both the subject and the object of the access request, in addition to the environmental properties in order to grant authorizations. Upon the reception of a service request, an access permission system allow or deny access to the requested service. When an access request is made, Attributes and Access Control Rules are evaluated by the Policy Enforcement and Decision mechanism to provide the access decision (Figure 34).

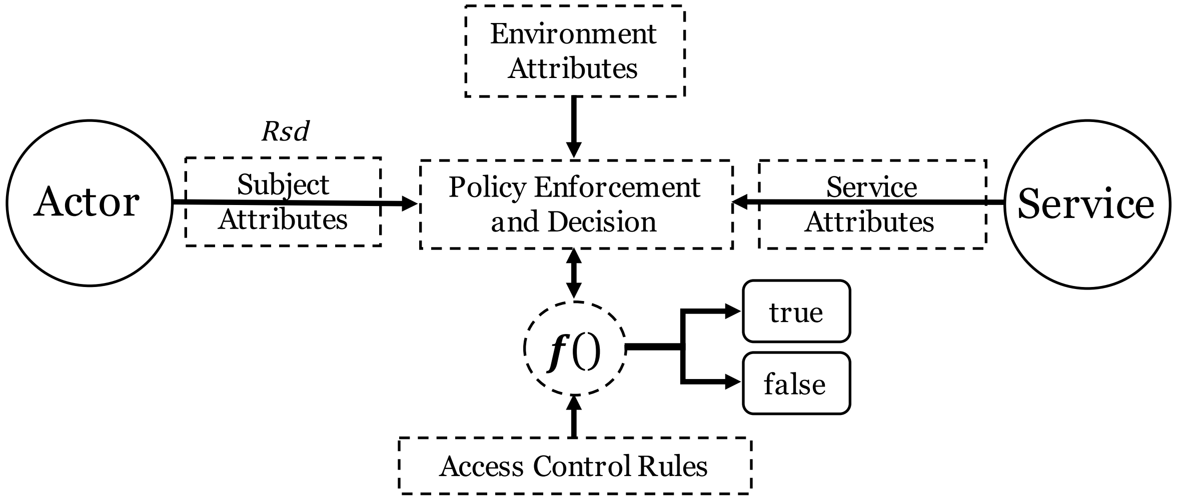

Figure 34.Collaborative Attribute-Bases Access Control Model

The following definitions extend the existing concepts required for the adaptation of ABAC to a collaborative Fog environment.

*2.2.1. **Definition: Context***

Context is the set of attributes describing the state of the environment, the user and the service subject of the current demand. Contextual attributes are for example location, time and so forth.

*2.2.2. **Definition: Access Control Rule***

Given a service request Rsd, and the context of this request Ctx, the access control rule determines whether the user who sent the request has the right to access a given service S.

The access rule function, denoted by $f()$, is defined as follows:

$$f(\quad): Rsd.A.\{attribute\} \times S.\{attribute\} \times Ctx.\{attribute\} \rightarrow [true|\, false]$$

The function returns a Boolean value that is equal to true when the access is granted, otherwise the value is equal to false.

2.3. Collaboration Access Models Comparison

We have evaluated the proposed access control models against the set of criteria deduced from the Fog requirements. Both models are adapting features of trusted and the largely known RBAC and ABAC models. Thus, they inherently support the well-known principles of Least Privilege and Separation of Duties. However, when it comes to the collaboration dimension, every model shows its particularities. Table 3 illustrates this comparison.

Table 3. Summary of the comparison between CollRbac and CollAbac

	CollRbac	CollAbac
Least Privilege Principle	Yes	Yes
Separation Of Duties	Yes	Yes
Scalability	Scalable to a certain extent. With the growth of actors in the collaborative network, the huge number of objects and services may lead to an explosion of roles.	Providing subject with attributes may have an overload on the framework. Services and context attributes are basic building blocks. Thus, no specialized mechanism are to be deployed for this purpose.
Dynamism Support	In relation with the scalability criterion, the constant movement of actors in the network may lead to an overload on the access-roles management process.	The active nature of the ABAC, makes it able to handle the dynamism of a collaborative system.
Contextual Information	Does not consider contextual information in the decision making mechanism	The context attributes provide a fairly representation of the contextual information.
Granularity	Low: lacks the ability to specify a fine-grained control on individual users in certain roles and on individual object instances.	High through attributes representation.

Flexibility	Low, regarding the responsiveness to the environment.	High due to its high granularity.

Indeed, from a collaborative perspective, the fact that access rules assignment is an application between groups of users on a set of objects is not fully sufficient. Often, a service in an instance of an actor might need specific permissions on an object at a particular time interval during the collaboration. Although the CollRbac access control has been augmented by the notion of operation, in comparison with the CollAbac it fails to provide the needed high level of fine-grained control. In addition, a comparison of the authorization mechanisms shows that CollAbac requires more complex trust relationships. In other words, CollAbac authorizations are derived directly from many sources such as the subject attributes, service context, and service data context.

3. Collaboration access policy process

As illustrated in Figure 34, CollAbac mechanism passes throughout the policy enforcement and decision. Indeed, within the authorization mechanism exist two main functions: the policy enforcement point (PEP) and the policy decision point (PDP). Given Rsd a request to access a service's data, the PEP extract the request's service attributes; a valid service request is a request that matches the demanded service definition. This match is founded on the satisfaction of the following conditions:

- $Rsd.data.d.t = S.data.d.t,$
- $Rsd.data.d.u = S.data.d.u,$
- $Rsd.data.frq.star \geq S.data.frq.start,$
- $Rsd.data.frq.end \leq S.data.frq.end,$
- $Rsd.data.frq.cron \subseteq S.data.frq.cron,$

- $\forall \text{opt}_i \in Rsd.data.\{opt\}, \exists \, opt_j \in S.data.\{opt\} : opt_i.attribute = opt_j.attribute \wedge opt_i.value (\subseteq \vee =) \, opt_j.value$,

Upon the validation of the request, the PDP makes the determination of whether or not to authorize the access. Such authorization is based on the extracted attributes and the application of the access rule function $f()$.

- $f() : Rsd.context \times S.context \times Rsd.A.\{attribute\} \rightarrow [true | \, false]$
- $\forall opt_i \in Rsd.context.\{opt\}, \exists \, opt_j \in S.context.\{opt\} : opt_i.attribute = opt_j.attribute \wedge opt_i.value \, (\subseteq \vee =) \, opt_j.value.$

Said differently, if the function returns *true* then grant subject $Rsd.A$ access to the service S.

4. CHAPTER CONCLUSION

The Internet of Things emerges as a new paradigm to provide communication, data consumption, and data analysis solutions for smart devices. The adoption of the IoT model has led malicious attacks to shift their targets from desktops and servers to IoT devices and objects. The main reason behind this behaviour lays on the weak protection of smart devices and sensors, in comparison to sophisticated servers. In addition, the nearness of such devices to the users makes them prone to leaking valuable information with catastrophic consequences. In this chapter, we proposed an Access Control approach for Fog Computing based on two well knows models: RBAC and ABAC. Across all the layers of the architecture, the Policy Enforcement Points component may be deployed to enforce access policies. The result of a policy evaluation may allow, deny, modify, or delay the execution requests in case policies controlling the respective request are deployed. This mechanism leverages the security of data through the architecture and the access authorizations of its components.

Chapter VII
INSTANTIATION OF COFOG, A PROOF OF CONCEPT AND EVALUATION

The Internet of Things is a tool for humanity, which is proving its worth in almost every area that touches our daily life activities. It is and will improve the efficiency of all applications where it is used. In order to show the importance and the added value of our approach, this chapter discusses a Fog-based smart parking system that instantiates the *CoFog* architectural framework. Simulations applied through this case study show the differences between the traditional Cloud-centric approach, and the Fog-based approach using the proposed architectural framework.

The *CoFog* architecture provides a generic Fog framework designed for any application in the Fog environment. Hence, no matter the size and the complexity of the targeted application scenario, it should be moderately easy to adapt the whole or part of the components of the different layers. Therefore, the case study described in this chapter toke the form of a feasibility study. Therefore, it has been led in three phases: the instantiation, the design and implementation, and the evaluation phases. The instantiation phase answered the question about how to adapt the architecture to a real use case. The design and implementation phase showed how the different components of the architecture could translate to classes and modules of the targeted platform. The later phase focused on the execution and the evaluation of the footprint of the developed application on the host system.

1. SMART CITY: A CASE STUDY ON SMART PARKING

Smart Cities are one of the fundamental use-cases of the Internet of Things. A Smart City, itself, is a combination of various use-cases ranging from Smart Waste management systems to managing Energy Grid systems in large city areas. In this section, we present a case study on a Fog-based Smart Parking Management System. This case study aims at demonstrating how the proposed reference architecture may be used in the context of Automotive and Smart Mobility scenarios. We believe that smart devices and Fog Computing can, indeed, offer parking providers a digital backbone to consolidate parking space availability across multiple locations. In addition, to providing services to publish real-time status and to improve reservation, use, visibility and efficiency of parking spaces. The Fog-based Smart Parking proof-of-concept, implements services for the delivery of on-trip information to a car driver. Such services are based on the data coming from the parking sensors.

The sensors deployed on parking spaces across the city, send data about wither a parking space is free or not. Those sensors may be heterogeneous and the data generated is formatted differently. This data pass through transformation and filtering processes.

As a result, the car driver receives information on the available parking sites around the position where the car is located at the moment.

In this case study, we compare the performance of applications based on our Fog infrastructure, versus the typical Cloud implementation. Figure 35 illustrates the logical network topology for simulating both Fog-based and Cloud-based scenarios.

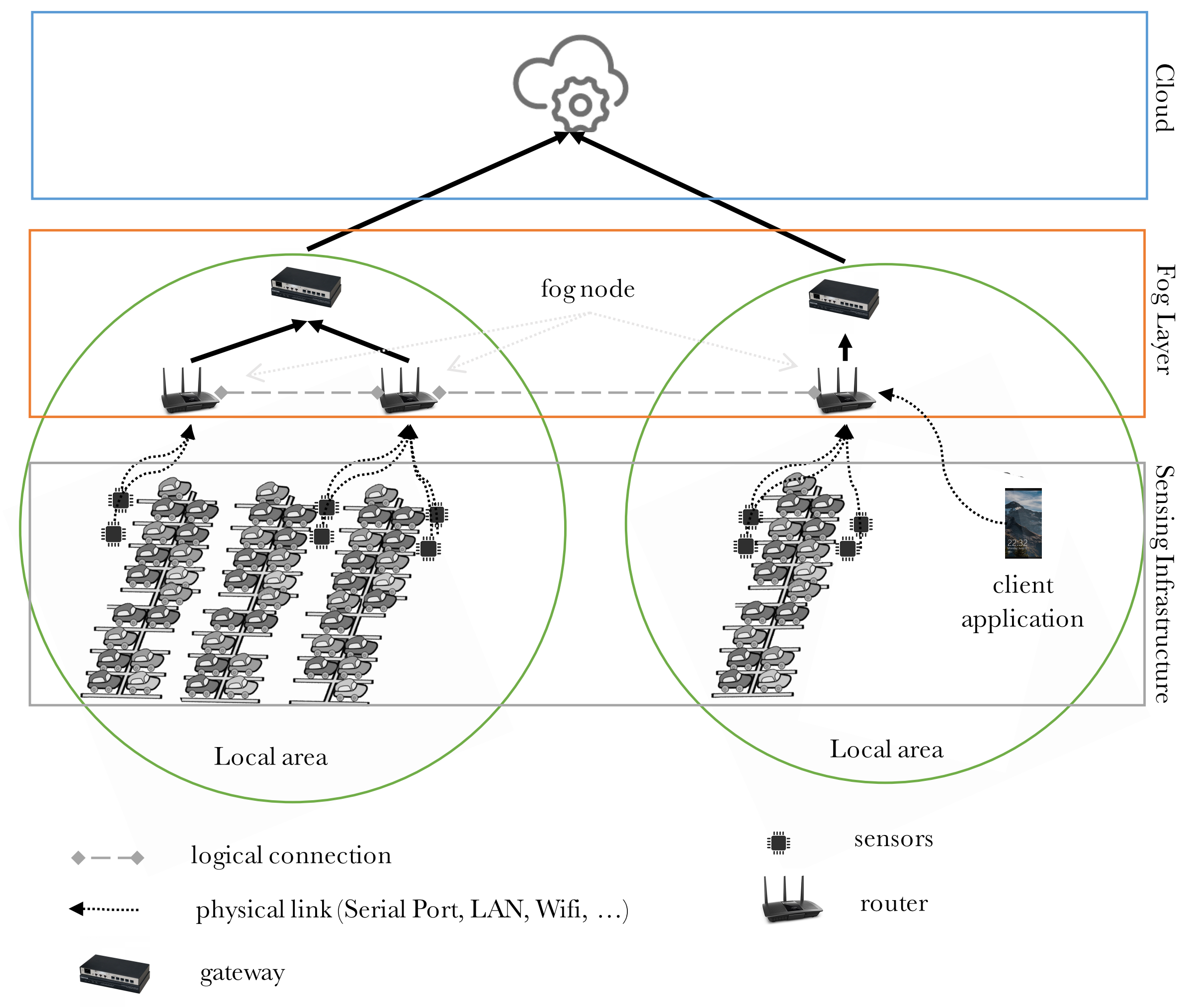

Figure 35. Logical structure for Cloud-based scenario

As described in the figure above (Figure 35), the network topology used for the simulation is organized as a hierarchical topology of Fog nodes and devices. The leaves of such tree-like topology are the edge devices (i.e., parking proximity sensors), and the root node represents the Cloud Computing infrastructure. Intermediate nodes in the tree represent intermediate network devices between the cloud and the edge –e.g., routers and gateways. Such devices –i.e., Fog nodes, are able to host applications by utilizing their compute, network and storage capacities.

This next sections gives further details on the experimental setup used to simulate both scenarios. In addition to design decisions that have been made to implement the reference architecture.

2. Experiment Setup

In the Cloud-based scenario, Cloud servers receives continuous update from the parking's sensing infrastructure. This data is then filtered, transformed and queued in order to be analysed. Upon receiving a request from the client application, parking services on the Cloud respond by delivering the appropriate and nearest available parking spot for the vehicle. In contrast, although both scenarios share the same network logical infrastructure, Fog-based applications do not need a direct communication with the Cloud. Indeed, in the Fog-based scenario, Fog nodes handle the workload for a certain geographical region. Hence, applications communicate directly with the Fog nodes in their local area.

In order to realize the intended scenarios as a set of computer simulations, many preparations had to be performed with regard to the following points.

2.1. Sensing Infrastructure:

The first goal regarding the sensing infrastructure is to simulate parking data as close to reality as possible. Hence, traffic data fed to simulation was obtained from the *Simulation of Urban Mobility* traffic simulator (SUMO) [93]. Despite being a traffic simulator, SUMO can be tweaked via configuration files to provide parking-like data (Figure 36).

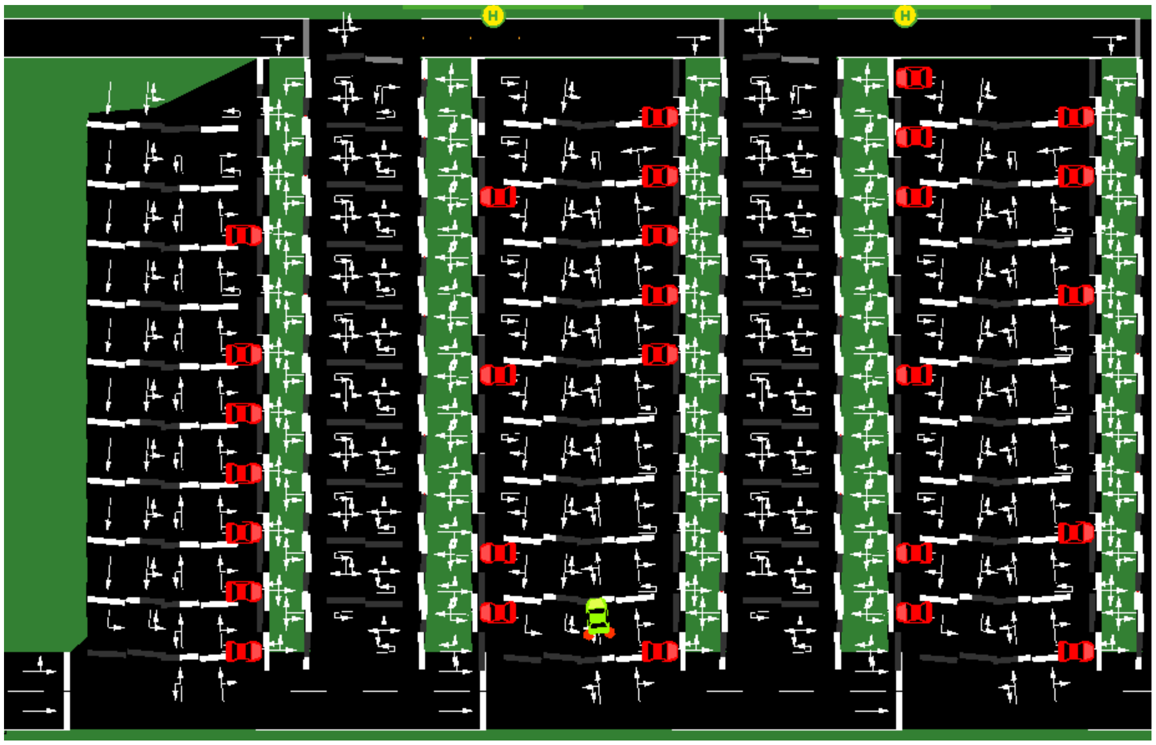

Figure 36. Tweaking SUMO for parking simulation

Many *Lane Area Detectors (E2)*[1] were defined and inserted in the parking space network. Such detectors act similar to tracking cameras –i.e., they save information about vehicles that cross over a certain position.

[1] http://sumo.dlr.de/wiki/NETEDIT#Lane_Area_Detectors_.28E2.29

2.2. CLOUD INFRASTRUCTURE:

The Cloud simulation environment was implemented using *CloudSim* Toolkit for Modeling and Simulation of Clouds [122]. The *CloudSim* framework provides users with means to model and simulate the execution of Cloud-based services. Therefore, by extending the basic entities in the original simulator (i.e., Datacentre, Host, Storage, and Cloudlet), it was possible to build a simulation backbone for the Cloud-based scenario. A set of features can be associated with each entity (e.g., CPU, RAM capacity, uplink network bandwidth), which provide resourceful measurements to be utilized for final comparisons.

2.3. OTHER PREPARATIONS:

To isolate the sensing activities, the SUMO simulator was deployed in a separate machine. We assumed that such separated execution of the Fog node and the sensing infrastructure, gives better understanding about the overload and response delay proper to the Fog node deployment. Furthermore, it includes factors like network link and data transmission delays between sensors and the fog node, leading to more realistic results.

In addition, to create a constrained execution environment for the Fog nodes, a platform virtualization software has been used. The Fog platform has been deployed on a virtual machine using VMware Fusion hypervisor [3][122], whilst the host machine is setup to connect to the network via Wi-Fi.

3. The Fog Platform: Design and Implementation

The implementation of the reference architecture as a suitable platform for the case study, has been realized through the instantiation of each layer as a module of the platform (Figure 37).

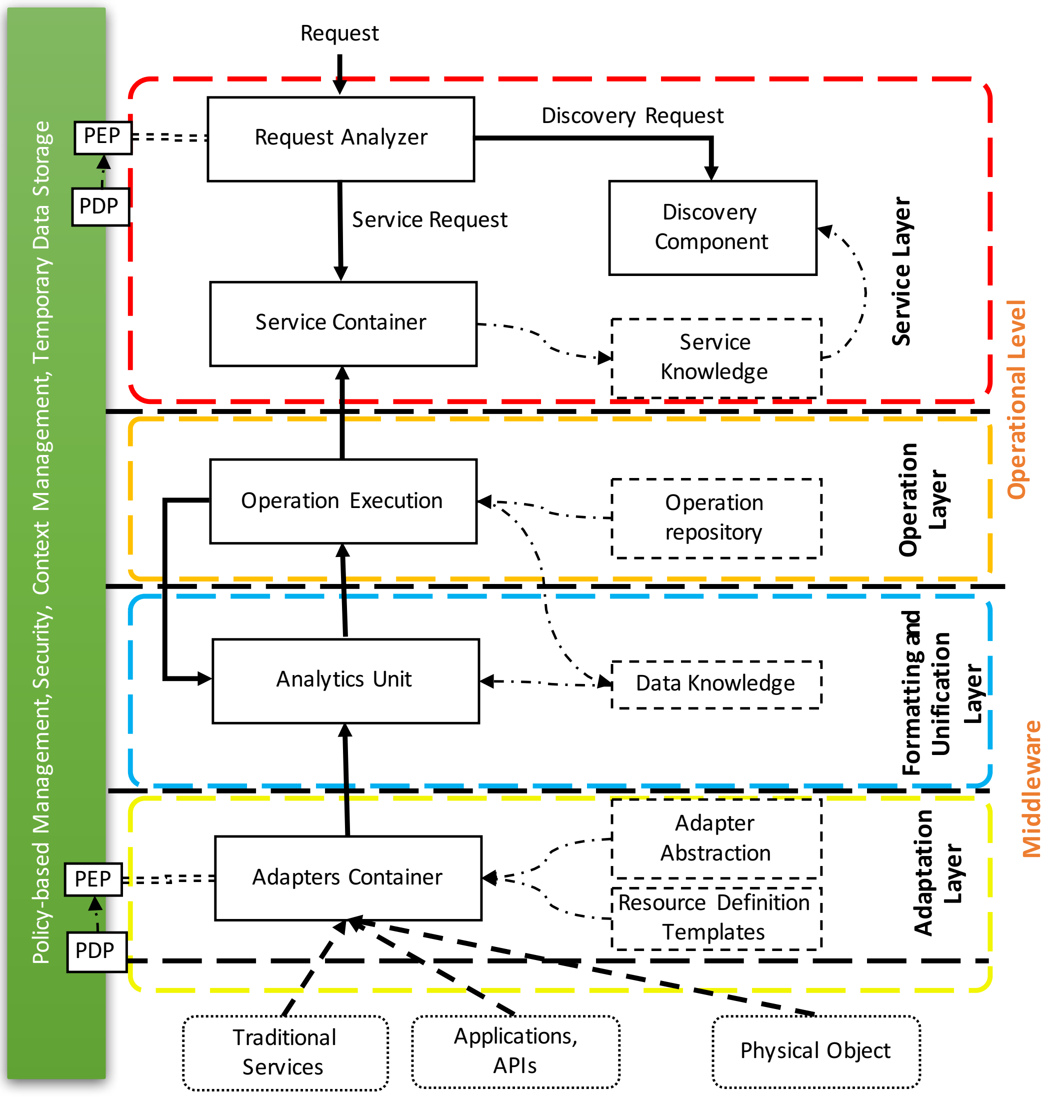

Figure 37. Instantiation of the Architecture

Hence, Figure 38 illustrates the resulting class diagram following the aforementioned decomposition.

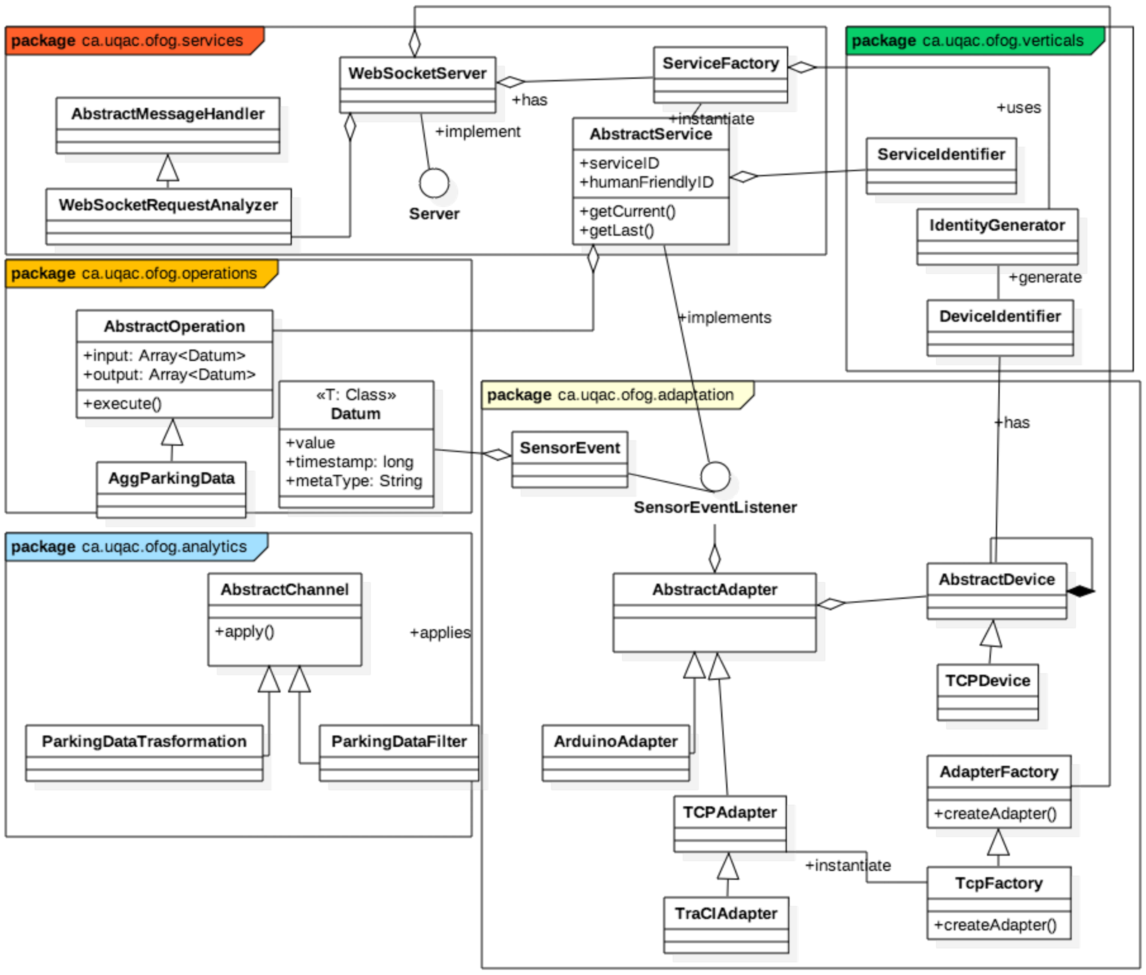

Figure 38. Class diagram for Fog nodes platform

As depicted in Figure 38, a number of *TraCIAdpater* adapters has been designed to intercept the upstream of data originated from the SUMO simulator via a *Connector* (Figure 39).

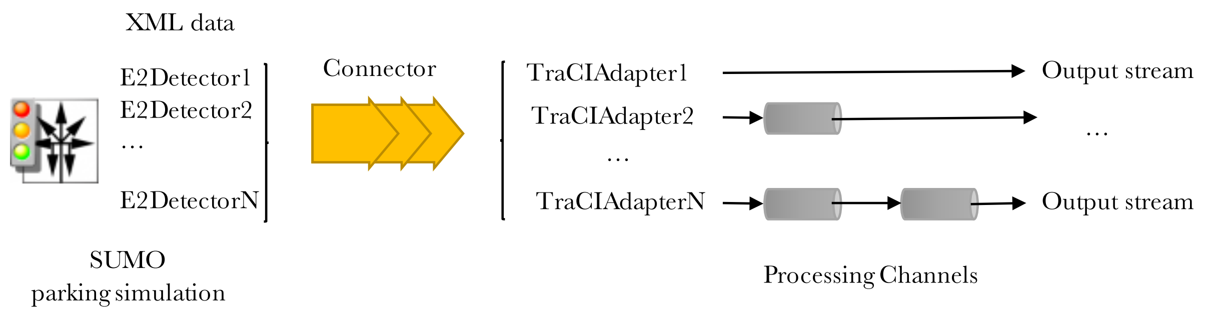

Figure 39. Data acquisition and filtering

In addition to intercepting data, a given adapter is instantiated using the corresponding sensor description file (Figure 40). Thus, incorporating all information about such sensor.

```
// Load Sensor from description
FileReader sensorDescriptionFile =
            new FileReader("sensrep/parking_detector.yml");
// Create Adapter

TraCIAdapter pSensorAdapter =

            ApdapterFactory.createAdapter(

                  TcpDevice.loadFromDescription(sensorDescriptionFile));
```

Figure 40. Loading sensors description files

As shown in Figure 39, acquired data passes through two Processing Channels: *Parking Data Transformation* and *Parking Data Filtering* (Appendix B), and they are defined as follows:

- Parking Data Transformation: this channel translates data from data about the halting time of vehicles, to whether or not a vehicle is present at a given moment. The new data format is more useful at the service level.

- Parking Data Filtering: this channel act as a filtering barrier that sends only the change in new data values. Hence, reducing the amount of data to be sent on the network.

The execution of the operations is straightforward as shown in Figure 41.

```
// Execute nested data filtering and transformation
Datum<String> nd =
      new ParkingDataTransformation().execute(
            new ParkingDataFilter().execute(d));
```

Figure 41. Filtering and Transformation channels

4. Performance Evaluation

The simulation of the Smart Parking system was carried out using the setup described in the first section of this chapter. This section presents and evaluates the results obtained, and demonstrates how different execution configurations affects the network overload and services latency. Indeed, to stress performance measurements – i.e., network overload, memory usage, and response time, each scenario has been simulated with and without data analytics. In addition, the simulations have been executed using five configurations, each of which having three, ten, fifteen, twenty and thirty connected sensors respectively.

4.1. Network Overload

The next two figures illustration the data overload on the network during the execution of both Cloud scenario and the scenario using only Fog nodes.

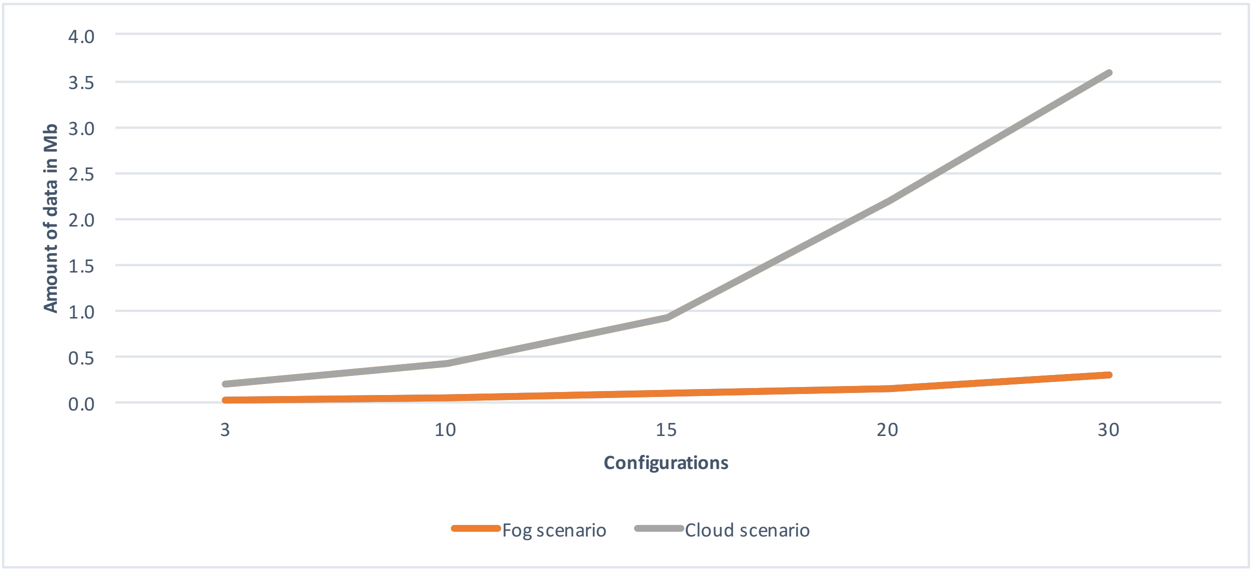

Figure 42. Network overload: Fog vs. Cloud scenario using raw data

Contrary to Figure 42, Figure 43 provides network overload measurement when data analytics operations are included in the simulation. Overall, the increase of connected-sensor number results in a significant growth of the load on the network. Still, the burden on the network while using the Cloud is largely significant than while using the Fog.

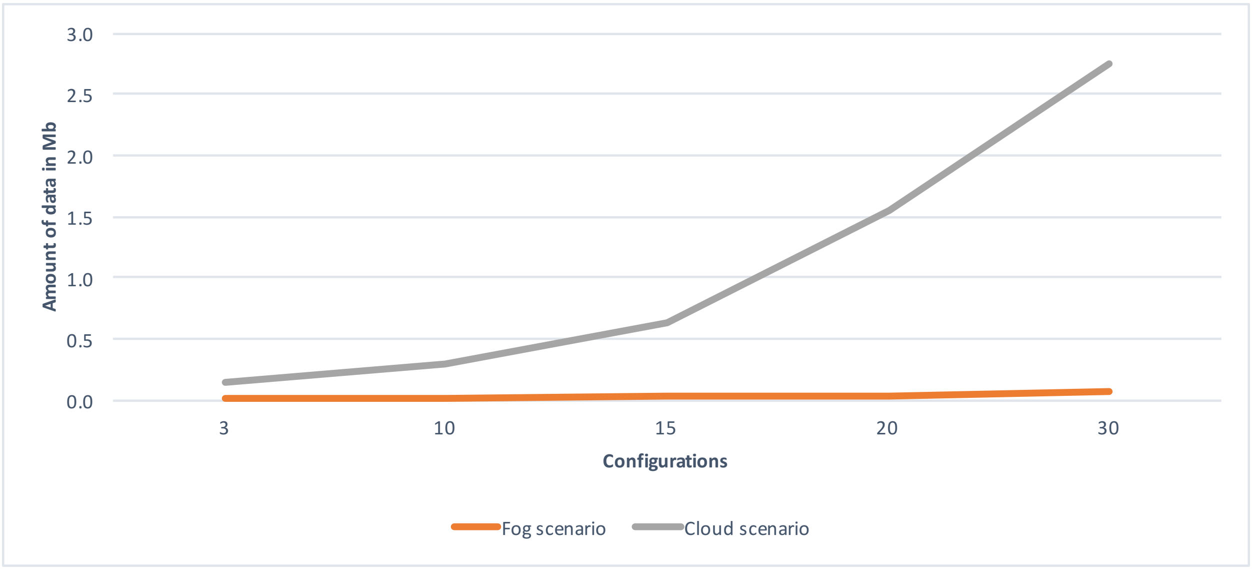

Figure 43. Network overload: Fog vs. Cloud scenario using analysed data

An additional observation lays in the decrease of data overload induced by the use of data analytics. Processing channels in the Fog-based simulations, in this case the Parking Data Filtering channel, considerably reduce the volume of data sent on the network.

4.2. MEMORY USAGE

To illustrate the footprint on memory of each scenario, Figure 44 and Figure 45 present simulations RAM consumption during the execution of both scenarios.

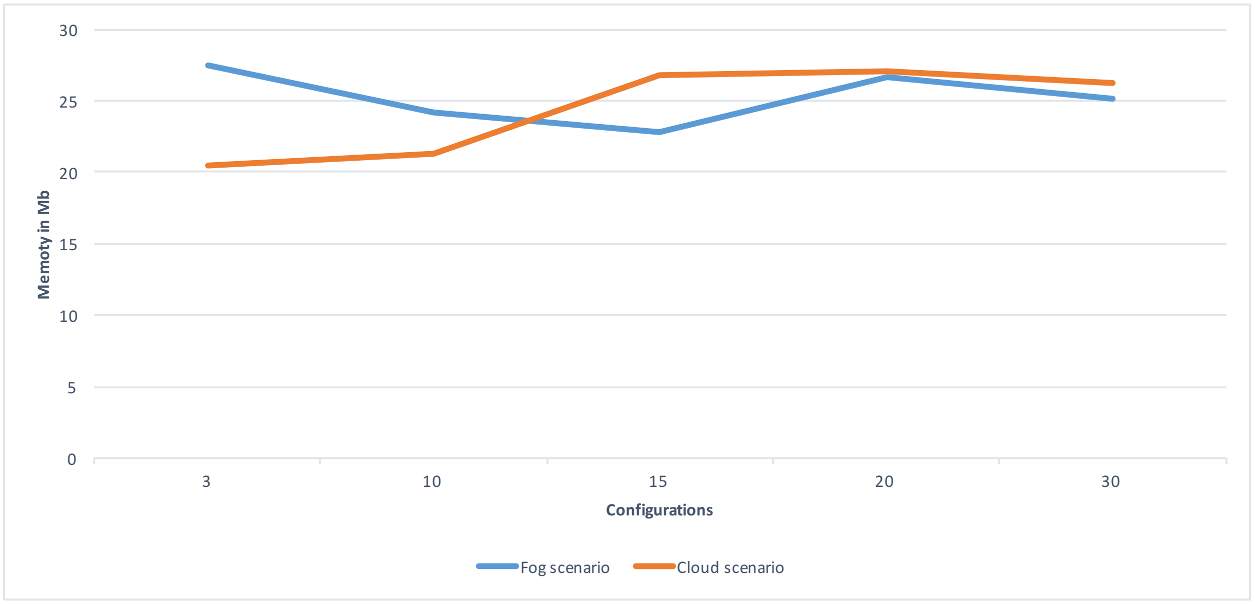

Figure 44. RAM consumption in simulations with raw data

The collected measurement data for the five configurations shows a slightly increase in memory consumption in both scenarios. In the first set of executions, where simulations use and provide only raw data, we did not notice significant difference gape between both scenarios. However, adding analytics functions, especially to the Fog node, result in a significant increase in the workload of the later.

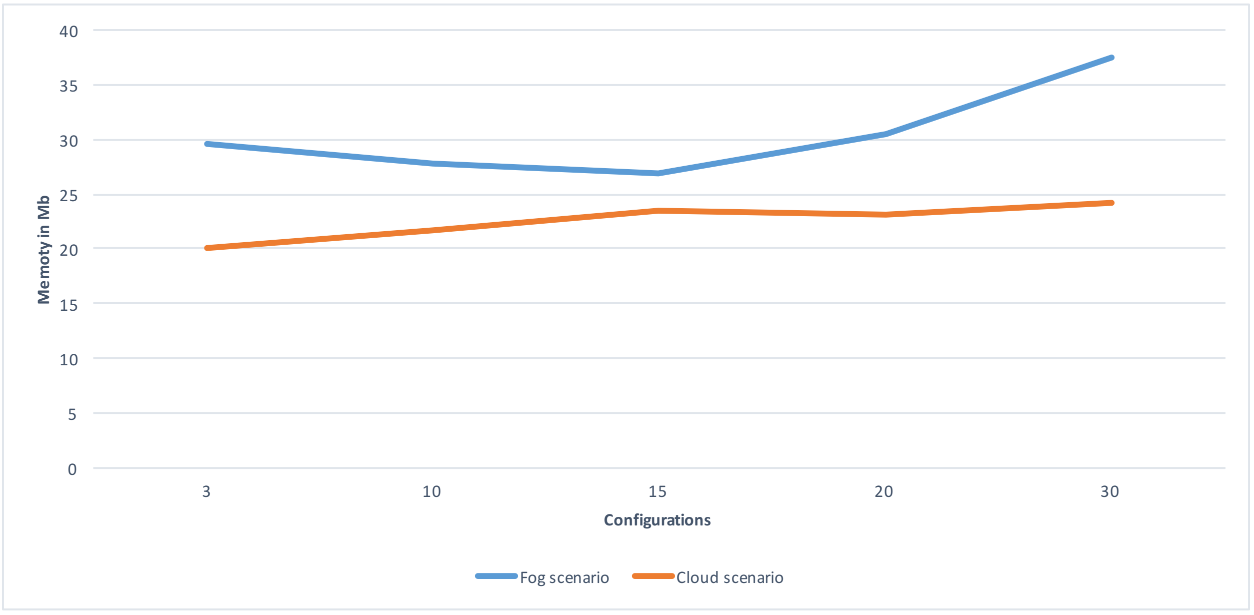

Figure 45. RAM consumption in simulations with data analysis

The aforementioned finding are direct consequences to the Fog node executing data acquisition and data analysis in the meantime. Besides, we noticed that executing the same simulation over the set of configurations, does not lead to drastic escalation of its memory usage. Thus, while the design itself demonstration a certain degree of scalability, the platform implementation need more refinement.

4.3. RESPONSE TIME

The nearness of Fog nodes to user applications is a major fact in reducing latency. Indeed, Figure 46 illustrates the average response time between the user application and the Cloud server, and between the application and the Fog node.

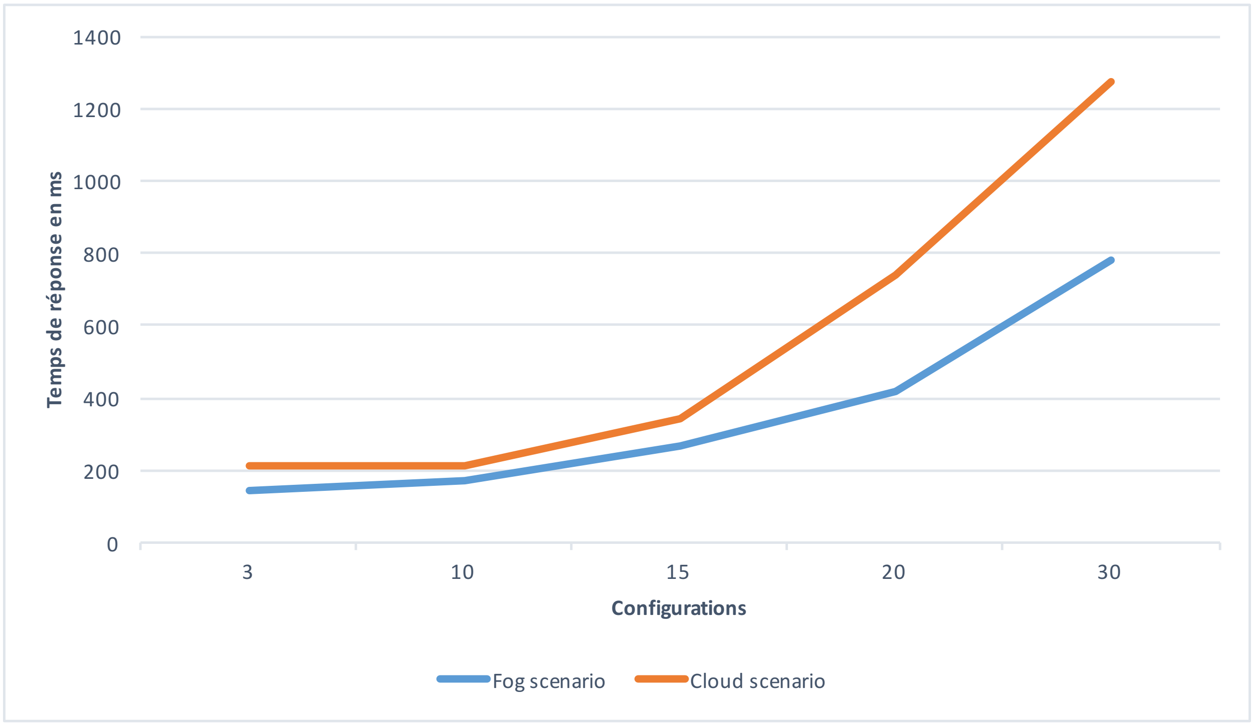

Figure 46. Comparing response time in Fog and Cloud application.

As the figure depicts, the results show that end-to-end network latency has been reduced when relaying on our Fog approach. In contrary, in the case of using Cloud-based strategy, one can notice a significant increase in latency. Cloud Data Centers constitute a bottleneck for simulation data.

4.4. SUMMARY

In this chapter, we introduced to the Smart Parking case study and how it constitutes a suitable use case to implement and validate the proposed architecture. Thus, showing the effectiveness of the proposed architecture. This case study was also used to show and compare the differences in using two scenarios: a traditional Cloud-based scenario and a Fog-based scenario using the proposed reference architecture. We presented the apparatuses used to deploy and run the simulation of each scenario, using five different configurations. In addition, this chapter emphasis on the platform design process and the implemented classes and components deduced from the reference architecture.

The findings of this simulation procedures reinforce our belief that a suitable adoption of the proposed architectural framework can indeed leverage many dimensions of the Fog Computing issues. These findings, either points of strength or limitations, along with future work of improvement are discussed in the next chapter.

Chapter VIII
CONCLUSION AND PROSPECTS

The Internet of Things has become, indeed, a reality. It fuels relentless transformation and convergence, comprises a new era of smart products, green initiatives, virtual reality, and augmented connectivity. This thesis anticipates the raising opportunities with this new paradigm, and presents an architectural model for collaboration in Fog Computing. We have demonstrated that, with automatic, resource-aware and domain-agnostic service-based collaboration, it is feasible to provide augmented services and support real-time applications at the edge of the Internet. This chapter reflects on the contributions of this work, discusses prospects and future research directions, and concludes.

1. OBJECTIVES SUMMARY

Can we transform the edge of the Internet into a nest of collaborative objects of the IoT? The work presented in this thesis answers this question with the affirmation, and transcends the theoretical proposition into providing a a proof-of-concept implementation. Indeed, following the idea of spreading intelligence to the edge of the Internet, our interest laid in extending the Fog Computing paradigm to embrace a thing collaborative model. In such computing model, objects would be enabled to exploit and collaborate with each other, in order to achieve common or distinct goals. We focused on delivering a architectural model that, in one hand, leverages the devices with services, and in the other augments data representation and consumption with local analytics. In addition, we have foreseen that aggregating sensing activities at a Fog level, constitutes an important building block to support more advanced collaborative scenarios. Therefore, powering Fog nodes with dynamic service creation was amongst our fixed objectives. Through composition, aggregation, transformation and other processes, new services would be dynamically created based on available services. A platform that follows such architectural model, would not see its full potential achieved, without objects being able to interact with each other. Hence, the goal of providing means of finding external functionalities has to follow some registration and look-up mechanism in a distributed fashion. Last but not the least, among the many facets of the non-functional requirements of the architecture, we focused our interest on security, chiefly access control management. In order to guaranty right access rights to the right subject on the right object, an access control model that adapt to the particularities of Fog Computing environment had to be provided. Hence, a comprehensive study has been planned to better select a suitable approach to tackle this problem in a constrained environment.

2. A Word on the Contributions

Indeed, the fundamental novelty of our research work is the introduction of an architectural model for collaboration in Fog Computing. The originality of this architecture resides in the benefits it bestows on Fog-based platforms and applications:

- The architectural design allows platforms to implement all or parts of the architecture. Hence, the ability to be deployed in variety of environments spanning from core servers to edge endpoints –e.g., routers [2].The architectural design allows platforms to implement all or parts of the architecture. Hence, the ability to be deployed in a variety of environments spanning from core servers to edge endpoints –e.g., routers [2].
- The high level of abstraction and the flexible virtualization mechanism of heterogeneous physical resources provide platforms with means to exploit physical devices, APIs, web services and other data sources. Furthermore, the analytics Units offer a given platform the possibility to not only capture data, but also the capabilities of performing local and direct analytics on real time data. Thus, among other advantages, freeing the network from the burden of the continuous torrents of data toward the Cloud.
- The mechanism of Operations leverages the platforms services with automatic, dynamic and on-demand service instantiation. It opens the doors wide for autonomous collaboration and more sophisticated applications [2].The mechanism of Operations leverages the platforms services with automatic, dynamic and on-demand service composition and instantiation. It opens the doors wide for autonomous collaboration and more sophisticated applications.

- The IOTCollab Access Control Model [2] is[6] is a new way to perceive access control for collaboration. Its design supports the scalability and the dynamic nature of Fog Computing. While providing fine-grained and contextual-aware rules needed for the constrained devices of IoT.

3. PROSPECTS BEYOND THE LIMITATIONS

We have only started scratching the surface of the possibilities of Fog Computing and the Internet of Things in general. While this work has taken major steps into unlocking some of these possibilities, the presented architectural framework exhibits limitations that would be interesting to explore in the future. In this section, we revisit these limitations and highlight some of the exciting avenues for future research.

In Chapter IV, we introduce the concept of Operation. Although its definition is formally abstracted, this concept need more expansion. In other words, the definition of formulas, within Operation, is limited to straightforward mathematical operations. Therefore, restraining the potential of such a concept. A future research direction would be toward defining a complete framework to express more sophisticated formulas. In addition, matching between a given request and a potential Operation, is based on strictly comparing request parameters against operation's outputs. Hence, it does not take into account the cases where request parameters constitute a subset of the operation outputs. An avenue for future work is to leverage this algorithm with Ontology based matching.

Although Things and applications in different Fog nodes have the ability to collaborate, the decision of when and what service to collaborate with is configured manually. Smart objects need to be fully autonomous in taking such decision, one direction of future research is to explore context-based inference mechanism.

As stated in the beginning of this chapter, the absence of another architecture to compare with. The case study was limited to an instantiation and an evaluation (simulation) at a small scale. Hence, a future work direction must be a large scale evaluation with real world sensors and data. In addition, the case study proved the adaptability and the easiness to instantiate the architecture in spite of the small scale of the application. Nevertheless, it is will be of great value to work more toward the automatization of the instantiation procedures. This automatization could, for example, take the form of a complete guide with detailed workflows.

To summarize, we believe that future IoT systems will rely on the edge of the Internet to deliver assistance that touches our everyday lives, similar to how the Cloud provides us with indispensable services. This thesis has made multiple strides in that direction. It also builds on a deep understanding that Fog Computing introduces new systems and new services that require redesigning the entire networking and computing stack, from the hardware to the applications. We believe that this approach will become a necessity, since devices are becoming ever-more ubiquitous and as their services keep expanding in the upcoming near future.

APPENDIX A

The Mobile Collaborative Computing Environment

The Mobile Collaborative Computing Environment (CCE) introduces a new way to perceiving mobile collaboration between devices. This model exploits the increasing capabilities and the decreasing costs of handhelds to address the portability, heterogeneity, and error handling in a device collaborative-based network. The collaborative environment was designed as a generic structure of layers. Therefore, the model is extensible to house different devices (smartphones, tablets, desktops…) and network infrastructures (LANs, Wi-Fi, Bluetooth, etc.). As illustrated in Figure 47, the environment is organized in two main layers: the Components Layer and the Collaboration Middleware.

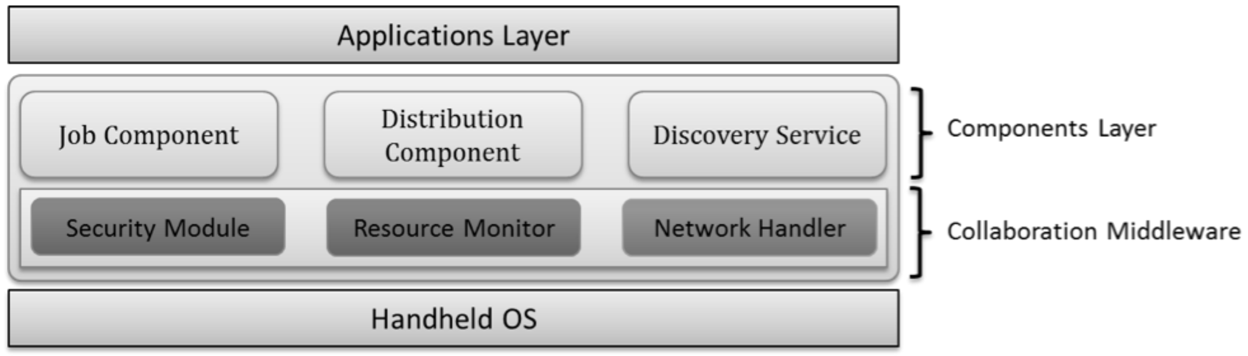

Figure 47. The CCE modular architecture

The first layer provides components to be used by the overlaying application layer, it contains:

- The Job Component that provides an execution environment where the received tasks are executed and managed,

- The Distribution Component implements a scheduling policy based on devices configuration to ensure appropriate task scheduling,
- The Task Tracking Component that distributes the task to be executed on available collaborators.

The second layer, the Collaboration Middleware is a platform-agnostic set of modules that abstracts the underlying operating system and networks architecture. It provides the minimum required to allow mobile collaboration to occur between mobile peers. Thus, the collaboration occurs between handhelds of many categories following the process depicted in Figure 48.

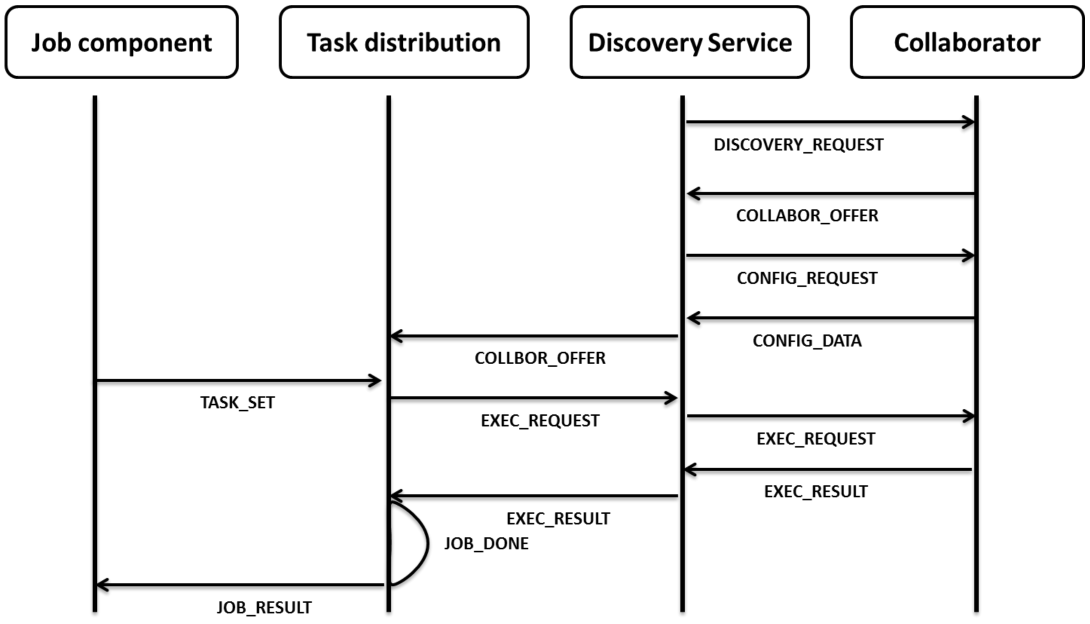

Figure 48. CCE's collaboration process workflow

The key points of strength of this collaborative environment lie in its:

- Portability: regardless of the architecture and the operating system of the device, the model ensures the communication between the top and the bottom of the system;
- Scalability: the model enables the collaborative network to scale with the dynamic change of users.
- Robustness: the decentralized network provides resistance and fault tolerance;
- Dynamic: collaboration networks construction and collaborators join are dynamic.

A simple prototype has been developed and deployed to illustrate the collaboration process over the network. This prototype tokes the form of an Android distributed-application calculating the value of π. Indeed, the application was deployed on three mobile devices running Android, using UDP protocol for transmitting collaboration messages and results. Each device acted as an independent node in charge of executing a portion of the algorithm. The following chart (Figure 49) shows the comparison between local and collaborative calculation time.

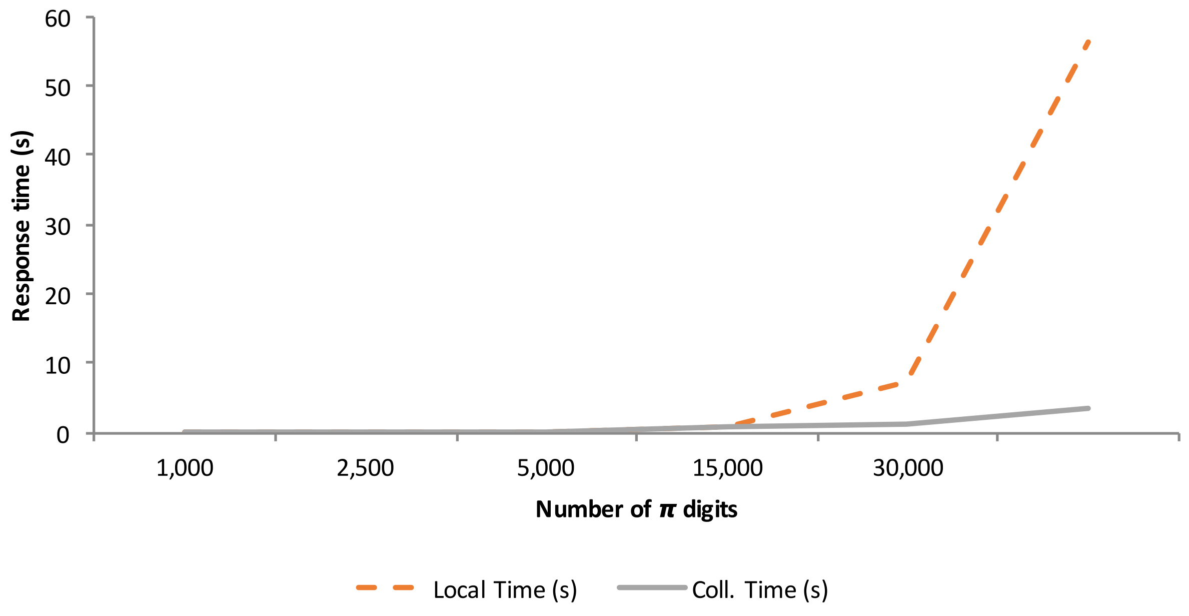

Figure 49. Comparison of response time between local and distributed execution using CCE.

BIBLIOGRAPHY

1. Jabril Abdelaziz, Mehdi Adda, and Hamid Mcheick. 2015. A Survey on Collaborative Internet of Things. *International Workshop on Healthcare systems and Internet of Things for Humanity (eHealthForHumanity'2015) Conjunction with the 6th conference MCETECH'2015 - Springer*, Bibliothèque et Archives nationales du Québec (BAnQ), 12–31.

2. Jabril Abdelaziz, Mehdi Adda, and Hamid Mcheick. 2016. Toward Service Aggregation for Edge Computing. *Procedia Computer Science* 94: 424–428.

3. Jabril Abdelaziz, Mehdi Adda, and Hamid Mcheick. 2018. An Architectural Model for Fog Computing. *Journal of Ubiquitous Systems and Pervasive Networks* 10, 1: 21–25.

4. Jabril Abdelaziz and Hamid Mcheick. 2012. Toward A Collaborative Computing Environment. *SYSCO 2012 : 1ère conférence francophone sur les Systèmes Collaboratifs*, 1–5.

5. Mervat Abu-Elkheir, Mohammad Hayajneh, and Najah Ali. 2013. Data Management for the Internet of Things: Design Primitives and Solution. *Sensors* 13, 11: 15582–15612.

6. Mehdi Adda, Jabril Abdelaziz, Hamid Mcheick, and Rabeb Saad. 2015. Toward an Access Control Model for IOTCollab. *Procedia Computer Science* 52, 1: 428–435.

7. Mehdi Adda and Rabeb Saad. 2014. A Data Sharing Strategy and a DSL for Service Discovery, Selection and Consumption for the IoT. *Procedia Computer Science* 37: 92–100.

8. Charu C. Aggarwal, Naveen Ashish, and Amit Sheth. 2013. *Managing and Mining Sensor Data*. Springer US, Boston, MA.

9. Tanveer Ahmed, Abhinav Tripathi, and Abhishek Srivastava. 2014. Rain4Service: An Approach towards Decentralized Web Service Composition. *2014 IEEE International Conference on Services Computing*, IEEE, 267–274.

10. Katherine Albrecht and Katina Michael. 2013. Connected: To Everyone and Everything. *IEEE Technology and Society Magazine* 32, 4: 31–34.

11. Kevin Ashton. 2009. That "Internet of Things" Thing: In the Real World Things Matter More than Ideas. *RFID Journal*.

12. Luigi Atzori, Antonio Iera, and Giacomo Morabito. 2010. The Internet of Things: A survey. *Computer Networks* 54, 15: 2787–2805.

13. Paramvir Bahl, Richard Y. Han, Li Erran Li, and Mahadev Satyanarayanan. 2012. Advancing the state of mobile cloud computing. *Proceedings of the third ACM workshop on Mobile cloud computing and services - MCS '12*, ACM Press, 21–21.

14. Martin Bauer, Mathieu Boussard, Nicola Bui, et al. 2013. IoT Reference Architecture. In *Enabling Things to Talk*. Springer Berlin Heidelberg, Berlin, Heidelberg, 163–211.

15. Martin Bauer, Nicola Bui, Jourik De Loof, et al. 2013. IoT Reference Model. In *Enabling Things to Talk*. Springer Berlin Heidelberg, Berlin, Heidelberg, 113–162.

16. Martin Bauer, Nicola Bui, Jourik De Loof, et al. 2013. IoT Reference Model. In A. Bassi, M. Bauer, M. Fiedler, et al., eds., *Enabling Things to Talk*. Springer Berlin Heidelberg, Berlin, Heidelberg, 113–162.

17. Oren Ben-Kiki, Clark Evans, and Ingy döt Net. 2009. YAML Ain't Markup Language (v1.2). Retrieved February 13, 2018 from http://www.yaml.org/spec/.

18. Flavio Bonomi, Rodolfo Milito, Preethi Natarajan, and Jiang Zhu. 2014. Fog Computing: A Platform for Internet of Things and Analytics. In 169–186.

19. Flavio Bonomi, Rodolfo Milito, Preethi Natarajan, and Jiang Zhu. 2014. Fog Computing: A Platform for Internet of Things and Analytics. In 169–186.

20. Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. 2012. Fog computing and its role in the internet of things. *Proceedings of the first edition of the MCC workshop on Mobile cloud computing - MCC '12*, ACM Press, 13.

21. Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. 2012. Fog computing and its role in the internet of things. *Proceedings of the first edition of the MCC workshop on Mobile cloud computing - MCC '12*, ACM Press, 13–13.

22. John Carney. 2011. *Why Integrate Physical and Logical Security?* San Jose, CA, USA.

23. Peter H. Carstensen and Kjeld Schmidt. 1999. Computer Supported Cooperative Work: New challenges to systems design. In *Handbook in Human Factors/Ergonomics*. Asakura Publishing, 619--636.

24. David Ferraiolo Richard Kuhn Ramaswamy Chandramouli. 2007. *Role-Based Access Control*. Artech House, Boston.

25. Cisco Inc. 2018. *Cisco GCI : Forecast and Methodology, 2016–2021*. San Jose, CA, USA.

26. Dragos-George Comaneci and Ciprian Dobre. 2011. Electronic ID: Services and Applications for Context-Aware Integrated Mobile Services. *2011 Developments in E-systems Engineering*, IEEE, 502–507.

27. R. W. Conway, W. L. Maxwell, and H. L. Morgan. 1972. On the implementation of security measures in information systems. *Communications of the ACM* 15, 4: 211–220.

28. Rudyar Cortés, Xavier Bonnaire, Olivier Marin, and Pierre Sens. 2015. Stream Processing of Healthcare Sensor Data: Studying User Traces to Identify Challenges from a Big Data Perspective. *Procedia Computer Science* 52: 1004–1009.

29. Directorate-General for the Information Society and Media. 2010. *Vision and challenges for realising the internet of things*. EU Publications.

30. Hani Y Diya, Hassan A Artail, and Haidar Safa. 2005. A Framework for Mobile Collaborative Environments. *The Third World Enformatika Conference, WEC'05*, 163–166.

31. A Dohr, R Modre-Opsrian, M Drobics, D Hayn, and G Schreier. 2010. The Internet of Things for Ambient Assisted Living. *2010 Seventh International Conference on Information Technology: New Generations*, IEEE, 804–809.

32. B P Douglass. 2003. *Real-time Design Patterns: Robust Scalable Architecture for Real-time Systems*. Addison-Wesley Professional.

33. ETSI. 2017. *Mobile Edge Computing (MEC); General principles for Mobile Edge Service APIs*. .

34. ETSI Technical Committee Machine-to-Machine and communications (M2M). 2013. Machine-to-Machine communications (M2M), Functional architecture (Technical Specification Standard). .

35. Dave Evans. 2011. *The Internet of Things How the Next Evolution of the Internet Is Changing Everything, A white paper*. San Jose, CA, USA.

36. Jose Oscar Fajardo, Fidel Liberal, Ioannis Giannoulakis, et al. 2016. Introducing Mobile Edge Computing Capabilities through Distributed 5G Cloud Enabled Small Cells. *Mobile Networks and Applications* 21, 4: 564–574.

37. Roy T. Fielding and Richard N. Taylor. 2002. Principled design of the modern Web architecture. *ACM Transactions on Internet Technology* 2, 2: 115–150.

38. Ying Gao, Wenlu Hu, Kiryong Ha, Brandon Amos, Padmanabhan Pillai†, and Mahadev Satyanarayanan. 2015. Are Cloudlets Necessary? .

39. Pedro Garcia Lopez, Alberto Montresor, Dick Epema, et al. 2015. Edge-centric Computing: Vision and Challenges. *ACM SIGCOMM Computer Communication Review* 45, 5: 37–42.

40. Pedro Garcia Lopez, Alberto Montresor, Dick Epema, et al. 2015. Edge-centric Computing. *ACM SIGCOMM Computer Communication Review* 45, 5: 37–42.

41. Jonathan Grudin. 1991. CSCW: the convergence of two development contexts. *Proceedings of the SIGCHI conference on Human factors in computing systems Reaching through technology - CHI '91*, ACM Press, 91–97.

42. Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. 2013. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems* 29, 7: 1645–1660.

43. Dominique Guinard, Vlad Trifa, Friedemann Mattern, and Erik Wilde. 2011. *Architecting the Internet of Things*. Springer Berlin Heidelberg, Berlin, Heidelberg, Heidelberg.

44. Dominique Guinard, Vlad Trifa, Friedemann Mattern, and Erik Wilde. 2011. From the Internet of Things to the Web of Things: Resource-oriented Architecture and Best Practices. In *Architecting the Internet of Things*. Springer Berlin Heidelberg, Berlin, Heidelberg, 97–129.

45. Guoping Zhang and Jiazheng Tian. 2010. An extended role based access control model for the Internet of Things. *2010 International Conference on Information, Networking and Automation (ICINA)*, IEEE, V1-319-V1-323.

46. S. Gusmeroli, S. Piccione, and D. Rotondi. 2012. IoT Access Control Issues: A Capability Based Approach. *2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, IEEE, 787–792.

47. Sergio Gusmeroli, Salvatore Piccione, and Domenico Rotondi. 2013. A capability-based security approach to manage access control in the Internet of Things. *Mathematical and Computer Modelling* 58, 5–6: 1189–1205.

48. D Hanes, G Salgueiro, P Grossetete, R Barton, and J Henry. 2017. *IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things*. Pearson Education.

49. Karthik Harihar and Stan Kurkovsky. 2005. Using Jini to enable pervasive computing environments. *Proceedings of the 43rd annual southeast regional conference on - ACM-SE 43*, ACM Press, 188.

50. Kirak Hong, David Lillethun, Umakishore Ramachandran, Beate Ottenwälder, and Boris Koldehofe. 2013. Mobile fog. *Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing - MCC '13*, ACM Press, 15.

51. Kirak Hong, David Lillethun, Umakishore Ramachandran, Beate Ottenwälder, and Boris Koldehofe. 2013. Mobile fog: a programming model for large-scale applications on the internet of things. *Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing - MCC '13*, ACM Press, 15.

52. Amine M. Houyou, Hans-Peter Huth, Christos Kloukinas, Henning Trsek, and Domenico Rotondi. 2012. Agile manufacturing: General challenges and an IoT@Work perspective. *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation (ETFA 2012)*, IEEE, 1–7.

53. Vincent C. Hu, David Ferraiolo, Rick Kuhn, et al. 2014. *Guide to Attribute Based Access Control (ABAC) Definition and Considerations*. Gaithersburg, MD, MD.

54. Ian Smith, Ken Sakamura, Anthony Furness, et al. 2009. *CASAGRAS Project "Final Report, RFID and the Inclusive Model for the Internet of Things."* .

55. IEEE-SA. 2015. Standard for an Architectural Framework for the Internet of Things (IoT). .

56. IoTivity Project. 2017. IoT Interoperability Framework. *The Open Connectivity Foundation*. Retrieved from https://www.iotivity.org/.

57. ITU Strategy and Policy Unit (SPU). 2006. *ITU Internet Reports, The Internet of Things*. Geneva, Switzerland.

58. ITU Strategy and Policy Unit (SPU). 2006. *ITU Internet Reports, The Internet of Things*. Geneva, Switzerland.

59. P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. 2003. Optimized link state routing protocol for ad hoc networks. *Proceedings. IEEE International Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century.*, IEEE, 62–68.

60. Chang-won Jeong, Dong-ho Kim, and Su-chong Joo. 2007. Mobile Collaboration Framework for u-Healthcare Agent Services and Its Application Using PDAs. In *Agent and Multi-Agent Systems: Technologies and Applications*. Springer Berlin Heidelberg, 747–756.

61. Florian Kerschbaum. 2010. An access control model for mobile physical objects. *Proceeding of the 15th ACM symposium on Access control models and technologies - SACMAT '10*, ACM Press, 193.

62. F. Khodadadi, A.V. Dastjerdi, and R. Buyya. 2016. Internet of Things: an overview. In *Internet of Things*. Elsevier, 3–27.

63. Gerd Kortuem, Fahim Kawsar, Vasughi Sundramoorthy, and Daniel Fitton. 2010. Smart objects as building blocks for the Internet of things. *IEEE Internet Computing* 14, 1: 44–51.

64. Matthias Kovatsch. 2013. CoAP for the Web of Things: From Tiny Resource-constrained Devices to theWeb Browser. *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication - UbiComp '13 Adjunct*, 1495–1504.

65. Matthias Kovatsch. 2013. CoAP for the Web of Things: From Tiny Resource-constrained Devices to theWeb Browser. *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication - UbiComp '13 Adjunct*, 1495–1504.

66. Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. 2012. Recent Development and Applications of SUMO – Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements* 5, 4: 128–138.

67. Bill Kuechler and Vijay Vaishnavi. 2008. On theory development in design science research: anatomy of a research project. *European Journal of Information Systems* 17, 5: 489–504.

68. Butler W. Lampson. 1974. Protection. *ACM SIGOPS Operating Systems Review* 8, 1: 18–24.

69. Chao Lee, Yunchuan Guo, and Lihua Yin. 2013. A Location Temporal based Access Control Model for IoTs. *AASRI Procedia*, Elsevier B.V., 15–20.

70. Edward A. Lee. 2015. Architectural Support for Cyber-Physical Systems. *ACM SIGPLAN Notices* 50, 4: 1–1.

71. Shancang Li, Li Da Xu, and Shanshan Zhao. 2015. The internet of things: a survey. *Information Systems Frontiers* 17, 2: 243–259.

72. David Linthicum. 2016. Responsive Data Architecture for the Internet of Things. *Computer* 49, 10: 72–75.

73. Yi Liu, Ke Xu, and Junde Song. 2013. A Task-Attribute-Based Workflow Access Control Model. *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, IEEE, 1330–1334.

74. Maarten Botterman. 2009. *Internet of Things: an early reality of the Future Internet*. Prague.

75. Parikshit N Mahalle, Bayu Anggorojati, Neeli R Prasad, and Ramjee Prasad. 2013. Identity Authentication and Capability Based Access Control (IACAC) for the Internet of Things. *Journal of Cyber Security and Mobility* 1, 4: 309–348.

76. P M Mell and Timothy Grance. 2011. *The NIST definition of cloud computing*. Gaithersburg, MD.

77. Stéphane Ménoret, Marc Roelands, Raffaele Giaffreda, and Swaytha Sasidharan. 2014. *iCore, Final architecture reference model.* .

78. Stéphane Ménoret, Marc Roelands, Raffaele Giaffreda, and Swaytha Sasidharan. 2014. *iCore, Final architecture reference model.* .

79. Microsoft Corporation. 2008. Distributed Component Object Model Specification (DCOM v6.1). Retrieved March 4, 2018 from https://msdn.microsoft.com/library/cc201989.aspx.

80. Mischa Möstl, Johannes Schlatow, Rolf Ernst, Henry Hoffmann, Arif Merchant, and Alexander Shraer. 2016. Self-aware systems for the internet-of-things. *Proceedings of the Eleventh IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis - CODES '16*: 1–9.

81. Andy Mulholland, Russ Daniels, Tim Hall, Mary Johnson, and Pete Chargin. 2008. *The Cloud and SOA, Creating an Architecture for Today and for the Future.* .

82. Nanjangud Narendra, Karthikeyan Ponnalagu, Aditya Ghose, and Srikanth Tamilselvam. 2015. Goal-Driven Context-Aware Data Filtering in IoT-Based Systems. *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, IEEE, 2172–2179.

83. Etri Ning, Kong Cnnic, Noel Crespi, Telecom Sudparis, and Ilyoung Chong. 2012. *The internet of things : concept and problem statement.* .

84. Takayuki Nishio, Ryoichi Shinkuma, Tatsuro Takahashi, and Narayan B. Mandayam. 2013. Service-oriented heterogeneous resource sharing for optimizing service latency in mobile cloud. *Proceedings of the first international workshop on Mobile cloud computing & networking - MobileCloud '13*: 19.

85. Object Management Group. 2012. *Common Object Request Broker Architecture (CORBA v3.3)*. Framingham, MA.

86. Se Won Oh and Hyeon Soo Kim. 2014. Decentralized access permission control using resource-oriented architecture for the Web of Things. *16th International Conference on Advanced Communication Technology*, Global IT Research Institute (GIRI), 749–753.

87. OpenFogConsortium. 2017. *OpenFog Reference Architecture for Fog Computing.* .

88. OpenFogConsortium. 2017. *OpenFog Reference Architecture for Fog Computing.* .

89. Jordan Pascual-Espada. 2012. Service Orchestration on the Internet of Things. *International Journal of Interactive Multimedia and Artificial Intelligence* 1, 7: 76.

90. Charith Perera, Prem Prakash Jayaraman, Arkady Zaslavsky, Dimitrios Georgakopoulos, and Peter Christen. 2014. MOSDEN: An Internet of Things Middleware

for Resource Constrained Mobile Devices. *2014 47th Hawaii International Conference on System Sciences*, IEEE, 1053–1062.

91. Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. 2013. Context Aware Computing for The Internet of Things: A Survey. 1–41.

92. Filip Perich, Anupam Joshi, Yelena Yesha, and Tim Finin. 2005. Collaborative joins in a pervasive computing environment. *The VLDB Journal* 14, 2: 182–196.

93. Sareh Fotuhi Piraghaj, Amir Vahid Dastjerdi, Rodrigo N. Calheiros, and Rajkumar Buyya. 2016. ContainerCloudSim: An environment for modeling and simulation of containers in cloud data centers. *Software: Practice and Experience*.

94. Jason Pontin. 2005. ETC: Bill Joy's Six Webs. *MIT Technology Review*.

95. Elli Rapti, Anthony Karageorgos, and Vassilis C. Gerogiannis. 2015. Decentralised service composition using potential fields in internet of things applications. *Procedia Computer Science* 52, 1: 700–706.

96. Smith Reid Garfield. 1980. The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. *IEEE Transactions on Computers* C–29, 12: 1104–1113.

97. Wayne Robbins and Schahram Dustdar. Collaborative Computing — Area Overview. In *Encyclopedia of Multimedia*. Springer-Verlag, New York, 59–70.

98. Rodrigo Roman, Javier Lopez, and Masahiro Mambo. 2016. Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges. *Future Generation Computer Systems* 78: 680–698.

99. D. Rotondi, S. Piccione, G. Altomare, et al. 2013. *Internet of Things at Work: Final framework architecture specification.* .

100. Luis Sanchez, Luis Muñoz, Jose Antonio Galache, et al. 2014. SmartSantander: IoT experimentation over a smart city testbed. *Computer Networks* 61: 217–238.

101. Mahadev Satyanarayanan. 2015. A Brief History of Cloud Offload: A Personal Journey from Odyssey Through Cyber Foraging to Cloudlets. *ACM SIGMOBILE Mobile Computing and Communications Review* 18, 4: 19–23.

102. Mahadev Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. 2009. The Case for VM-Based Cloudlets in Mobile Computing. *IEEE Pervasive Computing* 8, 4: 14–23.

103. Mahadev Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. 2009. The Case for VM-Based Cloudlets in Mobile Computing. *IEEE Pervasive Computing* 8, 4: 14–23.

104. Sensebridge Research and Collaboration Group. 2017. The Heart Spark heart beat tracker. *Noisebridge*. Retrieved from https://sensebridge.net/.

105. Alexandru Serbanati, Carlo Maria, and Ugo Biader. 2011. Building Blocks of the Internet of Things: State of the Art and Beyond. In *Deploying RFID - Challenges, Solutions, and Open Issues*. InTech.

106. Rob Shepherd, John Story, and Saad Mansoor. 2004. Parallel Computation in Mobile Systems Using Bluetooth Scatternets and Java. *International Conference on Parallel and Distributed Computing and Networks*, 159–164.

107. Jie Shi, Darren Sim, Yingjiu Li, and Robert Deng. 2012. SecDS: A Secure EPC Discovery Services System in EPCglobal Network. *Proceedings of the second ACM conference on Data and Application Security and Privacy - CODASKY '12*, ACM Press, 267.

108. João Nuno Silva, Luís Veiga, and Paulo Ferreira. 2008. SPADE: scheduler for parallel and distributed execution from mobile devices. *Proceedings of the 6th*

international workshop on Middleware for pervasive and ad-hoc computing - MPAC '08, ACM Press, 25–30.

109. Jatinder Singh, Thomas Pasquier, Jean Bacon, Julia Powles, and David Eyers. 2016. Big ideas paper : Policy-driven middleware for a legally-compliant Internet of Things. *Middleware, '16*.

110. Statista. 2018. Global statistics and survies - Statista Portal. Retrieved January 20, 2018 from https://www.statista.com.

111. Ivan Stojmenovic and Sheng Wen. 2014. The Fog Computing Paradigm: Scenarios and Security Issues. *Proceedings of the 2014 Federated Conference on Computer Science and Information Systems* 2: 1–8.

112. Ian J. Taylor and Andrew Harrison. 2009. *From P2P and Grids to Services on the Web*. Springer London, London.

113. Thiago Teixeira, Sara Hachem, Valérie Issarny, and Nikolaos Georgantas. 2011. Service Oriented Middleware for the Internet of Things: A Perspective. In *In Towards a Service-Based Internet*. 220–229.

114. Frederic Thiesse and Florian Michahelles. 2006. An overview of EPC technology. *Sensor Review* 26, 2: 101–105.

115. ThingSpeak. 2017. IoT analytics Platform Service. *ioBridge Inc.* Retrieved September 3, 2017 from https://thingspeak.com/.

116. Chai K Toh. 2002. *Ad Hoc Mobile Wireless Networks: Protocols and Systems*. Pearson Education.

117. William Tolone, Gail-Joon Ahn, Tanusree Pai, and Seng-Phil Hong. 2005. Access control in collaborative systems. *ACM Computing Surveys* 37, 1: 29–41.

118. Dieter Uckelmann, Mark Harrison, and Florian Michahelles. 2011. *Architecting the Internet of Things*. Springer Berlin Heidelberg, Berlin, Heidelberg, Heidelberg.

119. Luis M. Vaquero and Luis Rodero-Merino. 2014. Finding your Way in the Fog. *ACM SIGCOMM Computer Communication Review* 44, 5: 27–32.

120. O Vermesan and P Friess. 2013. *Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems*. River Publishers.

121. Luis Villasenor-Gonzalez, Ying Ge, and Louise Lamont. 2005. HOLSR: A hierarchical proactive routing mechanism for mobile ad hoc networks. *IEEE Communications Magazine* 43, 118–125.

122. VMware. 2017. VMware Fusion [Software Hypervisor]. *VMware, Inc.* Retrieved November 25, 2017 from www.vmware.com/products/fusion/.

123. Jim Waldo. 2000. *The Jini Specifications*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

124. Xively.com. 2018. IoT Platform for Connected Objects. *LogMeIn Inc.* Retrieved from https://www.xively.com/.

125. Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher, and Valerie Young. 2015. *Mobile Edge Computing A key technology towards 5G.* .

126. Guoping Zhang; and Jing Liu. 2011. A Model of Workflow-oriented Attributed Based Access Control. *International Journal of Computer Network and Information Security (IJCNIS)* 3, 1: 47–53.

127. Wang et al. - 2017 - A Survey on Mobile Edge Networks Convergence of C.pdf. .