THÈSE

pour obtenir le grade de

Docteur de l'Université Sorbonne Paris Nord

Discipline : "Ingénierie Informatique"

présentée et soutenue publiquement par

Hoan LE

le 22 Avril 2021

Une architecture de fournisseurs de services FOG pour la gestion des données de l'Internet des Objets

Directeur de thèse : **Dr. Nadjib Achir**

Co-encadrant de thèse : **Dr. Khaled Boussetta**

JURY

Sidi Mohammed Senouci,	Professeur, Université de Bourgogne	Président du Jury
Selma Boumerdassi,	Maître de conférences HDR, CNAM Paris	Rapporteuse
Yacine Ghamri-Doudane,	Professeur, Université La Rochelle	Rapporteur
Christophe Cérin,	Professeur, Université Sorbonne Paris Nord	Examinateur
Nadjib Achir,	Maître de conférences HDR, USPN	Directeur
Khaled Boussetta,	Maître de conférences HDR, USPN	Co-encadrant

Thesis

Submitted for the degree of Doctor of Philosophy of

Université Sorbonne Paris Nord

Specialization : "Computer Engineering"

presented and defended by

Hoan LE

22nd April 2021

A Fog Services Provider Architecture for Internet of Things data management

Supervisor : **Dr. Nadjib Achir**

co-supervisor : **Dr. Khaled Boussetta**

Committee

Sidi Mohammed Senouci,	Professor, Université de Bourgogne	Chairman
Selma Boumerdassi,	Associate professor, CNAM Paris	Reviewer
Yacine Ghamri-Doudane,	Professor, Université La Rochelle	Reviewer
Christophe Cérin,	Professor, Université Sorbonne Paris Nord.	Examiner
Nadjib Achir,	Associate professor, Université Sorbonne Paris Nord	Supervisor
Khaled Boussetta,	Associate professor, Université Sorbonne Paris Nord	Co-Supervisor

L2TI – EA3043
École doctorale Galilée – Université Sorbonne Paris Nord
99, avenue Jean-Baptiste Clément
93430 Villetaneuse.

Résumé

Les deux dernières décennies ont vu d'énormes changements dans la façon dont la vie des gens est affectée par les TIC. Les appareils mobiles ont subi une transformation significative des petits appareils avec une capacité limitée aux mini-ordinateurs mobiles. En parallèle, on assiste également à l'émergence de l'Internet des objets (IoT) comme nouveau paradigme. L'Internet des objets a apporté de nombreux avantages à la vie quotidienne des gens et a eu un impact positif sur beaucoup de domaines d'application tels que l'industrie, la santé, l'environnement, les transports, la ville intelligente, la maison intelligente, etc. L'une des approches adoptées pour améliorer les performances consiste à réduire la charge sur les appareils et leur permettre d'accéder à des ressources distantes dans le Cloud. Cela a donné naissance au paradigme du Cloud Computing. Malheureusement, même si les Clouds disposent d'importantes ressources de calcul et de stockage, ils sont généralement éloignés des utilisateurs mobiles, ce qui rend difficile de répondre aux exigences des applications sensibles aux délais. Pour surmonter cette limitation, une approche possible consiste à déplacer les services du Cloud vers la périphérie du réseau. Cela a conduit à l'émergence du concept de FOG Computing. Le FOG Computing donne la possibilité de déployer des services plus près des appareils IoT et fournit des solutions pour résoudre les problèmes d'hétérogénéité et d'interopérabilité.

Dans cette thèse, nous nous concentrons sur la conception d'une architecture de service, appelé **`Fog Services Provider`** (FSP), basée sur le paradigme FOG, pour fournir des services pour l'IoT. Notre première contribution est de définir les composants critiques du Fog Computing et de l'IoT dans notre architecture. Nous analysons ensuite les exigences et la conception détaillée des composants de l'architecture du service Fog et du mécanisme de gestion des données. Enfin, nous proposons un **`unified data model`** (FSPontex) qui résout le problème d'hétérogénéité et d'interopérabilité entre différents équipements IoT, applications ou encore d'autres systèmes IoT à l'aide du concept d'ontologie.

Abstract

The last two decades have seen tremendous changes in the way people's lives are affected by IT. Mobile devices have undergone a significant transformation from small devices with limited capacity to mobile mini-computers. In parallel, we are also witnessing the emergence of the Internet of Things (IoT) as a new paradigm. The Internet of Things (IoT) has brought many advantages to people's daily lives and positively impacted most application domains such as industry, healthcare, environment, transportation, smart city, smart home, etc. One of the approaches adopted to improve performance is to reduce the load on the devices and to allow them to access remote resources in the Cloud. This gave birth to the Cloud Computing paradigm. Unfortunately, even though Clouds have significant computing and storage resources, they are generally remote from mobile users, which makes hard to fulfill IoT latency-sensitive applications requirement. To overcome this limitation, one possible approach is to move services from the Cloud to the network edge. This leads to the emergence of the FoG computing paradigm. Fog computing brings the ability to deploy computing services closer to IoT devices and provides services that solve data heterogeneity and interoperability issues.

In this thesis, we focus on analyzing and designing a service architecture based on the Fog computing paradigm called **`Fog Services Provider`** (FSP). Our goal is to exploit the Fog computing system's useful characteristics and provide efficient support services for IoT devices. Our first contribution is to define some of the critical components in Fog computing and IoT. We then analyze the requirements and detailed design of the Fog service architecture components and the data management mechanism for IoT heterogeneous devices. Finally, we propose a **`unified data model`** (FSPontex) that solves the problem of heterogeneity and interoperability in different applications and with other IoT systems using web technology with the help of **`Ontology`**.

Acknowledgments

First of all, I would like to extend my great thanks to professor *Sidi Mohammed Senouci* and professor *Christophe Cérin* for their acceptance to be members of the committee, and professor *Selma Boumerdassi* and professor *Yacine Ghamri-Doudane* for their reviews of my thesis. I appreciate all their comments to make my thesis better.

I would like to express my great gratitude to my supervisors, *Nadjib Achir* and *Khaled Boussetta*, for all their guides and supports during the past over four years. From them, I have learned the serious working manner and attitude.

Vietnam Ministry of Education and Training who funds my whole the study in France. And my PhD study is also made possible thank to *Electric Power University*, particularly the *Faculty of Information Technology (FIT)* where I have been working as a lecture. I would like to thank my colleagues at FIT for their hard work to offer me the time for my study aboard.

I would like to thank all members of *L2TI* who have made my time in the laboratory remembering. I would like to thank to fellow doctoral students in L2TI whose sharing help me a lot improve my presentation skills in seminars.

Outside L2TI, I would like to express my warming thanks to my Vietnamese friends in *University Sorbonne Paris Nord* (*University Paris 13*). Son, Long, Viet, Y, Thuy, Hieu, Thi, Thai, Hoa, Gia, Tuan, Sy, Tram, Quang, and many others that I cannot list all. You really made my time in the university full of joys, funs and warmth. I will remember their considerateness and sharings as well as all travels and parties we made together.

My warmest appreciations go to my parents, parent's in-law, uncles, aunts, brothers

and sisters who always care of me and encourage me along my far-way living.

Finally, from the bottom of my heart, my love and gratitude go to my wife, *Thanh Huyen* who is always by my side, encourages and supports me in the good and hard time of my life. My daughter *Hoang Ngan* and my son *Hai Phong* are appreciated for their understanding and being good for me to throw myself in work.

Thank you very much everyone !

Villetaneuse, April 2021

Hoan LE

Contents

List of Tables

List of Figures

Abbreviations

6LoWPAN	IPv6 over Low Power Wireless Personal Area Networks
AP	Access Point
AMQP	Advanced Message Queuing Protocol
AWGN	Additive White Gaussian Noise
API	Application Programming Interface
BL-EST	Bottom Level Early Start Time
CDMA	Code Division Multiple Access
CoAP	Constrained Application Protocol
CPU	Central Pprocessing Unit
CRIU	Checkpoint/Restore In *U*serspace
CSV	Comma Separated Values
DAG	Directed Acyclic Graph
DC	Data Centers
DNS	Domain Name System
DOTA	Delay bounded Optimal egde server-cloudleT- user Association heuristic
DTN	Delay-Tolerant Networking
ESB	Enterprise Service Bus
FCO	Full Cloud Offloading policy
FSP	Fog Services Provider

HEM	Heavy Edge Merge
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
IBM	International Business Machines
ICN	Information Centric Networking
IRI	Internationalized Resource Identifier
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IoT	Internet of Things
IP	Internet Protocol
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
LAH	Lagrangian Adjustment Heuristic
LXC	LinuX Containers
NIST	National Institute of Standards and Technology
M2M	Machine to Machine
MANET	Mobile Adhoc Network
MCC	Mobile Cloud Computing
MCOP	Minimun Cost Offloading Partitioning
MEC	Mobile Edge Computing
MQTT	Message Queuing Telemetry Transport
NCO	Nearset edge server-Cloudlet- Offloading policy
NFC	Near Field Communication
OFDMA	Orthogonal Frequency-Division Multiple Access

OSI	Open Systems Interconnection
OS	Operating System
OWL	Web Ontology Language
P2P	Peer-to-Peer
PaaS	Platform as a service
PKI	Public Key Infrastructure
QAM	Quadrature Amplitude Modulation
QoS	Quality of Service
RDB	Relational Database
RDF	Resource Description Framework
REST	Representational State Transfer
RFID	Radio-Frequency Identification
RMI	Remote Method Invocation
RTT	Round Trip Time
SaaS	Software as a Service
SDA	Semantic Data Annotation
SDK	Software Development Kit
SDN	Software Defined Networking
SGS	Smart Gateway Service
SLAs	Service Level Agreements
SNR	Signal-to-Noise Ratio
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SW	Semantic Web
URI	Universal Resources Identifier
WFM	Wave Front Method
WLAN	Wireless Local Area Network

WSAN	**W**ireless **S**ensors and **A**ctuators **N**etwork
WSN	**W**ireless ensor **N**etworks
XML	**E**xtensible **M**arkup **L**anguage
XMPP	**E**xtensible **M**essaging and **P**resence **P**rotocol

Chapter 1

Introduction

> *Education is the most powerful weapon which you can use to change the world.*
>
> Nelson Mandela

Chapter content

The prominence of the Internet of Things (IoT) has increased applications in the domain of ubiquitous and context-aware computing. IoT will enable the interconnection and intercommunication billions of devices, known as smart objects or smart things which deployed pervasively and distributed geography. These things or devices caused an unprecedented generation of the enormous and heterogeneous amount of data. Traditional computing as Cloud computing has come and served as an efficient way to process and store these data. However, the integration of IoT with Cloud computing face many challenges, one of those challenges is the growth of real-time and latency-sensitive applications and the limitation of network bandwidth. This leads to unnecessary communication not only burdens the core network, but also the data center in the cloud. Therefore, a new paradigm that seems to be promisingly named in the literature as fog computing, it will complement the cloud solution. In essence, Fog computing extends the cloud services to the edge of the network and distributes these services such as storage, computation, control, and network closer to end-users with the purpose to optimize low-latency, mobility support, scalability, security and privacy, energy efficiency, data management. In this chapter, we present the motivation and challenges of fog computing architectures, applications, requirements, and open directions of research in this domain of computing.

1.1 Context

The Internet of things (IoT) [1, 2], first introduced by Kevin Ashton in 1998, refers to an emerging paradigm that consists in the interconnection of physical devices, vehicles, buildings and other items such as sensors and actuators to collect and exchange data. Theses ”things” connected to each other in order to form a much larger system and enabling new ubiquitous and pervasive computing services. This new paradigm is starting to transform how we live our lives, but all of the added convenience and increased efficiency comes at a cost. Indeed, according to Cisco, more than 50 billion devices will be connected to the Internet in 2020. Until 2022, 1 trillion networked sensors will be embedded in the world around us, in 20 years it will be up to 45 trillion [3]. As a consequence, the IoT is expected to generate a large amount of data, which in turn puts a tremendous strain on the Internet infrastructure. As a result, researchers are working to find ways to reduce that pressure and solve the problem of analyzing and processing a large amount of data. Among existing solutions, Cloud computing

will be a major part of future development of IoT, and especially by making all of the connected devices work together. Indeed, storing and processing the data locally in IoT devices is hardly possible.

The research community has envisioned that the Cloud [4, 5] can act as an intermediate layer between the "things"and the applications and thus hiding all the complexity of the resources management and the functionalities necessary to implement the latter. This intermediate layer will impact future application development, where information gathering, processing, and transmission will produce new challenges that have to be addressed. The integration of IoT into the cloud can be a good way to deal with some issues such as performance, reliability, security, and privacy [6]. The Cloud also simplifies the data flow processing of IoT by providing fast, low-cost installation and integration for complex data processing and deployments [7]. The benefits of integrating IoT into the Cloud computing can be summarized as the follows [6]:

- **`Communication`**: Widely deployed applications and data sharing are prominent features of a Cloud-based IoT paradigm. Ubiquitous applications can be implemented through the IoT environment, and use automated communications that can be leveraged to facilitate low-cost data delivery and collection.

- **`Storage`**: Data from billions of devices is generated in different type of data such as structured or non-structured data with several characteristic: (i) variety (data types), (ii) volume (data size), (iii) velocity (data generation frequency) that can be considered as Big Data. The Cloud is one of the most cost-effective and suitable solutions to store these data.

- **`Processing capabilities`**: IoT devices are essentially limited in terms of processing capabilities. Therefore, its generated data needs to be transferred to nodes that have high computing capabilities. Cloud usually provides services with unlimited processing capabilities through virtualization mechanisms and on-demand data usage model.

Although Cloud computing has solved several problems with support for IoT. However, the geographical distance to the Cloud can significantly reduce latency for some time-sensitive application domains such as healthcare, smart cities, smart home, video surveillance. To address this issue, this convergence between the IoT and Cloud Computing reveals a new paradigm that seems to be promising named in the literature

as Fog computing [8, 9]. Fog computing is known as a clever name, which can sometimes be referred to as edge computing. It allows data collection and processing at edge computing devices instead of using the cloud or at a remote data center. By leveraging this paradigm, data generated from the sensors and other connected devices is sent and instantly processed at a nearby edge computing device. This could be a gateway device, such as a switch or router that processors and analyzes this data. In Fog environments, it might play a role as a Fog node. These Fog nodes should be deployed in distributed geography to support time-sensitive applications. Moreover, handling this growth of the number of devices and information produced by them brings the need to have platforms and infrastructure that support different requirements for IoT System. Additionally, generated data from a large number of heterogeneous devices also need to be processed at local level to decrease a bottleneck for critical IoT applications by providing a unified data information model. Therefore, these IoT platforms need to provide several mechanisms not only to help to quickly connect devices and easily deploy services and applications but also to effectively manage produced data. The elastic resource provisioning and the ability to scale are the main requirements to deal with this big number of devices and to face the dynamic nature of IoT systems.

1.2 Challenges

Fog computing is a new computational paradigm and being the extension of cloud computing paradigm to handle IoT applications related issues at the edge of network. In nature, fog computing is similar to cloud computing but with different characteristics. Fog computing architecture will be able to provide possibilities of data processing, computation, storage, networking and application services to IoT systems accordingly and effectively [9]. However, Fog computing faces new challenges besides those inherited benefits from cloud computing. In Fog computing, computational nodes are heterogeneous and distributed as well as dealing with aspects of constrained environment. In this section, we will present several challenges in the perspective of fog computing.

- **`Heterogeneity`**: IoT is a complex ecosystem with many heterogeneous networks [10]. One of the first challenges that we have to deal with in IoT is the large heterogeneity of the "*things*". This heterogeneity can be divided according to a layered architecture form southern layer to northern layer as the follows:

1) **`Devices layer`**: Various devices use different communication technology standards to provide sensing data in different type of formats for the Fog nodes to process and make decisions. How to manage these devices in an integrated way?

2) **`Networking layer`**: This layer is used different topology such as star network, tree network, and hybrid network topology that allows data transfer from data sources to processing nodes. Therefore, it is necessary to build an effective topology to manage and secure data throughput and energy consumption issues.

3) **`Processing nodes layer`**: Nodes can be developed into an architecture, a platform. They are usually designed for deploying scalable, intelligent, and interoperable IoT applications in different domains. For example smart city is possible to have systems for specific domains such as, smart homes, smart traffic, public services, and smart industries. These systems can generate different kinds of context information in various formats and specifications. The context sharing can be a feasible solution in these processing nodes to provide interoperability.

4) **`Application layer`**: Existing IoT services and applications have been conceived as isolated vertical solutions, in which all system components are tightly coupled to specific application context, specific hardware, specific software, etc. In this case, one of the big challenges is to be able to unifying platforms and middleware, and also provide interoperable programming interfaces [11–13].

- **`Manageability`**: The objective of the heterogeneity issue should be considered in a deployable, scalable, and interoperable unified architecture. It tries to answer the question of what is its manageability? Manageability is the most practical way to able to bridge the gap between the organizational and technological silos in IoT. It can be managed in cross-platform applications, cross-platforms, or cross-domains. We define several criteria as main challenges for a manageable architecture as follows:

  1) **`Accessibility`**: a developed mechanism for the ability to support access to data generated by IoT platforms or applications. This means allowing

some web services such as RESTful, API to be able to access to services inside or outside of the architecture with authentication and authorization mechanisms.

2) **`Services management`**: functionalities may be developed as separate services which can be deployed independently on different platforms. Besides, the communication mechanism between services should also be considered to satisfy the internal and external cross communication capabilities.

3) **`Loosely coupled`**: a service-oriented distributed architecture with a single responsibility, without many dependencies, allowing teams to work and deploy independently, which can be developed based on a loosely coupled (also know as microservices) approach.

4) **`Standards-based`**: data management for heterogeneous IoT devices in a distributed architecture has to be considered. Instead of sending the data to the Cloud, resources placed very close to the data producers can be used for local processing, data analysis, and fast decision-making and thus provide better quality of service performances. Moreover, the IoT devices are heterogeneous in terms of communication protocols, data formats, and technologies. This not only causes issues of device interoperability but also its data interoperability and semantics interoperability issues [14]. Therefore, a standards-based data model (e.g. RDF [15], JSON-LD [16], and ontologies) is required to integrate these data sources.

There are some of the existing works on computing architecture for IoT based on typical characteristics of Fog computing paradigm to deal with issues for specific characteristics and not focus on designing a overview architecture to handle a heterogeneous set of devices and manage diverse data sources from these devices. For example, a Fog-based IoT resource management proposed to evaluate resource prediction, resource allocation and pricing [17]. To address several challenges that related to distributed devices and processing, according to [18], the authors have introduced a dataflow approach to evaluate for the development of IoT applications in the Fog by implementing a distributed dataflow programing model based on Node-RED (a programming tool for wiring together hardware devices, APIs and online service [19]). For cost-efficient provisioning limited resources, [20] focuses on some tasks such as data consumer association,

task distribution, and virtual machine placement to fowards a Fog computing based scheme through parameters such as service latency, power consumption, and cost.

In this thesis we tackle the Fog computing-based services architecture that should be provided in order to ensure such requirements as well as to bring other benefits such as management of heterogeneous IoT devices that use different technologies and investigate interoperability for these devices. On the other hand, we define several components and services in our architecture to support and build a unified data model for Fog computing architecture. In particular, we focus on to solve some challenges as the following:

1. How to design an efficient architecture that enables rapid deployment, management and scaling of IoT applications?

2. How to handle a heterogeneous set of devices?

3. How to effectively manage data and ensure a primary task is to build an information model with semantic interoperability for cross-IoT application domains?

4. Where to deploy services in a large set of fog nodes while ensuring the performance of IoT applications in terms of latency, energy consumption and scalability?

1.3 List of contributions

The variety of IoT applications has created many challenges as mentioned above. Support for large numbers of connected devices that can exchange data and interact with each other requires IoT architectures that are reliable, secure, energy-efficient, and capable of provides services for applications in other systems. These IoT architectures serve as an orchestrator that not only enable to combine IoT platforms and manage heterogeneous devices, services or applications, and users, but also provide data processing capabilities in different contexts in a cyber-physical world. The objective of this thesis is to study and design an efficient **Fog Services Provider** based on the Fog computing paradigm that represents a real-world test-bed and is used as an environment for execution, managing, and analysis of IoT services in a fog environment. Our main contributions are:

- We address the problem of handling a wide diversity of network access technologies such as Wi-Fi, Cellular, Lo-Ra, Zigbee, or Bluetooth by designing a multi-technology services architecture based on the Fog computing paradigm for IoT devices to perceive and manage flexible and heterogeneous multi-network

access technologies in ubiquitous environment. A novel aspect of this service architecture is to integrate a smart gateway service with a data scheme to describe and store information of IoT multi-technology devices through an access services. Besides, we also propose procedures to attach and detach IoT devices into fog architecture. The main result in our paper published in 10th International Conference on Networks of the Future (NoF2019) [21]

- We have extended the last proposal by elaborating on the basic notions, components, and services in the Fog computing architecture. This architecture not only allows heterogeneous IoT devices but also provides a data model based on semantic web technology with the help of ontology. In addition, the data model enables to support interoperability in IoT by organizing data sources generated from IoT devices based on a Resource Description Framework (RDF) which is used a triplet (Subject-Predicate-Object) to describe element in in a statement. Beside, we implement an algorithm to covert a relational database (RDB) to RDF to facilitate for management of heterogeneous IoT devices. The result in our paper published in 11th International Conference on Networks of the Future (NoF2020) [22].

- The last contribution focuses on the interoperability in IoT. The proposed approach supports to handle a large-scale heterogeneous data and process it in real-time or near real-time by using a common unified ontology. This proposal is extended from the NoF2020 with aiming at designing a unified semantic data model to manage different type of IoT devices in Fog environment. The submitted paper in 13th International Conference on Global Information Infrastructure and Networking Symposium (GIIS 2020) [23].

1.4 Thesis outline

In this chapter, we present new challenges and issues for fog computing environment. The rest of this manuscript is organized as follows:

- **Chapter 2 Backgrounds:** This chapter explains the background that is necessary for a better comprehension of our contributions in this thesis.

- **Chapter 3:** This chapter presents our proposed architecture based on Fog computing paradigm. A services architecture that enables flexible deployment, manage-

ment, and scaling for IoT applications. This chapter also details several the new concepts of components in the Fog architecture and introduce a data management mechanism for heterogeneous devices.

- **Chapter 4:** This chapter introduces data management management in Fog Services Provider by providing data processing approaches at a raw data stage and a semantic data stage.

- **Chapter 5:** This chapter discusses the challenges in the integration of different data sources which handled by Fog Services Provider by designing a unified data model. We presents in details a new schema between data objects and their relationships to address the need for interoperability and heterogeneity.

- **Conclusion 6:** This chapter has concluded this thesis and provides several prospects and upcoming works.

CHAPTER 2

Backgrounds

Life is like riding a bicycle.
To keep your balance,
you must keep moving.

Albert Einstein

Abstract

Internet of Things (IoT) systems connect the physical world with the Internet. In essence, IoT works by connecting real-world interfaces to the Internet, such as sensors that provide data and actuators that act on their environment. In effect, IoT systems provide the technologies and the tools to instrument, quantify, and actuate the physical world. In this chapter, we summarize some research works related to components and their architectures. We focus on several computing paradigms and prominent characteristics of Fog computing in IoT.

Chapter content

This chapter highlights the key issues and motivations involved in the importance of the fog computing paradigm. Subsequently, we will discuss the main characteristics of this emerging fog computing paradigm. Finally, the chapter addresses the major limitations and constraints essential to the analysis and design of fog computing architecture in Internet of things environments. Besides, the chapter introduces the state-of-the-art and fog computing challenges for the Internet of things. These aspects are regarded as requirements for comprehending the other work discussed in this paper.

Chapter 2 is organized as follows. Section 2.1 introduces the diversity of Internet of Things and its components. Section 2.2 discusses some communication protocols and standards in IoT. We presents some approaches that is used to develop IoT architectures in section 2.3 and section 2.4. One of the efficient paradigms to support IoT is cloud computing, introduced in section 2.5. Due to limited latency, several paradigms are proposed to overcome cloud computing issues, and these paradigms are presented in section 2.6. However, the extension for computing and interaction with cloud computing is still necessary. There is a need for appropriate solutions to address issues related to heterogeneity, interoperability, and scalability in IoT environments. Therefore, an efficient emerging solution is fog computing mentioned in section 2.7. The last section 2.8 draws a conclusion.

2.1 Introduction

Currently, IoT is a flourishing technology trend that is being discussed all over the world. The term “Internet of Things”was originally used in a presentation at Procter & Gamble [24] in June 1999 by Kevin Ashton [25], co-founder of the Auto-ID Center at MIT. At that time, in his presentation, he combined radio frequency identification (RFID) with the Internet, resulting in the term **`Internet of Things`**. It means the connection of enormous amounts of *things* to the Internet. These *things* are more and more intelligent and used today, including a range from sensors, actuators, wearables, cell phones, and on-board computers to micro-data centers that are called **`IoT devices`**. Many different communication technologies and protocols are used to connect IoT devices to the Internet, such as Wi-Fi, Bluetooth, ZigBee, LR-WPAN, Z-Wave, RFID, and Near Field Communication (NFC) [26]. In which, two RFID and NFC technologies are utilized in the near vicinity of ToT devices to be able to identify, track, and

authenticate [27].

With recent advances in miniaturization and falling costs of RFID, sensor networks, NFC, wireless communications, technologies, and applications, the IoT has become relevant to industry and end-users. Ubiquitous connectivity is one of the key requirements in IoT applications. These applications need to support a diverse set of devices and communication protocols. It not only provides support for tiny sensors that are capable of sensing and reporting desired environment events but must also provide full data to powerful back-end servers that are used for data analysis and extraction. This also requires integrating mobile devices, edge devices like routers and smart hubs, and humans in the loop as controllers. The detection of the physical state of things by sensors while gathering and processing detailed data enables an immediate response to changes in the real world. This fully interactive and responsive network provides a massive potential for citizens, consumers, and businesses.

Initially, RFID was the first introduced technology to attract attention to the IoT [28]. Until now, RFID is still one of the driving factors in the evaluation of IoT applications. This is the case because the overall vision of connecting everything to the Internet is easy to apply through called RFID tags [29]. RFID tags are small, uniquely identifiable, and programmable microprocessors connected to an antenna to communicate with RFID readers. However, with further technological achievements, wireless sensor networks (WSN) and Bluetooth-enabled devices accelerated the IoT trend's widespread adoption. The WSN approach is an interesting related technology concerning IoT's future trend. Although WSN and IoT can be considered as competing technologies, many similarities can be found. Sensor networks in a WSN are composed of wireless connected sensors that can communicate with each other. A standard communication approach applied in WSNs is Peer-to-Peer (P2P) technology [30]. It is necessary for sensors to detect other sensors in close proximity and to build a network that communicates with each other in a highly geo-distributed sensor network. A typical example of an application of this kind is presented in [31], where wireless sensors are placed within the emergency rooms of John Hopkins hospital to monitor in real-time the blood oxygen and heart rate of the patients. In this case, sensors send an emergency call either to a control middleware to recheck the emergency or directly to a doctor. According to [32–34], these communication technologies and IoT applications have been studied in recent years. Nevertheless, specific characteristics and requirements such as scalability, heterogeneity

support, full integration, and real-time query processing are not yet interested. To highlight these necessary advances, this chapter lists the challenges of IoT and promising approaches, taking into account recent research and progress made in the IoT ecosystem.

Industry 4.0 is another emerging technology concept that is currently being discussed in almost any company with a production context. IoT is very similar to the concept of Industry 4.0. Industry 4.0 is the vision of digitization and automation of the manufacturing process from the first draft of a product through the entire supply chain to the final product. This includes the fields of Big Data analytics, digitalization, manufacturing, self-adaptation, artificial intelligence, etc. [35]. In the present context, IoT comprises not only sensors and actuators but also, as mentioned above, a multitude of heterogeneous devices connected to the Internet. In addition, one of the most obvious motivating factors is the usability provided by the new applications, such as smart cities, smart houses, smart cars, smart grids, smart healthcare, smart logistics, autonomous cars, and robots, and virtual and augmented reality applications. Figure 2.1 shows graphically the sectors where companies started investing in IoT. Analysts are forecasting that the number of installed devices will be in the tens of billions over the next few years and that the number of sensors will grow to hundreds of billions in the near future [36]. In a forecast, by 2019-2030, the number of IoT-connected devices in the world will grow from 7.6 billion to 24.1 billion, with sales more than tripled from $465 billion to over $1.5 trillion [37].

The diversity of IoT in terms of devices, technologies, heterogeneous communication protocols and standards, and various applications poses many challenges for the research community. Not only does it challenge the design of common standards to address hardware problems, but also the challenges of application-level communication technologies. Moreover, architecture with the ability to manage these heterogeneities is an essential solution in IoT environments.

2.2 Communication protocols and standards

The Internet of Things (IoT) transfers data between devices and users using communication protocols. According to [38], IoT communication protocols can be classified into three types of architecture based on different aspects: OSI model hierarchy, IEEE 802 protocol standard or network types. The communication protocols in the OSI model hierarchy work mainly at layers, including the application layer (CoAP, ISA100.11a, MQTT,

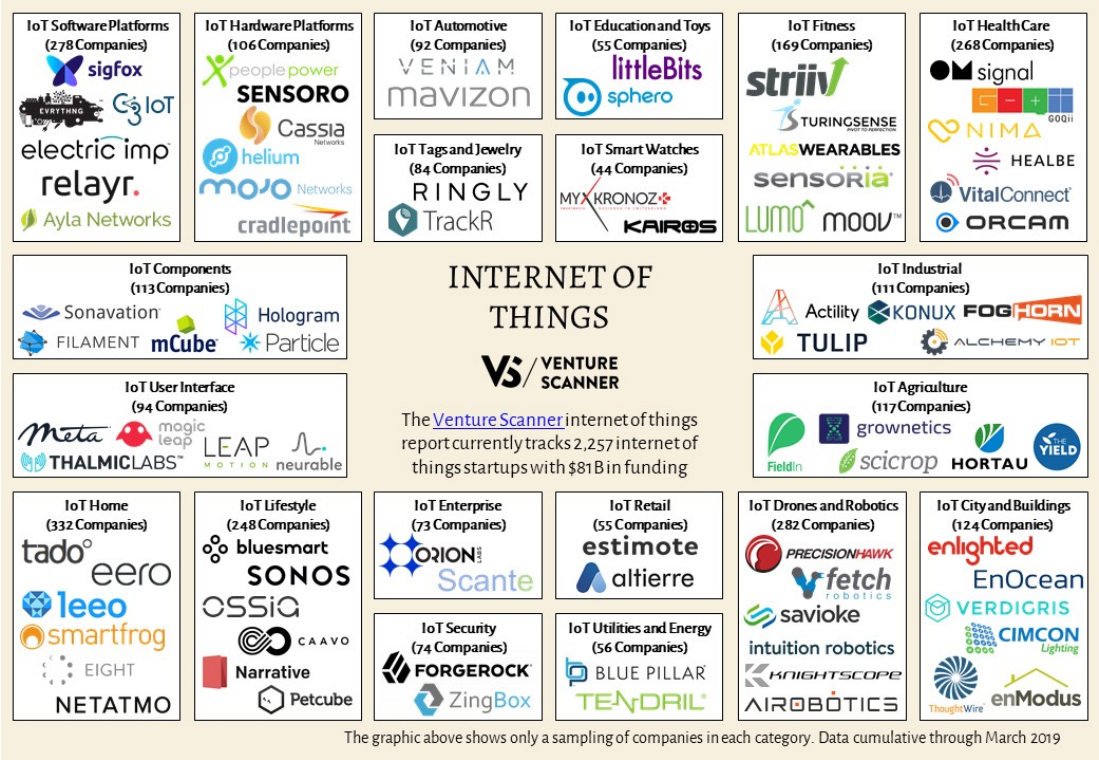

Figure 2.1: *Visual map of IoT companies group per type of business. Source: Venture Scanner, Internet of Things Sector Update.*

SOAP, Web Socket, etc.), the network layer (SMS/USSD protocol suite, TCP/UDP protocol suite, and wireless sensing protocols, and physical layer (short-range, long-range cellular, and long-range non-cellular communication protocols). IEEE 802 standard-based communication protocols include Wi-Fi, Bluetooth, ZigBee, UWB, etc. Network types divide IoT communication protocols into four types of networks: personal area network (PAN), local area network (LAN), wide area network (WAN), and mobile network. This division is necessary in resolving communication at each layer in the IoT architecture. However, providing interoperability between communication protocols is still a significant challenge.

In IoT we also need to address communication protocols in terms of topology formation, power consumption, optimization, integration, and cryptography, etc. [26, 38]. In terms of networks and communications perspective, IoT can be considered as an aggregation of different networks, including mobile networks (CDMA, 3G/4G/5G, etc.), WLANs, WSN, and Mobile Adhoc Network (MANET) [39, 40]. Seamless interconnectivity is an important prerequisite for IoT. The speed, reliability, and persistence of network communication will affect the entire IoT experience. With the appearance of high-speed mobile networks such as 5G and the increased acceptance of local and urban

network communication protocols such as Wi-Fi, Bluetooth, and WiMax, establishing an interconnected network of objects seems feasible, but dealing with different communication protocols that connect these environments is still a challenge. The means of communication and protocols are different depending on the specification of the device (CPU, memory, storage, battery life). However, the most common communication protocols and standards are listed as follows:

- **RFID** (e.g., ISO 18000 series, which includes file classes and two generations and covers both active and passive RFID tags).
- **IEEE 802** WLAN (802.11), Zigbee (802.15.4), Near Field Communication (NFC), Bluetooth (802.15.1).
- Low-power Wireless Personal Area Networks (**6LoWPAN**) standards by IETF (Internet Engineering Task Force).
- M2M protocols such as **MQTT** and **CoAP**.
- IP layer technologies, such as **IPv4**, **IPv6**.

The detailed specification for the network layer communication protocols mentioned above is presented in [41], and these are described in detail for the layers in the IoT protocol suite illustrated in Figure 2.2.

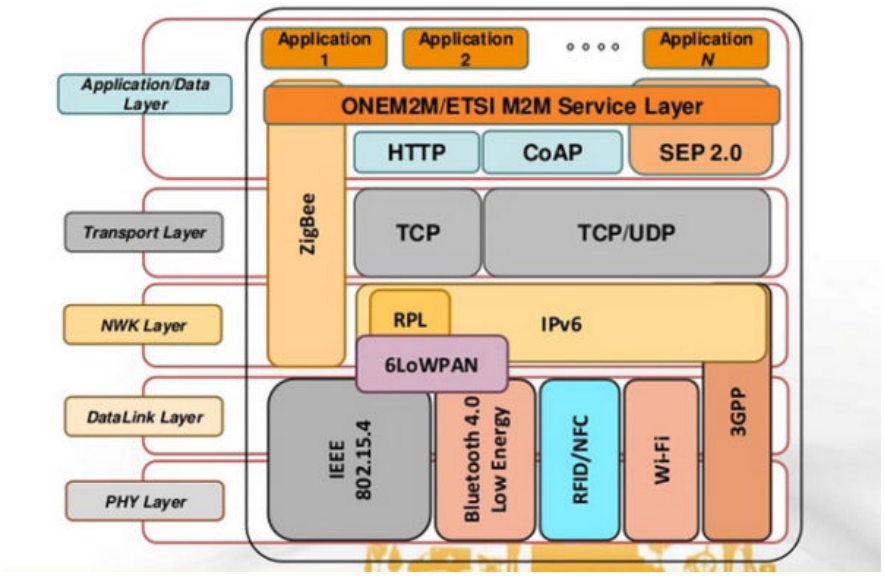

Figure 2.2: *Heterogeneous IoT standard protocols.*

Based on a classification of protocols, each communication protocol has its standards for providing some specific applications. The lack of communication protocol standards

can significantly affect the management of data from heterogeneous devices [42]. Furthermore, due to the complexity of data sources and data types, formats, and structures, there should be an effective architecture to protect these data. Several approaches proposed data protection by hiding information through semantic web technology with the help of ontology [43, 44]. IoT communication protocols can be hidden by several specific semantic properties (`annotation property`) that allow the hidden information to be transmitted securely by the communication channel.

2.3 IoT architectures

An IoT system consists of several basic components such as sensory devices, communication networks, services, applications, management, and context-aware processing of events [27]. However, combining a set of heterogeneous devices and their communication protocols into a unified architecture and create a knowledge system that makes them understandable and manageable by humans is a significant challenge. In addition, in a distributed environment, the interconnections between IoT systems and between these components are prerequisite requirements. A comprehensive system architecture for IoT must guarantee the appropriate operation of its components. Reliability is considered one of the most important design in IoT architecture [45]. In order to achieve this, it is necessary to carefully consider issues of heterogeneity, fault tolerance, and scalability in the design of IoT architecture. Because mobility and dynamic location changes with the prevalence of smart *things* have also become an important part of IoT systems, for this reason, a modern architecture needs to provide a level of adaptability to deal with dynamic impacts throughout the ecosystem.

Reference architectures and models give an overview of all the basic components of the system. It is independent of the vendor and does not specify a set of technologies. Its advantages allow other architectures to be relied on to provide a better and higher level of abstraction that can hide specific constraints or implementation details. These architectures are also introduced by some research groups such as IoT-A [46], IBM [47], and WSO2 [48]. The IoT-A focuses on developing and validating an integrated IoT network architecture and supporting building blocks, with the objective to be "the European Lighthouse Intergrated Project addressing the Internet of Things Architecture". IBM proposed an architecture capable of providing a platform for the industry for the purpose of building a consistent and consistent architecture even as the physical

capabilities change. IBM systems focus on scalability, security, usability, manageability, maintainability, etc. Figure 2.3 illustrates an outline of the extended version of a reference architecture for IoT of WSO2. The reference architecture consists of a set of components. Layers can be realized by means of specific technologies, and we will discuss options for realizing each component. There are also some cross-cutting/vertical layers such as access/identity management. The layers are Client/external communications - Web/Portal, Dashboard, APIs which is essential for defining and sharing system services and web-based dashboards for managing and accessing these APIs; Event processing and analytics (including data storage); Aggregation/bus layer – ESB and message broker built on top of communication and physical layers with relevant transports - MQTT/HTTP/XMPP/CoAP/AMQP, etc; and Devices. The cross-cutting layers are Device manager, Identity, and access management. They have the ability to uniquely identify objects and control their access level.

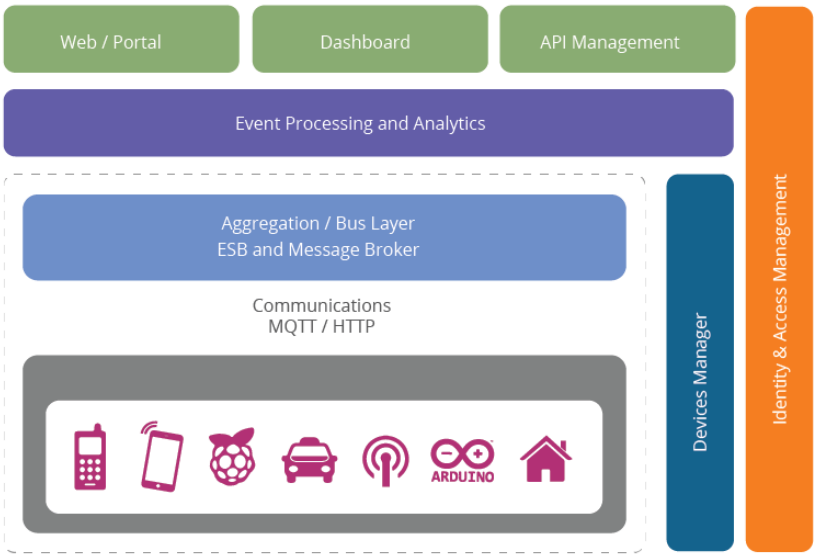

Figure 2.3: *Reference architecture for IoT [48].*

These reference architectures are essential in the design of an effective IoT architecture that can be provided both functionalities (device management, data management, services management, etc.) and non-functionalities (availability, maintainability, scalability, manageability, etc.). However, the development of architectures by any approach should also be considered. We mention in more detail several approaches in the next section.

2.4 Design approaches for IoT architecture

There are some kind of architectural models that have been discussed over time [27]. Firstly, it is a service-oriented architecture (SOA), where the capability of computation is based on the modularity of the end-to-end services. Secondly is an API-based architecture developed from different IoT-based platforms and computational needs. Finally, microservices architecture is focused on multiple, independent, self-contained application-level services that are lightweight and have their own unique data model [49]. These architectures present as follows:

2.4.1 Service-oriented architecture

In IoT, service-oriented architecture (SOA) might be essential for the service providers and users [50, 51]. SOA enables the interoperability between the heterogeneous devices [52, 53]. To illustrate this, according to several authors [54, 55] consider a generic SOA consisting of four layers with different functionalities as follows, as shown in Figure 2.4:

- **`Sensing layer`**: It is provided by hardware objects such as actuators, sensors, and RFID tags used to sense the status or change of the environment.
- **`Network layer`**: is equipped with a variety of infrastructures to support both wired and wireless connections of things from the "Sensing Layer" to "Service Layer."
- **`Service layer`**: provides various services to support IoT users and applications. Heterogeneous devices are interoperable through this layer, enabling useful services such as information search engines and communication, data storage, data exchange and management, and the ontology database.
- **`Interface layer`**: includes various communication approaches to support users and applications that facilitate interaction with objects and provide data information of interest in an understandable way.

In a service-oriented architecture, complex systems are often divided into subsystems that take advantage of loosely coupled and reusable features (modular disassembly features), so the overall system is easily maintained by servicing individual components [56]. Therefore, this ensures that other services or components of the system can still keep

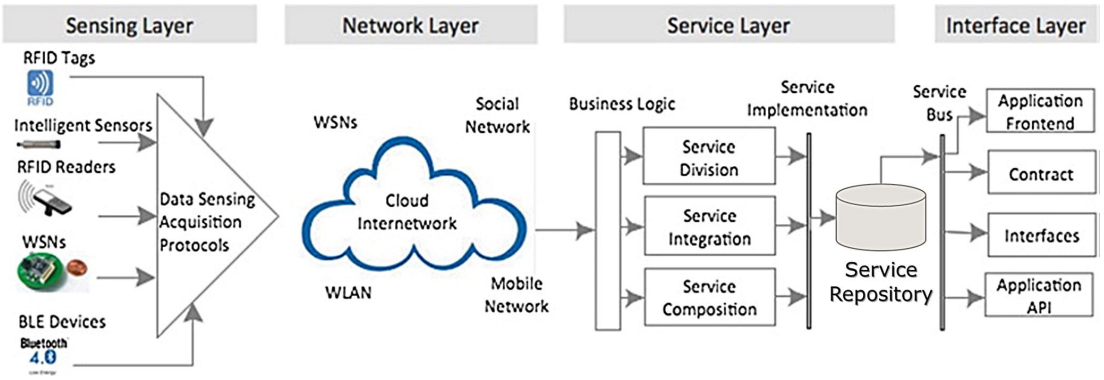

Figure 2.4: *Service-oriented Architecture for IoT [54].*

properly working in case of a failure. This is one of the significant advantages for designing an IoT architecture where reliability is the most important criterion.

The SOA middleware is designed to bridge the gap between the high-level requirements of various applications and the hardware limitations of WSNs [57]. By bringing these advantages to IoT, SOA has the potential to increase the degree of interoperability and scalability between objects in IoT. Furthermore, all services are abstracted into common propositions from the user's perspective, which eliminates additional complexity for the user when dealing with different layers and protocols [58]. In addition, the capability to build diverse and complex services by assembling different functions of the system (i.e., modular composability) through service compositions that are appropriate to the heterogeneous nature of IoT, where the accomplishment of each task requires a series of service requests to all the different entities distributed across multiple locations [59].

2.4.2 API-oriented architecture

The Simple Object Access Protocol (SOAP) and Remote Method Invocation (RMI) approaches are conventional approaches to developing service-oriented solutions and are used to describe, discover, and invoke services. However, considering the complexity and overhead of these techniques, Web API (Application Programming Interface)- and Representational State Transfer (REST)-based methods have been introduced as promising alternatives. Demanding resources range from network bandwidth to compute and storage capacity and are triggered by the request-response data transformation that periodically occurs during service calls. Lightweight data exchange formats such as JSON (JavaScript Object Notation) can reduce the mentioned overhead, mainly for smart devices and sensors with constrained resources, by providing a replacement for

large XML (Extensible Markup Language) files describing services. As a result, the communication channel can be leveraged, and devices' performance can be processed more efficiently.

Similarly, building APIs for IoT applications allows service providers to gain more consumers while focusing on the functionality of their products instead of the presentation. More importantly, the security features of modern Web APIs such as OAuth facilitate multi-tenancy, APIs that effectively support the service exposition and commercialization of an organization. Moreover, providing more efficient tools for service monitoring and pricing than previous service-oriented approaches [60].

One approach enables the efficient discovery of IoT devices and their exposed services, taking into account humans as the main actors [61]. This new approach facilitates communication between the entities involved by creating a ubiquitous environment of IoT elements described by standard human-readable files that can be easily located and invoked by promoted IoT services through standard RESTful Web APIs

Similarly, Bo Cheng et al. [62] proposes a lightweight IoT service mashup middleware based on a REST-like architecture for IoT applications and designs a unified framework for sensor device access and dynamic protocol stack management, Proposes a distributed publish/subscribe based message distribution service and a situational mashup approach for IoT services that can be easily integrated to create a new composite and situational applications, and also applies the REST principles to define an extensible interface for creating comprehensive and situational mashup applications. With the proposed service mashup middleware, the end-user can easily integrate applications and services.

2.4.3 Microservices-based architecture

The microservice approach has made a turning-point in software development. Before development, monolithic software systems faced several issues related to maintainability and scalability limitations. This approach leverages service-oriented architecture along with best practices and recent developments in software virtualization to overcome those issues with the idea of dividing the application into a smaller set of connected services [63]. Each service typically implements one or more features or functionalities. For example, data management service, monitoring management service, etc. Furthermore, message communication between services is handled by a number of lightweight mechanisms such as HTTP, API, and REST [49]. The essence in the IoT environment is decentralized, so

services also need to be deployed in a distributed way to support IoT heterogeneous devices.

On the other hand, applications are divided into much smaller, loosely coupled services usually performed asynchronous ways. To handle a large number of remote access services, load balancing, and monitoring capabilities, this architecture enables messages that may communicate to work together by providing an intermediary component called **`API Gateway`** [64]. In IoT, microservice architecture model can obtain some advantages compared to other methods such as *scalability*, *development*, *changeability*, *integration*, *communication*, *maintenance*, *upgrades* [65]. These advantages provides replicating microservices across nodes depending on demand (**`scalability`**), supports different programming languages in one system (**`development`**). The flexible change on services when errors occur or change services without affecting other services (**`changeability`**). Distributed services can be integrated with each other from the nodes of other systems with services placement mechanisms (**`integration`**) and support many various communication interfaces between services (**`communication`**). The maintainability of the microservices-based approach through updating the smaller source code in services is also easier than the monolithic method (**`maintenance`**). The ability to upgrade and the agility to integrate or change new services without impacting the entire system is also a significant advantage of this approach (**`upgrades`**).

Based on the characteristics of each approach, the design and development of an IoT architecture depend on the requirements and analysis of the context in the IoT applications. Although, a service of the SOA architecture can be a large one and can be implemented in various functions. On the other hand, microservices are small, and each of them performs to one service only. Communications between SOA services typically take place using bus message protocols such as MQTT, XMPP, CoAP, HTTP, AMQP [66]. Alternatively, the microservices-based architecture uses APIs for this. In terms of data processing, services are shared in the SOA, which causes data dependencies. This contrasts with the microservices architecture, where data processing services are independent and decomposed. For services to associate, the SOA architecture uses XML and SOAP protocols. Differently, API-based architecture uses REST and JSON. This means that an API-based architecture is suitable for service interoperability both internally and externally, while an SOA-based architecture is for internal only. Moreover, to build an efficient IoT architecture, APIs can be implemented in both SOA-based and

microservices-based architectures. These approaches are motivations for researchers to develop a comprehensive architecture. One of the most effective IoT architecture platforms is the cloud computing paradigm [67].

2.5 Cloud computing paradigm for IoT

Cloud computing has been introduced in the definition provided by the National Institute of Standard and Technologies (NIST) [5]: "*Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction*". The IoT represents a diverse set of technologies that enable ubiquitous and pervasive computing scenarios by deploying widely distributed devices with limited storage and processing capabilities. On the other hand, cloud computing has unlimited capabilities due to the use of virtualization technologies and almost overcome the problems of the IoT. Therefore, the integration of the two IoT and Cloud technologies can bring many advantages such as storage, computation, processing, and communication [68]. However, we have realized that the complexity of cloud computing faces several challenges for each application (especially time-sensitive applications) currently being investigated by the research community. Several typical challenges depended on the application scenarios as follows:

- **`Heterogeneity`**: One of the main challenges in Cloud-based IoT is related to the large heterogeneity of devices, operating systems, platforms, and services that are available and potentially deployed for new or enhanced applications. Typically, IoT services and applications are designed as independent vertical solutions whereby all system components are tightly coupled to the specific application context. This challenge includes several aspects, with solutions being studied with a focus on unifying platforms and middleware to deal with data diversity [69].

- **`Performance`**: In many cases, Cloud-based IoT applications implement specific performance and QoS requirements at various levels (i.e., communication, compute, and storage aspects), and in certain particular scenarios, requirements may not be easily achieved. Indeed, in several scenarios (e.g., when mobility is required), the provision of data and services must be highly responsive [70]. Therefore, timeliness

can be severely affected by unpredictability issues for real-time applications.

- **`Reliability`**: Deployment of Cloud-based for mission-critical applications typically leads to reliability issues, e.g., in the context of intelligent transportation, as vehicles are frequently on the road and in-vehicle networking and communication are often sporadic or unreliable. When applications are deployed in resource-constrained environments, they face a number of challenges related to devices failure or devices that are not always accessible [70].

- **`Large scale`**: Cloud-IoT platforms facilitate the design of next-generation applications targeted at the integration and analysis of information from real devices [71, 72]. However, the distribution of IoT devices makes monitoring tasks more difficult because they have to deal with latency dynamics and connectivity problems.

- **`Big data`**: By 2020, an estimated 50 billion devices will be connected to the Internet. Especially attention must be focused on the transfer, storage, access, and processing of the huge amounts of data they will generate. Managing this data is a critical challenge, as the overall application performance depends heavily on the properties of the data management service [73].

- **`Security and privacy`**: As sensitive IoT applications migrate to the Cloud, issues such as lack of trust in the service provider, knowledge of service level agreements (SLAs), and knowledge of the physical location of data raise concerns. The distributed system is vulnerable to several possible attacks (e.g., Session Riding, SQL Injection, Cross Site Scripting, and Side-Channel). As a result, new challenges require special attention [74].

Cloud computing and the Internet of Things (IoT) are two very different technologies, and both of them are already part of our lives. It is expected that their adoption and utilization will become increasingly widespread, making them important components of the future Internet. A novel paradigm in which Cloud and IoT are merging is considered potentially pervasive and enables many application scenarios. However, the unpredictable growth of IoT devices poses several challenges related to the management and processing of these devices to ensure low latency, flexible storage systems, diverse connectivity, and real-time data analysis. Therefore, it is essential to develop and deploy a paradigm that is close to the edge of the network to meet these challenges. In the

next section, we discuss some paradigms that can effectively overcome issues in the IoT environment.

2.6 Related paradigms and technologies

The term "pay-as-you-go"in the Cloud computing model is the origination of computing technology to manage private data centers (DCs) for enterprise end-users [8]. With the limitless capacities of computing, storage, and networking resources, cloud computing has opened a new turning-point. There are several computation service providers such as Amazon, IBM, Google, Microsoft, etc., are taking advantage of this computing paradigm as a benefit. They have deployed cloud-based services such as Infrastructure as a Service (IaaS), Platform as a Service (Paas) as Software as a Service (SaaS), etc., to manage huge enterprises and end-users who interact with related issues at the same time.

However, data centers are geographically centralized and located far from the edge of end users/devices. As a result, the responses of cloud datacenters often dissatisfied latency and real-time computation applications and services requests and cause large round-trip delays, network congestion, quality of service reduction. Therefore, there are some concepts similar to resolve these issues that have been existing besides cloud computing. Figure 2.5 shows the ecosystem of computing in the context of IoT.

2.6.1 Edge computing

This new concept has been proposed as a novel paradigm [76] that will push the frontier of computing applications, data, and services away from centralized node to the edge devices of the network. Data generated from IoT devices is sent to edge devices. It can be processed directly on these devices without being sent to the centralized Cloud. Figure 2.6 present the edge computing general architecture. Edge computing's essence is to bring the computing facilities closer to the source of the data. It enables data processing at the edge network [77]. Edge computing's network system is a set of edge nodes (e.g., base stations, switches, border routers, set-top boxes, etc.), end devices (e.g., smart objects, mobile phones, vehicles, etc.), edge servers, etc. These components are acted as localized computing paradigm with the capabilities for supporting edge computation. Consequently, Edge computing has resolved the faster response problems from computational services and application requests. However, cloud-based services such as IaaS, PaaS, SaaS unsuitable with Edge computing when deployed services

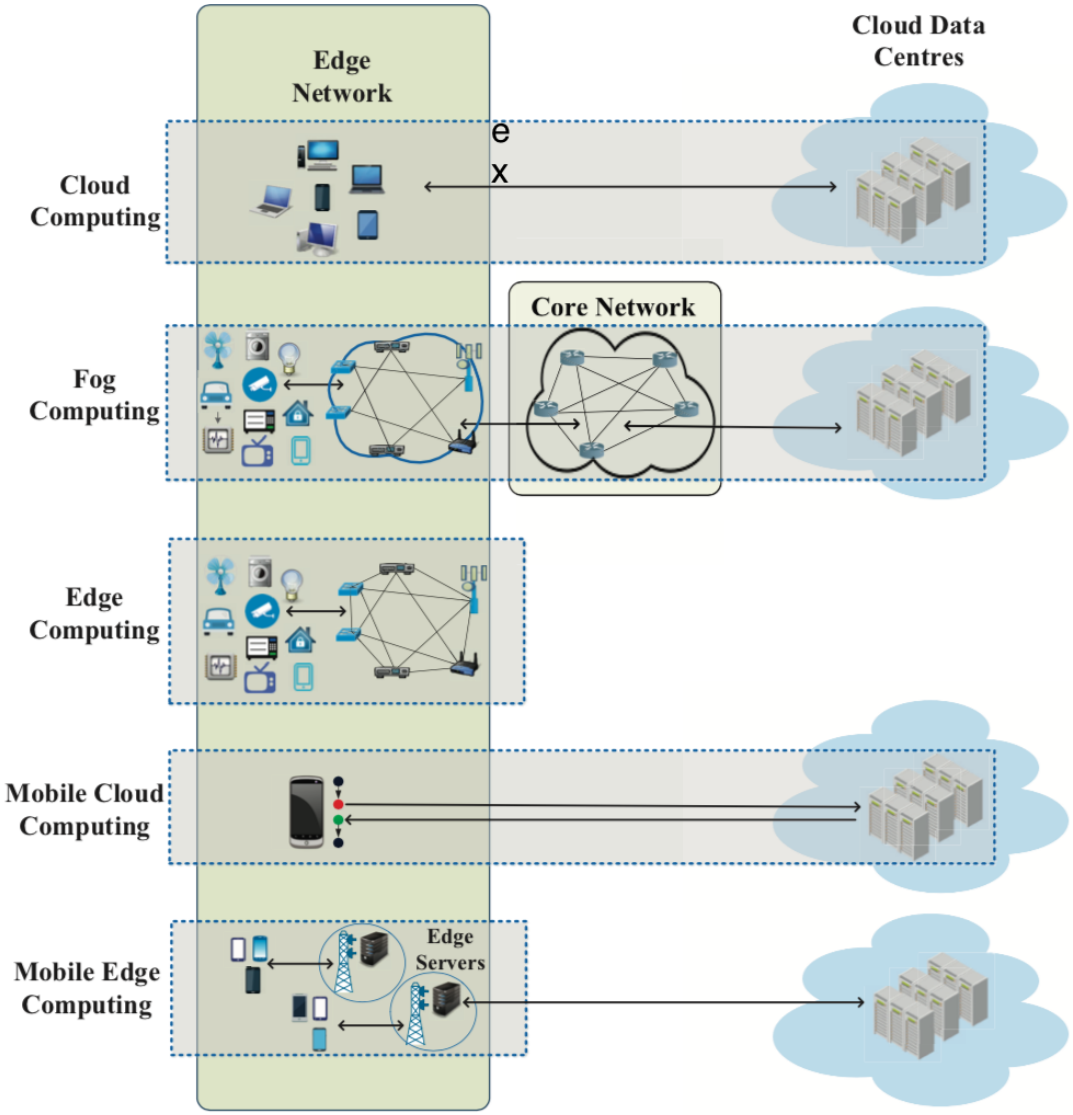

Figure 2.5: *The ecosystem of computing [75].*

autonomously and concentrate more towards the end devices side [78].

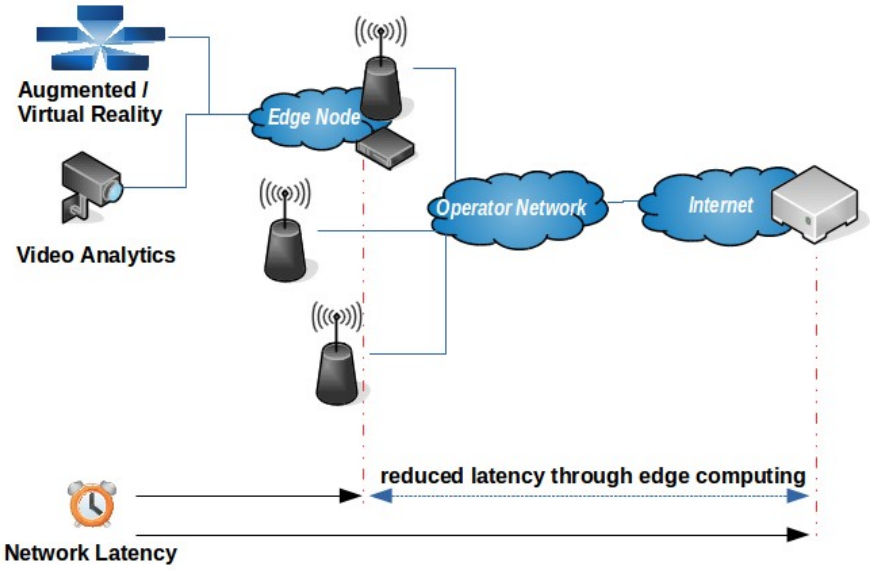

Figure 2.6: *Edge computing architecture [79].*

2.6.2 Mobile cloud computing (MCC)

MCC is originally used based on the concept of the combination between cloud computing and mobile computing, where it relies on wireless networks to provide rich computational resources to mobile users. The objective of MCC is to enable rich mobile applications to be executed across multiple mobile devices, with a rich user experience and to provide business opportunities for network operators, as well as cloud computing providers [80], as shown in Figure 2.7. MCC uses four kinds of Cloud-based resources, named distant immobile clouds, proximate immobile computing entities, proximity mobile computing entities, and hybrid (a combination of the other three models)[81, 82]. Besides, most smart mobile devices are often designed with the computational resource, storage, and energy constraints. Therefore, MCC processes and stores data outside of mobile devices by providing necessary computational resources to support remote execution of rich mobile applications at the Cloud, such as offloaded mobile applications in closer proximity of end users [83].

However, in the context of IoT, devices are very diverse and heterogeneous with many different network access technologies. The proposed MCC to manage this huge set of devices is not feasible since it is designed to deal with mobile phones.

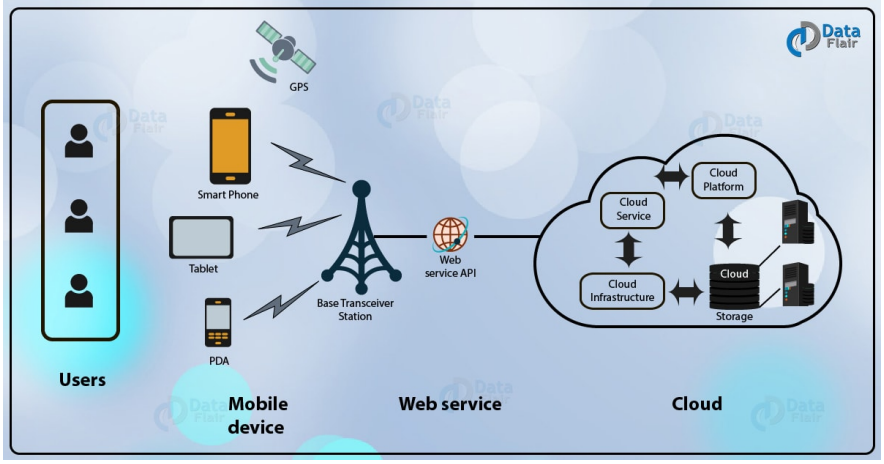

Figure 2.7: *Mobile Cloud Computing architecture [84].*

2.6.3 Mobile edge computing (MEC)

MEC is considered as the potential extension of Cloud and Edge computing. It provides an IT service environment and cloud computing capabilities at the edge of cellular network [85]. The fundamental idea of MEC is to perform processing tasks and run applications closer to the cellular customer. Thus network congestion is significantly reduced, and applications perform better by combining edge servers and cellular base stations. Due to being designed and implemented at cellular base stations, MEC technology enables flexible and rapid deployment of new services and applications for customers and enhances network efficiency. The architecture of MEC illustrates as Figure 2.8. During a recent period, MEC also allows cellular providers to open their radio access network to distribute contents and develop applications so that it can support 5G communication [86, 87].

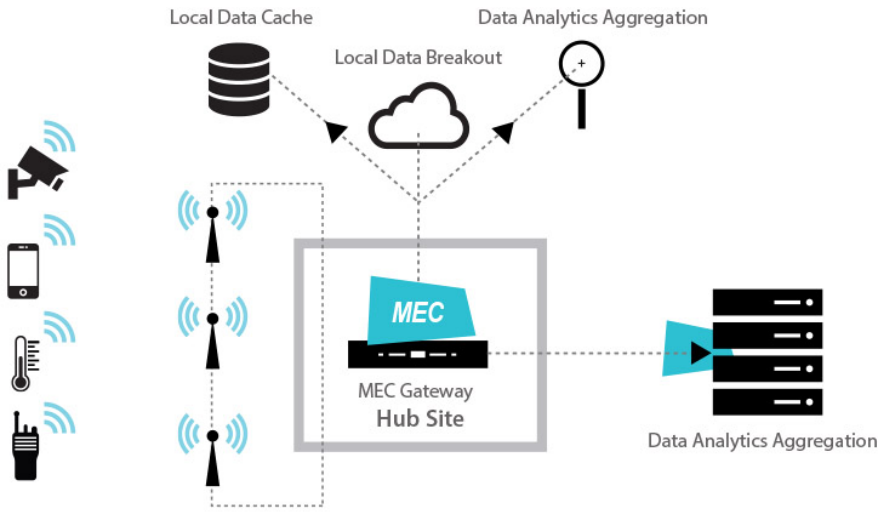

Figure 2.8: *Mobile Edge Computing architecture [88].*

Similar to MCC, MEC was also introduced to address access mechanisms for mobile networks (cellular) [89]. While the devices are increasingly diverse with different access technologies such as Bluetooth, Wi-Fi, mobile networks. Consequently, there is necessary to propose an effective architecture that can overcome these issues.

2.6.4 Cloudlet

Cloudlet [90] is a lightweight cloud or a data center in a box is placed at the edge network. Cloudlet, along with cloud computing and mobile devices, focuses more on providing services and deploying application platforms to latency-sensitive, bandwidth-limited for rich mobile applications in proximity. Figure 2.9 illustrates the cloudlet architecture: a cloudlet is a small-scale server with specific resources assigned to it. It can be accessed with mobile devices within a particular coverage area. The cloudlet architecture is presented in three layers [91], namely the component layer, the node layer, and the cloudlet layer. The component layer can be started and stopped by an Execution Environment (EE) and deployed services by providing interfaces to higher layers. The node layer manages one or multiple Execution Environments that run on top of an operating system (OS) or Node Agent (NA). A set of nodes clustered from the cloudlet layer managed by Cloudlet Agent (CA) to migrate components between cloudlets in order to satisfy low latency [92].

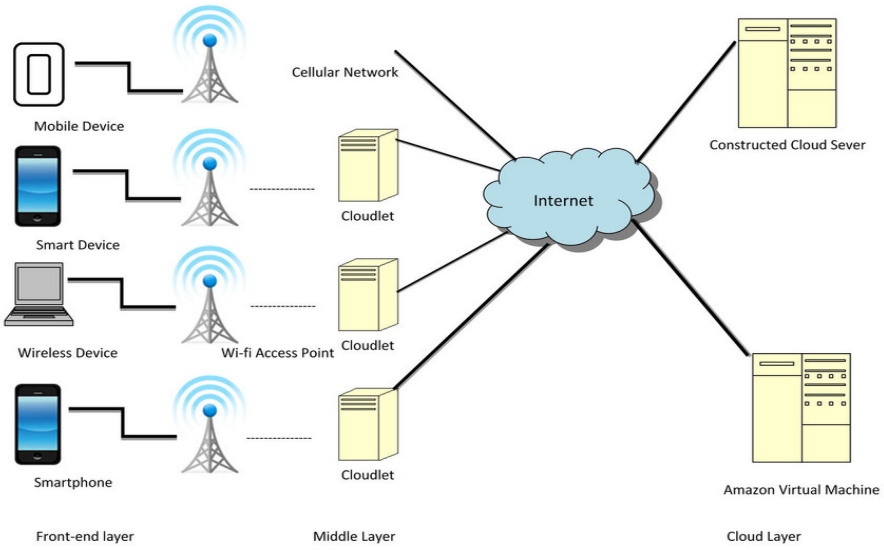

Figure 2.9: *Cloulet architecture [93].*

Although, cloudlet has many advantages in terms of support for physical proximity devices, the power consumed optimization and reduced computation time. However,

providing a computational environment for heterogeneous devices to ensure interoperability is still limited. Especially for non-IP supported devices as BLE and ZigBee, it may not have enough features to support a wider range of end devices [89].

The dramatically increasing number of heterogeneous devices and diversified data traffic in the IoT technology is posing a significant burden on computing paradigms and uncontrollable service latency. Solutions like cloud computing can provide unlimited possibilities for data storage and processing, but they can not meet the sensitive-latency and context-aware requirements of IoT applications. Some computing paradigms are proposed to address these problems by moving computation from the cloud center to distributed edge computing, such as edge computing, mobile cloud computing, mobile edge computing, and cloudlet. However, these paradigms still face some challenges when dealing with heterogeneity, scalability, interoperability, data processing, and vertical and horizontal communication issues [89]. Therefore, a paradigm that distributed resources and services of computing, storage, and networking need to be deployed everywhere. It supports edge devices and can provide an intermediary layer that can complement cloud services. In this thesis, we argue that the mentioned issues make the appropriate `Fog computing` paradigm for the number of critical IoT applications.

2.7 Fog computing paradigm

The term *Fog computing* was proposed in 2012 by researchers from Cisco Systems Bonomi et al.[8]. It is a highly virtualized platform that provides compute, storage, and networking services between IoT devices and traditional cloud computing data centers, typically but not exclusively located at the edge of network. The IoT is bringing more than an explosive proliferation of endpoints [9]. It requires latency-aware computation for real-time application processing. In IoT environments, interconnected things produce a huge amount of data. The data generated by IoT devices are typically processed in a cloud infrastructure due to the on-demand services and scalability characteristics of the cloud computing paradigm. However, the integration of IoT with Cloud computing faces many challenges. One of those challenges is the growth of real-time and latency-sensitive applications and network bandwidth limitations. This leads to unnecessary communication burdens the core network and the data center in the Cloud. Therefore, a new paradigm that seems to be promisingly named in the literature is fog computing. It is designed to support IoT applications characterized by latency limitations and

requirements for mobility and geo-distribution [8, 94, 95]. This section will introduce the key motivations and issues behind the importance of the Fog computing paradigm.

2.7.1 Overview of Fog computing

Fog computing extends the Cloud computing paradigm to the edge of the network by enabling a new range of applications and services directly deployed to close IoT devices. In general, Fog computing is an intermediate architecture that provides compute, storage, networking services between end-devices and traditional Cloud Computing Data Centers [8].

Provisioning resources at the edge of networks (closer to end-devices) brings several benefits such as low latency and enables the provisioning of new applications such as mobile data offloading. This section discusses and contrasts the key concepts that enable application provisioning at the edge. The next subsections introduce the need for fog computing. The last subsection illustrated and discusses the similarities and differences between these concepts [96].

2.7.2 Need of Fog computing

Cloud computing has made many conveniences for companies by providing their customers with a pool of computing services. Cloud computing technology has been driven by the economic model that customers will charge according to the time they spend or called with the terminology "*pay-as-you-go*". For example, users can rent 2-3 hours to run a temporary application on Amazon EC2 [97]. End-users and enterprises can demand or customize resources from Cloud computing based on their consumptions such as storage resources, computation, and networking components. However, cloud computing is not an optimal solution. Because there are billions of *Things* or physical objects connected to the Internet, and there is a lot of application applied by the industries, factories, the government, or the public services, etc. For instance, smart transport system applications, smart homes, smart cities, etc. Therefore, enormous amounts of data are being generated by those connected devices and sent throughout the network to the Internet. Especially in the Internet of Things environment, there are still problems unresolved because IoT-based services and applications are becoming popular rapidly and usually require low latency, mobility support, geo-distribution, and location-awareness. According to [98], those are reasons that the Cloud computing paradigm can hardly satisfy their requirements of latency, mobility support, and location

awareness.

Fog computing enable directly processing and computing services at the edge of the network. It is really the computing paradigm to support and address the above problems. Similar to Cloud, fog computing also provides compute, storage, network, and application services to end-users. Moreover, this advanced infrastructure supports to deploy on heterogeneous devices which act as Fog nodes including end-user devices, set-top box, access points, edge routers, and switches, or on constrained devices such as Raspberry, Arduino, etc. Figure 2.10 shows a basic model of Fog computing that plays a role as immediate layer (`fog layer`) between `things layer` and `cloud layer`.

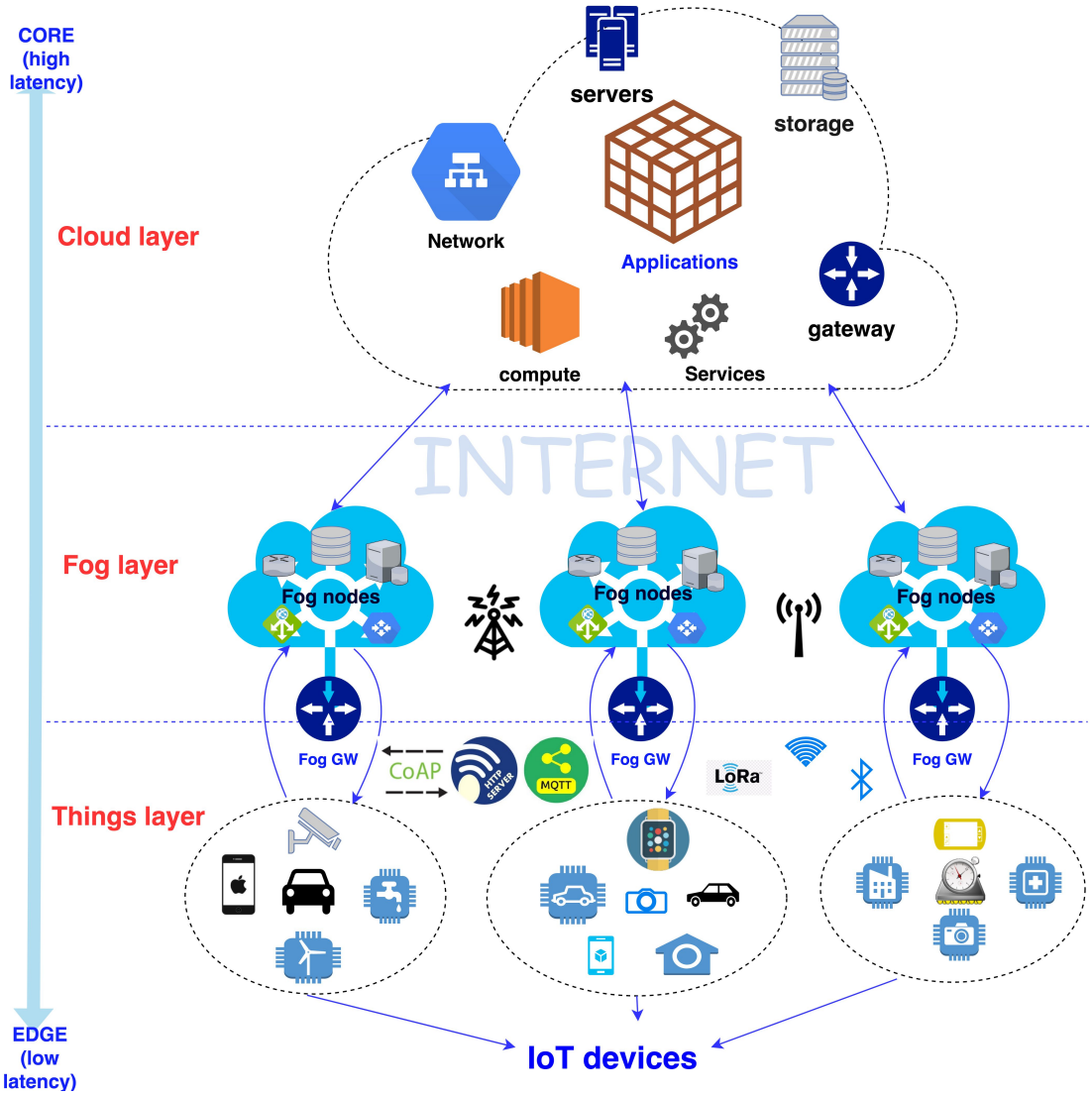

Figure 2.10: *A model of Fog computing.*

The main difference of Fog computing compared with Cloud computing by some characteristics as proximity to the end-user, the density of geographical distribution,

and support of mobility. Especially, Fog Computing supports applications in the IoT environment that demand real-time or latency-sensitive. Doing so will reduce service latency and improve Quality of Services (QoS), resulting in enhanced user experience. Cloud computing could help by providing on-demand and scalable storage and processing services which can scale to requirements of IoT applications. However, for applications requiring emergency response, health monitoring, and latency-sensitive, data transmission from IoT devices to the cloud and back to applications affected by delay is unacceptable [99].

Fog computing was introduced to use computing resources close IoT devices for local storage and data pre-processing to address these issues. This may reduce network congestion, as well as accelerating analysis and decision-making. Fog computing can take advantage of the flexibility of providing resources at the edge of the network by using cloud resources and coordinating geographically distributed edge devices.

2.7.3 Definition of fog computing

Fog Computing is a distributed computing paradigm where processing is performed seamlessly with the cloud infrastructure at the edge of the network. It provides a computing mechanism for IoT environments or other latency-sensitive application scenarios. Transmitting all data from all connected devices for processing in the Cloud will require massive amounts of bandwidth and storage. These devices are not connected to the controller via IP but via some other industrial IoT protocols [26]. As a result, a translation process is also required to handle or store information from IoT devices. Many researchers have defined fog computing in different approaches. The following are some examples:

- **`Bonomi et al.`** [8]: “Fog computing is a highly virtualized platform that provides compute, storage, and networking services be- tween IoT devices and traditional cloud computing data centers, typically, but not exclusively located at the edge of network”.

- **`Ahmed Banafa-IBM`** [100]: “The term Fog computing or Edge Computing means that rather than hosting and working from a centralized cloud, Fog systems operate on network ends. It is a term for placing some processes and resources at the edge of the Cloud, instead of establishing channels for cloud storage and utilization”.

- **`Vaquero et al.`** [101]: "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. Users leasing part of their devices to host these services get incentives for doing so".

- **`Naha et al.`** [102]: "Fog computing is a distributed computing platform where most of the processing will be done by the virtualized and non-virtualized end or edge devices. It is also associated with the Cloud for non-latency-aware processing and long-term storage of useful data by residing in between users and the cloud".

For the first definition of fog computing, Bonomi et al. [8] proposed the computing paradigm as a highly virtualized platform. But, some IoT devices, such as smartphones, are not virtualized, although they could be part of the Fog infrastructure since part of the processing could still be performed. The definition given by IBM [100] represents Edge and Fog computing as the same computing paradigm. It addresses an important issue in cloud computing, reducing bandwidth requirements by not sending every information bit through cloud channels but rather aggregating them at specific access points. Vaquero and Rodero-Merino [101] is a controversial issue, and a definition that can clearly distinguish between fog computing and other related computing paradigms is still needed. In Naha's definition [102] considered all devices with computing and storage capacity as fog devices and also defined the role of the Cloud in the fog computing environment more precisely. In brief, Table 2.1 summarizes the definitions of fog computing provided by several research works.

According to the definitions above, we define Fog computing as follows:

- *Fog Computing is a platform that allows a huge number of heterogeneous devices which is characterized by a wide diversity of network access technologies, including Wi-Fi, Cellular, Lo-Ra, Zigbee, or Bluetooth (included the edge network devices) is connected in a decentralized environment and to be intelligently ability for data computing and processing. Tasks and services will operate in an isolated environment to solve several issues such as latency requirement, geo-distributed, support mobility, flexible storage, computation, and data processing. Therefore,*

fog system distributes these services and takes to a huge scale of end-user in IoT environments.

In our definition, we address a definition for fog computing that covers a large number of different types of devices and the capacity of data processing to support the interoperability in IoT environments.

Table 2.1: *Summary of Fog computing definitions*

Defined by	Characteristics
Bonomi et el.[8]	Highly virtualized Reside between IoT devices and cloud Not exclusively located at the edge
Ahmed Banafa[100]	Defined Fog and Edge computing as similar Not depends on centralized cloud Resides at network ends Place some resource and at the edge of the cloud
Vaquero et el. [101]	Heterogeneous, ubiquitous and decentralised devices communication Storage and processing done without third party invention Run in a sandboxed environment Leasing part of users devices and provide incentive
Naha et al.[102]	Virtualization and non-virtualization characteristics Association with the cloud for non-latency-aware processing and storage. Any edge device with available processing power and storage capability can be act as a Fog device. Always resides between end users and cloud.

2.7.4 Fog computing architecture for IoT

To illustrate several literatures which have been published, we have separated to formally define fog computing and the related works in Table 2.2. The selected criteria for the classification are the following:

- **Fog computing architecture (FAC**: This part will provides a big picture which related to the fog computing paradigm including architectures, concepts, and applications. These features will have to interpret issues such as *low latency and location-awareness*; *mobility support*; *high geographical distribution*; *large-scale sensors and actuators network*; *heterogeneous devices*

- **Internet of Things (IoT)**: In this section, we only focus on the research works which need to consider handling of the emerging of IoT heterogeneous devices,

which use many various communication technologies when these IoT devices connected to the Internet.

- **`Programming model (PM)`**: A distributed architecture as Fog computing is the optimal solution for IoT applications, one of the criteria which response real-time applications is low latency, and capability of resource utilization at the edge of the network. Furthermore, a suitable data management model for heterogeneous devices is needed to provide interoperability and scalability in the IoT environment.

- **`Resource provisioning management (RPM)`**: Fog computing is the basic paradigm of the IoT, where a set of heterogeneous devices can cooperate with fog nodes or other IoT devices to provide resources. Fog nodes take responsibility for executing IoT services close to the data sources or gateways. Thus, the **`RPM`** can help to decrease delays as well as better utilization of already available computation, storage, and networking resource in fog computing.

- **`Dynamic application topology (DAT)`**: DAT is an application topology where distributed services can be deployed and interconnected on any selected execution environment [103]. This criterion needs to take dynamic restructuring of the system topology into account and adapt to the underlying topology during runtime.

- **`Security and Privacy (S&P)`** are always a key challenge in any systems, and fog computing is not also an exception, especially in IoT environment with a multitude of heterogeneous devices and a various set of protocol communications. One of the main s&p unsolved problems in fog computing is authentication capability at different levels. For example, attackers can fake a gateway or fog nodes to compromise with users to achieve their purpose (ex: man-in-the-middle attack). Thus, secure communications need to be granted permissions at each level to guarantee data privacy at the edge of the network.

- **`Reliability (REL)`**: In the context of real-time IoT applications, we can take full advantage of fog computing to achieve correct results by minimizing the overall delay and with high reliability. In fog landscapes, using the edge devices can handle reliable communication and reliable computation near the end devices.

- **`Energy efficiency (EEF)`**: Energy management has an important play in IoT applications such as microgrids, homes, and buildings by taking full advantage of fog

computing paradigm, which provides the flexibility, interoperability, connectivity, data privacy, and real-time features required for energy efficiency management.

- **`Simulation and Mathematics (SIM&MATH)`** are terminologies which related to the experiments of the paper authors.

The main challenge of this thesis is to design a platform that enables the management of the available resources in a fog environment, as described above. Moreover, dealing with a wide of network access technologies, including Wi-Fi, Cellular, Lo-Ra, Zigbee, or Bluetooth, is still very challenging. In the next chapter, we focus on more details of the contributions which can resolve the limited challenges from cloud computing.

2.8 Conclusions

The growth of the *Internet of Things* (IoT) devices has created a large network with sensors and actuators which provide a latency-sensitive response to end-users. Therefore, traditional computing as cloud computing needs many changes. Moreover, these large and heterogeneous numbers of geographically distributed IoT devices required several real-time responses such as location-aware services, mobility support, reliability, low latency, etc. So the Cloud cannot provide the requirements. Fog computing is really a high-potential computing paradigm that is emerging rapidly to satisfy the fast development of IoT, CPS, and Mobile Internet. With the broadly computational capability at the edge of the network, fog computing will push more applications and services from the Cloud to the network edge. This is a great reason to decrease the data transfer time and the amount of network transmission and effective response to the demands of real-time or latency-sensitive applications. This chapter focuses on the concept of fog computing, the services architectures, applications, and its characteristics. We have classified the existing literature to find the latest research works and the different scenarios of fog computing.

Table 2.2: *Classification of the related work*

Works	FAC	IoT	PM	RPM	DAT	S&P	REL	EEF	SIM	MATH
Bonomi et al. [8, 9, 104]	✓	✓	x	x	x	x	x	x	x	x
Vaquero et al. [101]	✓	✓	x	x	x	x	x	x	✓	x
Luan et al. [105]	✓	x	x	x	x	x	x	x	✓	x
Yi et al. [106, 107]	✓	x	x	x	x	x	x	x	✓	x
Madsen et al. [98]	✓	✓	x	x	x	x	✓	x	x	x
Stojmenovic et al. [108]	✓	x	x	x	x	x	✓	x	x	x
Dastjerdi et al. [109]	✓	✓	x	x	x	x	✓	✓	✓	x
Sarkar et al. [92]	✓	✓	x	x	x	x	x	x	x	✓
Hong et al. [99]	x	✓	✓	✓	x	x	x	x	✓	x
Habibi et al. [110]	✓	✓	✓	✓	x	✓	✓	x	x	x
Faruque et al. [111]	✓	✓	x	x	x	x	x	✓	x	✓
Saures et al. [112]	✓	✓	✓	✓	x	x	x	x	✓	x
Skarlat et al. [113]	✓	✓	x	✓	x	x	x	x	✓	✓
Aazam et al. [17, 114]	✓	✓	x	✓	x	x	x	x	✓	x
Singh et al. [115]	x	x	x	✓	x	x	x	x	x	x
Sarkar et al. [92]	✓	✓	x	x	x	x	x	✓	x	✓
Zhan et al. [116]	x	x	x	✓	x	x	x	x	x	x
Antonescu et al. [103]	x	x	x	x	✓	x	x	x	x	x
Okafor et al. [117]	✓	✓	x	x	✓	x	x	x	✓	x
Shanhe Yi. [118]	✓	x	x	x	x	✓	x	x	x	x
Yigitoglu et al. [119]	✓	✓	✓	✓	x	x	x	x	x	x
Rahman et al. [120]	✓	✓	✓	✓	x	x	x	x	✓	✓
Brito et al. [121]	✓	✓	✓	✓	x	x	x	x	x	x
Fog Computing Platform	✓	✓	✓	✓	✓	✓	✓	✓	x	✓

CHAPTER 3

Fog Services Provider for the Internet of Things

Stay hungry, stay foolish.

Steve Jobs

Abstract

In this chapter, we will study a multi-technology service architecture based on Fog computing paradigm for IoT devices to perceive and manage flexible and heterogeneous multi-network access technologies in ubiquitous environments. The main objective is to propose a microservice-based architecture named Fog Services Provider (FSP) to deal with the heterogeneity and interoperability of IoT devices [21, 22].

[21] Hoan Le, Nadjib Achir, and Khaled Boussetta. "Fog Computing Architecture with Heterogeneous Internet of Things Technologies". In: *2019 10th International Conference on Networks of the Future (NoF)*. 2019 10th International Conference on Networks of the Future (NoF). Oct. 2019, pp. 130–133. DOI: 10.1109/NoF47743.2019.9014960

[22] H. Le, N. Achir, and K. Boussetta. "Fog Services Provider Architecture for IoT". in: *2020 11th International Conference on Network of the Future (NoF)*. 2020 11th International Conference on Network of the Future (NoF). Oct. 2020, pp. 8–15. DOI: 10.1109/NoF50125.2020.9249177

Chapter content

3.1 Introduction

The prominence of Internet of Things (IoT) devices is characterized by a wide diversity of network access technologies, including Wi-Fi, Cellular, Lo-Ra, ZigBee, or Bluetooth. Dealing with such heterogeneity is still very challenging. Current cloud computing solutions can sustain a considerable amount of diverse data generated by geographically spread IoT devices. However, this task is quite challenging for time-sensitive services such as healthcare, augmented reality, a cognitive system, and gaming [109, 122]. Moreover, IoT heterogeneous devices are always managed by different application systems [123, 124], and there are no unified storage and management solutions for information on IoT devices. This diversity raises several interoperable issues. Besides, data packets created from *things* can have various syntax, types, formats, and the meaning of this data also varies from device to device. Consequently, a high-level fog computing architecture is required to cope with heterogeneous IoT devices generating a large and diverse type of data for sensitive-latency problems.

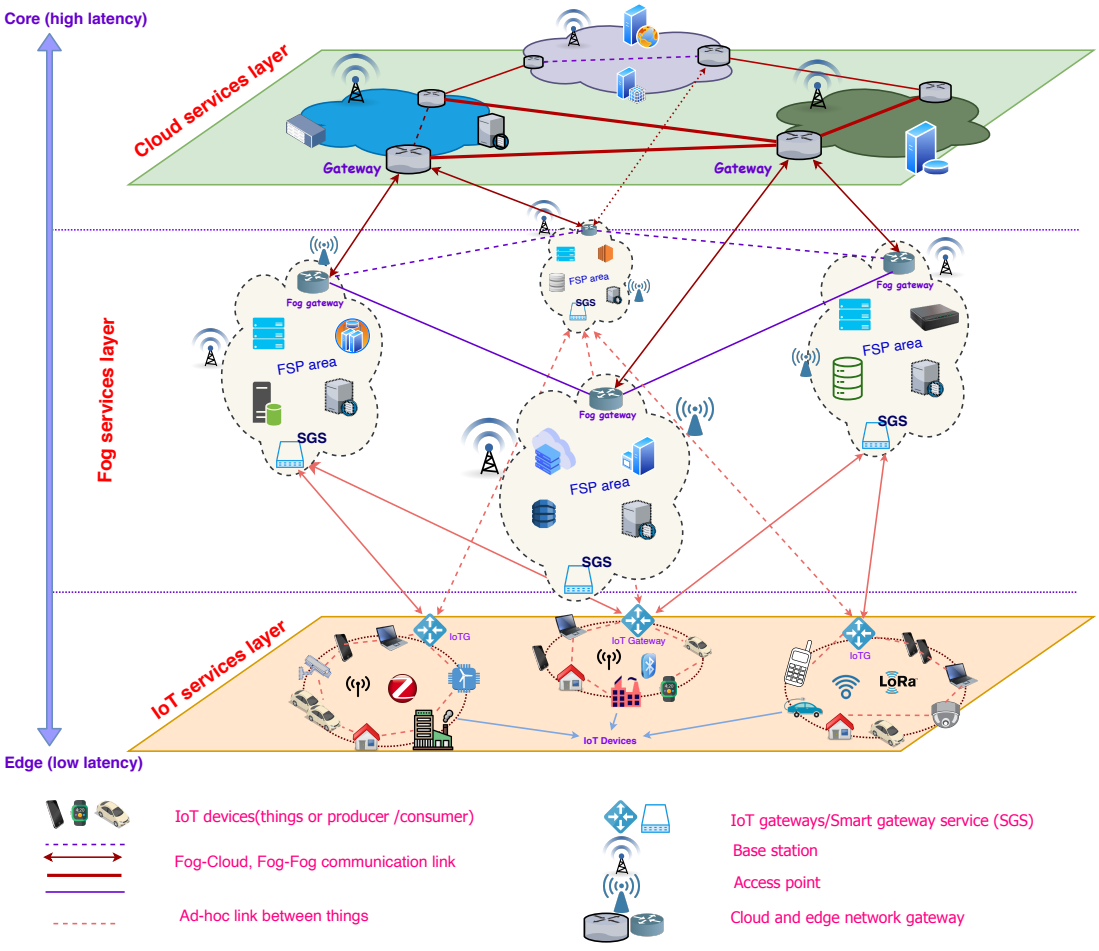

Figure 3.1: *The internet of things and Fog computing*

Recently, several works have studied architectures for fog computing in IoT environments. Proposed architectures [125, 126] rely on a gateway to integrate an IoT network and use an instance of Information-Centric Networking (ICN) to introduce a fog gateway in the sensors network. Some advanced techniques and services, such as embedded data mining, distributed storage, and notification services, are used at the edge of the network to trigger events that related to data transmission at fog layer [127]. However, these architectures neither mention how to organize the data structure and management on IoT connected devices nor provide an approach to handle IoT multi-network technologies. To achieve semantic interoperability in a heterogeneous IoT environment, [120, 128] introduced a semantic Fog model different functionalities, such as data composing, aggregation, modeling, linking, reasoning to model raw data. They also integrate conversion techniques for processed data to annotate them into Resource Description Framework (RDF [129]) format. Nevertheless, the relationship between data objects and mapping is not mentioned. Therefore, as shown in Figure 3.1, providing services that are deployed closer with the end-users and supporting data processing for the IoT layer from various technologies at the edge of the network are necessary. Fog computing architectures provide and manage more simply computing, storing, and networking services by complementing and extending resources of the Cloud to close the edge of network and endpoints [9]. Generally, it provides the capability for distributed deployment of applications that require for these services across different layers as follows:

1. **`IoT services layer`**: This is the lowest layer or the closest layer to the IoT devices and physical environment. It encompasses all various IoT devices such as sensors, mobile phones, smart vehicles, smartwatches, cameras, etc. Devices that act at IoT services layer usually equip low-processing capability and limited storage. They can use several different types of connectivity technologies such as Bluetooth, ZigBee, Wi-Fi, LoRa, 4G/5G, etc. Data that has been collected from these devices and not processed for use are considered raw data. The raw data or events collecting from IoT devices do not contain any semantic annotation and demand extensive manual effort by using practical applications.

2. **`Fog services layer`**: The intermediate layer is also known as the fog computing layer. This layer is located on the edge of the network. It consists of a distributed

number of fog nodes, including routers, gateways, switches, access points, base stations, micro data centers, fog servers, etc. The fog services layer provides a flexible and easy way to analyze and process raw data at the network's edge instead of the Cloud. Moreover, this layer can model processed data to make semantic contexts and provide interoperability at the data annotation level through concepts, attributes, and the definition of relationships between data objects. These modeled data then are mapped into a triplestore (a triple is a data entity composed of *subject-predicate-object*) or RDF (resource description framework) store by taking advantage of a web ontology language (OWL, a standard of W3C [130])

3. **`Cloud services layer`**: The cloud computing layer is the top layer in this architecture. This layer consists of multiple high-performance servers, data centers, and network systems capable of processing and storing enormous data and providing different applications and services.

In this chapter, we propose a services architecture based on Fog computing paradigm named **`Fog Services Provider (FSP)`**. This architecture perceives and manages heterogeneous IoT devices with various network access technologies, service components, and inter-process communications in ubiquitous environments. As in previous works [121, 131], we assume that heterogeneous devices and services at the fog node level only can interact with each other through the M2M standard. However, unlike those works, we define and propose a microservices-based architecture with both communication mechanisms that enable non-M2M devices to be compatible with M2M devices through a combination of access services and the ability to manage data information using message protocols.

The FSP architecture is designed to support heterogeneity, interoperability, scalability in different applications and domains. It consists of a collection of service layers with diverse communication capabilities:

1. **`Device layer`**: provides heterogeneous device management services and access services to ensure interoperability and scalability at the network level.

2. **`Intermediate layer`**: provides various supported services and communication protocols to ensure interoperability and scalability at the exchange level.

3. **`Application layer`**: provides a wide variety of application services to ensure interoperability and scalability at the application level.

The remainder of the chapter is organized as follows. Section 3.2 introduces the requirements for designing Fog architecture. Section 3.3 presents the design of Fog architecture with a general model of service layers. Our architecture is detailed in the section 3.4. The interaction model in Fog architecture is illustrated in section 3.5. The evaluation and comparison with other architecture are discussed in section 3.7. The last section 3.8 draws a conclusion.

3.2 Requirements for Fog architecture

In this section, the comprehensive approaches of the fog computing architecture are proposed and specified requirements and design decisions. The Fog Services Provider architecture implements an architecture that allows the execution of IoT applications in the Fog environment. The FSP architecture developed in this thesis provides the basic functionalities, including the heterogeneous producers and consumers management mechanism, a unified data model approach, and service placement approach, among other fundamental tasks necessary for the fog landscape to work as intended. These basic functionalities will present in the FSP management services layer (sub section 3.4.4). Moreover, these specified functionalities can be adapted and extended to fit the particular application's need. In addition to these functionalities, the Fog computing architecture also has to meet the following non-functional requirements [102]: (i) scalability, (ii) extensibility, (iii) portability, (iv) interoperability, (v) maintainability. Non-functional requirements are detailed as follows:

- **`Scalability`**: IoT devices are growing very fast every day worldwide, leading to a recent issue of scalability. Consequently, the scaling of devices and services in the Fog environment must be addressed. Scalability provides the capability to handle increasing appropriately (and decreasing) demands on Fog system resources. Some research works [132, 133] proposed to use Cloud computing to support the scalability of IoT applications. However, deploying applications in a Cloud environment is inefficient and impractical because of the nature of continuously moving devices. Moreover, with the tremendous increase of IoT devices, the fog architecture design must also provide horizontal and vertical strategies. In vertical scaling, additional resources are added to a single Fog node. As a result, the Fog node can control more tasks and provide better additional capacity (CPU faster, more memory). Meanwhile, horizontal scaling adds more

Fog nodes to the entire system. Therefore, scalability is an essential requirement for distributed systems such as Fog computing. Specific non-functional requirements for the FSP architecture are: (i) resource provisioning and service placement can be scaled to ensure that CPU utilization is maintained under a certain threshold, (ii) optimize fog resource efficiency to reduce cloud cost, (iii) facilitate a fast service deployment.

- **`Extensibility`**: A Fog architecture can be deployed if components and services can be provided in the proximity of the data source without too much effort. This means that the fog architecture should be designed in such a way that it can be extended by some features in the service-oriented software development principle [134] such as *loose coupling* and *high cohension*. The *loose coupling* feature would imply that components or services might be made as independent as possible from other ones so that changes to components or services do not heavily impact others. The *high cohension* generally indicates that changes in one functionality will require changes in other related functionality if a service provides the functionality that logically belongs together. Besides, the Fog architecture needs to be clearly defined APIs between different components and services. Furthermore, utilizing standards-based communication technologies helps to simplify the extensibility of Fog services

- **`Portability`**: Portability is the ability of a service to be deployed in different system environments, i.e., operating systems, without the necessity to adapt the service manually, and should be built with platform-independent technologies that are preferably fast to deploy and migrate. Moreover, platform-independent communication technologies are necessary to separate data from specific technologies.

- **`Interoperability`**: According to IEEE [135], interoperability is defined as "*the ability of two or more systems or components to exchange information and to use the information that has been exchanged*". Thus, interoperability can be defined in the IoT context as the ability of two or more systems, devices, platforms, or networks to understand and interact with each other [136]. It enables heterogeneous devices or systems to communicate to achieve a common objective. The Fog services are deployed in a distributed environment and close to devices, which allows Fog nodes to quickly respond to applications with minimal latency. However,

what will happen if an increasing number of multiple latency-sensitive application requests are transmitted to the Fog node at concurrency. As a result, the Fog node cannot process such numerous requests because it will cause network congestion. Thus, interoperability and federation among clusters of Fog nodes and Fog servers are necessary to work and seamlessly exchange data with other Fog systems for required application.

- **`Maintainability`**: Software architectures are often developed with modifiable capabilities and always need new features or bug fixes. The software is easily maintainable and extensible, which encourages its quality improvement. In [137] defined maintainability as "*The ease with which a software system or component can be modified to correct faults, improve performance, or other attributes, or adapt to a changed environment*". The organizational approach of Fog-based services architecture allows developers to quickly and easily fix a bug without creating a new error, add new features without introducing bugs, improve usability, increase performance by providing code documentation and writing readable code for the general functions and interactions between the different parts of the Fog architecture.

Further, non-functional requirements such as security, data integrity, usability, and reliability are still to be addressed [138].

3.3 Design for Fog architecture

We design our FSP in a hierarchical way. This allows processing, networking, and storage at each level according to the topology and services running on the FSP. IoT will be a major service-oriented technology. Therefore, it must be easy to organize all levels of the fog hierarchy through many different classes of stakeholders. The advantage is that the FSP architecture is easy to create, modify, and maintain. As we highlight in section 2.7.4, we also need to take into consideration some requirements such as data processing at low layer and semantically annotation to make them more sense and interoperable for IoT applications. This architecture provides a semantic data annotation (SDA) service, which is the ability to map meaning to raw data and obtain knowledgeable information. To ensure semantic interoperability in heterogeneous IoT environments, we decide to use Semantic Web (SW) technologies [139], a standard of W3C, to interpret and integrate

data reading from heterogeneous data sources on IoT networks to enhance the quality of data and increase interoperability. In addition, the SDA service allows to model IoT data based on shared vocabularies that different Fog nodes can interpret by using several SW standards such as OWL, RDF. To design conceptual models, we use **`Ontology`** as the main approach to provide a semantic annotation of raw sensory data.

Based on the advantages of a **`microservices-based`** architecture model such as *services discovery*, *loose coupling* and *high cohesion*, the FSP architecture is proposed to integrate services that take advantage of the outstanding features of this model to manage a kind of multi-network technology of heterogeneous IoT devices and components in the FSP and are ubiquitously located in Fog areas. In the FSP service architecture model, a set of service instances often change dynamically to be assigned dynamic network locations because of auto-scaling, failures, and upgrades. Therefore, when a request is sent from a client, it must use a service discovery mechanism [140]. To ensure auto-scaling, service instances must be highly cohesive for groups of similar functionalities and loosely coupled [141].

To meet the above criteria, we propose a four-layer services architecture and the two underlying system services for IoT environments. These include the IoT producer layer, the FSP infrastructure, the FSP management services, the FSP support services, the security/private sphere management and the analysis and data management, and the consumer layer. Figure 3.2 shows a simplified view of our FSP architecture.

3.4 Fog Services Provider architecture

This section provides some basic notions necessary for the rest of the chapter. *IoT devices*, *IoT things*, or *IoT objects* will be referred to as *Producers and Consumers*, while FSP is the main core and responsible for providing services to Producers and Consumers.

3.4.1 Preliminaries

1. **`Producer`**: In IoT context, *producers* are devices or virtual devices designed to detect events in the environment and change in its environment and also can generate data. It is made or adapted for a particular scenario with the role of physical devices such as sensors, actuators, or virtual devices like web services, etc. Producers are deployed on the IoT services layer. Data generated from producers

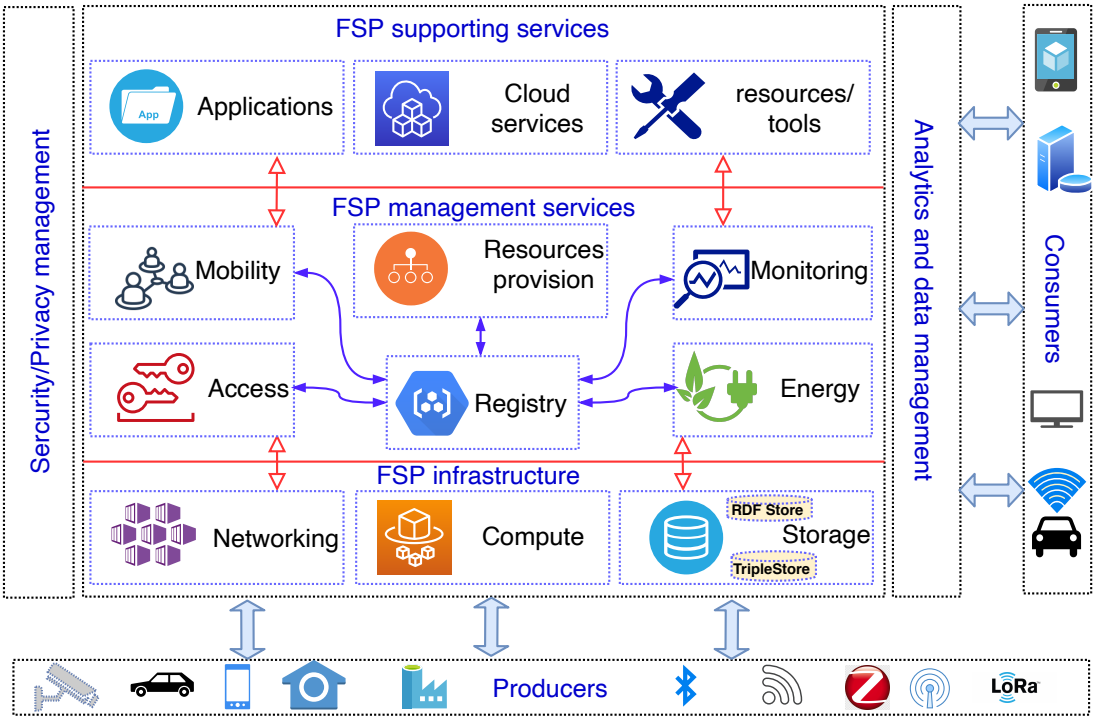

Figure 3.2: *Microservices architecture for FSP*

will be processed and transmitted to Fog services layers.

2. **`Consumer`**: A *consumer* is a software entity (microservices, user application) that enables us to analyze and process data generated from producers. It provides utilities for clients using Fog services. Furthermore, it also allows Fog service provider's applications running on a cluster of federated Fog nodes. This type of service is similar to Software as a Service (SaaS) in cloud computing, and means that IoT devices can access the capabilities of Fog nodes through either synchronous or asynchronous communication mechanisms (e.g. API, messages protocols). For example, animal tracking software installed on smartphones or computers might be used to track some properties of wildlife animals to know their location and movements. These producers periodically send information to that software, and a message is triggered to be sent to the wildlife's protector.

3. **`FSP node`**: The FSP node is the core component of our architecture. It is an entity that provides services for producers and consumers who can communicate with FSP. It includes both hardware components (e.g., gateways, switches, routers, servers, etc.) and software components or virtual components (e.g., virtualized switches, virtual machines, etc.). It can be tightly interacted at a high level and low level

with the smart end-devices or access networks and provide computing resources to these devices. Additionally, FSP nodes provide some form of data management and communication services between the network edge layer where end-devices reside. The FSP nodes can operate in a centralized or decentralized manner. They can be configured as stand-alone fog nodes to be able to communicate and deliver services to each other or can be federated to form clusters that provide horizontal scalability over distributed geographics through mirroring or extension mechanisms [142].

3.4.2 IoT producers layer

This layer includes a set of devices or virtual devices that detects several measurable phenomena of the physical world and transforms producer-generated data into a processable output. Producers use various characteristics in terms of functions, processing, and connectivity network technologies. They need to directly communicate with the edge of the network to collect data. They are managed over IoT protocols such as Wi-Fi, 4G/5G, LoRa, Bluetooth, NFC, Zigbee, or any other communication or transport layer protocols.

3.4.3 FSP infrastructure layer

The architecture infrastructure is Fog hardware, including networking, compute, and storage elements. It enables the deployment of easier services based on technologies such as Containers (i.e., Docker) or Virtual Machines. Moreover, providing computing services at this layer is leveraged not only by the constraint of processing power at the IoT services layer but also by the desired location of computing to satisfy better system requirements and reduce network transmission [143].

3.4.4 FSP management services layer

The core architecture of the FSP includes a set of flexible services to support low-layer and high-layer. In our architecture, we adopt a services architecture based on a microservices approach. As the name indicates, the microservice architecture is an approach to develop a server application in a series of small services [144]. This allows a micro-services architecture to focus primarily on the back-end, although the approach is also used for the front-end. All services run in a separate process and communicate with each other via protocols like HTTP/HTTPS, MQTT (Message Queue Telemetry

Transport), WebSockets, or AMQP (Advanced Message Queuing Protocol) [145]. These IoT protocols provide a set of principles for enabling communication and define the syntax, semantics, error recovery capabilities, synchronization of data between different services of Fog architecture. Also, each micro-service deploys a particular end-to-end domain or capability within a specific context boundary, and each must be autonomously developed and independently deployable. Ultimately, each microservice should have its own associated domain data model and domain logic (convenience and decentralized data management) and could be based on several data storage technologies (SQL, NoSQL) and different programming languages [144]. We describe the microservices architecture of the FSP as a set of small services as follows:

1. **`Access service`** are responsible for handling the communication from IoT producers and consumers to FSPs via their native protocols. The two most important services are *producer service* and *consumer service*. The *producer service* plays a role as connectors that interact with the IoT producers. It can simultaneously serve one or more producers, including sensors, actuators, etc. In order to ensure semantic interoperability, this service provides a semantic methodology that focuses on interoperable IoT provisioning by converting the raw data produced and transmitted by the IoT producers into a common data structure called **`RDF triplestore`** [146]. The generated data of a physical entity network is stored in a graphical database. Graphical databases are a form of RDF triplestore that can reason and discover new information and existing relationships. The flexibility and dynamism of the RDF triplestore allow linking of diverse data types, indexing for semantic search, and enriching data by analyzing text to create knowledge graphs [147]. Then, this converted data is sent to the storage services at the *FSP infrastructure layer* and can provide other services at other layers of the FSP node. The *consumer services* are the means of extracting, processing, and converting event or reading data from FSP and sending it to an endpoint or can be processed by the end-user's choice as filtering or compressing the data they are interested in. Besides, the *consumer service* also provides applications or SDKs (Software Development Kit) as templates that can be used to build applications by integrating built-in functions or customizable functions depends on the end-user's decision-making to develop specific services.

2. **`Registry service`** should be used in any service architecture to solve auto-scaling, failures, and upgrades issues [141]. This service registration is a key part of the service discovery feature. The objective of the registry is to enable services to discover and communicate with each other. When each service starts up, it must register itself with the service registration, and this functionality continues checking its availability periodically via a specified health check endpoint. For example, when one of the consumer services needs to connect to the storage service to require data information (e.g. temperature or humidity), it connects to the *Registry service* to retrieve the available hostname and port number of the storage service and then invokes the storage service to interact.

3. **`Monitoring service`** undertakes to analyze the monitoring data and dismiss events when previously specified QoS thresholds. This microservice can provide useful information such as workload, usage, and energy to help with decision-making and pricing. We highlight this component in fog computing architecture since it gives crucial information to other components.

4. **`Resource provisioning service`** provides functions for orchestration, provision, and monitoring resources in the associated FSP nodes. It includes microservices such as *Resource computing service*, *Service placement*, *Resource reallocating*. To create an allocating plan to execute received task requests according to a specified resource allocating approach (selected FSP to use) performs by *Resource computing service*. *Service placement* is in charge of handling task requests by deploying services and depends on the *Resource computing service*. *Resource reallocating* enables the calculation of a resource allocating plan according to the events (a new producer joins or leaves the FSP node or a producer with a hardware failure).

5. **`Mobility support service`**: Fog computing expands the Cloud towards the edge of the network, and the distributed resources and services of computation are closer to the producer to provide some outstanding features such as location awareness, low latency, geographical distribution, real-time interactions, heterogeneity, and support for mobility [8]. This proximity provides some advantages, such as time-sensitive latency, but producers frequently move from one location to another based on different types of communication and connections. The mobility of producers will affect the Fog node's ability since when a producer moves, the

distance between the producer and the services provided in the Fog node will increase. Therefore, migration of Fog services may be one of the possible ways that allow the services always to be close enough to the producer [148]. The FSP node allows continuous connectivity of data sources if network access is temporarily interrupted by keeping a location list of the producers to provide services such as motion tracking, location information sharing, and maintaining its state with the targe FSP node. Additionally, our FSP node's mobility service enables keeping the network locations with geographical information of producers by using a services placement mechanism.

6. **`Energy efficiency service`**: Energy management is one of the issues to be considered when deploying applications and managing services and data in IoT environments such as smart homes, smart cities, especially in smart grid (micro-grids and nanogrids). It is required to handle power generation and consumption in these application domains. Cloud computing is introduced as a solution for centralized computing models based on requirements for data processing [149]. However, controlling the heterogeneous producers and diverse data from these producers is a significant challenge for the Cloud computing in terms of scalability, interoperability, and adaptability to respond to time-sensitive services or applications [111]. Therefore, the reduction of energy consumption can be achieved by remotely monitoring smart devices based on feedback on their energy consumption. The FSP architecture is proposed based on the Fog computing paradigm aimed to develop services that are deployed close to the edge of the network. It is also designed based on a mathematical model to study offloading services from FSP nodes to producers enables moving computing, control, data storage, and processing by introducing a dynamic speed scaling mechanism at the edge of the network [150]. Furthermore, the efficient energy management service also depends on the optimal services placement to accommodate producers' scalability and heterogeneity and reduce network power consumption. Therefore, an energy efficiency service is needed to manage and maintain IoT producers' computation.

3.4.5 FSP supporting services layer

This layer provides the ability to get data generated from producers to other services or external systems. It allows performing the act of delivering the data to registered consumers. In the FSP architecture, to trigger other systems (external systems associated with the FSP), a notification mechanism is required. These notices are informational in the event of unusual, important, or urgent alerts to be immediately addressed. In addition, application services are also the means to get data from the FSP nodes to external systems or vice versa. Its main purpose is to solve scalability problems for consumer services as well as provide a flexible solution for exporting data outside of the FSP node.

Fog computing is about extending computing capabilities from cloud computing to near the edge of the network. However, in some applications related to storage capacity, cloud services are still an irreplaceable solution [151], especially in applications such as smart cities, smart transportation where large data storage and processing with high computing power are required. Thus, the FSP architecture provides APIs with flexible capabilities for application implementation and an environment for developers with a variety of tools and resources (operating systems, programming languages).

3.4.6 Security and privacy management

We aim to develop the FSP architecture with some security components both inside and outside the FSP node, such as data security and connection security. Protects the data and control of producers, and other objects managed by the FSP. The FSP administrator would initialize the security components, set up a security service operating environment, manage user access control, and create JWT(JSON Web Token) for resource access for other services.

3.4.7 Analytics and data management

The FSP architecture data consists of several steps that start from the data collection at the producer level. The data is generated, then processed and stored in the FSP node and sent feedback to the producer layer to execute the commands required by producers.

As illustrated in Figure 3.3, we describe the main steps in data management include, *Data collection*, *Data pre-processing*, *Data annotation*, *Data analytics*, *Data exchange*,

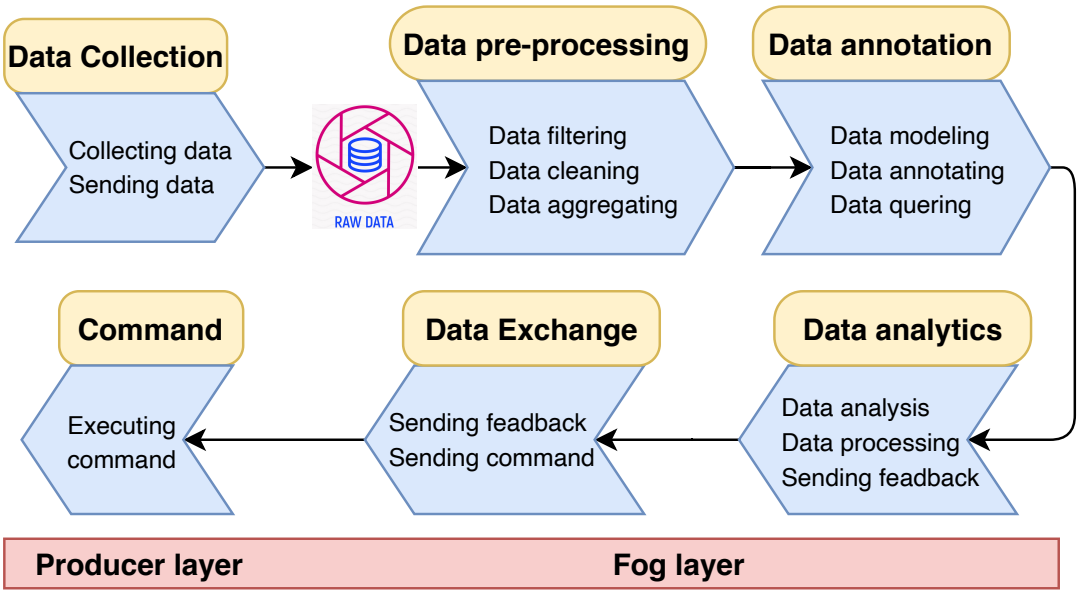

Figure 3.3: *Data management in the FSP architecture*

and *Command* execution. By providing the capabilities at all levels to gather, process, analyze, and act on data generated by connected producers at global scale, the FSP architecture allows its data management to be more efficient and lightweight.

3.4.8 Consumers

The FSP architecture allows end-users to register and retrieve data of interest. It defines user applications as *Consumers* to provide the end-user with information on how to get data, what data to get, and where to get data. *Consumers* also allow the end-user to use several features such as filtering data (i.e., by `ID`, or `name`), getting data in a specific format such as JSON-LD, XML, compressed data (i.e., ZIP, GZIP), or providing some standard-based protocols, which support for getting data such as MQTT, HTTP, RESTful, etc.

3.5 Interaction model in FSP

The microservices-based FSP architecture is a distributed system that runs on multiple processes and services. Each service instance is a typical process that can run on many different FSP nodes. Therefore, services need to be communicated with each other through an inter-process communication protocol [144]. This protocol provides the interaction between synchronous communications, which uses HTTP-based REST API, and can be used for asynchronous communications with message patterns such as MQTT, AMQP.

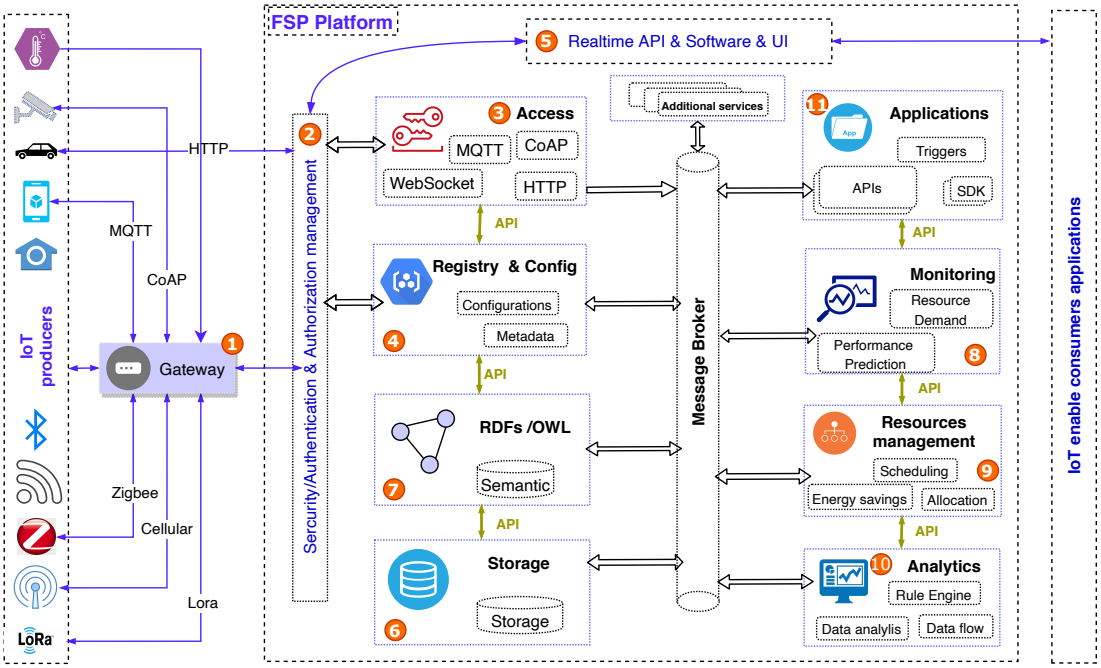

Figure 3.4: *Interaction model in FSP*

Figure 3.4 illustrates a communication mechanism in FSP architecture. The beginning of this mechanism is started when *producers* want to join the *FSP*, they use some different access technologies and protocols such as Zigbee, Cellular, LoRa, and some message protocols like MQTT, HTTP, CoAP to connect to the FSP. These communications are handled by **`gateway (FSP gateway)`** and be controlled by a security *authentication or authorization* management unit at the second step. At the third step, these heterogeneous devices are managed by an *Access service* with many synchronous (WebSocket, HTTP) and asynchronous (MQTT, AMQP) communication mechanisms. For the services to communicate, they must be registered to the *Registry & Config service* in step four. At step five *consumers* interact with APIs or User Interfaces to get interesting information. After successful access and registration, data is generated as raw data and processed and stored in a unified format (RDF/OWL) by six and seven steps, respectively. Steps eight, nine, ten, and eleven are services for monitoring, managing resources, and analyzing data.

3.5.1 Synchronous communication

Synchronous communication in FSP architecture is usually related to a request/response interaction style. One service makes a request to another and waits a while to process the result and send the response back. HTTP-based REST API is a synchronous protocol.

In this case, when a service sends and waits for a response, tasks can only continue after receiving the response from other services. For example, the *consumer* sends a request and waits for a response from the *data processing and analytic service*. The independence of these two processes can be synchronous. This can result in the thread being blocked while waiting for the service to process data and send a response back to the *consumer*. However, the synchronous communication mechanism ensures some advantages in communications of the FSP architecture such as *statelessness*, *uniform interfaces*, *uniform resource identifiers (URIs)*, and *self-describing messages* [152].

1. **`Statelessness`**: The microservices-based FSP architecture is stateless if it does not store state information in internal storage (memory); instead, it is stored externally (databases or files). This is in contrast to statefulness services, which store all of the communications between services in the memory. It means that sessions or cookies will be stored in memory to establish a state between a *consumer* request and an *FSP node* handler. This difference has several advantages of statelessness when comparing statefulness in some cases, such as an unexpected restart or termination of service or FSP node.

2. **`Uniform interfaces`**: Every action on a service such as *create*, *read*, *update*, and *delete* are performed through well-defined commands including **`PUT`**, **`GET`**, **`POST`**, and **`DELETE`**, respectively. These commands allow for resource management with a clean, uncluttered, and straightforward interfaces.

3. **`Uniform resource Identifiers (URIs)`**: Identifiers are a standard in the IoT [153], including *object identifier*, *communication identifier*, and *application identifier*. The object identifier is on behalf of physical or virtual objects such as *barcodes* and *RFID*. The communication identifier is used to uniquely identify the nodes on the network that are capable of communicating with an *IP address*. The application identifier represents resources in service layer. These resources are executed by *uniform interfaces* through calling unique URIs.

4. **`Self-describing messages`**: Resources should be formatted with messages that can separate the structure and content of the resource. These messages can be translated using different data types such as XML, JSON without losing information, portability, safety, security, and human-readability. The example below shows the structure of a message of the producer profile by a JSON format:

```
{
   name: "iotproducer1"
   Serial: "90035229"
   memory: "1024 MB"
   processor: "Broadcom BCM2837"
   model: "Model B Pi 3"
   manufacturer: "Sony UK"
}
```

3.5.2 Aynchronous communication

The asynchronous communication is implemented in the FSP architecture when services exchange with other services use a message model (or publish/subscribe model). This model is based on an event-bus interface or message broker that serves as an intermediate access point for services [154]. FSP is a distributed architecture that is designed to support interfaces for heterogeneous devices, services, and applications in cross-domains with a diversity of data. Therefore, to ensure consumer requests are not blocked, it needs to be subscribed to events on topics.

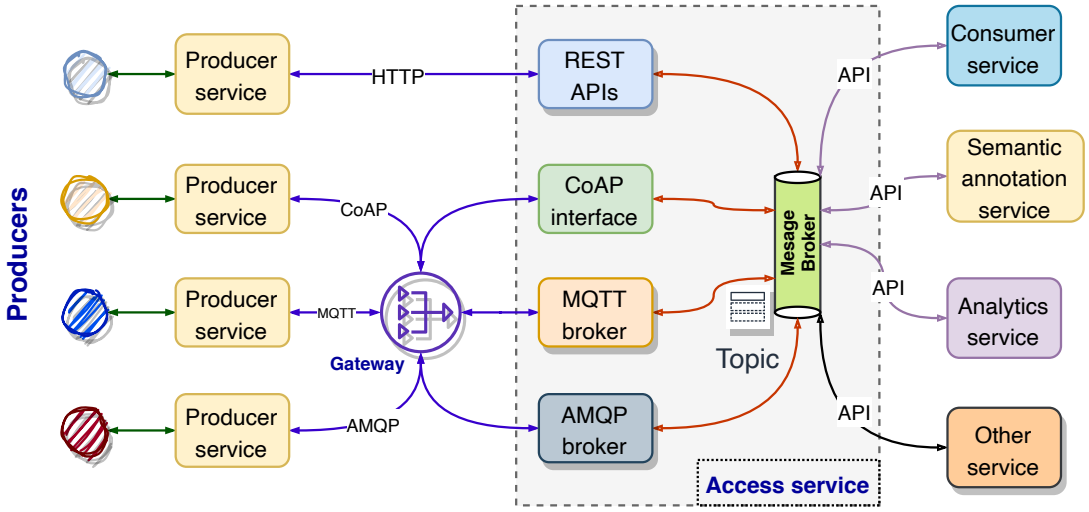

Figure 3.5: *A communication mechanism in FSP.*

Figure 3.4, the **access service** provides APIs based on several communication protocols such as HTTP, MQTT, and CoAP that are implemented on the **gateway** to enables *producers* which can communicate with the FSP node depending on their protocols. Once the **access service** receives messages from the *producers*, it analyzes and pushes a **message queue** that is configured in a **message broker**. The **message**

queue is an asynchronous form of communication between services designed to provide a lightweight publish/subscribe communication mechanism [155]. Moreover, the **message queue** also enables services in the FSP architecture to ensure low power consumption, minimized data packets sent, and efficiently distributed data to *consumer* or other services. Figure 3.5 illustrates the communication mechanism in the FSP architecture to ensure interoperability and scalability.

3.6 Services deployment

The FSP architecture is built on the microservices model with its loosely-coupled architecture that enables the deployment of decentralized services and tiered resource computation. Unlike monolithic applications, deployment optimization is only for a single specific case. The microservices-based architecture needs to scale many services that can be implemented in many different programming languages, and each service has its own dependencies. Therefore the architecture must have high reliability in the deployment methods to provide new functionalities or add new services without affecting the overall architecture or causing critical failures.

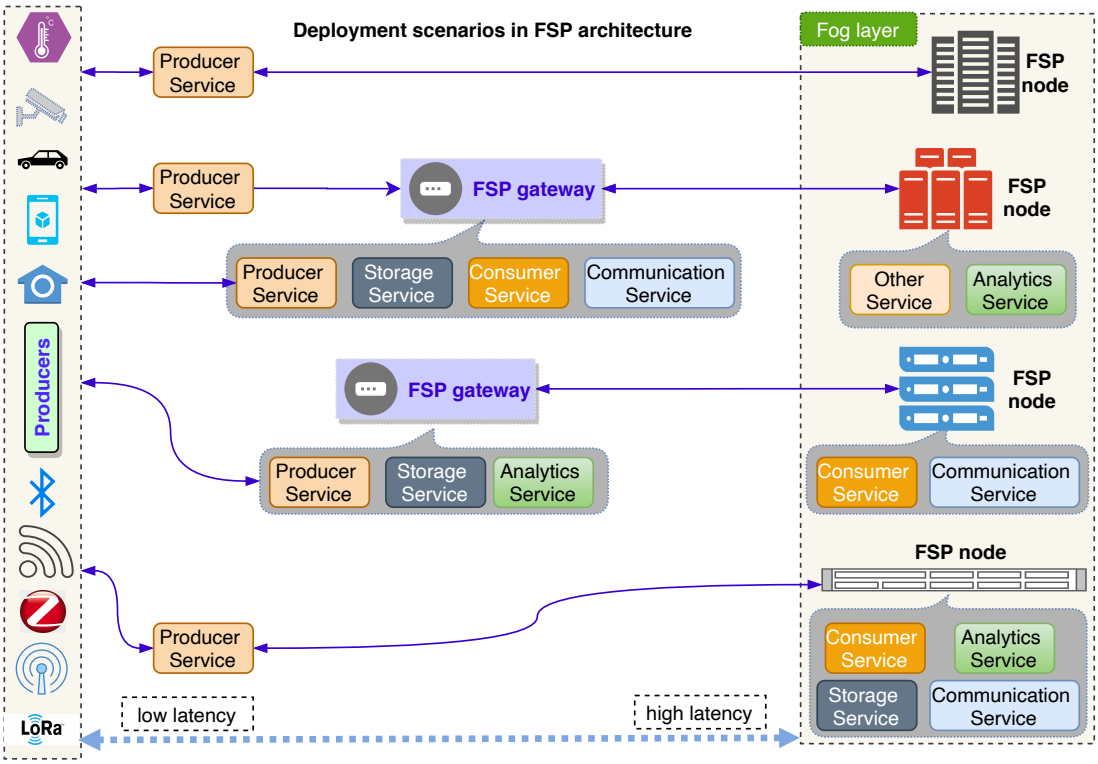

Figure 3.6: *Services deployment model in FSP*

The FSP architecture is designed to allow services to be deployed at different levels

with the objective of minimizing the cost of deploying new services, and optimizing resource utilization, as illustrated in Figure 3.6. Moreover, a service can be distributed across several cross FSP nodes or at gateways. This allows the FSP architecture to take advantage of computing, storage, or network resources wherever they are deployed on the edge. For example, a **`producer service`** can be implemented at the gateway, or in smart producers while other services such as **`storage service`**, **`communication service`**, **`analytics service`**, or **`consumer service`** are deployed at the associated FSP nodes to maximize resources.

Although it takes advantage of the strong deployment capabilities in microservices-based architectures. However, we also see that there are some issues that need to be considered when deploying services. The issues that are presented as below can be:

- How to optimize the services deployed on devices with limited resources.
- The decision issue to place the service in a heterogeneous set of FSP nodes.
- A reliable and cost-effective way to deploy the application.

3.7 Evaluation and Comparison

In the FSP architecture, each service is considered a small problem, and services need to be designed and operated as independent functionalities without being dependent on another service. It must be sustainable and not affected when an error occurs. Services should also provide their interface components externally and support sharing information structures and data constraints through standards-based data schemas. Therefore, the FSP architecture can be highly interconnected, scalable, and extensible.

Each service are separated about the database, have its own bussiness logic, and communicate with other services or applications through the REST API (synchronous communication), as show in Figure 3.7. Besides, services can also communicate with each other through events (asynchronous communication). These events are divided into two types of services such as the **`receiver service`** (*data analytic service*, *consumer service*) that listen from other services and the **`sender service`** (*producer service*), which send events to the **`receiver service`** by providing names for events through topics. From this design, we will proceed to build each service with the corresponding internal components. The FSP architecture supports REST APIs services that are performed by four operations such as *create*, *read*, *update*, and *delete* through well-defined

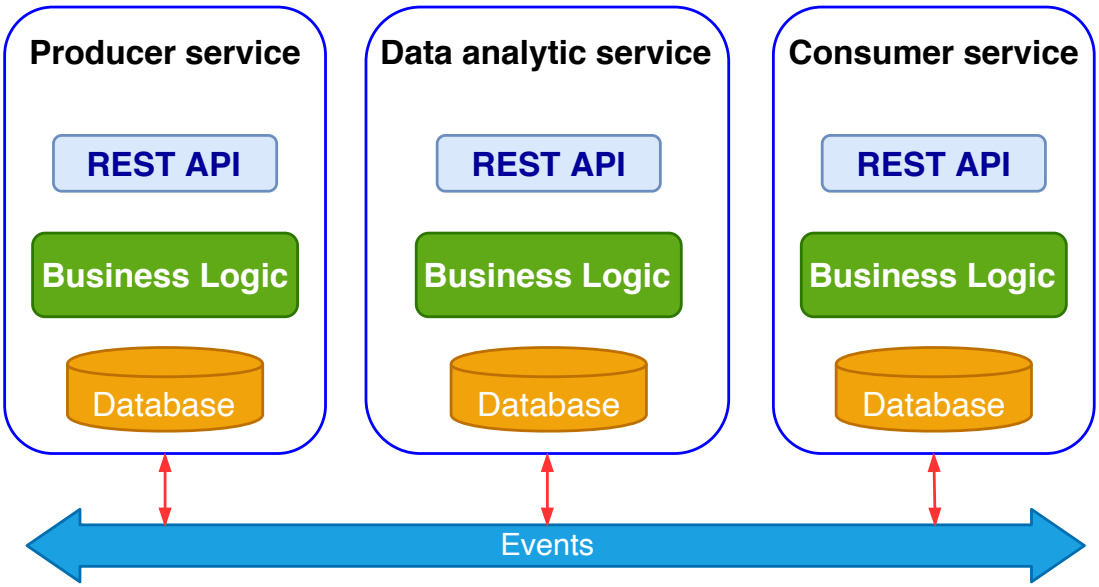

Figure 3.7: *Design model for services*

Table 3.1: *FSP architecture REST APIs*

Methods	Endpoint	Description
POST	/api/objects/	add a new object
GET	/api/objects	get all objects
	/api/objects/:id	get an object by id
PUT	api/objects/:id	update an object by id
DELETE	/api/objects	remove all objects
	/api/objects/:id	remove an object by id

PUT, GET, POST, DELETE commands, respectively, as shown in Table 3.1. Where objects can be any entities accessed by the service such as *producers*, *consumers*.

And the code for the APIs to be implemented looks like this:

```
@api_view(['GET', 'POST', 'DELETE'])
def producer_list(request):
# GET list of producers, POST a new producer, DELETE all producers


@api_view(['GET', 'PUT', 'DELETE'])
def producer_detail(request, pk):
# find producer by primary key (id)
# GET / PUT / DELETE producer


@api_view(['GET'])
def producer_list_published(request):
# GET all published producers
```

In the development and implementation of the FSP architecture, we use several

technologies such as Docker and Django Rest Framework [156] as microservices. We select the Django REST framework to build APIs for its powerful and flexible toolkit that supports serialization both object-relational mapper (SQL, MYSQL databases) and non-object-relational mapper (NoSQL databases) data sources. Asynchronous protocols were developed using several protocols such as MQTT, AMQP, XMPP, ActiveMQ that ensure stability, providing interoperability and integration for a wide variety of data [157].

To evaluate the advantages of FSP architecture, we also compare the FSP architecture with some other architecture that are proposed by some authors (introduced in Table 2.2) based on several criteria such as *heterogeneity*, *manageability*, *interoperability*, *scalability*, and *DevOps*, as illustrated in Table 3.2.

1. **`Heterogeneity`**: This criterion is considered in the context of diversity in the IoT environment. It deals with heterogeneity from the lowest level (IoT producer layer) to the higher processing layers (FSP node). Therefore, Fog architectures should have a processing mechanism to optimize the data since it is created from the lowest layer. This criterion is evaluated at the support levels such as **`partial`** or **`full`** support.

2. **`Manageability`**: The Fog architecture design must ensure that it is manageable to not only handle access from heterogeneous *producers* that use various network access technologies and *consumers* but also facilitate the fast deployment of services or applications to support this textit producers and *consumers*. The architecture should provide flexible and dynamic services management with a combination of both synchronous and asynchronous communication with authentication and authorization mechanisms. The (✓) symbol indicates that it is supported by both types of communications.

3. **`Interoperability`**: This criterion enables the components of the Fog landscape to be exchanged with each other in terms of producers, APIs, data format, and shared resources between IoT cross-applications and cross-domains. We classify it into two groups: full interoperability (**`full_int)`** and partial interoperability (**`par_int`**). The full interoperability attempts to address interoperability at various levels such as communication technologies, data formats, and heterogeneous service interchange. The partial interoperability is provided only at one level and can be

within a domain or an IoT application.

4. **Scalability**: A scalable architecture is an architecture that can expand horizontally and vertically to adapt to increased workloads. Vertical scaling allows the deployment of services on a Fog node with higher hardware capabilities than existing ones. Horizontal scaling is the way of adding services in the application deployed in the running Fog node. The (✓) symbol indicates that functionalities in the fog architecture should be handled in parallel in multiple services.

5. **DevOps**: According to [158], DevOps is a combination of **Development** and **Operations** in the software development process. It allows to ensures fast planning, continuous integration, continuous delivery, and monitoring of application processes that can be closely linked to shortening the software development life cycle. Therefore, DevOps was born with the idea of combining **Dev** and **Ops** for a common objective is *speed* and *stability*. Fog architectures can be developed to achieve purposes such as implemented fast code and rapid deployment, few services failures, or immediate recovery from failures. Due to the nature of DevOps is stability and speed. Therefore, whether an architecture can be developed openly depends on the design level of architecture such as **hard**, **difficult**, or **easy** .

3.8 Conclusion

In this chapter, we propose an analytical and design approach for a **Fog Services Provider** architecture based on Fog computing paradigm for supporting heterogeneous devices and interoperable components. We also introduce several essential concepts such as **producer**, **consumer**, **FSP node**, and a set of services that enables rapid development, management and scaling of IoT applications. Moreover, the FSP is designed based on microservices architecture to provide interoperability and enable services to work independently and scalable in different applications domains. Besides, it also allows managing **producers** which use different network access technologies by **access services** and provide synchronous and asynchronous communications for horizontal and vertical scaling of **consumers** and **FSP node**.

Table 3.2: *Comparison resultes*

Criteria	Bonomi et al [9]	Brito et al [121]	Habibi et al. [110]	Semantic-Fog [120]	**FSP architecture**
Heterogeneity	(**full**) - supported by fog abstraction layer within the level of fog nodes and physical resources.	(**partial**) - supported by fog orchestrator agents within the level of fog nodes.	(**full**) - handled by using virtualization and abstraction layers but detail is not clear.	(**full**) - supported by a controller within fog nodes at L2-Fog and L1-Fog devices through interfaces.	(**full**) - supported by *producer services* at lowest layer, data and applications at fog nodes (*SDA service*).
Manageability	(✓) - supported by Fog service orchestration layer with APIs and distributed message bus.	(**-**) - provided the interactions between components through interfaces but not specify any approach.	(✓) - supported by the hierarchical SDN approach with providing services through the fog controller.	(**-**) - based on iFogSim simulator, managed services and components in fog nodes are not mentioned.	(✓) - supported by interfaces with REST APIs, an *access service*, and a *message broker*.
Interoperability	(**par_int**) - using generic APIs but detail is not clear and lack of data processing mechanism.	(**pat_int**) - M2M interoperability through standard communication within fog nodes.	(**full_int**) - federated SDN controllers by using interfaces REST APIs and Openstack.	(**par_int**) - supported by Semantic technology and ontology. However, it is only handled at data level.	(**full_int**) - supported by the *FSP gateway* at network level, the *FSPontex* at data level, and applications at microservices.
Scalability	(✓)- provided the distributed message bus to address scalable management.	(**N/A**) - not mentioned.	(**-**) - supported only at management and control plane using hierarchical architecture.	(**N/A**) - not mentioned.	(✓) - provided by both synchronous and asynchronous communication
DevOps	(**easy**) - a reference architecture but it is not clear about design approaches.	(**difficult**) - proposed architecture is based on FO and FOA components. However, it resides on the same machine.	(**difficult**) - the number of devices managed by SDN controller is limited and difficult to scalable.	(**hard**) - due to using an simulator, it is only suitable for mathematical theory and difficult to develop.	(**easy**) - based on Microservices-base approach.

CHAPTER 4

Data management in Fog Services Provider

Honesty is the first chapter in the book of wisdom.

Thomas Jefferson

Abstract

In this chapter, we focus on the data management mechanism in the FSP architecture to ensure the data generated by producers that can be handled in an organized and flexible approach. Furthermore, with the design and development of microservices-based FSP architecture, we also solve some issues of how to process and manage data across multiple services and communication between services to achieve interoperable and semantic data for different application domains by using semantic web technologies [21, 22].

[22] H. Le, N. Achir, and K. Boussetta. "Fog Services Provider Architecture for IoT". in: *2020 11th International Conference on Network of the Future (NoF)*. 2020 11th International Conference on Network of the Future (NoF). Oct. 2020, pp. 8–15. DOI: 10.1109/NoF50125.2020.9249177

Chapter content

4.1 Introduction

Fog Services Provider architecture proposes to address computing, storage, and networking issues in large-scale deployment of IoT applications. It is a different computing paradigm for cloud computing where computing, storage, and networking services are deployed close to the network's edge in a distributed and possibly collaborative way. The **FSP** architecture is designed based on an independent service management approach (**microservices**) that provides a flexible and effective management mechanism for several issues such as heterogeneity, diverse interoperability at multiple levels (networks, data, and application). **Producers** are characterized by a wide diversity of network access technologies, including Wi-Fi, Cellular, Lo-Ra, ZigBee, Bluetooth, or based on messaging technologies such as MQTT, CoAP, AMQP, etc. Dealing with such generated data from these producers is still very challenging.

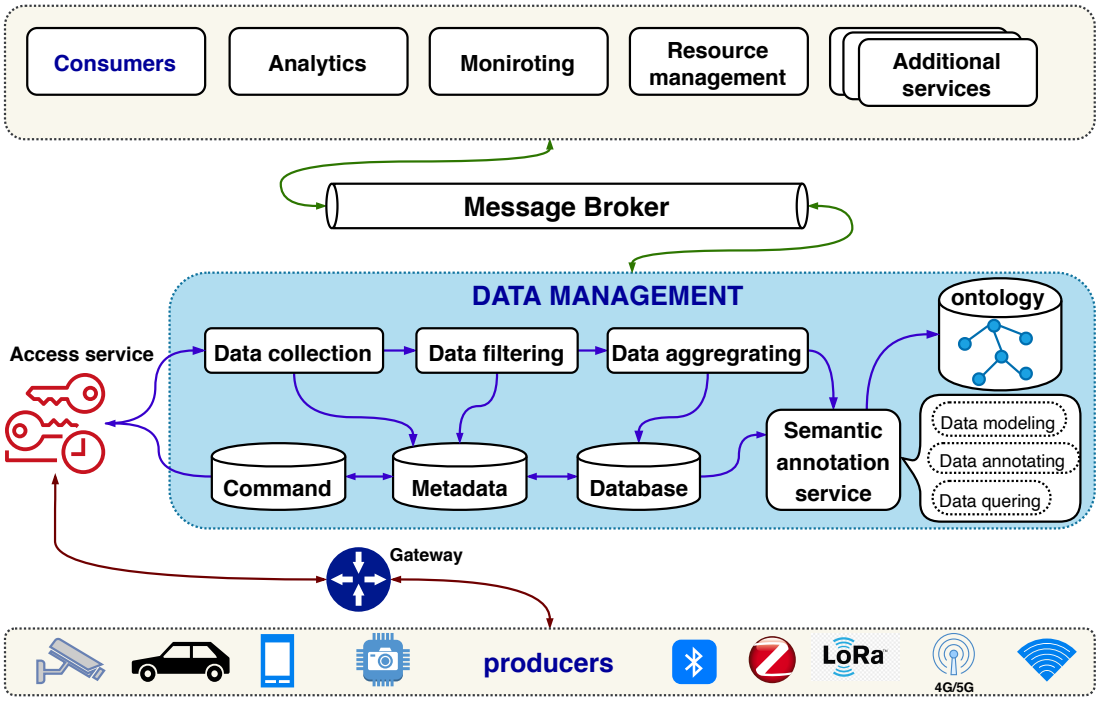

Figure 4.1: *Data management in Fog Services Provider*

Although the FSP microservices-based architectures have solved many issues it also brings a new set of problems. One of them is the ability to maintain data consistency across services [159]. Many data is being generated from billions of IoT producers and periodically report certain events or abnormal phenomena. After this data is collected and processed, it optimizes industrial environments, manufacturing, and monitoring systems through a Fog-based architecture, which is deployed nearly by data sources. The

objectives of FSP architecture integrates several different data processing approaches such as *data collection*, *data filtering*, *data aggregating*, and *semantic data annotation* to deal with data heterogeneity, interoperability, and synchronous and asynchronous communication protocols in IoT environments [160], as demonstrated in Figure 4.1. Data management in the FSP architecture can help improve application quality and simple deployment since services are developed independently of their own data. However, efficient data management also needs to consider in some cases [161]:

- Data type requirements must be accessed by the services and enforced by the regulations, security through defined APIs.
- Data transformation services need to communicate over standard messages based on the combination of a message queue and defined APIs.
- Data access control policies that need to be stored depend on data sensitivity. Therefore, services must also be used by the appropriate role for the application component.
- Data that needs to be stored in storage must be considered some issues such as the volume of data, the velocity of data, how the data is used, and the security and resilience requirements.
- Data traffic should be monitored and logged information, and controlled who accesses what data to ensure that security is properly operated.

In this chapter, we propose a data management approach for **`Fog Services Provider`** to handle data generated from heterogeneous **`producers`**, and the interaction of **`consumers`** and **`FSP nodes`** with data in ubiquitous environments. In the last chapter, we represent a microservices-based architecture and provide a set of services to manage functionalities and ensure interoperability and scalability. However, the proposed architecture's data management and data life cycle are essential concepts. Furthermore, we define and propose a semantic data annotation service that the FSP can support for data to be shared and interoperable with cross-domain or applications. A novel aspect of this service architecture is to build a general data management mechanism to describe and store data of IoT heterogeneous devices based on Semantic Web technologies. We also propose a data mapping algorithm from relational databases to a unified format. For effective management for registration and access of these devices into the FSP

architecture, we demonstrate several procedures that enable automated integration and data processing with the FSP.

The rest of the chapter is organized as follows. Section 4.2 introduces data management mechanism in Fog Services Provider architecture by processing approaches that are separated into two stages, include data organization for raw data processing and data annotation for semantic data processing. To manage producers and consumers, we implement some services to evaluate their interactive capability with FSP architecture that is presented in more detail in section 4.3 and section 4.4. The last section 4.5 draws a conclusion.

4.2 Data management in Fog Services Provider

With a considerable amount of data generated from many heterogeneous and distributed IoT producers, Fog computing plays an essential role in latency-sensitive applications to handle real-time data management. However, processing such large amounts of data and responding to requests on-time is challenging with Fog computing architectures. Moreover, data generated from heterogeneous producers often have different formats and syntax. This leads to interoperability and understanding issues in other IoT domains. Interoperability in IoT can be considered based on across-domain applications or services such as devices interoperability, networking interoperability, platform interoperability, semantic interoperability, and syntactic interoperability [14].

For efficient data management, we propose an FSP architecture to resolve heterogeneity between producers and how to manage them through a necessary information model. This model can facilitate semantic interoperability, such as understanding what data means and sharing Semantic technologies. Semantic technologies enable computers to understand the meaning of data and process data correctly. Therefore, the application of semantic such as Semantic Web to IoT is becoming a crucial prerequisite for the integration of data and the future development of intelligent IoT applications and services. IoT producers generate an enormous amount of raw data that is processed and made smarter by transforming the information from raw data into semantic annotations using `Ontology`. The *producer service* is in charge of processing low-level raw data information with knowledge application services by enabling interoperability at the data modeling level to facilitate the heterogeneity between IoT producers and the FSP and provide APIs for interaction with different types of devices.

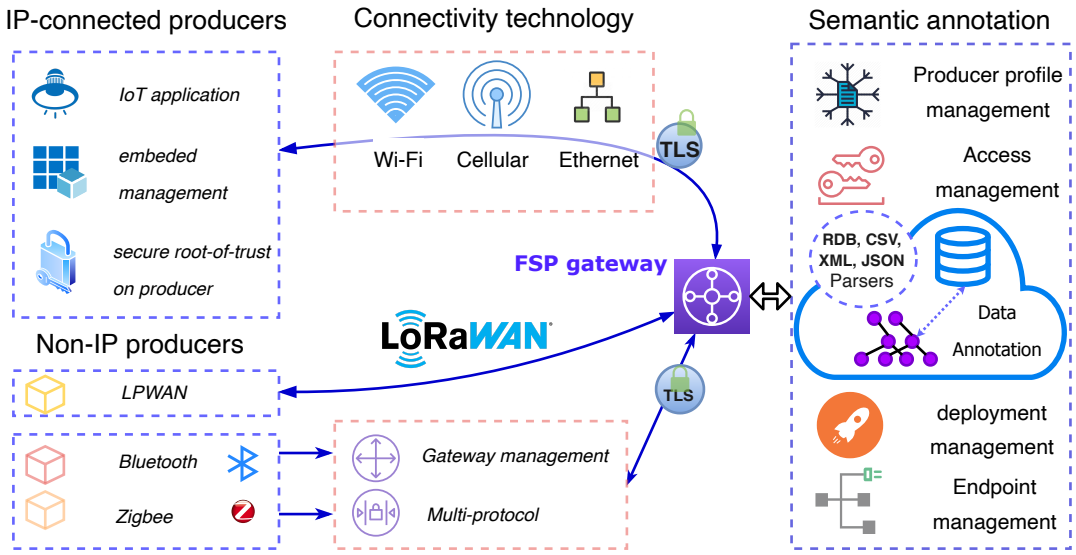

Figure 4.2: *General interaction model in IoT producer layer*

Figure 4.2 shows a general interaction model for heterogeneous producers. The low-processing components are mainly categorized into three major elements: IP and non-IP producers, connectivity technology, and semantic annotation. Typically, IP and non-IP producers represent the lowest level and are composed of very limited resources with different connectivity technologies such as Wi-Fi, Cellular, Lo-Ra, ZigBee, and Bluetooth, etc. These heterogeneous producers are supported to connect to the FSP by a gateway called the **`FSP gateway`**. The **`FSP gateway`** is responsible for ensuring secure and efficient communication with producers. The proliferation and the velocity of data increase dramatically in IoT systems. To control this amount of data, these IoT systems usually rely on message protocols for data exchange [157]. Messages can be exchanged within the **`FSP gateway`** by integrating message protocols such as MQTT, CoAP, WebSockets, HTTP/HTTPS.

Although the processed data can solve heterogeneous producers' interoperability issues can be effectively managed through the FSP gateway. However, the heterogeneity and diversity from multiple data sources, exploiting the potential advantages in generated data would not be fully achieved without a suitable method to support data links and information exchange. IoT systems still lack accessibility to semantic data [162]. **`Semantic annotation service`** is responsible for processing these semantic data by using ontologies to describe and annotate the data information with the help of Semantic Web techniques such as the Linked Data model and the SPARQL query language

(SPARQL Protocol and RDF Query Language) [163]. Data management that in the FSP is performed by several functionalities such as *data collection*, *aggregation*, *modeling*, *mapping*, *linking*, and *querying* [164]. These functionalities will be introduced in more detail next section.

4.2.1 Data organization

The data organization mechanism plays a vital role in the FSP architecture. It includes the steps of collecting, storing, preparing, and aggregating data for the pre-processing task. This task is responsible for analyzing and cleaning the producers' raw data by filtering redundant data and converting the rest of the data into JSON, XML, CSV, or a relational database (RDB). Pre-processed data will help optimize resource computation and facilitate the data annotation step. In this first stage, the data management in the FSP is performed by three functionalities as follows:

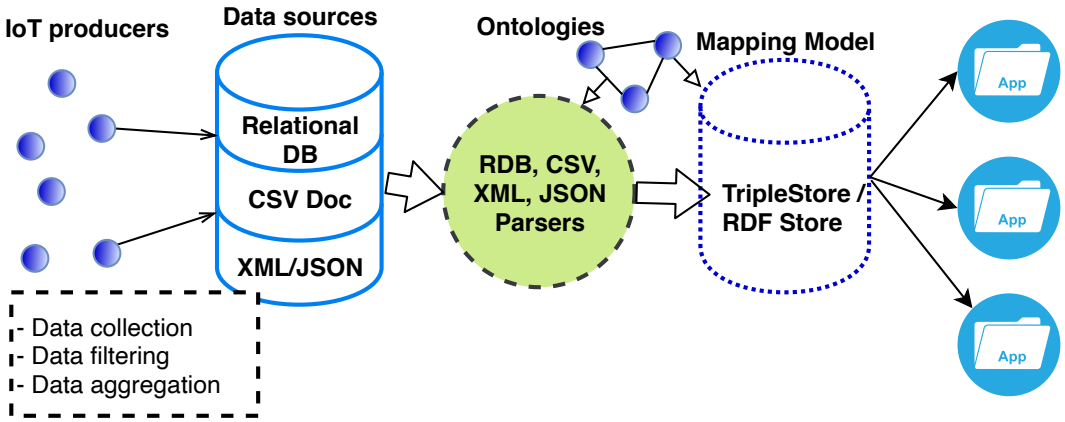

Figure 4.3: *Data pre-processing in the FSP architecture*

1. `Data collection`: the role of this functionality is to collect data of any type, source, and structure to make it easily accessible to heterogeneous producers. It supports several solutions such as databases (SQL, Non-SQL), CSV documents, or XML/JSON, etc.

2. `Data aggregation`: the aggregation functionality receives data-streams from producers and stores them in temporary storage for analyzing. The aggregated data, after being stored in different types of databases, is filtered through profiles (i.e., `producerId`, `producerName`, `producerModel`, etc.) that we have configured and sent the producer's description to the FSP gateway. By doing this, we have minimized the data size during data transmission between producers and the FSP

gateway.

3. **`Data filtering`**: the purpose of this functionality is to reduce the amount of data to be annotated in a further step and save resources required for annotation. It includes some rules, which contain a set of predefined rules to filter data from the data aggregation step and removing duplicated and unnecessary data.

The RDB, CSV, XML, JSON parsers then receive these aggregated necessary data to process further by modeling these aggregated data to make the relationship of data objects. This modeled data is converted into a Resource Description Framework (RDF) and linked with semantic annotations using the ontology. The resource RDF can be described by statements that define attributes and values. The RDF uses a "`triplet`"or "`RDF statements`"to describe different elements in a statement. The *subject-predicate-object* format allows to take any subject and link it to any other object by using a predicate (verb) to indicate the type of relationship that exists between the subject and the object.

A Triplet (subject->predicate->object) is described as a node-arc-node link [15] whose direction starting from *subject* to *object*. A set of such triples is called an RDF graph. An RDF graph can be visualized as a node and directed-arc diagram. For example: *"The FSP is a microservice architecture"* can be stored as an RDF statement in a triplestore and it describes the relationship between the subject and object of the sentence, where the subject is "`The FSP`", and "`microservice architecture`"is the object. The predicate (or verb) "`is`"illustrates how the *subject* and the *object* are linked.

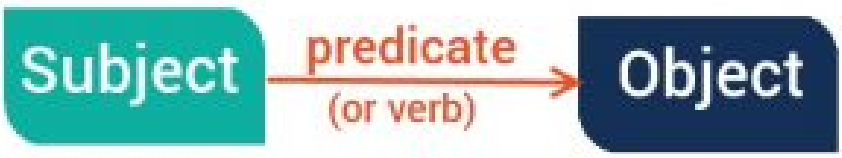

Figure 4.4: *The RDF graph*

- *Subject* is the element that identifies the entity to which the statement refers the source of an arc in an RDF graph (i.e, **`The FSP`**).
- *Predicate* is an element that defines the attribute of the subject in the statement or the property part of a triple (i.e, **`is`**).

- *Object* is an element that determines the value of an attribute, it can be a resource (URI), or a value (Literal) (i.e, `microservice architecture`).

Statements can be interpreted as resources that associate data entities defined in the RDF triplestore and used as the Universal Resources Identifier (URI). This URI is a global identification system and a unique ID. Data pre-processed of heterogeneous producers managed by the FSP node is stored as an RDB. A set of linked triplet creates a graph-based RDF model, nodes in the graph can be `Subject` or `Object`, and edges is `Predicate`. Thus, converting from various data formats to the RDF format needs to preserve the integrity and meaning of data.

Algorithm 1: Converting RDB to RDF graph

```
   Input  : RDB T ∈ Ω
1  PK : primarykey, FK : foreignkey
   Output : RDF graph G
2  Procedure convertRDB2RDF()
3      S ← T.size()
4      for i ← 1 to S
5          C[i] ← ClassRDF(T)
6          S_r ← T.Row[i].size()
7          for j ← 1 to S_r
8              R ← T.Row[j]
9              Triple() ← Tp
10             Tp.Subject ← R.id
11             S_c ← R.Cells.size()
12             for k ← 1 to S_c
13                 if (ColHead(R.Cells[i]) not in (PK, FK))
14                     P ← Predicate(ColHead(R.Cells[i]))
15                     Tp.Predicate[k] = P
16                     O ← Object(Value(R.Cells[i]))
17                     Tp.Object[k] = O
18                 if (ColHead(R.Cells[i]) in FK)
19                     foreach tp in RelatedTable(T)
20                         P ← T.Name ∪ tb.Name
21                         Tp.Predicate[k] = P
22                         Triple(tb) ← ColHead(R.Cells[i])
23                         URI = tb.Subject
24                         Tp.Object[k] = URI
25             G.AddTriple(Tp)
26     return G
```

We propose an algorithm allowing the conversion of the relational database model to a new data model using Semantic Web technology, which is represented by RDF

graph with triples. In the RDB, each triplet corresponds to a row in a table, and the values refer to each column. When converted to the RDF diagram, the classes (**Class**) correspond to each table, each row in the table is represented by a statement in the form of `triples (S, P, O)` to form the RDF diagram, where a `subject (S)` is the key attribute value, `predicates (P)` correspond to the labels of each column, `object (O)` is the value at the column corresponding to the `predicate (P)`. This algorithm is used in semantic annotation services. It is responsible for converting many data types in different formats into RDF triplestore.

Algorithm 1 represents a detailed transformation from a relational database to an RDF graph. Where, Ω is a set of tables in a database D. T is the table in the set Ω.

- *PK(T)* is a set of attributes that makes primary keys in the table T
- *FK(T)* is a set of attributes that makes foreign keys in the table T
- *FK(T_i,T_j)* is a foreign key, named FK, of the table T_j that refer to the primary key of table T_j.
- *A(T)* is a set of attributes that is not in *PK(T)* and *FK(T)*. *A(T)* is corresponding in each cells in a table T: *A(T)* = `ColHead(`*R.Cells[i]*`)` (Line 13).
- *R(T)* is a set of records in the table T; R is a record, $R.A$ is a value of the A attribute in R.

In details, the algorithm first creates a RDF class for each table (Line 5). Each row of table T corresponds to a Triple (*Tp*) in RDF (Line 8,9). The *subject(S)* of *Tp* is key value of each row that corresponds to a identifier of a row *Tp.Subject* = *R.id* (Line 10). Attributes that is not the primary key and foreign key will be assigned for the *predicate(P)*, and its values is assigned for the *object(O)*. In relational databases, there are always functional dependencies (FDs) between relational tables. FDs is used as a measure to evaluate a good relationship. The FDs and keys are used to define standard forms of relationships and data constraints derived from meaning and relationships between attributes. Therefore, in order to define a FDs for a relation *r(A)*, it is defined as follows:

Definition 4.1. *Let r(A), where r is the relationship and A is the set of attributes. Let A_1,$A_2 \subseteq A$, functional dependencies $X \rightarrow Y$ (read as X define Y) is defined as:*

> $\forall$ t, t' $\in$ r *if* $t.X = t'.X$ *then* $t.Y = t'.Y$. *(Meaning: If two sets have the same value* X *then have the same value* Y*)*

The definition 4.1 enables creating a triple for functional dependencies between two relations. Where a value of a key attribute of a table is associated with a foreign key of another table, thus, assignment of subjects, predicates, and objects is executed by browsing tables in a relationship with T (Line 18-24). The last step adds the triple into G graph (Line 25).

4.2.2 Data annotation

Handling such large-scale heterogeneous data in real-time will be a key factor in building smart applications. Ontology-based semantic approaches have been used to solve these issues related to large-scale heterogeneity and interoperability [165]. The ontology describes technical details necessary for a producer registration, producer connection, and producer data provisioning and can also be used as a meta-data source by other microservice. The semantic data annotation (SDA) service processes each producer received data from the parsers, as illustrated in Figure 4.5. Afterwards, it implements the annotation process to tag these data by defined concepts from domain ontology.

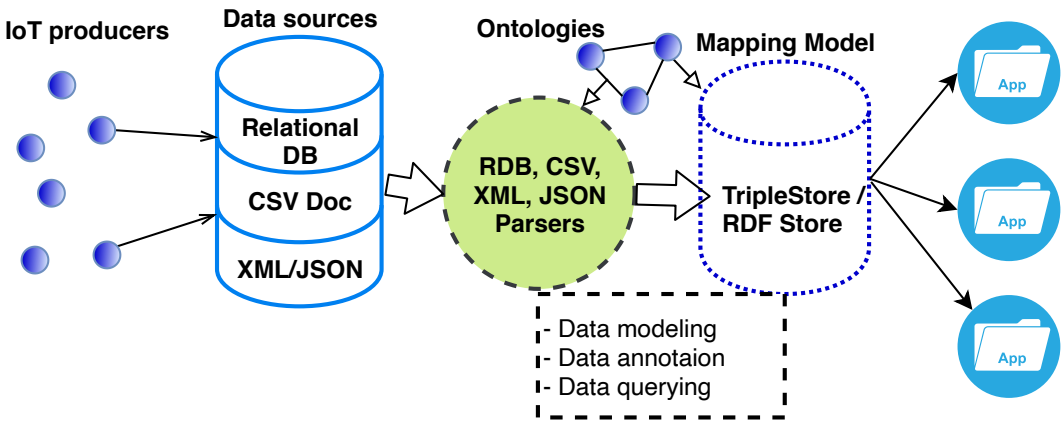

Figure 4.5: *The semantic data annotation (SDA) service*

Typically, producers are at the lowest level and are composed of limited resources that their major task is only to collect data and send it to the gateway. Producers at the gateway level have more computing resources compared to the IoT producer level. The SDA's main objective is to analyze and pre-process the raw data sent by producers by filtering out redundant data and converting them to a particular format such as XML or JSON format. Therefore, minimizing computing resources required for the annotation processes. In this stage, the SDA service is composed of three functionalities, as follows:

1. **`Data modeling`**: The filtered data received from the filtering functionality will be analyzed by parsers and performs a converting mechanism to model the composed data and define the relationship to other data objects. These modeled data are converted to a resource description framework (RDF) with ontology.

2. **`Data annotation`**: This functionality receives data from the above steps. It implements the annotation process to tag these data by concepts resulted from mapping a domain ontology and reference ontologies.

3. **`Data querying`**: Using semantic data queries, it is possible to retrieve both explicit and implicitly derived information based on syntactic, semantic, and structural information contained in data. It is designed to provide precise results (possibly the uniquely selected piece of information) or answer fuzzy and broadly open questions by matching patterns and digital reasoning.

The output of the SDA service is an RDF triplestore database file. The producers and their related components are defined in the ontology links along with their functionalities and the measured values. The ontology is a light-weight file and provides semantic annotation for the producers' data. By storing the data as such, allows the data to become more semantic, which is effective for the integration of heterogeneous and interoperable data [166]. Furthermore, it will enable consumers to collect semantically desired information, reducing the burden on developers integrating data from different data sources.

4.3 Producers and Consumers management

Any advanced and complex IoT system should support a wide variety of producers and integrate producer management functions into its architecture. IoT producers are often deployed in hostile environments. They require active monitoring, and in the event of a failure, they may need to be replaced or upgraded to maintain their operation in these environments. Producer Management is developed to protect the producers and their data by making it convenient to secure and monitor them. Producers management capabilities enable IoT developers to handle IoT producers by performing tasks such as resetting devices to the factory if necessary or applying updates to fix security issues or errors. Furthermore, in the FSP architecture, these producers' active monitoring should be handled and managed by consumers. It is defined as the software entities that

allow obtaining information related to the producers. Therefore, consumer management consists of two tasks, one that allows the end-users to register to get the data of interest. The second task provides a mechanism for filtering and transforming the data and then physically retrieving the registered end-users' data.

Because the producers might communicate with different protocols and data formats, the complexity of IoT systems increases with producers' proliferation, which makes managing IoT producers even more critical and challenging. This section introduces two services to manage producers and consumers and how they access the FSP.

4.3.1 Producer access service

To manage a heterogeneous set of producers and their different native protocols, the FSP architecture provides a `producer access service`. It represents one of the producers as a bridge for communication with the FSP architecture. In the first step, producers need to provide their profiles to register to the FSP node. A `producer profile` can be considered as a template or as a type of classification for heterogeneous producers. The type of producer, type of protocol, and type of data are provided in a producer profile. Detailed producers' information is unnecessary because it is contained in the producer ontology. In Table 4.1, we propose a set of attributes used to define a producer.

Table 4.1: *Producer profile*

Attribute	Required/Obtional	Remark
`producerId`	required	Producer identifer
`manufacturer`	required	Producer manufacturer
`name`	optional	Producer name
`location`	optional	Producer coorditate
`model`	optional	Producer model
`description`	optional	Producer description

The producer register procedure is performed, as illustrated in Figure 4.6 according to the following steps: (1) the producer starts by asking information about the domain name of FSP via a DNS service; (2,3) The DNS service responses based on the received data with the IP address of the FSP node; (4) the producer, then send their profile to register to the FSP node using the received IP address; (5) The FSP node forwards this profile to the semantic data annotation service to process and update the producer's profile into the corresponding format; (6) Finally, the FSP node confirms the successful registration and the producer can send data to the FSP node.

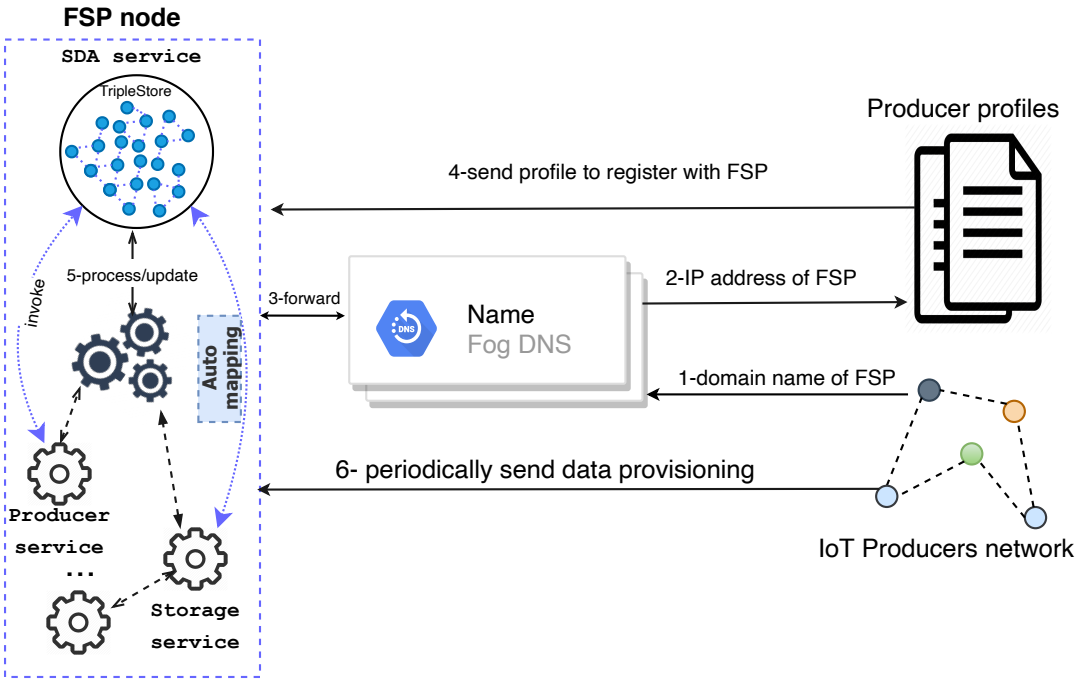

Figure 4.6: *IoT producers registration service.*

4.3.2 Consumer access service

Similar to the producer access service introduced above, it is necessary to provide a management mechanism for consumers through a service called `consumer access service`. The consumer entity is defined in section 3.4.

Table 4.2: *Consumer profile*

Attribute	Required/Obtional	Remark
`name`	required	Consumer name
`company`	required	Company name
`email`	required	Email address
`postalcode`	required	Postal code
`mobilenumber`	required	Mobile number

We use the concept of *Consumer*, which is defined in the ontology with attributes and the relationships with data objects. This concept enables access to the FSP node resources through authenticating and authorizing mechanisms. To register into the FSP architecture, we also use the profile of the *Consumer* as necessary information, as illustrated in Table 4.2.

In Figure 4.7, we illustrate a sequence diagram related to both the registration procedure and resources request to an FSP. This procedure is granted through two authentication and authorization mechanisms after being successfully registered with

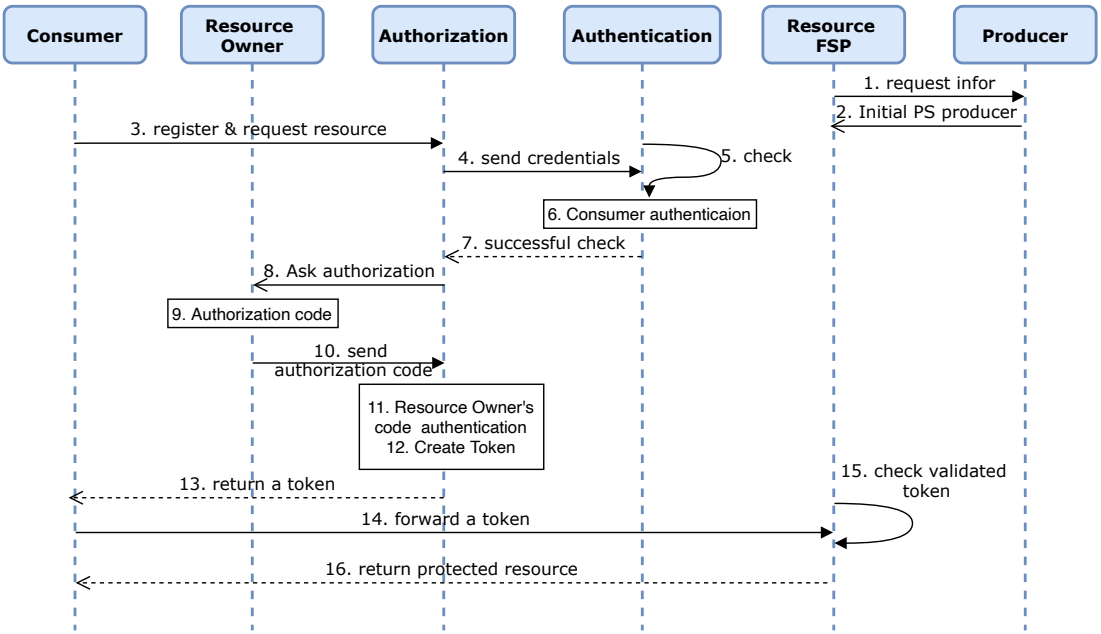

Figure 4.7: *The register and access procedure for consumers.*

consumers' profile information. The procedure starts when the consumer wants access to the FSP node data for a given producer. In steps from 3 to 13, the consumer sends a request to the *authorization service* to asks for authentication. *Authentication service* checks the validity of consumer by asking them to provide a credential (`username` and `password`). This credential is provided by a *resource owner* (i.e, the end-user who has registered via the consumer service). The *Resource Owner* grants access to the consumer by sending an authorization code. An authorization code delivers to the *Authorization service* to verify and release a token which contains the details of the consent provided to the *Consumer*. The *Consumer* forwards the token to the *Resource FSP* to check the validity of the received token and provide the protected resource.

4.4 Experimental testbed

In order to demonstrate an experimental testbed, as shown in Figure 4.8, we develop the main functionalities of the proposed architecture such as registration procedures for producers and consumers as well as providing services to manage generated data from heterogeneous producers using the ontology. The IoT producer network includes Raspberry Pis 3 Model B with an operating system (*Hypriot*) and built-in Wi-Fi. Hypriot [167] enables the execution of Docker on a Raspberry Pi and deploys services in a Docker environment. Those are also reasons why we use Docker technology to deploy distributed services. In this testbed experiment, we use Pi as an FSP gateway,

and it is responsible for processing data coming from a DHT11 sensor (the DHT11 is a basic, ultra low-cost digital temperature and humidity sensor). The DHT11's profile is also registered and sent through the FSP gateway to the FSP node by the producer service. The FSP node is configured and executed on a laptop computer. The PIs will also run a Python script to collect data from the DHT11 sensor to a storage service (PostgresSQL). A registry service's objective is to allow services to find and communicate with each other. When each service is initialized, it registers with the *Registry*. Therefore, once successfully registered with the *Registry*, the SDA service will proceed to call the producer service to get the data for the semantic annotation step. This is similar when the consumer service (access service) that wants to get the data of interest must also be registered with the *Registry* before calling the SDA service.

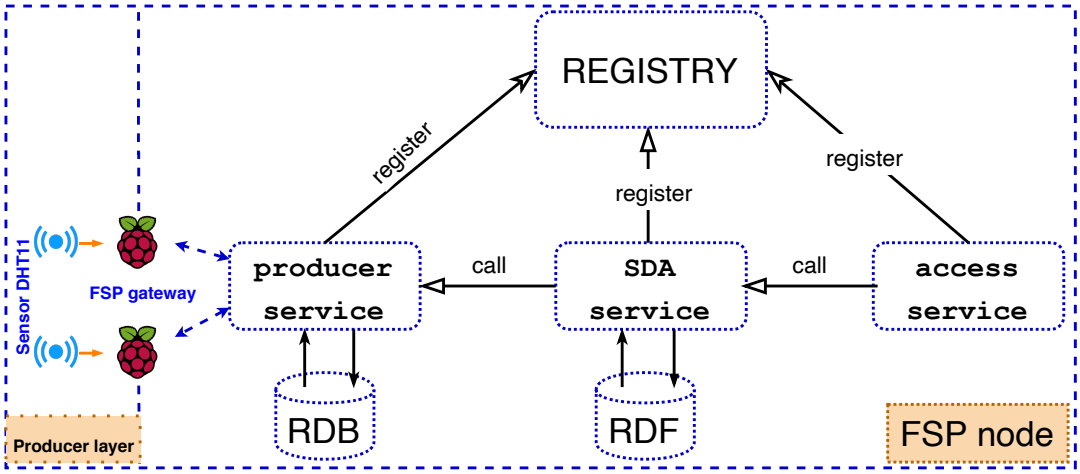

Figure 4.8: *Evaluation setup.*

We also show the example result of mapping from RDB to RDF graph through *Algorithm* 1 with related tables in a producer management relational database. The primary key of relational tables is the bolded properties, as described in the following:

PRODUCER(**producerID**, producerName, producerModel, producerManufacturer)

```
{90035229,iotproducer1,Raspberry B 3,UK }
{90035230,iotproducer2,Raspberry B 3,FR }
```

SENSOR(**sensorID**, sensorName, sensorType)

```
{sen01, DHT11, Temperature }
{sen02, DHT22, Humidity }
```

OBSERVATION(**observationID**, **producerID**, timestamp)

```
{obsen01, 90035229, 30/4/2020 }
```

`{obsen02, 90035230, 29/4/2020 }`

DETAILOBSERVATION(**observationID**, **sensorID**, location, value)

`{obsen01, sen01, RoomLiving, 20 C }`

`{obsen01, sen02, Kitchen, 70 % }`

The data converted from an RDF graph into the data model for semantical annotation is stored as a TripleStore, as shown in Table. 4.3. The namespace **`fsp = http://<fsp-microservices-ip>:<port>/fspdb/`** indicates that the elements with the `fsp` prefix belong to a triplet (S, P, O). The execution of the query data in the RDF store is supported by SPARQL language.

Table 4.3: *The RDF store triplet*

Subject	Predicate	Object
fsp/sensor/sen01	fsp/sensor/sensorID	sen01
fsp/sensor/sen01	fsp/sensor/sensorName	DHT11
fsp/sensor/sen01	fsp/sensor/sensorType	Temperature
fsp/sensor/sen02	fsp/sensor/sensorID	sen02
fsp/sensor/sen02	fsp/sensor/sensorName	DHT22
fsp/sensor/sen02	fsp/sensor/sensorType	Humidity
fsp/producer/90035229	fsp/producer/producerID	90035229
fsp/producer/90035229	fsp/producer/producerName	iotproducer1
fsp/producer/90035229	fsp/producer/producerModel	Rasperry B 3
fsp/producer/90035229	fsp/producer/producerManufacturer	UK
fsp/producer/90035230	fsp/producer/producerID	90035230
fsp/producer/90035230	fsp/producer/producerName	iotproducer2
fsp/producer/90035230	fsp/producer/producerModel	Rasperry B 3
fsp/producer/90035230	fsp/producer/producerManufacturer	FR
fsp/observation/obsen01	fsp/observation/observationID	obsen01
fsp/observation/obsen01	sp/observation/timestamp	30/04/2020
fsp/observation/obsen01	sp/observation/producerID	fsp/producer/90035229
fsp/observation/obsen02	fsp/observation/observationID	obsen02
fsp/observation/obsen02	sp/observation/timestamp	29/04/2020
fsp/observation/obsen02	sp/observation/producerID	fsp/producer/90035230

Continued on next page

Table4.3 – continued from previous page

Subject	Predicate	Object
fsp/detailobservation/ obsen01sen01	fsp/detailobservation/ observationID	fsp/observation/obsen01
fsp/detailobservation/ obsen01sen01	fsp/detailobservation/ sensorID	fsp/sensor/sen01
fsp/detailobservation/ obsen01sen01	fsp/detailobservation/ location	RoomLiving
fsp/detailobservation/ obsen01sen01	fsp/detailobservation/ value	20 ℃
fsp/detailobservation/ obsen01sen02	fsp/detailobservation/ observationID	fsp/observation/obsen01
fsp/detailobservation/ obsen01sen02	fsp/detailobservation/ sensorID	fsp/sensor/sen02
fsp/detailobservation/ obsen01sen02	fsp/detailobservation/ location	Kitchen
fsp/detailobservation/ obsen01sen02	fsp/detailobservation/ value	70 %

Table. 4.4 and Table. 4.5 are API endpoints between the FSP node and microservices. The former depicts the endpoint to be used for processing the register producer's profile in the FSP. The later is the endpoint to be applied to access the resource by an external user.

Table 4.4: *Producer register service*

Request/Response	
URL	POST http://fsp-microservices-ip:8080/fspdb/register
JSON	{"id":6,"serial":"90035229","name":"iotproducer1","model":"Raspberry B 3", "manufacture":"UK"}

Table 4.5: *Resource access service*

Request/Response	
URL	GET http://fsp-microservices-ip:8080/fspdb/resource/access
JSON	[[27, "RoomLiving", "20", 4, 6],[33, "Kitchen", "70", 5, 7]]

4.5 Conclusion

IoT is a global network of heterogeneous producers, these producers are data sources that provides changes in the environment or a certain event. The volume and the velocity of data from these producers needs to be collected, processed, and decision-making in real time through a computing paradigm that is deployed close to the edge of the network. In this chapter, we present a data management approach provided by the **`FSP architecture`**. In this data management approach, the data be managed depends on the service provision for each application. Therefore, we use both data management ways such as database-per-services and shared database with several processing techniques, including **`data collection`**, **`data filtering`**, **`data aggregating`** (for raw data), and **`data modeling`**, **`data annotating`**, and **`data querying`** (for semantic data). In addition, we have also developed a data mapping algorithm that converts raw data from diverse data sources into a light-weight storage with efficient linked data through **`RDF triplestore`**.

CHAPTER 5

A unified and semantic data model for fog computing

Keep your eyes on the stars and
your feet on the ground.

Theodore Roosevelt

Abstract

This chapter investigates a solution for handling IoT data's heterogeneity and facilitating interoperability and contextual information management. The solution comprises a semantic data model for the generic description of elements in our proposed fog computing platform, namely Fog Services Provider (FSP), to support IoT. Besides, this data model is designed for managing any device using different communication technologies that are fully described and formalized in an ontology format called `FSPontex` [23].

[23] H. Le, K. Boussetta, and N. Achir. "A Unified and Semantic Data Model for Fog Computing". In: *2020 Global Information Infrastructure and Networking Symposium (GIIS)*. 2020 Global Information Infrastructure and Networking Symposium (GIIS). Oct. 2020, pp. 1–6. DOI: 10.1109/GIIS50753.2020.9248482

Chapter content

5.1 Introduction

The last chapter introduces the fog computing architecture and the data management mechanism in the IoT environment. Fog computing is an essential part of a large and time-sensitive data management system for IoT. However, the IoT faces some challenges in managing huge amounts of data and responding accordingly. Indeed, the high growth rate of data generation in the IoT environment is a considerable challenge. According to IBM estimates, 2.5 trillion bytes of data are generated daily [168]. In particular, analysis of a healthcare IoT application with 30 million users presented data streams up to 25.000 records per second [109]. Data processing time and transmission delay in Cloud computing lead to the high-latency that affects performance, which is unacceptable for IoT applications like e-Health. Late responses regarding a critical or urgent situation can cause a risk to a person's health.

In addition, IoT producers periodically generate raw data since this can result in unusable, spurious, or repetitive records. However, the transmission of huge amounts of data leads to increased errors, packet loss, and a high risk of data congestion [169]. Besides, processing and storing these data also wastes resources without benefit. Therefore, applications with large generated data have to reduce end-to-end delay and reach real-time data processing and analysis. Furthermore, the data generated by the producers is usually presented in different formats, types, and for different applications, which poses many challenges for machines to process and understand the meaning of the data. As a consequence, the addition of semantics to the data of the IoT producers becomes an interesting trend [170]. The semantic not only provides the potential and the ability for machines to discover and deepen hidden information, but it can also provide a unified model that can describe data, suggest information, and interact knowledge between variable IoT producers. On the other hand, to achieve semantics of data from different data sources and respond quickly to time-sensitive applications, these data should be processed locally close to the edge of the network. Fog-based semantic models were introduced to provide data processing techniques at Fog nodes, such as the off-loading technique that reduces system service latency and the energy consumption of the nodes [120].

In our architecture, defined in last chapter, we use a fog computing architecture with three basic layers, including device layer, fog layer, and cloud layer. The main objective of Fog layer is responsible for temporary data storage, some preliminary processing, and

analytics. These functionalities focus on data management, processing, virtualization, service provisioning, and data analytics on data generated from IoT devices [20, 171, 172]. Moreover, the definition of data is essential for processing data. It provides to define the processing model on streams of data and enables the definition of data-driven reactive behaviors that can effectively use data models to dynamically drive the processing of data streams while leveraging resources at the edges of the network [173].

Chapter 5 is structured as follow. Section 5.2 discusses the interoperability issues in the FSP architecture and categorizes them into different interoperability levels. Section 5.3 mentions some characteristics of data that pose challenges in the development of the data model. We discuss several state-of-the-art works and select some criteria to specify data information in IoT in section 5.4. To overcome interoperability and context sharing information issues in IoT, we present a unified semantic data model by proposing the FSPontex in section 5.5. The last section 5.6 draws a conclusion about the importance of a data model for a fog computing architecture to address issues and challenges to improve the performance of the existing architectures.

5.2 Interoperability issues

One of the major challenges in designing the data information model for IoT systems is to meet interoperability requirements. Interoperability refers to the ability of two or more devices, systems, platforms, or networks to collaborate [14]. It enables the communication between heterogeneous IoT producers or other systems to achieve a shareable purpose. However, these producers and systems are fragmented in items of network access technologies, communication protocols, and data formats. Therefore, these diverse features cause communication and data sharing problems for the producers and systems in the IoT network and lead to the lack of interoperability.

As shown in Figure 5.1, interoperability of IoT environments can be considered in several different aspects such as IoT producers, IoT data, networking, consumers, platforms, and semantic.

1. **`IoT producers interoperability`**: The IoT producers consist of a diverse set of devices with different functionalities. They use different communication protocols and different access technologies (i.e., NFC, Bluetooth, ZigBee, 4G/5G, IEEE 802.15.4, LoRa, etc.). Allowing these heterogeneous devices to exchange information through various communication protocols is challenging. Further-

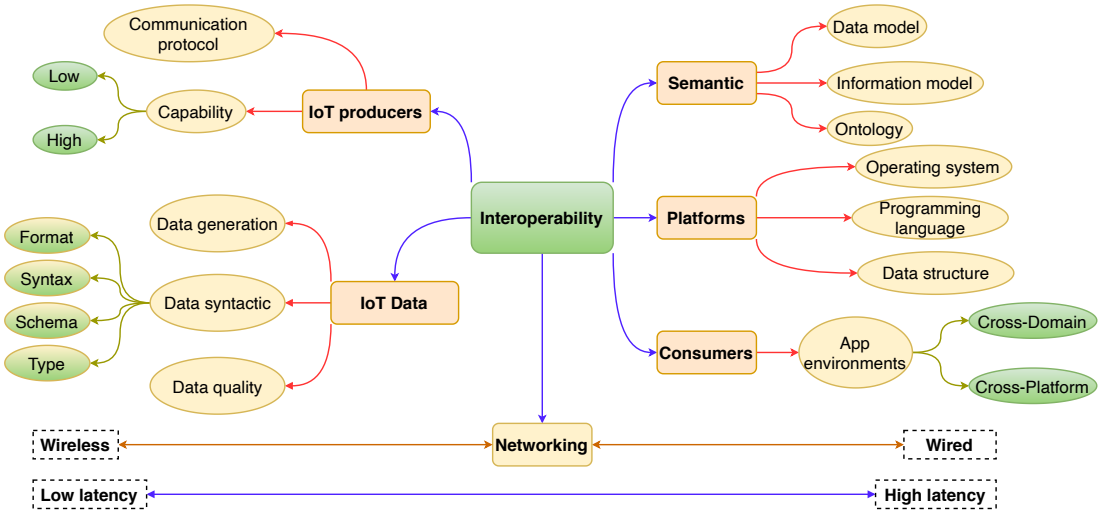

Figure 5.1: *Interoperability in Internet of things: an overview*

more, integrating a new device into any IoT platform is not feasible if there is no standard definition of communications for them. Therefore, producers may enable the integration and interoperability with supported standards to satisfy these challenges.

2. **`IoT data interoperability`**: Data generated by producers is based on different data structures and formats. In order to use such data types, the IoT system needs to provide unified data interoperability through processing raw data, eliminating redundant data, and giving syntactic data in terms of format, syntax, schema, and type data to any applications or services.

3. **`Networking interoperability`**: The heterogeneity of devices also leads to their operation in different services and providers. They operate on network technologies and communicate less reliable and fragmented short-range [174]. Moreover, due to the heterogeneous and dynamic network environment in IoT systems, ensuring seamless data exchange between different networks is very important. As a result, network interoperability should be considered to handle issues such as routing, addressing, resource optimization, and mobility [175].

4. **`Platforms interoperability`**: There is no line of sight to consistent choices between operating systems in IoT environments. The variety of programming languages and data structures poses several issues for developers to implement cross-platforms. Interoperability platforms might allow developers who use or define specific APIs and unified information models to adapt their applications

from one platform to another.

5. **`Consumers interoperability`**: The complexity of data sources from heterogeneous devices in distributed IoT environments leads to their applications, which need to be implemented more complicated to analyze and process the large volumes of data. Furthermore, integrating functionalities in pre-existing applications or new functionalities in different application domains also poses many software development challenges. Interoperability in consumers enables functionalities in various applications to exchange information, share resources, and use the same protocols. For example, in the java programming language, interoperability is considered a feature that allows running and executing on any program with a java virtual machine (JVM) [176].

6. **`Semantic interoperability`**: According to the W3C [177]: *Semantic interoperability is about enabling different agents, services, and applications to exchange information, data, and knowledge in a meaningful way, on or off the Web.* It provides the ability to connect understanding and sharing between cross platforms and cross-application domains by providing data models, information models, and semantic web technologies with the help of ontology.

This chapter addresses issues related to requirements to achieve semantic interoperability by providing a unified and information model for the FSP architecture.

5.3 IoT data characteristics

IoT producers are widely diverse primarily because of the different hardware and operating systems used. The heterogeneity of the information provided by the underlying producers is one of the most highlight features of the IoT domain. These heterogeneous producers communicate through a variety of protocols at low power networking (ZigBee, ZWare, LoRaWAN, and Bluetooth), or traditional networking protocols (Ethernet, Cellular, WiFi), and even application protocols such as CoAP (constrained application protocol), MQTT (message queuing telemetry transport), XMPP (extensible messaging and presence protocol), and AMQP (advanced message queuing protocol). These protocols are designed for domain-specific applications with particular features. Moreover, these devices are expected to be deployed in different areas of applications to observe the environment and generate enormous data continually—differences in the data formats,

types, or syntax result in interoperability issues between the applications. Due to the lack of common understanding between different producers and platforms with resource constraints in IoT as shown in Figure 5.2, it is not easy to understand the exact meaning (semantics) of the exchanged content. Thus, much work has to be done to ensure interoperability.

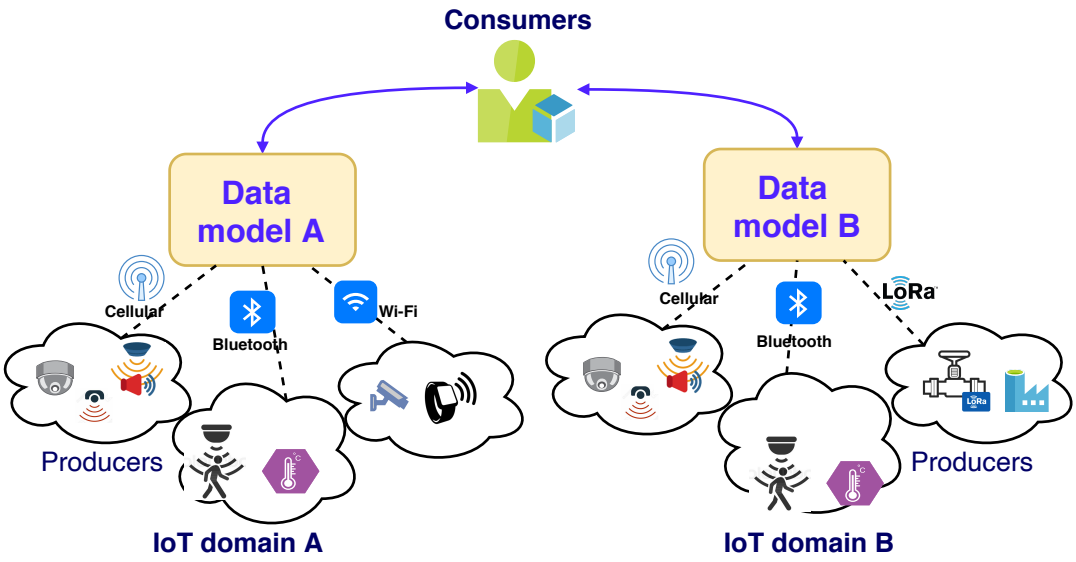

Figure 5.2: *Challenges in across domains*

Data is a valuable resource in the IoT paradigm used to provide insights and a means of communication. Data collected from the global deployment of smart producers is fundamental to making intelligent decisions and providing services. Therefore, if the data is of poor quality, these decisions can not be accurately guaranteed. Moreover, data quality is also an important requirement for any consumer to provide IoT services for end-users. It also depends on the data characteristics that need to be defined and considered in the data management process. The IoT data characteristics are introduced in [178] and are associated with data in the IoT, including heterogeneous characteristics such as uncertain, erroneous, noisy, distributed, voluminous, and context-dependent characteristics such as smooth variation, continuous, correlation, periodicity, and Markovian behavior. According to to [168] the IoT data characteristics are classified into three categories, including data generation, data quality, and data interoperability. The data generation characteristics include *velocity*, *scalability*, *dynamics*, and *heterogeneity* that might be increased in speed and data volumes. Then the data are collected in storage and processed relying on analytic requirements. The data quality characteristics need to be analyzed to filter and reduce data redundancy and ambiguity. Besides,

the data interoperability enables the combination of data from different types of data sources as well as the data structure used in any exchanged services and information models between heterogeneous IoT systems. Therefore, the major characteristics of the IoT data and related issues might be handled in the **FSP** data management process to achieve high quality and applicability requirements. In the following, we consider some characteristics that motivate for designing a unified data model:

1. **Heterogeneity**: Managing distributed heterogeneous producers connected to the internet poses significant challenges in providing services to respond quickly to them. Furthermore, the data generated from these diverse producers can be very different in terms of data structures and formats [179, 180].

2. **Interoperability**: IoT systems have always required scalability, especially in advanced and time-varying systems to adapt to new components, services, and applications. These systems usually come with a specific data model and standards such as IPSO [181], oneM2M [182], but are not compatible with each other. Therefore interoperability of data and information is essential to integrate data from different types of data sources to obtain large aggregations of useful data from different cross-domains [14].

3. **Semantic**: Data generated by producers is often raw data and lacks semantic. The main objective of an IoT data model is to enhance semantic interoperability. Inspired by the Semantic Web technologies, to enable machines that are readable and understandable, i.e., a shared understanding of what the data means, the raw data may need to be injected with some information and relationships between data objects through semantic so that they can understand data of human and share knowledge with other applications [168, 183]. It is performed with the help of ontologies for the formal specification of the defined concepts' semantics and interaction.

4. **Scalability**: A large and heterogeneous number of IoT producers with a high data collection rate that can be available in different scenarios may generate a tremendous amount of data [169]. They require the provision of services and applications to meet adaptability as the environment changes. Moreover, the scalability in the context of IoT is also considered in two aspects such as vertical scalability and horizontal scalability [184]. Vertical scalability is referred to as

scaling up, which provides the ability of existing FSP nodes by adding more resources to support these producers. The horizontal scalability is referred to as scaling out, which provides the capacity of IoT systems by federating FSP nodes to be interchanged.

5.4 IoT data model: state of the art

The traditional IoT service model's paradigm provides the software agent with raw sensor data collected by the devices. This raw data does not contain semantic annotation and requires a high manual overhead to implement practical applications. This diversity raises several interoperability issues. Because of these service providers' approaches, the IoT domain can be viewed as vertical silos of different IoT applications without horizontal connectivity between them. One of the main challenges is the lack of **`interoperability with independent services`**, especially for applications capable of benefiting from multiple devices, thus risking the widespread adoption and acceptance of the IoT domain. One possible solution is to achieve semantic interoperability in a heterogeneous IoT environment is to use semantic annotation of raw sensing data using; for instance, the ontology approach. In general, all semantic model features are deployed centrally in the cloud. While the cloud's computing capabilities are essential, the response time for user requests is not always as fast as required by the applications. Besides, semantic annotation algorithms to such amounts of data at the centralized layer lead to significant consumption of resources, which probably affects their performance. Consequently, a high-level fog computing architecture is required to cope with heterogeneous IoT devices generating large and diverse data types. Concurrently, providing a unified data model to support interoperability in IoT by using Semantic Web (SW) technologies.

In the literature, fog computing-based architectures and applications are available and can be considered as potential solutions to the heterogeneity problems [18, 21, 185]. However, semantic-based approaches are not commonly applied in the Fog computing architectures. There are many approaches in progress to define IoT data models [186]. These approaches are divided into some criteria to specify data information in IoT. The criteria are presented as follows:

- **`Modeling (Mol)`**: It is responsible for converting the data model into a predefined format. The modeling process helps to facilitate the interpretation of the data model. There are several modeling techniques used in the literature, such as

key-value modeling (kv), text-based modeling (tex), ontology-based modeling (ont), object-oriented modeling (oob).

- **`Data distribution (Dis)`**: Each IoT system has a policy to distribute its data. Data distribution is a statistical data processing phase where collected data is aggregated and shared with the end-users through these policies. In the IoT perspective, there are two ways of data distribution [186]: static (sta) and dynamic (dyn). In the first method [187], data needs to be stored and predefined its information into a list (i.e., a distributed database system or a schema) for applying to the specific application. On the other hand, the dynamic method [188] may use several reasoning rules to define requirements depended on context information.

- **`Interoperatbility (Int)`**: The interoperability issue is different from most IoT systems that only contain an application-specific system (i.e., a vertical domain); shared IoT systems usually have a heterogeneous characteristic due to the available ability of diverse operating systems, data structures, format, and access mechanisms. Currently, there is no standard format for data annotations, and it isn't easy to create a format with a unified data model for all IoT domains. Therefore, cross-platform interoperability is enabled by federating heterogeneous domains to build horizontal IoT domains. In order to ensure complete interoperability, IoT applications should categorize into two parts: full interoperability (ful), and partial interoperability (par) [186]. The full interoperability deals with both data format and communication technologies. The partial interoperability only provides some predefined data information and similar characteristics between objects in IoT domains.

- **`Scalability (Sca)`**: In IoT environments are characterized by a large number of concurrent requests, enormous amounts of data information must be processed and handled efficiently and with an acceptable response time. In addition, this is also caused by a huge number of different communication technologies. Consequently, analyzing and sharing data models for IoT application domains can minimize the processing and the communication overhead in IoT platforms.

- **`Architecture model (Arc)`**: The approach for the architectural prospect of data models can vary depending on the characteristics of the application domain. As a result, considering the IoT environments, some architectures are more appropriate

than others. There are three main architectures for supporting to design data models in IoT environments: cloud-based (clo), fog-based (fog), and decentralized-edge (dec). In the cloud-based, computational requirements are mostly done in the cloud, and it depends significantly on the network performance. The fog-based places the computing close to the IoT producers, and its processing capabilities still need a central point of computation that acts as fog nodes. Finally, in the decentralized-edge, the computation is executed by some devices. These devices may lack some features such as processing power and storage capability due to design limitations. However, in some systems, it will not depend on the two data models mentioned above. It can adapt (adt) itself to the new application environment, which means it may have a combination of cloud, fog, or edge-based methods.

- **`Context information (Con)`**: Context in IoT environments is commonly represented semantically and used to define the status of an environment entity for a specified domain such as smart city, smart home, smart health, or vehicles. The information can be easily understood by humans and can share knowledge for other IoT domains by providing a unified data model. Therefore, to provide context information, we use some techniques (e.g., rules that combine with reasoning and ontology).

The criteria selected above are based on several research topics related to a data model for IoT. The IoT is the key driver to develop a comprehensive, unified data model for Fog computing. Consequently, the description of the data from these heterogeneous producers plays an important role in aggregating them into a single **`modeling`** which needs to be taken into account. Another important issue to consider is that IoT **`data distribution`** is frequently geographically distributed. The criterion regarding **`interoperability`** is chosen because the changing environment should also be considered when developing a fog computing architecture. To ensure scaling up and out capabilities in an IoT environment, the **`scalability`** is the essential criterion. Developing a fog computing architecture with a unified data model to adapt to diverse applications and the ability to understand context information in the IoT environment also needs to be addressed and analyzed carefully by criteria **`architecture model`** and **`context infromation`**. The selected criteria for the taxonomy with the related works are shown in Table 5.1.

Table 5.1: *Data models taxonomy*

Works	Mol	Dis	Int	Sca	Arc	Con
SSN-XG [189]	ont	dyn	par	✓	adt	-
Xue et al [190]	ont	dyn	par	-	-	✓
IoT-O [191]	ont	dyn	ful	-	dec	-
IoT-Lite [192]	ont	dyn	par	✓	clo	-
M3/M3-Lite [193, 194]	ont	dyn	par	-	adt	✓
LiO-IoT [195]	ont	dyn	ful	✓	adt	-
CS-Sharing [196]	kv	dyn	par	✓	dec	-
Bluewave [197]	tex	sta	par	✓	cen	-
PSW [198]	ont	dyn	ful	✓	adt	-
SCS [199]	-	dyn	ful	✓	fog	✓
SE-TSDB [200]	ont	dyn	ful	-	adt	-
FIESTA-IoT [201]	ont	dyn	ful	✓	adt	-

The symbol (✓) is used to indicate that the research is proposed to support the stated criteria.

The different specifications usually address various application domains and distinguish in terms of characteristics, definitions, terminology, and scope. In order to solve the heterogeneity and interoperability in IoT producers, we proposed a simple fog-based unified and semantic data model in our previous work, in Chapter 3. In this extended version, we propose a semantic-based annotation approach to handle such large-scale heterogeneous data and processes for latency-sensitive applications by using a common unified ontology. According to [202], the objective of the ontology is a formal, explicit description of concepts in a domain of application that has the ability of reasoning, also known as classes (sometimes called **concepts**). The properties of each class describe different features and attributes of the class (called **properties** or **roles**). This unified ontology model should be described to meet the concepts in both horizontal and vertical silos, as illustrated in Fig 5.3. The concepts that are constituted the horizontal silo (**core concepts**) of semantic description are used for an IoT system, while concepts (**specific concepts**) that are established for specific applications or specific domains are provided for the vertical silo.

To develop an ontology for the IoT, we use 4WH1 methodology [203]. This common methodology describes basic events or situations in a physical environment. To define core concepts, the authors have proposed to answer five basic questions, including four Ws (What, Where, When, Who), and one H (How). Based on this approach, we present these types of questions to define core concepts as follows:

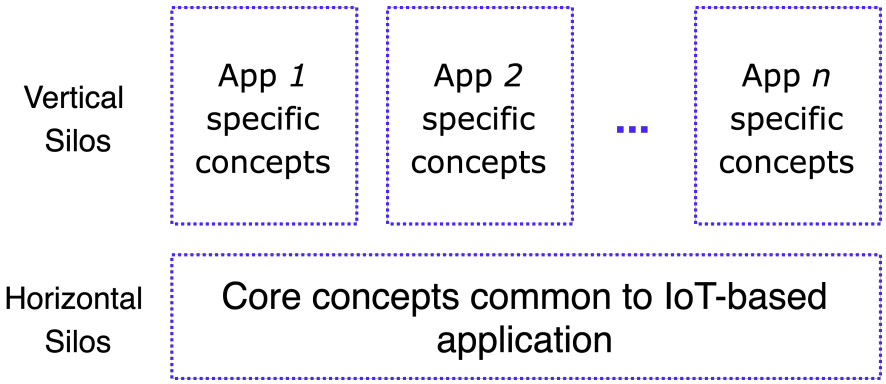

Figure 5.3: *Core concepts for IoT domains [203]*

- **`Who`**: who is providing the information required to develop the IoT application? To answer this question, concepts need to be defined to identify how the data source is generated for IoT applications. This data source is **`Producers`** which is managed by the FSP architecture (or Platform). Thus, an IoT ontology must include concepts **`Producers`**, and **`Platform`**

- **`What`**: what are the conditions required for the source to collect data? Answering this question requires the ontology to define concepts related to the producers' deployment **`Context`** (i.e., deploying applications of producers, whether the operating environment is static or dynamic).

- **`Where`**: where should the data come from? The generated data sources must always determine their geographical location. Therefore, the concept of **`Location`** is essential in the IoT ontology.

- **`When`**: when should the data collection happen? Data collected from producers should be determined through **`Timestamps`**. These parameters need to be defined and supported in several formats of time through the concepts of **`Observation`**.

- **`How`**: how should the data be exposed to the developer for building the IoT application? The generated data needs to be processed and analyzed through FSP architecture services. Therefore, the answer to this question should be supported by the **`Services`** concept.

5.5 Fog Services Provider ontology context (FSPontex)

Producers are becoming smarter due to the growing popularity of the IoT. This leads to more pervasive computing environments becoming smarter as well. These environments

are considered one of the potential research fields in ubiquitous computing technology. To manage data as well as to support efficient context information management in various IoT application domains, ontology-based methods can be considered as the backbone for these management [204]. It has an essential role in understanding the contexts of environmental entities (i.e., person, place, application, or device), and it is also represented for semantic information [205]. The objective of data management and dissemination information is to provide an efficient ontology that allows handling heterogeneous data and the ability to share context information with services related to different application domains. This section provides more detailed data information to describe FSP architecture elements. As illustrated in Figure 5.4, these elements are core concepts that associated with **`producer`** and **`consumer`** elements a to build a *semantic data model*, namely **`FSPontex`** that enables the sharing of context information and interoperability criteria.

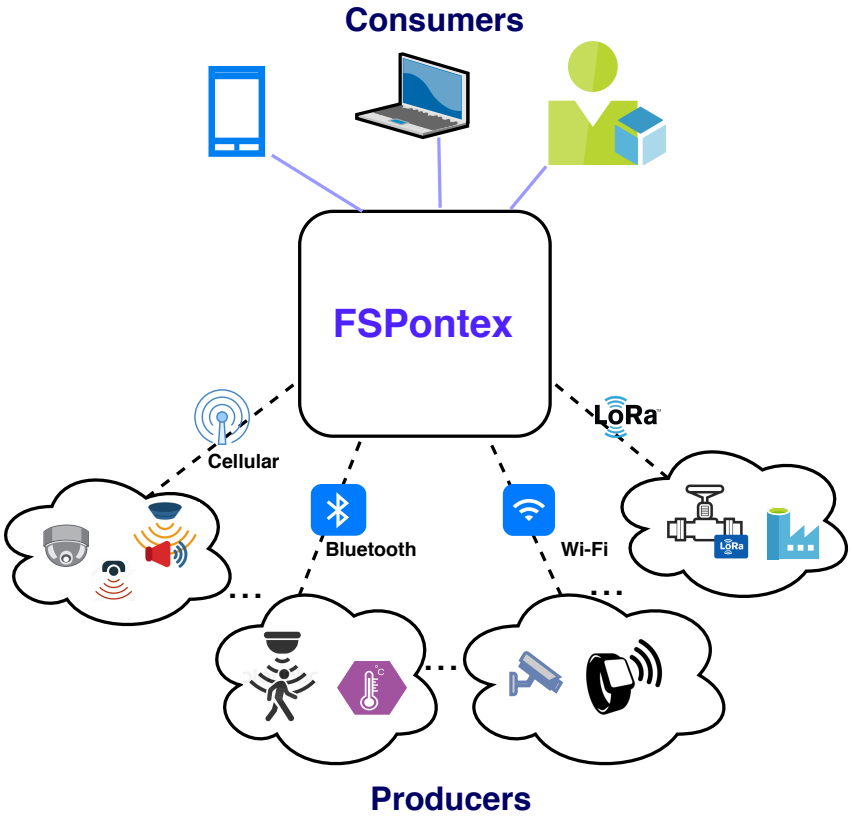

Figure 5.4: *The FSPontex model.*

5.5.1 Core concepts

Implementing a completely unified ontology for IoT could be a challenge because of the existence of more than 200 domain ontologies [206]. There are specific concepts for most ontologies inherent to IoT application domains, while all IoT platforms share some of the concepts used. To provide a contextual data information model, we reuse the ontology based on the FIESTA-IoT ontology [201], which is an existing unified ontology for IoT. It provides most of the concepts identified in [203] with criteria on core concepts when designing an ontology. However, it lacks concepts for annotating context information to share knowledge and has a limited notion for *Software*, *Hardware*, and *Communication*.

From defined concepts of existing ontologies, we borrow the concepts of *Sensor*, *Service*, *Location, Observation* and build an ontology called **FSPontex** and extend them with new core concepts, including *Producer*, *Consumer*, *NetworkInterface*, *Communication*, *Network*, and *ContextEntity* to describe objects in heterogeneous environments and the interoperability of different IoT application domains.

1. **ContextEntity**: is the root entity to describe the contextual information of a single node. It allows the specification of information for a particular context that provides context-sharing capabilities for different application domains such as smart cities, smart health, or smart home, etc. This concept includes several attributes to characterize it such as `contextId`, `contextName`, `contextType`. The **ContextEntity** is a superclass of the **Platform** class that represents other classes such as *Producer* and *Consumer* will connect to the FSP architecture. In the FSP architecture, the *Platform* class has a role as an **FSP node**.

2. **Platform**: is a core component of FSP architecture. It represents a fog node or an FSP node that provides services, components, and modules for *Producer* and *Consumer*. It is also responsible for managing data information and sharing context information with other entities. Moreover, this element enables the model of processed data to make semantic contexts and provide interoperability at the data annotation level through concepts, attributes, and the relationships between data objects.

3. **Producer**: is designed to perform a particular task such as event detection or changes in its environment. It consists of two elements: hardware, on behalf of

a physical entity like *Sensor*, *Actuator*, *SmartPhone*, and software is a computational data element representing a physical entity, including *API*, *Virtual Device*, *Webservice*, or *Microservice*.

4. **`Consumer`**: is a software entity that enables analysis, process data generated from *Producers*. It can be a user's applications that provide some authentication and authorization mechanisms to allow access from outside through a *Service* concept class.

5. **`NetworkInterface`**: is the point of interconnection between a *Producer* and a private or public network. This concept enables to manage heterogeneous communications of *Producer* either both low-layer networking (ZigBee, Bluetooth, LoRaWAN, etc.) and high-layer networking (MQTT, CoAP, XMPP, or AMQP) protocols.

6. **`Communication`**: describes the state of a Producer's communication via protocol stack. The state can be in terms of the general quality, efficiency, security, frequency, or availability of communication to provide appropriate contexts and share information for other applications.

7. **`Network`**: this concept provides the state of IoT platform's network based on exchanged *Communication* contexts as well as from deployed network management for IoT producers. Increased awareness of the network's state can offer more effective solutions—especially those resulting from inherent constraints (e.g., resource constraints).

5.5.2 Context information

The **`context information`** is used to define the state of an environment entity (application, device, place, person) and to characterize its situations with semantic information that is easily understandable by humans when reading it [207]. In section 5.4, we also introduce the 4W1H approach to characterize the situation of classes used in the FSP architecture. This characteristic allows data generated from *producers* to be analyzed to provide a high-level data (semantic) and turn them into contextual information. The **`FSPontex`** is built on Semantic Web technology with the help of ontology, not only capable of providing contextual information based on **`profiles`** of *producers* and *consumers* that use to interact between concepts in the data model. Furthermore, the

FSPontex can play a vital role for a **generic ontology** that may be utilized as a formal context repository to facilitate reasoning for specific application domains. This reasoning technique is important in providing shared context information [186]. It allows using inference rules, logical data association, relationships between concepts, and annotating these relationships.

In the **FSPontex**, the **ContextEntity** concept is used to address different contexts, such as a **generic context** and a **specific context**. The *generic context* is used as a data store that stores *profiles* with basic attributes that can be referenced for specific applications. It can be deployed in any smart context-aware system, while the specific context is used for a specific smart domain, as a smart city, smart health, etc. The **contextType** attribute provides information to identify these contexts. To provide a semantic data model with the ability to provide context information, we propose a step-by-step data management mechanism described in section **??**. Raw data is collected and processed from *producers* through data collection, data filtering, data aggregation steps; then it is converted to semantic data (or context data) to form a semantic data model with some steps: data modeling, data annotation, and data querying by using a modeling technique with the help of ontology.

In this chapter, we focus on the modeling step, and it is also essential to convert the described data into a predefined format (RDF or OWL formats) that enables the sharing of context information to other smart application domains. An example of sharing context information from a producer profile in a specific application (smart home), as shown in Table 5.2. The data is generated by a *producer* to measure the room temperature in JSON format. This data is then handled and described through the *FSPontex*, and processed by applying a reasoning technique (provides `temperature`, `time`, and `location` status that is utilized to make the decision) to provide the context information of the room's status. The context information is stored in the generic context used to share it with other applications.

5.5.3 Components of the FSPontex

Handling such large-scale heterogeneous data processing and organizing them by their relationships with each other might be a key factor in building smart applications. Ontology-based semantic approaches are capable of mitigating heterogeneity issues and promoting interoperability between IoT systems [208]. The benefits of this semantic

Table 5.2: *Sharing specific producer's profile into a generic context*

Specific Producer Profile	Generic context
`name: raspberry`	
`manufacturer: UK`	
`model: B`	HighTempLevel
`description: room temperature`	InsideE206Room
`serialnumber:90035229`	AffernoontPeriod
`temperature of room: 36 C`	
`date time: 2020-07-29T14:23:45:152Z`	
`longitude: 41.24336`	
`longitude: -0.593154`	
`altitude: 2sdFloor`	

ontology-based data management allow for a common understanding of the meaning of *producers*. Furthermore, the common understanding also allows different IoT systems, data sources, and applications to communicate more efficiently and productively. This can be obtained by modeling data of *producers* based on shared vocabularies that are used to interpret by *consumers* in different IoT systems. This modeling approach is called semantic annotation. The semantic annotation for data collected by *producers* is an essential step towards developing smarter and interoperable IoT applications [209]. The principal information model focuses on modeling data generated from producers and the relationships between elements in the FSP. Each element is described by several components in the **FSPontex** such as *classes*, *properties*, and *relationships*. These classes reflect the core concepts as mentioned in the previous section. *Properties* are the parameters used to describe producers' characteristics. *Relationships* are the ways in which classes and instances of a class can be related to one another.

As illustrated in Figure 5.5, the data is central to the FSPontex model that needs to be processed. Therefore, the **producer** class is considered to represent a data source where a data stream originated. This class needs to be linked with other classes. However, the classes alone will not provide enough information to answer the questions mentioned in section 5.4. On the other hand, once classes have been defined, it is necessary to describe these classes' internal structures. This internal structure can be properties of classes, including **datatype property**, **object property**, and **annotation property**, as follows:

- **Datatype property** refers to sets of data values. It contains data values that related to literal data such as boolean, strings, numbers, datetimes, etc. The datatype is a kind of data range, which allows them to be used in restrictions [210].

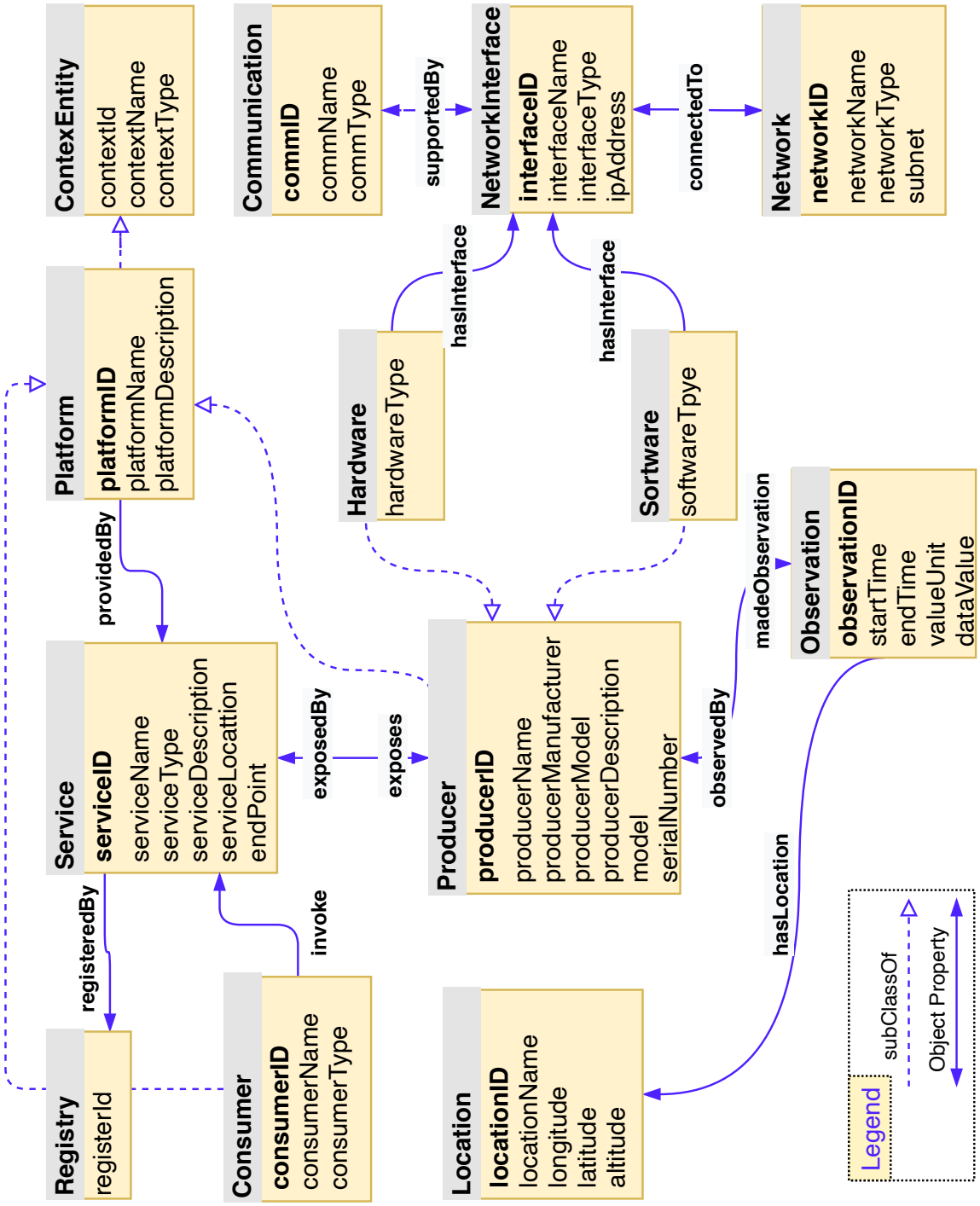

Figure 5.5: *Data model for Fog Services Provider*

If we denote A as an attribute and **xsdIRI**(A) is the IRI (Internationalized Resource Identifier) of the *xsd* data type (*xsd* is a data type of XML schema - XML Schema Definition). it corresponds to the data type of attribute A. The syntax for using data types in the ontology is `Datatype := IRI`. For instance, the datatype `xsd:integer` denotes the set of all integers. For example, a `producer` class has attributes and datatypes such as:

`producerId xsd:int`,

`producerName xsd:string`,

`producerManufacturer xsd:string`,

`producerModel xsd:string`,

`producerDescription xsd:string`,

`serialNumber xsd:byte`,

`isMobile xsd:boolean`

- `Object property` is made to represent relationships between objects. Unlike databases and object-oriented programming languages, properties in OWL are defined independently of classes. When they are used, objects are identified as belonging to the class (domain) and value (range) of the properties. For example, a described relationship between `Producer` and `Observation` is a complementary combination by defined object properties that are `observedBy` property and `madeObservation` property, respectively.

- `Annotation property` allows adding annotations on individuals, class names, property names, and ontology names. It is responsible for explaining the relationship between the classes and establishing the relations between the data required for efficient interoperability processing for IoT applications. It may make the information more readable for data analysts and the object of an annotation property must be either a data literal, a URI reference, or an individual [211]. Fig. 5.6 illustrates an instance of a data model with a combination of object properties, datatype properties, and annotation properties. The below shows the *Annotation Property* for two classes `Producer` and `Observation`:

```
<owl:AnnotationProperty rdf:about="&fsp;observedBy">
<rdfs:label xml:lang="en">observed by</rdfs:label>
</owl:AnnotationProperty>
```

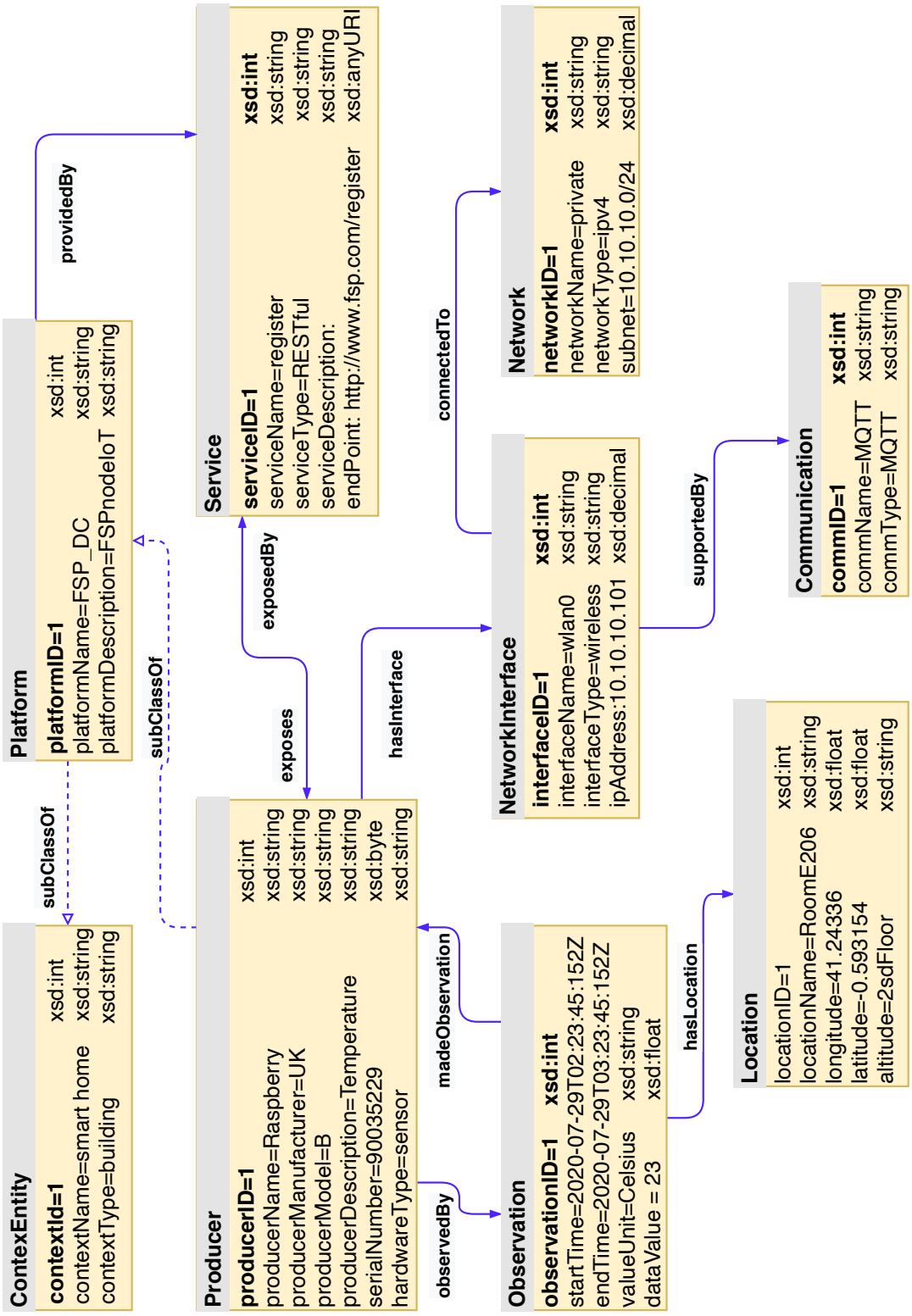

Figure 5.6: *An example of a data model annotated with the proposed ontology*

```
<owl:AnnotationProperty rdf:about="&fsp;madeObservation">
<rdfs:label xml:lang="en">made observation</rdfs:label>
</owl:AnnotationProperty>
```

5.5.4 Solution for interoperability

In order to build a unified data semantic model that provides interoperability, we first need to solve the interoperability problem by dividing it into three different levels: network interoperability level, communication interoperability level, and data interoperability level.

1. **`Network interoperability`**: As mentioned in section 4.2 of chapter 4, **`Producers`** and **`Consumers`** can access the FSP architecture by using their network technologies to connect through the **`access service`** that is provided by the FSP architecture. The **`producer service (PS)`** ensures network interoperability on **`FSP gateways`**, accordingly with their **`producer profiles`**, which define a producer's network, hardware, and software technologies, as shown in Figure 5.7. The related management information for producers and consumers is stored as **`METADATA`**, which is used to provide parameters of one network to the parameters of another network.

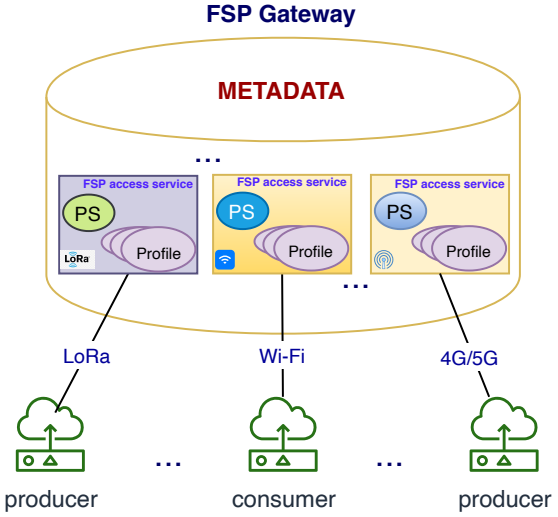

Figure 5.7: *FSP gateway for network interoperability*

2. **`Communication interoperability`**: At the communication level, interoperability can be realized into high-level (application-level) network protocols. There are many protocols at this level, such as CoAP, MQTT, XMPP, and AMPQ that

have introduced and become standards to provide interoperability [155, 212, 213]. These protocols have special characteristics and message architectures useful for different types of IoT applications and services. In the FSP architecture, message exchange processing is performed by the **access service**. Besides, the scalable FSP architecture also operates flexibly and independently with these protocol standards to provide integration and transmission of messages between various protocols.

3. **Data interoperability**: The data generated from **producers** is usually raw data and contains no semantics, as well as interoperability for other applications or services. To provide data interoperability, we propose to process these data through six steps: *data collection*, *data filtering*, *data aggregation*, *data modeling*, *data annotation*, and *data querying*. Where, the first three steps are responsible for analyzing and pre-processing raw data by filtering out redundant data, removing duplicated and unnecessary data. In order to achieve data interoperability, data needs to be described with their *concepts*, *properties*, and *relationships* with the help of the ontology, as mentioned in section 5.5.3. it is performed in data modeling step. The data annotation step implements the annotation process by tagging data processed by **annotation property**. Finally, the process of obtaining semantic data is done through data querying that is supported by the SPARQL language [214].

We present about the data interaction in the FSP architecture, as illustrated in Figure 5.8. Services of the producer layer are **register service**, **data collection service**, and **command service**. *Producers* can join or leave the FSP network dynamically via this service. This service is necessary for sending and receiving messages. Registered producers collect information from the environment and generate raw data and send it to the FSP layer through the FSP-producer interaction. The authentication service searches the list of registered producers on storage and finds related key and ID to authenticate the incoming messages. Received data will send to a **data annotation service** after pre-processing data with some techniques such as filtering, reducing, and removing unnecessary data to convert processed data and combine with the FSPontex ontology and other ontologies to create a **semantic data model**.

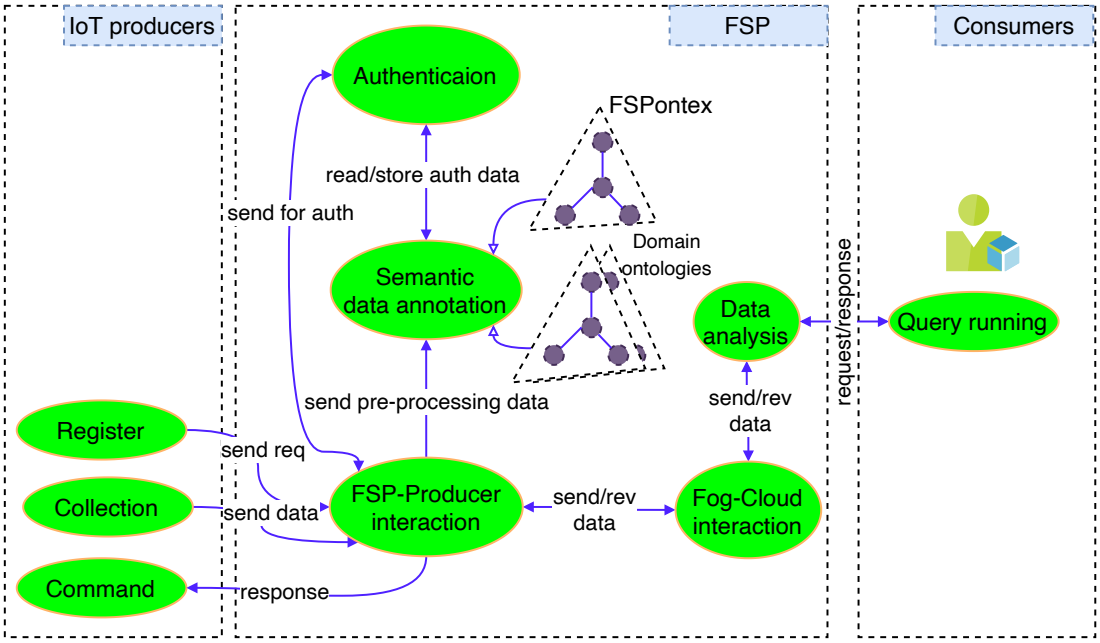

Figure 5.8: *Data interaction in the FSP architecture*

.

5.5.5 Description language and query in FSPontex

The most used and well-known language to describe ontologies is Ontology Web Language (OWL) proposed as a standard by W3C's Web Ontology Working Group [130]. In order to describe data information, we use the Protégé tool [Protégé, version 5.5.0]. It is developed by the Stanford Research Center based on Java language. One important advantage of Protégé is the higher compatibility with different ontology description languages such as WebOnto [215] and OntoEdit [216]. Besides, Protégé enables users to build, edit classes and properties, different import ontologies, visualize ontologies in various techniques, reasoning ability, create rules, and execute queries using a configurable graphical user interface (GUI). Fig. 5.9 illustrates the relationship between classes in the FSPontex.

By providing semantic annotation to producer data, the **`FSPontex`** ontology is stored in a single *RDF* or *OWL* file that is light-weight and supported by SPARQL query language. This language allows users to query information from databases or any other data source that can be mapped to a "key-value"data format, that based on the RDF specification of the W3C. Therefore, all data will be stored as a set of `subject-predicate-object` triples. This is similar to using the "document-key-value"terminology of some NoSQL databases, such as MongoDB. The following query returns names and a serial number of every *Producer* in the dataset:

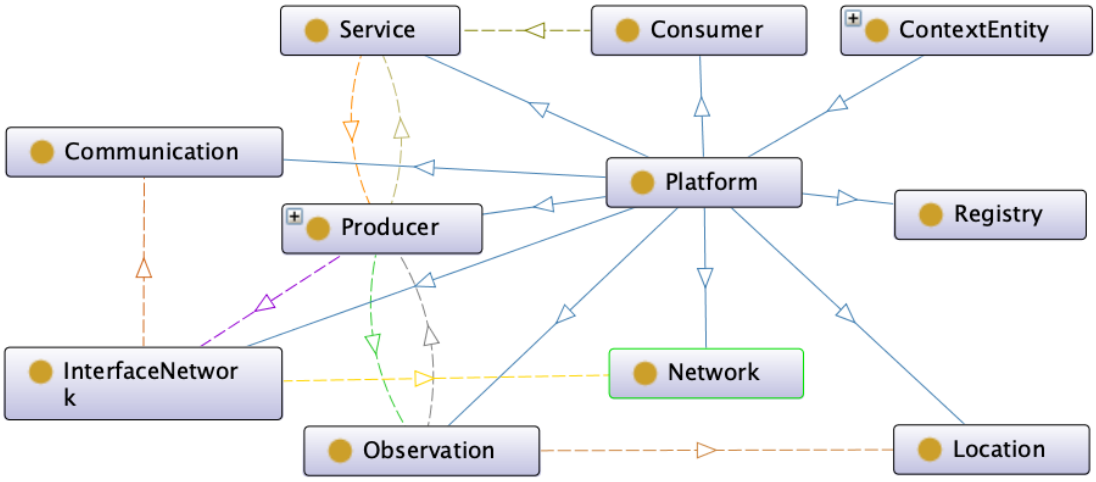

Figure 5.9: *A visualization functionality of the FSPontex*

```
PREFIX fsp: <http://www.fsp.com/fsp/ontologies/fspontex#>
SELECT ?producerName
?serialnumber
WHERE
{
   ?producer p                      fsp:Producer .
   ?producer fsp:producerName ?producerName .
   ?producer fsp:serialnumber   ?serialnumber .
}
```

This query combines all of the triplets together with a matching `subject`, in which the type `predicate`, "p", is a *producer* `(fsp:Producer)`, and the *Producer* has one or more names `(fsp:producerName)` and serial number `(fsp:serialnumber)`.

5.6 Conclusion

Interoperability in IoT is difficult to achieve due to its heterogeneous nature and the lack of standard architecture. The available IoT ontologies are not adequate for semantic interoperability when an interaction between devices is limited to a specific domain of IoT. However, ontologies designed in various contexts cannot validate the semantic interoperability between heterogeneous IoT devices. This chapter presents a view on how to achieve interoperability at the data and knowledge levels to support smart applications in IoT domains. We have proposed a new semantic data model for fog computing platform with help of the ontology, named *FSPontex*, and defined

several new concepts to build a complete ontology, including *Producer*, *Consumer*, *Communication*, *NetworkInterface*, *Network*, and *ContextEntity* which enables sharing contextual information between IoT applications. The FSPontex is an easily accessible and understandable semantic model and can apply to IoT-supported platforms. In the future, we will need to be conducted to finalize the validation of our ontology and to investigate its more complex test scenarios.

Chapter 6

Conclusions and Perspectives

> *Nothing in life is to be feared, it's to be understood. Now is the time to understand more, so that we may fear less.*
>
> Marie Curie

Chapter content

6.1 Conclusions

This thesis presents our research on designing a services architecture to support a heterogeneous set of devices IoT devices that use many different types of communication technologies based on the Fog computing paradigm. Fog computing is considered as one of the most suitable solutions to support the expansion of services from Cloud computing to the edge of the network to provide IoT devices with an effective response in terms of low latency and real-time interactions. By taking advantage of the outstanding features of Fog computing such as geographical distribution, heterogeneity, local data processing capabilities, interoperability, mobility support, and optimize energy efficiency, we have focused on research and analysis to propose a Fog-based architecture to address some of the challenges as introduced in Chapter 1.

As contributions, in Chapter 3 we first identify the initial challenges that IoT systems must face to deal with IoT devices. That is the dynamic nature and diversity of IoT heterogeneous devices. Therefore, the flexible provisioning of resources and the ability to scale within a Fog-based architecture are core requirements for dealing with these large numbers of heterogeneous devices.

Second, we propose an open services architecture to provide support for these heterogeneous devices called the **`Fog Services Provider (FSP)`**. In this FSP architecture, we analyzed the required functional and non-functional requirements as introduced in Section 3.2. These requirements enable our architecture that can be built and developed in a scalable fashion and deployed in different IoT system environments. Based on these analyzed requirements, we have designed a generalized architecture of services hierarchically to ensure support for features at each level such as compute, storage, and network according to topology and services running on the FSP. Furthermore, we also define some of the concepts required to clarify the components and services involved in the architecture such as the **`Producer`**, **`Consumer`**, and the **`FSP node`**. These components are introduced in detail to provide the most comprehensive overview of the IoT context.

Third, we also present a data management mechanism for these diverse and heterogeneous producers through the FSP architecture by providing an access service (**`producer access service`**) that connect to Producers. This service enables the handling of different types of producers by defining their profiles and manages through a gateway called the FSP gateway. This FSP gateway is responsible for ensuring efficient and reliable communication for Producers. Data generated from Producers will be processed

according to the ordered steps in Section 3.4.7. We divide these steps into two stages to facilitate data processing. The first stage includes the steps of processing raw data such as data collecting, data filtering, and data aggregating. Data processing from this stage ensures the elimination of redundant data and unnecessary repetition. The second stage focuses on processing the data so it becomes more meaningful. These steps consist of data modeling, data annotating, and data querying. This allows data to be shared and used to provide interoperability in IoT cross applications. On the other hand, the data can be provided to end-users by providing a **`consumer access service`**. This service provides an accessible way for the end-users to get data of interest. The FSP architecture also ensures that access to this service is secure through authentication and authorization mechanisms.

Fourth, we also implement an algorithm to convert the relational database into RDF data. This RDF data uses a **`triple`** or **`RDF statement`** to describe the different elements in a statement. Storing in this RDF data allows ensuring that semantic data and understanding are shared rather than relational database which only focuses on pure storage.

To provide a more comprehensive algorithm that allows data to be transformed in various forms such as CSV, XML, and JSON, we need to provide a general information model for the elements of the FSP architecture.

Out the fifth contribution is a unified data model detailed in Chapter 5. We designed a data model based on Semantic Web technology with the help of ontology by using several already existing ontologies and defining new core concepts to build our semantic data model called **`FSPontex`** in Section 5.5. This data model not only supports heterogeneous producers but also provides interoperability for applications in IoT cross-domains. Moreover, the consideration of context information in an IoT environment is necessary to provide shared information about contexts to relevant applications. Hence in this design model also shows some properties that allow context sharing detailed in Table 5.1.

Finally, we also consider a services placement mechanism in the FSP architecture by proposing a mathematical model to place services among the huge number of FSP nodes, while ensuring the performance of IoT applications in terms of latency, power consumption, and scalability.

6.2 Perspectives

Fog computing is an advanced technology and is an outstanding area of research, there is a lot of interest and many open areas of research which are promising challenges for future works. This section discusses some perspectives in analyzing and designing an IoT architecture based on the Fog computing paradigm.

In this thesis, a fog computing architecture to support a large number of heterogeneous IoT devices has been designed and some necessary functionalities were implemented and evaluated. However, the detailed design of the complete architecture, and the evaluation of its its performances in realistic scenarios, requires further research works. We have listed some open issues and future research directions as the following:

`Improvements in a general architecture`: A Fog architecture designed to fundamentally address some issues such as resource management, failure management, communication management, energy management, security and privacy.

1. **`Resource management`**: In a Fog landscape, resources are always dynamic and heterogeneous due to the variety of devices and their available resources. Therefore, fog nodes are responsible for providing services to meet the needs of these devices. Resource computation elasticity can predict the change and long-term activities of services or applications to provide to devices. Thus, the resource allocation and scheduling mechanism in a Fog environment is a significant challenge and needs to be addressed.

2. **`Failure management`**: Due to the devices are often geographically distributed and the management of these devices is not centralized. The probability of failure is always high due to some reasons such as hardware failure, software error, or even user error. Furthermore, these errors can be caused by connectivity, mobility, and power supply. Besides, devices are often connected according to their protocol. The change of position to continue to be seamlessly connected is also difficult to identify errors. Therefore, a service-level agreements(SLAs) mechanism is needed to ensure some parameters for QoS in terms of quality, availability, and role of services.

3. **`Communication management`**: In Fog architecture, fog nodes always ensure a seamless connection with devices, especially devices in time-sensitive applications. Combined fog nodes to form a cluster of federal fog nodes is very important.

Moreover, the communication between devices with Fog, Fog-Fog, and Fog-Cloud is also different in terms of connection type and protocol. Hence, this issue should also be considered.

4. **`Energy management`**: In the fog environment requires the deployment of a large number of fog nodes, essentially the computation is distributed and may be less energy efficient than the centralized model in the Cloud. Therefore, reducing energy consumption in fog is a significant challenge. A model of energy consumption and latency-related features should be considered to ensure performance and scalability issues.

5. **`Security and privacy`**: Devices are usually manufactured by different brands and have their security mechanisms. However, in order to join the Fog system, there is a need to compromise between the fog services provider and the owner of the devices. Correspondingly, security and trust policies need to be enforced within the Fog architecture of data and connectivity.

`Infrastructure-related issues`: Fog architectures are usually defined by hierarchical approaches. This helps ensure latency between levels. However, the decision to deploy will need to be guaranteed based on some criteria of each level in terms of application and resource requirements, scalability or shrinkage without interrupting services. As a result, the service placement issue also affects the implementation of the Fog architecture.

`Special application domains`: The variety of a set of heterogeneous devices requires a set of applications or services to support. The approach of a standard form of connection protocols, as well as a unified data model should also be addressed for issues such as device interoperability, network interoperability, data interoperability, platform interoperability, and semantic interoperability.

Publications

Based on the research work presented in this thesis, some papers have been published in international conferences and journals:

i. **Conferences:**

1. Hoan Le, Nadjib Achir, and Khaled Boussetta. "Fog Computing Architecture with Heterogeneous Internet of Things Technologies". In: *2019 10th International Conference on Networks of the Future (NoF)*. 2019 10th International Conference on Networks of the Future (NoF). Oct. 2019, pp. 130–133. DOI: 10.1109/NoF47743.2019.9014960
2. H. Le, N. Achir, and K. Boussetta. "Fog Services Provider Architecture for IoT". in: *2020 11th International Conference on Network of the Future (NoF)*. 2020 11th International Conference on Network of the Future (NoF). Oct. 2020, pp. 8–15. DOI: 10.1109/NoF50125.2020.9249177
3. H. Le, K. Boussetta, and N. Achir. "A Unified and Semantic Data Model for Fog Computing". In: *2020 Global Information Infrastructure and Networking Symposium (GIIS)*. 2020 Global Information Infrastructure and Networking Symposium (GIIS). Oct. 2020, pp. 1–6. DOI: 10.1109/GIIS50753.2020.9248482

ii. **Journals:**

Bibliography

[1] Rolf H. Weber. "Internet of Things – Need for a New Legal Environment?" In: *Computer Law & Security Review* 25.6 (Nov. 1, 2009), pp. 522–527. ISSN: 0267-3649. DOI: 10.1016/j.clsr.2009.09.002 *Cited on page 2.*

[2] Rolf H. Weber. "Internet of Things: Privacy Issues Revisited". In: *Computer Law & Security Review* 31.5 (Oct. 1, 2015), pp. 618–627. ISSN: 0267-3649. DOI: 10.1016/j.clsr.2015.07.002 *Cited on page 2.*

[3] Vala Afshar, ContributorChief Digital Evangelist, and Salesforce. *Cisco: Enterprises Are Leading The Internet of Things Innovation*. HuffPost. —400–Aug. 2017. URL: https://www.huffpost.com/entry/cisco-enterprises-are-leading-the-internet-of-things_b_59a41fcee4b0a62d0987b0c6 *Cited on page 2.*

[4] Tharam Dillon, Chen Wu, and Elizabeth Chang. "Cloud Computing: Issues and Challenges". In: *2010 24th IEEE International Conference on Advanced Information Networking and Applications*. 2010 24th IEEE International Conference on Advanced Information Networking and Applications. Apr. 2010, pp. 27–33. DOI: 10.1109/AINA.2010.187 *Cited on page 3.*

[5] Peter Mell and Tim Grance. *The NIST Definition of Cloud Computing*. NIST Special Publication (SP) 800-145. National Institute of Standards and Technology, Sept. 28, 2011. DOI: 10.6028/NIST.SP.800-145 *Cited on pages 3, 24.*

[6] H. F. Atlam et al. "Integration of Cloud Computing with Internet of Things: Challenges and Open Issues". In: *2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE

Smart Data (SmartData). June 2017, pp. 670–675. DOI: 10.1109/iThings-GreenCom-CPSCom-SmartData.2017.105 *Cited on page 3.*

[7] Sanjit Dash, Subasish Mohapatra, and P.K. Pattnaik. "A Survey on Applications of Wireless Sensor Network Using Cloud Computing". In: *International Journal of Computer Science & Emerging Technologies* 1 (Jan. 1, 2010) *Cited on page 3.*

[8] Flavio Bonomi et al. "Fog Computing and Its Role in the Internet of Things". In: *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*. MCC '12. New York, NY, USA: Association for Computing Machinery, Aug. 17, 2012, pp. 13–16. ISBN: 978-1-4503-1519-7. DOI: 10.1145/2342509.2342513 *Cited on pages 4, 26, 31, 32, 34–36, 39, 53.*

[9] Flavio Bonomi et al. "Fog Computing: A Platform for Internet of Things and Analytics". In: *Big Data and Internet of Things: A Roadmap for Smart Environments*. Ed. by Nik Bessis and Ciprian Dobre. Studies in Computational Intelligence. Cham: Springer International Publishing, 2014, pp. 169–186. ISBN: 978-3-319-05029-4. DOI: 10.1007/978-3-319-05029-4_7 *Cited on pages 4, 31, 39, 44, 65.*

[10] T. Qiu et al. "How Can Heterogeneous Internet of Things Build Our Future: A Survey". In: *IEEE Communications Surveys Tutorials* 20.3 (thirdquarter 2018), pp. 2011–2027. ISSN: 1553-877X. DOI: 10.1109/COMST.2018.2803740 *Cited on page 4.*

[11] Benoit Christophe et al. "The Web of Things Vision: Things as a Service and Interaction Patterns". In: *Bell Labs Technical Journal* 16.1 (June 1, 2011), pp. 55–61. ISSN: 1089-7089. DOI: 10.1002/bltj.20485 *Cited on page 5.*

[12] Nathalie Mitton et al. "Combining Cloud and Sensors in a Smart City Environment". In: *EURASIP Journal on Wireless Communications and Networking* 2012.1 (Aug. 8, 2012), p. 247. ISSN: 1687-1499. DOI: 10.1186/1687-1499-2012-247 *Cited on page 5.*

[13] Roy Want, Bill N. Schilit, and Scott Jenson. "Enabling the Internet of Things". In: *Computer* 48.1 (Jan. 2015), pp. 28–35. ISSN: 1558-0814. DOI: 10.1109/MC.2015.12 *Cited on page 5.*

[14] Mahda Noura, Mohammed Atiquzzaman, and Martin Gaedke. “Interoperability in Internet of Things: Taxonomies and Open Challenges”. In: *Mobile Networks and Applications* 24.3 (June 1, 2019), pp. 796–809. ISSN: 1572-8153. DOI: 10.1007/s11036-018-1089-9 *Cited on pages 6, 71, 90, 94.*

[15] RDF Triplet. *RDF 1.1 Concepts and Abstract Syntax (Accessed Dec 2020).* URL: https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/#data-model *Cited on pages 6, 74.*

[16] JSON-LD. *JSON-LD 1.1.* 2020. URL: https://www.w3.org/TR/json-ld/ *Cited on page 6.*

[17] Mohammad Aazam and Eui-Nam Huh. “Fog Computing Micro Datacenter Based Dynamic Resource Estimation and Pricing Model for IoT”. In: *2015 IEEE 29th International Conference on Advanced Information Networking and Applications.* 2015 IEEE 29th International Conference on Advanced Information Networking and Applications. Mar. 2015, pp. 687–694. DOI: 10.1109/AINA.2015.254 *Cited on pages 6, 39.*

[18] Nam Ky Giang et al. “Developing IoT Applications in the Fog: A Distributed Dataflow Approach”. In: *2015 5th International Conference on the Internet of Things (IOT).* 2015 5th International Conference on the Internet of Things (IOT). Oct. 2015, pp. 155–162. DOI: 10.1109/IOT.2015.7356560 *Cited on pages 6, 95.*

[19] *Node-RED.* URL: https://nodered.org/ *Cited on page 6.*

[20] Hamid Reza Arkian, Abolfazl Diyanat, and Atefe Pourkhalili. “MIST: Fog-Based Data Analytics Scheme with Cost-Efficient Resource Provisioning for IoT Crowdsensing Applications”. In: *Journal of Network and Computer Applications* 82 (Mar. 15, 2017), pp. 152–165. ISSN: 1084-8045. DOI: 10.1016/j.jnca.2017.01.012 *Cited on pages 6, 90.*

[21] Hoan Le, Nadjib Achir, and Khaled Boussetta. “Fog Computing Architecture with Heterogeneous Internet of Things Technologies”. In: *2019 10th International Conference on Networks of the Future (NoF).* 2019 10th International Conference on Networks of the Future (NoF). Oct. 2019, pp. 130–133. DOI: 10.1109/NoF47743.2019.9014960 *Cited on pages 8, 41, 67, 95, 119.*

[22] H. Le, N. Achir, and K. Boussetta. "Fog Services Provider Architecture for IoT". In: *2020 11th International Conference on Network of the Future (NoF)*. 2020 11th International Conference on Network of the Future (NoF). Oct. 2020, pp. 8–15. DOI: 10.1109/NoF50125.2020.9249177 *Cited on pages 8, 41, 67, 119.*

[23] H. Le, K. Boussetta, and N. Achir. "A Unified and Semantic Data Model for Fog Computing". In: *2020 Global Information Infrastructure and Networking Symposium (GIIS)*. 2020 Global Information Infrastructure and Networking Symposium (GIIS). Oct. 2020, pp. 1–6. DOI: 10.1109/GIIS50753.2020.9248482 *Cited on pages 8, 87, 119.*

[24] Gamble Procter. *Procter & Gamble*. 1999, Jun. URL: https://us.pg.com/ *Cited on page 13.*

[25] Kevin Ashton. *That 'Internet of Things' Thing | RFID JOURNAL*. RFiD Journal, 22(7):97–114. June 2009. URL: https://www.rfidjournal.com/that-internet-of-things-thing *Cited on page 13.*

[26] S. Al-Sarawi et al. "Internet of Things (IoT) Communication Protocols: Review". In: *2017 8th International Conference on Information Technology (ICIT)*. 2017 8th International Conference on Information Technology (ICIT). May 2017, pp. 685–690. DOI: 10.1109/ICITECH.2017.8079928 *Cited on pages 13, 16, 34.*

[27] P. P. Ray. "A Survey on Internet of Things Architectures". In: *Journal of King Saud University - Computer and Information Sciences* 30.3 (July 1, 2018), pp. 291–319. ISSN: 1319-1578. DOI: 10.1016/j.jksuci.2016.10.003 *Cited on pages 14, 18, 20.*

[28] Alessandro Bassi and Geir Horn. *Internet of Things in 2020: A Roadmap for the Future. European Commission: Information Society and Media, 2008*. The IT Law Wiki *Cited on page 14.*

[29] X. Jia et al. "RFID Technology and Its Applications in Internet of Things (IoT)". In: *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*. 2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet). Apr. 2012, pp. 1282–1285. DOI: 10.1109/CECNet.2012.6201508 *Cited on page 14.*

[30] Chiara Buratti et al. “An Overview on Wireless Sensor Networks Technology and Evolution”. In: *Sensors (Basel, Switzerland)* 9.9 (Aug. 31, 2009), pp. 6869–6896. ISSN: 1424-8220. DOI: 10.3390/s90906869. pmid: 22423202 *Cited on page 14.*

[31] JeongGil Ko, Tia Gao, and Andreas Terzis. “Empirical Study of a Medical Sensor Application in an Urban Emergency Department”. In: *Proceedings of the Fourth International Conference on Body Area Networks*. BodyNets ’09. Brussels, BEL: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Apr. 1, 2009, pp. 1–8. ISBN: 978-963-9799-41-7. DOI: 10.4108/ICST.BODYNETS2009.5947 *Cited on page 14.*

[32] Sajjad Hussain Shah and Ilyas Yaqoob. “A Survey: Internet of Things (IOT) Technologies, Applications and Challenges”. In: *2016 IEEE Smart Energy Grid Engineering (SEGE)*. 2016 IEEE Smart Energy Grid Engineering (SEGE). Aug. 2016, pp. 381–385. DOI: 10.1109/SEGE.2016.7589556 *Cited on page 14.*

[33] P. Victer Paul and R. Saraswathi. “The Internet of Things — A Comprehensive Survey”. In: *2017 International Conference on Computation of Power, Energy Information and Commuincation (ICCPEIC)*. 2017 International Conference on Computation of Power, Energy Information and Commuincation (ICCPEIC). Mar. 2017, pp. 421–426. DOI: 10.1109/ICCPEIC.2017.8290405 *Cited on page 14.*

[34] Karandeep Kaur. “A Survey on Internet of Things – Architecture, Applications, and Future Trends”. In: *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*. 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC). Dec. 2018, pp. 581–583. DOI: 10.1109/ICSCCC.2018.8703341 *Cited on page 14.*

[35] Mohammad Aazam, Sherali Zeadally, and Khaled A. Harras. “Deploying Fog Computing in Industrial Internet of Things and Industry 4.0”. In: *IEEE Transactions on Industrial Informatics* 14.10 (Oct. 2018), pp. 4674–4682. ISSN: 1941-0050. DOI: 10.1109/TII.2018.2855198 *Cited on page 15.*

[36] *Internet of Things*. In: *Wikipedia*. Sept. 10, 2020 *Cited on page 15.*

[37] Matt Hatton. *IoT News - The IoT in 2030: 24 Billion Connected Things Generating $1.5 Trillion*. IoT Business News. May 20, 2020. URL: https://iotbusinessnews.com/2020/05/20/03177-the-iot-in-2030-24-billion-connected-things-generating-1-5-trillion/ *Cited on page 15.*

[38] Yonghong Cheng, Huajun Zhang, and Yi Huang. "Overview of Communication Protocols in Internet of Things: Architecture, Development and Future Trends". In: *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*. 2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI). Santiago: IEEE, Dec. 2018, pp. 627–630. ISBN: 978-1-5386-7325-6. DOI: 10.1109/WI.2018.00-25 *Cited on pages 15, 16.*

[39] Pallavi Sethi and Smruti R. Sarangi. *Internet of Things: Architectures, Protocols, and Applications*. Journal of Electrical and Computer Engineering. Jan. 26, 2017 *Cited on page 16.*

[40] Angelo P. Castellani et al. "Architecture and Protocols for the Internet of Things: A Case Study". In: *2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*. 2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops). Mar. 2010, pp. 678–683. DOI: 10.1109/PERCOMW.2010.5470520 *Cited on page 16.*

[41] Tara Salman and Raj Jain. "Networking Protocols and Standards for Internet of Things". In: *Internet of Things and Data Analytics Handbook*. Wiley, 2017, pp. 215–238. ISBN: 978-1-119-17363-2. DOI: 10.1002/9781119173601.ch13 *Cited on page 17.*

[42] S. A. Al-Qaseemi et al. "IoT Architecture Challenges and Issues: Lack of Standardization". In: *2016 Future Technologies Conference (FTC)*. 2016 Future Technologies Conference (FTC). Dec. 2016, pp. 731–738. DOI: 10.1109/FTC.2016.7821686 *Cited on page 18.*

[43] Pedro Gonzalez-Gil, Juan Antonio Martinez, and Antonio F. Skarmeta. "Lightweight Data-Security Ontology for IoT". In: *Sensors (Basel, Switzerland)* 20.3 (Feb. 1, 2020). ISSN: 1424-8220. DOI: 10.3390/s20030801. pmid: 32024127 *Cited on page 18.*

[44] Mayke Ferreira Arruda and Renato Freitas Bulcão-Neto. "Toward a Lightweight Ontology for Privacy Protection in IoT". In: *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. SAC '19. New York, NY, USA: Association for Computing Machinery, Apr. 8, 2019, pp. 880–888. ISBN: 978-1-4503-5933-7. DOI: 10.1145/3297280.3297367 *Cited on page 18.*

[45] Samuel J. Moore et al. “IoT Reliability: A Review Leading to 5 Key Research Directions”. In: *CCF Transactions on Pervasive Computing and Interaction* 2.3 (Oct. 1, 2020), pp. 147–163. ISSN: 2524-5228. DOI: 10.1007/s42486-020-00037-z *Cited on page 18.*

[46] Alessandro Bassi and Sebastian Lange. “The Need for a Common Ground for the IoT: The History and Reasoning Behind the IoT-A Project”. In: *Enabling Things to Talk: Designing IoT Solutions with the IoT Architectural Reference Model*. Ed. by Alessandro Bassi et al. Berlin, Heidelberg: Springer, 2013, pp. 13–16. ISBN: 978-3-642-40403-0. DOI: 10.1007/978-3-642-40403-0_2 *Cited on page 18.*

[47] IBM. *Internet of Things Architecture: Reference Diagram - IBM Cloud Architecture Center*. 2020. URL: https://www.ibm.com/cloud/architecture/architectures/iotArchitecture/reference-architecture/ *Cited on page 18.*

[48] Paul Fremantle. *A Reference Architecture for the Internet of Things*. Oct. 2015. URL: https://wso2.com/whitepapers/a-reference-architecture-for-the-internet-of-things/ *Cited on pages 18, 19.*

[49] B. Butzin, F. Golatowski, and D. Timmermann. “Microservices Approach for the Internet of Things”. In: *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*. 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA). Sept. 2016, pp. 1–6. DOI: 10.1109/ETFA.2016.7733707 *Cited on pages 20, 22.*

[50] Sabrina Sicari et al. “A Security-and Quality-Aware System Architecture for Internet of Things”. In: *Information Systems Frontiers* 18.4 (Aug. 1, 2016), pp. 665–677. ISSN: 1572-9419. DOI: 10.1007/s10796-014-9538-x *Cited on page 20.*

[51] Prem Prakash Jayaraman et al. “Privacy Preserving Internet of Things: From Privacy Techniques to a Blueprint Architecture and Efficient Implementation”. In: *Future Generation Computer Systems* 76 (Nov. 1, 2017), pp. 540–549. ISSN: 0167-739X. DOI: 10.1016/j.future.2017.03.001 *Cited on page 20.*

[52] John Strassner and Wael William Diab. “A Semantic Interoperability Architecture for Internet of Things Data Sharing and Computing”. In: *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*. 2016 IEEE 3rd World Forum on Internet

of Things (WF-IoT). Dec. 2016, pp. 609–614. DOI: 10.1109/WF-IoT.2016.7845422 *Cited on page 20.*

[53] Dominique Guinard et al. "Interacting with the SOA-Based Internet of Things: Discovery, Query, Selection, and On-Demand Provisioning of Web Services". In: *IEEE Transactions on Services Computing* 3.3 (July 2010), pp. 223–235. ISSN: 1939-1374. DOI: 10.1109/TSC.2010.3 *Cited on page 20.*

[54] Shancang Li, Li Da Xu, and Shanshan Zhao. "The Internet of Things: A Survey". In: *Information Systems Frontiers* 17.2 (Apr. 1, 2015), pp. 243–259. ISSN: 1572-9419. DOI: 10.1007/s10796-014-9492-7 *Cited on pages 20, 21.*

[55] Sarra Hammoudi, Zibouda Aliouat, and Saad Harous. "Challenges and Research Directions for Internet of Things". In: *Telecommunication Systems* 67.2 (Feb. 1, 2018), pp. 367–385. ISSN: 1572-9451. DOI: 10.1007/s11235-017-0343-y *Cited on page 20.*

[56] Anwer Al-Dulaimi, Xianbin Wang, and Chih-Lin I. "Service-Oriented Architecture for IoT Home Area Networking in 5G". In: *5G Networks: Fundamental Requirements, Enabling Technologies, and Operations Management*. IEEE, 2018, pp. 577–602. ISBN: 978-1-119-33394-4. DOI: 10.1002/9781119333142.ch16 *Cited on page 20.*

[57] Alshinina R and Elleithy K. "Performance and Challenges of Service-Oriented Architecture for Wireless Sensor Networks." In: *Sensors (Basel, Switzerland)* 17.3 (Mar. 8, 2017). ISSN: 1424-8220. DOI: 10.3390/s17030536. pmid: 28282896 *Cited on page 21.*

[58] V. A. Vasil'ev et al. "Service-Oriented Architecture and Its Application to Smart Capabilities of Sensors". In: *2017 International Siberian Conference on Control and Communications (SIBCON)*. 2017 International Siberian Conference on Control and Communications (SIBCON). June 2017, pp. 1–4. DOI: 10.1109/SIBCON.2017.7998462 *Cited on page 21.*

[59] Yang Zhang, Jun-Liang Chen, and Bo Cheng. "Integrating Events into SOA for IoT Services". In: *IEEE Communications Magazine* 55.9 (Sept. 2017), pp. 180–186. ISSN: 1558-1896. DOI: 10.1109/MCOM.2017.1600359 *Cited on page 21.*

[60] Soumya Kanti Datta, Christian Bonnet, and Navid Nikaein. “An IoT Gateway Centric Architecture to Provide Novel M2M Services”. In: *2014 IEEE World Forum on Internet of Things (WF-IoT)*. 2014 IEEE World Forum on Internet of Things (WF-IoT). Mar. 2014, pp. 514–519. DOI: 10.1109/WF-IoT.2014.6803221 *Cited on page 22.*

[61] Farzad Khodadadi, Amir Vahid Dastjerdi, and Rajkumar Buyya. “Simurgh: A Framework for Effective Discovery, Programming, and Integration of Services Exposed in IoT”. In: *2015 International Conference on Recent Advances in Internet of Things (RIoT)*. 2015 International Conference on Recent Advances in Internet of Things (RIoT). Apr. 2015, pp. 1–6. DOI: 10.1109/RIOT.2015.7104910 *Cited on page 22.*

[62] Bo Cheng et al. “Lightweight Service Mashup Middleware With REST Style Architecture for IoT Applications”. In: *IEEE Transactions on Network and Service Management* 15.3 (Sept. 2018), pp. 1063–1075. ISSN: 1932-4537. DOI: 10.1109/TNSM.2018.2827933 *Cited on page 22.*

[63] M. Amaral et al. “Performance Evaluation of Microservices Architectures Using Containers”. In: *2015 IEEE 14th International Symposium on Network Computing and Applications*. 2015 IEEE 14th International Symposium on Network Computing and Applications. Sept. 2015, pp. 27–34. DOI: 10.1109/NCA.2015.49 *Cited on page 22.*

[64] Dongjin Yu et al. “A Survey on Security Issues in Services Communication of Microservices-Enabled Fog Applications”. In: *Concurrency and Computation: Practice and Experience* 31.22 (2019), e4436. ISSN: 1532-0634. DOI: 10.1002/cpe.4436 *Cited on page 23.*

[65] Petar Krivic et al. “Microservices as Agents in IoT Systems”. In: *Agent and Multi-Agent Systems: Technology and Applications*. Ed. by Gordan Jezic et al. Smart Innovation, Systems and Technologies. Cham: Springer International Publishing, 2017, pp. 22–31. ISBN: 978-3-319-59394-4. DOI: 10.1007/978-3-319-59394-4_3 *Cited on page 23.*

[66] Sagar Jaikar and R. Kamatchi. “A Survey of Messaging Protocols for IoT Systems”. In: (Feb. 5, 2018) *Cited on page 23.*

[67] Jiehan Zhou et al. “CloudThings: A Common Architecture for Integrating the Internet of Things with Cloud Computing”. In: *Proceedings of the 2013 IEEE 17th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. Proceedings of the 2013 IEEE 17th International Conference on Computer Supported Cooperative Work in Design (CSCWD). June 2013, pp. 651–657. DOI: 10.1109/CSCWD.2013.6581037 *Cited on page 24.*

[68] Alessio Botta et al. “Integration of Cloud Computing and Internet of Things: A Survey”. In: *Future Generation Computer Systems* 56 (Mar. 1, 2016), pp. 684–700. ISSN: 0167-739X. DOI: 10.1016/j.future.2015.09.021 *Cited on page 24.*

[69] Menachem Domb. “Smart Home Systems Based on Internet of Things”. In: *Internet of Things (IoT) for Automated and Smart Applications* (Feb. 28, 2019). DOI: 10.5772/intechopen.84894 *Cited on page 24.*

[70] Wu He, Gongjun Yan, and Li Da Xu. “Developing Vehicular Data Cloud Services in the IoT Environment”. In: *IEEE Transactions on Industrial Informatics* 10.2 (May 2014), pp. 1587–1595. ISSN: 1941-0050. DOI: 10.1109/TII.2014.2299233 *Cited on pages 24, 25.*

[71] Riccardo Petrolo, Valeria Loscrí, and Nathalie Mitton. “Towards a Smart City Based on Cloud of Things”. In: *Proceedings of the 2014 ACM International Workshop on Wireless and Mobile Technologies for Smart Cities*. WiMobCity ’14. New York, NY, USA: Association for Computing Machinery, Aug. 11, 2014, pp. 61–66. ISBN: 978-1-4503-3036-7. DOI: 10.1145/2633661.2633667 *Cited on page 25.*

[72] Mihai T. Lazarescu. “Design of a WSN Platform for Long-Term Environmental Monitoring for IoT Applications”. In: *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 3.1 (Mar. 2013), pp. 45–54. ISSN: 2156-3365. DOI: 10.1109/JETCAS.2013.2243032 *Cited on page 25.*

[73] C. Dobre and F. Xhafa. “Intelligent Services for Big Data Science”. In: *Future Generation Computer Systems*. Special Section: Innovative Methods and Algorithms for Advanced Data-Intensive Computing 37 (July 1, 2014), pp. 267–281. ISSN: 0167-739X. DOI: 10.1016/j.future.2013.07.014 *Cited on page 25.*

[74] Tapalina Bhattasali, Rituparna Chaki, and Nabendu Chaki. “Secure and Trusted Cloud of Things”. In: *2013 Annual IEEE India Conference (INDICON)*. 2013

Annual IEEE India Conference (INDICON). Dec. 2013, pp. 1–6. DOI: 10.1109/INDCON.2013.6725878 *Cited on page 25.*

[75] Redowan Mahmud, Ramamohanarao Kotagiri, and Rajkumar Buyya. "Fog Computing: A Taxonomy, Survey and Future Directions". In: *Internet of Everything: Algorithms, Methodologies, Technologies and Perspectives*. Ed. by Beniamino Di Martino et al. Internet of Things. Singapore: Springer, 2018, pp. 103–130. ISBN: 978-981-10-5861-5. DOI: 10.1007/978-981-10-5861-5_5 *Cited on page 27.*

[76] Pedro Garcia Lopez et al. "Edge-Centric Computing: Vision and Challenges". In: *ACM SIGCOMM Computer Communication Review* 45.5 (Sept. 30, 2015), pp. 37–42. ISSN: 0146-4833. DOI: 10.1145/2831347.2831354 *Cited on page 26.*

[77] Blesson Varghese et al. "Challenges and Opportunities in Edge Computing". In: *2016 IEEE International Conference on Smart Cloud (SmartCloud)*. 2016 IEEE International Conference on Smart Cloud (SmartCloud). Nov. 2016, pp. 20–26. DOI: 10.1109/SmartCloud.2016.18 *Cited on page 26.*

[78] Weisong Shi et al. "Edge Computing: Vision and Challenges". In: *IEEE Internet of Things Journal* 3.5 (Oct. 2016), pp. 637–646. ISSN: 2327-4662. DOI: 10.1109/JIOT.2016.2579198 *Cited on page 28.*

[79] Open Edge Computing. *Open Edge Computing Initiative*. Open Edge Computing Initiative. URL: https://www.openedgecomputing.org/ *Cited on page 28.*

[80] Atta ur Rehman Khan et al. "A Survey of Mobile Cloud Computing Application Models". In: *IEEE Communications Surveys Tutorials* 16.1 (First 2014), pp. 393–413. ISSN: 1553-877X. DOI: 10.1109/SURV.2013.062613.00160 *Cited on page 28.*

[81] Saeid Abolfazli et al. "Cloud-Based Augmentation for Mobile Devices: Motivation, Taxonomies, and Open Challenges". In: *IEEE Communications Surveys Tutorials* 16.1 (First 2014), pp. 337–368. ISSN: 1553-877X. DOI: 10.1109/SURV.2013.070813.00285 *Cited on page 28.*

[82] Atta ur Rehman Khan et al. "Context-Aware Mobile Cloud Computing and Its Challenges". In: *IEEE Cloud Computing* 2.3 (May 2015), pp. 42–49. ISSN: 2325-6095. DOI: 10.1109/MCC.2015.62 *Cited on page 28.*

[83] Zohreh Sanaei et al. “Heterogeneity in Mobile Cloud Computing: Taxonomy and Open Challenges”. In: *IEEE Communications Surveys Tutorials* 16.1 (First 2014), pp. 369–392. ISSN: 1553-877X. DOI: 10.1109/SURV.2013.050113.00090 *Cited on page 28.*

[84] DATAFLAIR TEAM. *Mobile Cloud Computing - 4 Unbelievable Benefits of MCC*. DataFlair. Dec. 8, 2018. URL: https://data-flair.training/blogs/mobile-cloud-computing-tutorial/ *Cited on page 29.*

[85] Yifan Yu. “Mobile Edge Computing towards 5G: Vision, Recent Progress, and Open Challenges”. In: *China Communications* 13 (Supplement2 2016), pp. 89–99. ISSN: 1673-5447. DOI: 10.1109/CC.2016.7833463 *Cited on page 29.*

[86] Yuyi Mao et al. “A Survey on Mobile Edge Computing: The Communication Perspective”. In: *IEEE Communications Surveys Tutorials* 19.4 (Fourthquarter 2017), pp. 2322–2358. ISSN: 1553-877X. DOI: 10.1109/COMST.2017.2745201 *Cited on page 29.*

[87] Eleonora Cau et al. “Efficient Exploitation of Mobile Edge Computing for Virtualized 5G in EPC Architectures”. In: *2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*. 2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud). Mar. 2016, pp. 100–109. DOI: 10.1109/MobileCloud.2016.24 *Cited on page 29.*

[88] Druid Software. *Mobile Edge Computing | Druid Software*. URL: https://www.druidsoftware.com/mobile-edge-computing/ *Cited on page 29.*

[89] K. Dolui and S. K. Datta. “Comparison of Edge Computing Implementations: Fog Computing, Cloudlet and Mobile Edge Computing”. In: *2017 Global Internet of Things Summit (GIoTS)*. 2017 Global Internet of Things Summit (GIoTS). June 2017, pp. 1–6. DOI: 10.1109/GIOTS.2017.8016213 *Cited on pages 30, 31.*

[90] Mahadev Satyanarayanan et al. “The Role of Cloudlets in Hostile Environments”. In: *IEEE Pervasive Computing* 12.4 (Oct. 2013), pp. 40–49. ISSN: 1558-2590. DOI: 10.1109/MPRV.2013.77 *Cited on page 30.*

[91] Tim Verbelen et al. “Cloudlets: Bringing the Cloud to the Mobile User”. In: *Proceedings of the Third ACM Workshop on Mobile Cloud Computing and Services*. MCS ’12. New York, NY, USA: Association for Computing Machinery, June 25,

2012, pp. 29–36. ISBN: 978-1-4503-1319-3. DOI: 10.1145/2307849.2307858 *Cited on page 30.*

[92] Subhadeep Sarkar and Sudip Misra. "Theoretical Modelling of Fog Computing: A Green Computing Paradigm to Support IoT Applications". In: *IET Networks* 5.2 (2016), pp. 23–29. ISSN: 2047-4962. DOI: 10.1049/iet-net.2015.0034 *Cited on pages 30, 39.*

[93] Ramasubbareddy Somula and Sasikala R. "Round Robin with Load Degree: An Algorithm for Optimal Cloudlet Discovery in Mobile Cloud Computing". In: *Scalable Computing: Practice and Experience* 19.1 (Mar. 11, 2018), pp. 39–52. ISSN: 1895-1767. DOI: 10.12694/scpe.v19i1.1392 *Cited on page 30.*

[94] Tuan Nguyen Gia et al. "Fog Computing Approach for Mobility Support in Internet-of-Things Systems". In: *IEEE Access* 6 (2018), pp. 36064–36082. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2848119 *Cited on page 32.*

[95] Yuanguo Bi et al. "Mobility Support for Fog Computing: An SDN Approach". In: *IEEE Communications Magazine* 56.5 (May 2018), pp. 53–59. ISSN: 1558-1896. DOI: 10.1109/MCOM.2018.1700908 *Cited on page 32.*

[96] Yang Liu, Jonathan E. Fieldsend, and Geyong Min. "A Framework of Fog Computing: Architecture, Challenges, and Optimization". In: *IEEE Access* 5 (2017), pp. 25445–25454. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2766923 *Cited on page 32.*

[97] H. Zhao et al. "Optimal Resource Rental Planning for Elastic Applications in Cloud Market". In: *2012 IEEE 26th International Parallel and Distributed Processing Symposium*. 2012 IEEE 26th International Parallel and Distributed Processing Symposium. May 2012, pp. 808–819. DOI: 10.1109/IPDPS.2012.77 *Cited on page 32.*

[98] H. Madsen et al. "Reliability in the Utility Computing Era: Towards Reliable Fog Computing". In: *2013 20th International Conference on Systems, Signals and Image Processing (IWSSIP)*. 2013 20th International Conference on Systems, Signals and Image Processing (IWSSIP). July 2013, pp. 43–46. DOI: 10.1109/IWSSIP.2013.6623445 *Cited on pages 32, 39.*

[99] Kirak Hong et al. “Mobile Fog: A Programming Model for Large-Scale Applications on the Internet of Things”. In: *MCC ’13*. 2013. DOI: 10.1145/2491266.2491270 *Cited on pages 34, 39.*

[100] Ahmed Banafa. *What Is Fog Computing?* Cloud computing news. Aug. 25, 2014. URL: https://www.ibm.com/blogs/cloud-computing/2014/08/25/fog-computing/ *Cited on pages 34–36.*

[101] Luis M. Vaquero and Luis Rodero-Merino. “Finding Your Way in the Fog: Towards a Comprehensive Definition of Fog Computing”. In: *ACM SIGCOMM Computer Communication Review* 44.5 (Oct. 10, 2014), pp. 27–32. ISSN: 0146-4833. DOI: 10.1145/2677046.2677052 *Cited on pages 35, 36, 39.*

[102] Ranesh Kumar Naha et al. “Fog Computing: Survey of Trends, Architectures, Requirements, and Research Directions”. In: *IEEE Access* 6 (2018), pp. 47980–48009. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2866491 *Cited on pages 35, 36, 46.*

[103] Alexandru-Florian Antonescu, Philip Robinson, and Torsten Braun. “Dynamic Topology Orchestration for Distributed Cloud-Based Applications”. In: *2012 Second Symposium on Network Cloud Computing and Applications*. 2012 Second Symposium on Network Cloud Computing and Applications. Dec. 2012, pp. 116–123. DOI: 10.1109/NCCA.2012.14 *Cited on pages 37, 39.*

[104] Jiang Zhu et al. “Improving Web Sites Performance Using Edge Servers in Fog Computing Architecture”. In: *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*. 2013 IEEE Seventh International Symposium on Service-Oriented System Engineering. Mar. 2013, pp. 320–323. DOI: 10.1109/SOSE.2013.73 *Cited on page 39.*

[105] Tom H. Luan et al. *Fog Computing: Focusing on Mobile Users at the Edge*. Mar. 29, 2016. arXiv: 1502.01815 [cs]. URL: http://arxiv.org/abs/1502.01815 *Cited on page 39.*

[106] Shanhe Yi et al. “Fog Computing: Platform and Applications”. In: *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*. 2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies

(HotWeb). Nov. 2015, pp. 73–78. DOI: 10.1109/HotWeb.2015.22 *Cited on page 39.*

[107] Shanhe Yi, Cheng Li, and Qun Li. "A Survey of Fog Computing: Concepts, Applications and Issues". In: *Proceedings of the 2015 Workshop on Mobile Big Data*. Mobidata '15. New York, NY, USA: Association for Computing Machinery, June 21, 2015, pp. 37–42. ISBN: 978-1-4503-3524-9. DOI: 10.1145/2757384.2757397 *Cited on page 39.*

[108] Ivan Stojmenovic. "Fog Computing: A Cloud to the Ground Support for Smart Things and Machine-to-Machine Networks". In: *2014 Australasian Telecommunication Networks and Applications Conference (ATNAC)*. 2014 Australasian Telecommunication Networks and Applications Conference (ATNAC). Nov. 2014, pp. 117–122. DOI: 10.1109/ATNAC.2014.7020884 *Cited on page 39.*

[109] Amir Vahid Dastjerdi and Rajkumar Buyya. "Fog Computing: Helping the Internet of Things Realize Its Potential". In: *Computer* 49.8 (Aug. 1, 2016), pp. 112–116. ISSN: 0018-9162. DOI: 10.1109/MC.2016.245 *Cited on pages 39, 43, 89.*

[110] P. Habibi et al. "Virtualized SDN-Based End-to-End Reference Architecture for Fog Networking". In: *2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA)*. 2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA). May 2018, pp. 61–66. DOI: 10.1109/WAINA.2018.00064 *Cited on pages 39, 65.*

[111] Mohammad Abdullah Al Faruque and Korosh Vatanparvar. "Energy Management-as-a-Service Over Fog Computing Platform". In: *IEEE Internet of Things Journal* 3.2 (Apr. 2016), pp. 161–169. ISSN: 2327-4662. DOI: 10.1109/JIOT.2015.2471260 *Cited on pages 39, 54.*

[112] Enrique Saurez et al. "Incremental Deployment and Migration of Geo-Distributed Situation Awareness Applications in the Fog". In: *Proceedings of the 10th ACM International Conference on Distributed and Event-Based Systems*. DEBS '16. New York, NY, USA: Association for Computing Machinery, June 13, 2016, pp. 258–269. ISBN: 978-1-4503-4021-2. DOI: 10.1145/2933267.2933317 *Cited on page 39.*

[113] Olena Skarlat et al. “Resource Provisioning for IoT Services in the Fog”. In: *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*. 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA). Nov. 2016, pp. 32–39. DOI: 10.1109/SOCA.2016.10 *Cited on page 39.*

[114] Mohammad Aazam and Eui-Nam Huh. “Dynamic Resource Provisioning through Fog Micro Datacenter”. In: *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*. 2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops). Mar. 2015, pp. 105–110. DOI: 10.1109/PERCOMW.2015.7134002 *Cited on page 39.*

[115] Sukhpal Singh and Inderveer Chana. “QoS-Aware Autonomic Resource Management in Cloud Computing: A Systematic Review”. In: *ACM Computing Surveys* 48.3 (Dec. 22, 2015), 42:1–42:46. ISSN: 0360-0300. DOI: 10.1145/2843889 *Cited on page 39.*

[116] Zhi-Hui Zhan et al. “Cloud Computing Resource Scheduling and a Survey of Its Evolutionary Approaches”. In: *ACM Computing Surveys* 47.4 (July 21, 2015), 63:1–63:33. ISSN: 0360-0300. DOI: 10.1145/2788397 *Cited on page 39.*

[117] K. C. Okafor et al. *Leveraging Fog Computing for Scalable IoT Datacenter Using Spine-Leaf Network Topology*. Journal of Electrical and Computer Engineering. Apr. 24, 2017 *Cited on page 39.*

[118] Shanhe Yi, Zhengrui Qin, and Qun Li. “Security and Privacy Issues of Fog Computing: A Survey”. In: *In International Conference on Wireless Algorithms, Systems and Applications (WASA*. 2015 *Cited on page 39.*

[119] E. Yigitoglu et al. “Foggy: A Framework for Continuous Automated IoT Application Deployment in Fog Computing”. In: *2017 IEEE International Conference on AI Mobile Services (AIMS)*. 2017 IEEE International Conference on AI Mobile Services (AIMS). June 2017, pp. 38–45. DOI: 10.1109/AIMS.2017.14 *Cited on page 39.*

[120] Hafizur Rahman and Md. Iftekhar Hussain. “Fog-Based Semantic Model for Supporting Interoperability in IoT”. In: *IET Communications* 13.11 (2019),

pp. 1651–1661. ISSN: 1751-8636. DOI: 10.1049/iet-com.2018.6200 *Cited on pages 39, 44, 65, 89.*

[121] M. S. de Brito et al. "A Service Orchestration Architecture for Fog-Enabled Infrastructures". In: *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*. 2017 Second International Conference on Fog and Mobile Edge Computing (FMEC). May 2017, pp. 127–132. DOI: 10.1109/FMEC.2017.7946419 *Cited on pages 39, 45, 65.*

[122] Fatema Tuz Zohora et al. "Enhancing the Capabilities of IoT Based Fog and Cloud Infrastructures for Time Sensitive Events". In: *2017 International Conference on Electrical Engineering and Computer Science (ICECOS)*. 2017 International Conference on Electrical Engineering and Computer Science (ICECOS). Aug. 2017, pp. 224–230. DOI: 10.1109/ICECOS.2017.8167139 *Cited on page 43.*

[123] Yong Li and Wei Gao. "Interconnecting Heterogeneous Devices in the Personal Mobile Cloud". In: *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*. IEEE INFOCOM 2017 - IEEE Conference on Computer Communications. May 2017, pp. 1–9. DOI: 10.1109/INFOCOM.2017.8057083 *Cited on page 43.*

[124] Amit Banerjee et al. "Centralized Framework for Controlling Heterogeneous Appliances in a Smart Home Environment". In: *2018 International Conference on Information and Computer Technologies (ICICT)*. 2018 International Conference on Information and Computer Technologies (ICICT). Mar. 2018, pp. 78–82. DOI: 10.1109/INFOCT.2018.8356844 *Cited on page 43.*

[125] Sripriya Srikant Adhatarao, Mayutan Arumaithurai, and Xiaoming Fu. "FOGG: A Fog Computing Based Gateway to Integrate Sensor Networks to Internet". In: *2017 29th International Teletraffic Congress (ITC 29)*. 2017 29th International Teletraffic Congress (ITC 29). Vol. 2. Sept. 2017, pp. 42–47. DOI: 10.23919/ITC.2017.8065709 *Cited on page 44.*

[126] Jeevan Kharel, Haftu T. Reda, and Soo Y. Shin. "An Architecture for Smart Health Monitoring System Based on Fog Computing". In: *Journal of Communications* (2017). ISSN: 17962021. DOI: 10.12720/jcm.12.4.228-233 *Cited on page 44.*

[127] Prabal Verma and Sandeep K. Sood. "Fog Assisted-IoT Enabled Patient Health Monitoring in Smart Homes". In: *IEEE Internet of Things Journal* 5.3 (June 2018), pp. 1789–1796. ISSN: 2327-4662. DOI: 10.1109/JIOT.2018.2803201 *Cited on page 44.*

[128] John Soldatos et al. "OpenIoT: Open Source Internet-of-Things in the Cloud". In: *Interoperability and Open-Source Solutions for the Internet of Things*. Ed. by Ivana Podnar Žarko, Krešimir Pripužić, and Martin Serrano. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, pp. 13–25. ISBN: 978-3-319-16546-2. DOI: 10.1007/978-3-319-16546-2_3 *Cited on page 44.*

[129] *RDF - Semantic Web Standards*. URL: https://www.w3.org/RDF/ *Cited on page 44.*

[130] *OWL - Semantic Web Standards*. URL: https://www.w3.org/OWL/ *Cited on pages 45, 110.*

[131] Yu Hsiang Chien and Fuchun Joseph Lin. "Distributed Semantic Reasoning Enabled by Fog Computing". In: *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. 2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData). July 2019, pp. 1033–1040. DOI: 10.1109/iThings/GreenCom/CPSCom/SmartData.2019.00179 *Cited on page 45.*

[132] Y. Xu and A. Helal. "Scalable Cloud–Sensor Architecture for the Internet of Things". In: *IEEE Internet of Things Journal* 3.3 (June 2016), pp. 285–298. ISSN: 2327-4662. DOI: 10.1109/JIOT.2015.2455555 *Cited on page 46.*

[133] H. Y. Shwe and P. H. J. Chong. "Scalable Distributed Cloud Data Storage Service for Internet of Things". In: *2016 Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*. 2016 Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big

Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld). July 2016, pp. 869–873. DOI: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0137 *Cited on page 46.*

[134] doodlemania2. *Design for Change - Azure Application Architecture Guide*. URL: https://docs.microsoft.com/en-us/azure/architecture/guide/design-principles/design-for-evolution *Cited on page 47.*

[135] "IEEE Standard Glossary of Software Engineering Terminology". In: *IEEE Std 610.12-1990* (Dec. 1990), pp. 1–84. DOI: 10.1109/IEEESTD.1990.101064 *Cited on page 47.*

[136] J. Kiljander et al. "Semantic Interoperability Architecture for Pervasive Computing and Internet of Things". In: *IEEE Access* 2 (2014), pp. 856–873. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2014.2347992 *Cited on page 47.*

[137] C. Chen et al. "Why Is It Important to Measure Maintainability and What Are the Best Ways to Do It?" In: *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C). May 2017, pp. 377–378. DOI: 10.1109/ICSE-C.2017.75 *Cited on page 48.*

[138] Bruce R. Maxim and Marouane Kessentini. "Chapter 2 - An Introduction to Modern Software Quality Assurance". In: *Software Quality Assurance*. Ed. by Ivan Mistrik et al. Boston: Morgan Kaufmann, Jan. 1, 2016, pp. 19–46. ISBN: 978-0-12-802301-3. DOI: 10.1016/B978-0-12-802301-3.00002-8 *Cited on page 48.*

[139] web semantic. *Semantic Web - W3C*. URL: https://www.w3.org/standards/semanticweb/ *Cited on page 48.*

[140] Chris Richardson. *Service Discovery in a Microservices Architecture*. NGINX. Oct. 12, 2015. URL: https://www.nginx.com/blog/service-discovery-in-a-microservices-architecture/ *Cited on page 49.*

[141] Rotem-Gal-Oz Arnon. *SOA Patterns*. Manning Publications. URL: https://www.manning.com/books/soa-patterns *Cited on pages 49, 53.*

[142] Michaela Iorga et al. "Fog Computing Conceptual Model". In: *Special Publication (NIST SP)* (Mar. 14, 2018). DOI: 10.6028/NIST.SP.500-325 *Cited on page 51.*

[143] Pengfei Hu et al. “Fog Computing Based Face Identification and Resolution Scheme in Internet of Things”. In: *IEEE Transactions on Industrial Informatics* 13.4 (Aug. 2017), pp. 1910–1920. ISSN: 1941-0050. DOI: 10.1109/TII.2016.2607178 *Cited on page 51.*

[144] nishanil. *Microservices .NET. Architecture pour les applications .NET en conteneur*. URL: https://docs.microsoft.com/fr-fr/dotnet/architecture/microservices/ *Cited on pages 51, 52, 56.*

[145] Istabraq M. Al-Joboury and Emad H. Al-Hemiary. “IoT Protocols Based Fog/-Cloud over High Traffic”. In: *The ISC International Journal of Information Security* 11.3 (Aug. 1, 2019), pp. 173–180. ISSN: 2008-2045. DOI: 10.22042/isecure.2019.11.3.23 *Cited on page 52.*

[146] Triplestore. *Category:Triple Store - Semantic Web Standards*. URL: https://www.w3.org/2001/sw/wiki/Category:Triple_Store *Cited on page 52.*

[147] Ontotext. *What Is an RDF Triplestore? | Ontotext Fundamentals*. Ontotext. URL: https://www.ontotext.com/knowledgehub/fundamentals/what-is-rdf-triplestore/ *Cited on page 52.*

[148] C. Puliafito, E. Mingozzi, and G. Anastasi. “Fog Computing for the Internet of Mobile Things: Issues and Challenges”. In: *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*. 2017 IEEE International Conference on Smart Computing (SMARTCOMP). May 2017, pp. 1–6. DOI: 10.1109/SMARTCOMP.2017.7947010 *Cited on page 54.*

[149] Yousef Allahvirdizadeh, Mohsen Parsa Moghaddam, and Heidarali Shayanfar. “A Survey on Cloud Computing in Energy Management of the Smart Grids”. In: *International Transactions on Electrical Energy Systems* 29.10 (2019), e12094. ISSN: 2050-7038. DOI: 10.1002/2050-7038.12094 *Cited on page 54.*

[150] Asfa Toor et al. “Energy Efficient Edge-of-Things”. In: *EURASIP Journal on Wireless Communications and Networking* 2019.1 (Mar. 29, 2019), p. 82. ISSN: 1687-1499. DOI: 10.1186/s13638-019-1394-4 *Cited on page 54.*

[151] X. Masip-Bruin et al. “Foggy Clouds and Cloudy Fogs: A Real Need for Coordinated Management of Fog-to-Cloud Computing Systems”. In: *IEEE Wireless Communications* 23.5 (Oct. 2016), pp. 120–128. ISSN: 1558-0687. DOI: 10.1109/MWC.2016.7721750 *Cited on page 55.*

[152] Richard John Anthony. “Chapter 6 - Distributed Systems”. In: *Systems Programming*. Ed. by Richard John Anthony. Boston: Morgan Kaufmann, Jan. 1, 2016, pp. 383–474. ISBN: 978-0-12-800729-7. DOI: 10.1016/B978-0-12-800729-7.00006-6 *Cited on page 58.*

[153] Haris Aftab et al. “Analysis of Identifiers in IoT Platforms”. In: *Digital Communications and Networks* 6.3 (Aug. 1, 2020), pp. 333–340. ISSN: 2352-8648. DOI: 10.1016/j.dcan.2019.05.003 *Cited on page 58.*

[154] nishanil. *Communication in a Microservice Architecture*. 2020. URL: https://docs.microsoft.com/en-us/dotnet/architecture/microservices/architect-microservice-container-applications/communication-in-microservice-architecture *Cited on page 59.*

[155] N. Naik. “Choice of Effective Messaging Protocols for IoT Systems: MQTT, CoAP, AMQP and HTTP”. In: *2017 IEEE International Systems Engineering Symposium (ISSE)*. 2017 IEEE International Systems Engineering Symposium (ISSE). Oct. 2017, pp. 1–7. DOI: 10.1109/SysEng.2017.8088251 *Cited on pages 60, 109.*

[156] REST framework Django. *Django REST Framework*. 2020. URL: https://www.django-rest-framework.org/ *Cited on page 63.*

[157] E. Al-Masri et al. “Investigating Messaging Protocols for the Internet of Things (IoT)”. In: *IEEE Access* 8 (2020), pp. 94880–94911. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2993363 *Cited on pages 63, 72.*

[158] Sam Guckenheimer. *What Is DevOps? - Azure DevOps*. 2018. URL: https://docs.microsoft.com/en-us/azure/devops/learn/what-is-devops *Cited on page 64.*

[159] R. Laigner et al. “From a Monolithic Big Data System to a Microservices Event-Driven Architecture”. In: *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). Aug. 2020, pp. 213–220. DOI: 10.1109/SEAA51224.2020.00045 *Cited on page 69.*

[160] Muhammad Saqlain et al. “Framework of an IoT-Based Industrial Data Management for Smart Manufacturing”. In: *Journal of Sensor and Actuator Networks* 8.2 (2 June 2019), p. 25. DOI: 10.3390/jsan8020025 *Cited on page 70.*

[161] Sandra Tucker and Beth Ackerman. *Data Management Considerations for Microservices Solutions - IBM Garage Practices*. 2020. URL: https://www.ibm.com/garage/method/practices/code/data-management-for-microservices/ *Cited on page 70.*

[162] Hoan Nguyen Mau Quoc et al. "A Performance Study of RDF Stores for Linked Sensor Data". In: (Aug. 1, 2019) *Cited on page 72.*

[163] SPARQL Linked Data. *Data - W3C*. URL: https://www.w3.org/standards/semanticweb/data *Cited on page 73.*

[164] Erno Kovacs et al. "Standards-Based Worldwide Semantic Interoperability for IoT". In: *IEEE Communications Magazine* 54.12 (Dec. 2016), pp. 40–46. ISSN: 1558-1896. DOI: 10.1109/MCOM.2016.1600460CM *Cited on page 73.*

[165] Garvita Bajaj et al. *A Study of Existing Ontologies in the IoT-Domain*. July 1, 2017. arXiv: 1707.00112 [cs]. URL: http://arxiv.org/abs/1707.00112 *Cited on page 77.*

[166] V. Sugumaran. "Chapter 14 - Semantic Technologies for Enhancing Knowledge Management Systems". In: *Successes and Failures of Knowledge Management*. Ed. by Jay Liebowitz. Boston: Morgan Kaufmann, Jan. 1, 2016, pp. 203–213. ISBN: 978-0-12-805187-0. DOI: 10.1016/B978-0-12-805187-0.00014-0 *Cited on page 78.*

[167] HypriotOS. *Docker Pirates ARMed with Explosive Stuff · Docker Pirates ARMed with Explosive Stuff*. URL: https://blog.hypriot.com/ *Cited on page 81.*

[168] Yongrui Qin et al. "When Things Matter: A Survey on Data-Centric Internet of Things". In: *Journal of Network and Computer Applications* 64 (Apr. 1, 2016), pp. 137–153. ISSN: 1084-8045. DOI: 10.1016/j.jnca.2015.12.016 *Cited on pages 89, 93, 94.*

[169] Tina Samizadeh Nikoui, Amir Masoud Rahmani, and Hooman Tabarsaied. "Data Management in Fog Computing". In: *Fog and Edge Computing*. John Wiley & Sons, Ltd, 2019, pp. 171–190. ISBN: 978-1-119-52508-0. DOI: 10.1002/9781119525080.ch8 *Cited on pages 89, 94.*

[170] Feifei Shi et al. "A Survey of Data Semantization in Internet of Things". In: *Sensors* 18.1 (1 Jan. 2018), p. 313. DOI: 10.3390/s18010313 *Cited on page 89.*

[171] Pavlos Charalampidis, Elias Tragos, and Alexandros Fragkiadakis. "A Fog-Enabled IoT Platform for Efficient Management and Data Collection". In: *2017 IEEE 22nd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*. 2017 IEEE 22nd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD). June 2017, pp. 1–6. DOI: 10.1109/CAMAD.2017.8031527 *Cited on page 90.*

[172] J. He et al. "Multitier Fog Computing With Large-Scale IoT Data Analytics for Smart Cities". In: *IEEE Internet of Things Journal* (2018). DOI: 10.1109/JIOT.2017.2724845 *Cited on page 90.*

[173] Eduard Gibert Renart, J. Montes, and M. Parashar. "Data-Driven Stream Processing at the Edge". In: *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)* (2017). DOI: 10.1109/ICFEC.2017.18 *Cited on page 90.*

[174] O. Hahm et al. "Operating Systems for Low-End Devices in the Internet of Things: A Survey". In: *IEEE Internet of Things Journal* 3.5 (Oct. 2016), pp. 720–734. ISSN: 2327-4662. DOI: 10.1109/JIOT.2015.2505901 *Cited on page 91.*

[175] Oladayo Bello, Sherali Zeadally, and Mohamad Badra. "Network Layer Inter-Operation of Device-to-Device Communication Technologies in Internet of Things (IoT)". In: *Ad Hoc Networks* 57 (Mar. 2017), pp. 52–62. ISSN: 15708705. DOI: 10.1016/j.adhoc.2016.06.010 *Cited on page 91.*

[176] JVM. *Java Virtual Machine*. In: *Wikipedia*. Dec. 13, 2020 *Cited on page 92.*

[177] W3C. *Achieving Semantic Interoperability Using RDF and OWL — V10. Accessed 02/01/2021.* URL: https://www.w3.org/2001/sw/BestPractices/OEP/SemInt/ *Cited on page 92.*

[178] Aimad Karkouch et al. "Data Quality in Internet of Things". In: *Journal of Network and Computer Applications* 73.C (Sept. 1, 2016), pp. 57–81. ISSN: 1084-8045. DOI: 10.1016/j.jnca.2016.08.002 *Cited on page 93.*

[179] Shree Krishna Sharma and Xianbin Wang. "Live Data Analytics With Collaborative Edge and Cloud Processing in Wireless IoT Networks". In: *IEEE Access*

5 (2017), pp. 4621–4635. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2682640 *Cited on page 94.*

[180] Viacheslav Kulik and Ruslan Kirichek. "The Heterogeneous Gateways in the Industrial Internet of Things". In: *2018 10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*. 2018 10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT). Nov. 2018, pp. 1–5. DOI: 10.1109/ICUMT.2018.8631232 *Cited on page 94.*

[181] IPSO Alliance. *IPSO Alliance*. In: *Wikipedia*. July 21, 2020 *Cited on page 94.*

[182] oneM2M. *oneM2M - Home*. URL: https://www.onem2m.org/ *Cited on page 94.*

[183] Altti Ilari Maarala, Xiang Su, and Jukka Riekki. "Semantic Reasoning for Context-Aware Internet of Things Applications". In: *IEEE Internet of Things Journal* 4.2 (Apr. 2017), pp. 461–473. ISSN: 2327-4662. DOI: 10.1109/JIOT.2016.2587060 *Cited on page 94.*

[184] Anisha Gupta, Rivana Christie, and R Manjula. "Scalability in Internet of Things: Features, Techniques and Research Challenges". In: *International Journal of Computational Intelligence Research* 13 (2017), p. 12. ISSN: 0973-1873 *Cited on page 94.*

[185] Harishchandra Dubey et al. *Fog Data: Enhancing Telehealth Big Data Through Fog Computing*. June 1, 2016. arXiv: 1605.09437 [cs] *Cited on page 95.*

[186] Everton de Matos et al. "Context Information Sharing for the Internet of Things: A Survey". In: *Computer Networks* 166 (Jan. 15, 2020), p. 106988. ISSN: 1389-1286. DOI: 10.1016/j.comnet.2019.106988 *Cited on pages 95, 96, 103.*

[187] Mervat Abu-Elkheir, Mohammad Hayajneh, and Najah Abu Ali. "Data Management for the Internet of Things: Design Primitives and Solution". In: *Sensors (Basel, Switzerland)* 13.11 (Nov. 14, 2013), pp. 15582–15612. ISSN: 1424-8220. DOI: 10.3390/s131115582. pmid: 24240599 *Cited on page 96.*

[188] G. Daneels et al. "Real-Time Data Dissemination and Analytics Platform for Challenging IoT Environments". In: *2017 Global Information Infrastructure and Networking Symposium (GIIS)*. 2017 Global Information Infrastructure and

Networking Symposium (GIIS). Oct. 2017, pp. 23–30. DOI: 10.1109/GIIS.2017.8169799 *Cited on page 96.*

[189] *Semantic Sensor Network Ontology. https://www.w3.org/TR/vocab-ssn/ Cited on page 98.*

[190] Lingling Xue et al. "An Ontology Based Scheme for Sensor Description in Context Awareness System". In: *2015 IEEE International Conference on Information and Automation*. 2015 IEEE International Conference on Information and Automation. Aug. 2015, pp. 817–820. DOI: 10.1109/ICInfA.2015.7279397 *Cited on page 98.*

[191] Nicolas Seydoux et al. "IoT-O, a Core-Domain IoT Ontology to Represent Connected Devices Networks". In: *20th International Conference on Knowledge Engineering and Knowledge Management - Volume 10024*. EKAW 2016. Bologna, Italy: Springer-Verlag, Nov. 19, 2016, pp. 561–576. ISBN: 978-3-319-49003-8. DOI: 10.1007/978-3-319-49004-5_36 *Cited on page 98.*

[192] Maria Bermudez-Edo et al. "IoT-Lite: A Lightweight Semantic Model for the Internet of Things". In: *2016 Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*. 2016 Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld). July 2016, pp. 90–97. DOI: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0035 *Cited on page 98.*

[193] Amelie Gyrard et al. "Cross-Domain Internet of Things Application Development: M3 Framework and Evaluation". In: *2015 3rd International Conference on Future Internet of Things and Cloud*. 2015 3rd International Conference on Future Internet of Things and Cloud. Aug. 2015, pp. 9–16. DOI: 10.1109/FiCloud.2015.10 *Cited on page 98.*

[194] *M3-Lite, https://lov.linkeddata.es/dataset/lov/vocabs/m3lite - Oct 2018*. URL: https://lov.linkeddata.es/dataset/lov/vocabs/m3lite *Cited on page 98.*

[195] Hafizur Rahman and Md. Iftekhar Hussain. “LiO-IoT: A Light-Weight Ontology to Provide Semantic Interoperability in Internet of Things”. In: *SSRN Electronic Journal* 2 (Mar. 29, 2019), pp. 571–575 *Cited on page 98.*

[196] Kun Xie et al. “Decentralized Context Sharing in Vehicular Delay Tolerant Networks with Compressive Sensing”. In: *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*. 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS). June 2016, pp. 169–178. DOI: 10.1109/ICDCS.2016.83 *Cited on page 98.*

[197] Adrian A. de Freitas et al. “Bluewave: Enabling Opportunistic Context Sharing via Bluetooth Device Names”. In: *Proceedings of the 8th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. EICS ’16. New York, NY, USA: Association for Computing Machinery, June 21, 2016, pp. 38–49. ISBN: 978-1-4503-4322-0. DOI: 10.1145/2933242.2933248 *Cited on page 98.*

[198] Michele Ruta et al. “Knowledge Discovery and Sharing in the IoT: The Physical Semantic Web Vision”. In: *Proceedings of the Symposium on Applied Computing*. SAC ’17. New York, NY, USA: Association for Computing Machinery, Apr. 3, 2017, pp. 492–498. ISBN: 978-1-4503-4486-9. DOI: 10.1145/3019612.3019701 *Cited on page 98.*

[199] Diptendu Sinha Roy et al. “A Context-Aware Fog Enabled Scheme for Real-Time Cross-Vertical IoT Applications”. In: *IEEE Internet of Things Journal* 6.2 (Apr. 2019), pp. 2400–2412. ISSN: 2327-4662. DOI: 10.1109/JIOT.2018.2869323 *Cited on page 98.*

[200] Wenxi Zeng et al. “Invited Paper: Semantic IoT Data Description and Discovery in the IoT-Edge-Fog-Cloud Infrastructure”. In: *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*. 2019 IEEE International Conference on Service-Oriented System Engineering (SOSE). Apr. 2019, pp. 106–10609. DOI: 10.1109/SOSE.2019.00024 *Cited on page 98.*

[201] Rachit Agarwal et al. “Unified IoT Ontology to Enable Interoperability and Federation of Testbeds”. In: *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*. 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT). Dec. 2016, pp. 70–75. DOI: 10.1109/WF-IoT.2016.7845470 *Cited on pages 98, 101.*

[202] Natalya F. Noy and Deborah L. Mcguinness. *Ontology Development 101: A Guide to Creating Your First Ontology*. 2001 *Cited on page 98.*

[203] Garvita Bajaj et al. "4W1H in IoT Semantics". In: *IEEE Access* 6 (2018), pp. 65488–65506. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2878100 *Cited on pages 98, 99, 101.*

[204] Mounir Achouri et al. "SMART FOG COMPUTING FOR EFFICIENT SITUATIONS MANAGEMENT IN SMART HEALTH ENVIRONMENTS". In: *Journal of Information and Communication Technology* 17.4 (Oct. 1, 2018), pp. 537–567. ISSN: 2180-3862 *Cited on page 100.*

[205] Everton de Matos, Leonardo Albernaz Amaral, and Fabiano Hessel. "Context-Aware Systems: Technologies and Challenges in Internet of Everything Environments". In: *Beyond the Internet of Things: Everything Interconnected*. Ed. by Jordi Mongay Batalla et al. Internet of Things. Cham: Springer International Publishing, 2017, pp. 1–25. ISBN: 978-3-319-50758-3. DOI: 10.1007/978-3-319-50758-3_1 *Cited on page 100.*

[206] Amelie Gyrard et al. "LOV4IoT: A Second Life for Ontology-Based Domain Knowledge to Build Semantic Web of Things Applications". In: *2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*. 2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud). Aug. 2016, pp. 254–261. DOI: 10.1109/FiCloud.2016.44 *Cited on page 101.*

[207] Charith Perera et al. "Context Aware Computing for The Internet of Things: A Survey". In: *IEEE Communications Surveys Tutorials* 16.1 (First 2014), pp. 414–454. ISSN: 1553-877X. DOI: 10.1109/SURV.2013.042313.00197 *Cited on page 102.*

[208] Ahmed Bali, Mahmud Al-Osta, and Gherbi Abdelouahed. "An Ontology-Based Approach for IoT Data Processing Using Semantic Rules". In: *SDL 2017: Model-Driven Engineering for Future Internet*. Ed. by Tibor Csöndes, Gábor Kovács, and György Réthy. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 61–79. ISBN: 978-3-319-68015-6. DOI: 10.1007/978-3-319-68015-6_5 *Cited on page 103.*

[209] Mahmud Al-Osta, Bali Ahmed, and Gherbi Abdelouahed. "A Lightweight Semantic Web-Based Approach for Data Annotation on IoT Gateways". In: *Procedia Computer Science*. The 8th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2017) / The 7th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2017) / Affiliated Workshops 113 (Jan. 1, 2017), pp. 186–193. ISSN: 1877-0509. DOI: 10.1016/j.procs.2017.08.339 *Cited on page 104.*

[210] Datatype. *OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition)*. URL: https://www.w3.org/TR/owl2-syntax/#Datatypes *Cited on page 104.*

[211] Annotations. *OWL Web Ontology Language Reference*. URL: https://www.w3.org/TR/owl-ref/#Annotations *Cited on page 106.*

[212] M. Dave, J. Doshi, and H. Arolkar. "MQTT- CoAP Interconnector: IoT Interoperability Solution for Application Layer Protocols". In: *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*. 2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC). Oct. 2020, pp. 122–127. DOI: 10.1109/I-SMAC49090.2020.9243377 *Cited on page 109.*

[213] H. Wang et al. "A Lightweight XMPP Publish/Subscribe Scheme for Resource-Constrained IoT Devices". In: *IEEE Access* 5 (2017), pp. 16393–16405. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2742020 *Cited on page 109.*

[214] *SPARQL Query Language for RDF. https://www.w3.org/TR/rdf-sparql-query/*. https://www.w3.org/TR/rdf-sparql-query/. URL: https://www.w3.org/TR/rdf-sparql-query/ *Cited on page 109.*

[215] John Domingue and Milton Keynes. "TADZEBAO AND WEBONTO: DISCUSSING, BROWSING, AND EDITING ONTOLOGIES ON THE WEB". In: (), p. 21 *Cited on page 110.*

[216] York Sure-Vetter et al. "OntoEdit: Collaborative Ontology Development for the Semantic Web". In: vol. 2342. June 9, 2002, pp. 221–235. DOI: 10.1007/3-540-48005-6_18 *Cited on page 110.*

Titre: **Une architecture de fournisseurs de services FOG pour la gestion des données de l'Internet des Objets**

Mots clefs: FOG computing, Informatique en nuage, Internet des Objets, gestion des données, modèles de données, hétérogénéité, interopérabilité, web sémantique.

Résumé: L'Internet des objets a apporté de nombreux avantages à la vie quotidienne des gens et a eu un impact positif sur beaucoup de domaines d'application tels que l'industrie, la santé, l'environnement, les transports, la ville intelligente, la maison intelligente, etc. L'une des approches adoptées pour améliorer les performances consiste à réduire la charge sur les appareils et leur permettre d'accéder à des ressources distantes dans le Cloud. Cela a donné naissance au paradigme du Cloud Computing. Malheureusement, même si les Clouds disposent d'importantes ressources de calcul et de stockage, ils sont généralement éloignés des utilisateurs mobiles, ce qui rend difficile de répondre aux exigences des applications sensibles aux délais. Pour surmonter cette limitation, une approche possible consiste à déplacer les services du Cloud vers la périphérie du réseau. Cela a conduit à l'émergence du concept de FOG Computing. Le FOG Computing donne la possibilité de déployer des services plus près des appareils IoT et fournit des solutions pour résoudre les problèmes d'hétérogénéité et d'interopérabilité.
Dans cette thèse, nous nous concentrons sur la conception d'une architecture de service, appelé `Fog Services Provider` (FSP), basée sur le paradigme FOG, pour fournir des services pour l'IoT. Notre première contribution est de définir les composants critiques du Fog Computing et de l'IoT dans notre architecture. Nous analysons ensuite les exigences et la conception détaillée des composants de l'architecture du service Fog et du mécanisme de gestion des données. Enfin, nous proposons un `unified data model` (FSPontex) qui résout le problème d'hétérogénéité et d'interopérabilité entre différents équipements IoT, applications ou encore d'autres systèmes IoT à l'aide du concept d'ontologie.

Title: **A Fog Services Provider Architecture for IoT data management**

Keywords: Fog computing, Cloud computing, Internet of Things, data management, data models, heterogeneity, interoperability, semantic web.

Abstract: The Internet of Things (IoT) has brought many advantages to people's daily lives and positively impacted most application domains such as industry, healthcare, environment, transportation, smart city, smart home, etc. One of the approaches adopted to improve performance is to reduce the load on the devices and to allow them to access remote resources in the Cloud. This gave birth to the Cloud Computing paradigm. Unfortunately, even though Clouds have significant computing and storage resources, they are generally remote from mobile users, which makes hard to fulfill IoT latency-sensitive applications requirement. To overcome this limitation, one possible approach is to move services from the Cloud to the network edge. This leads to the emergence of the FoG computing paradigm. Fog computing brings the ability to deploy computing services closer to IoT devices and provides services that solve data heterogeneity and interoperability issues.
In this thesis, we focus on analyzing and designing a service architecture based on the Fog computing paradigm called `Fog Services Provider` (FSP). Our goal is to exploit the Fog computing system's useful characteristics and provide efficient support services for IoT devices. Our first contribution is to define some of the critical components in Fog computing and IoT. We then analyze the requirements and detailed design of the Fog service architecture components and the data management mechanism for IoT heterogeneous devices. Finally, we propose a `unified data model` (FSPontex) that solves the problem of heterogeneity and interoperability in different applications and with other IoT systems using web technology with the help of `Ontology`.

École doctorale Galilée, Université Sorbonne Paris Nord,
Laboratoire de Traitement et Transport de l'Information (L2TI).
Villetaneusse, France