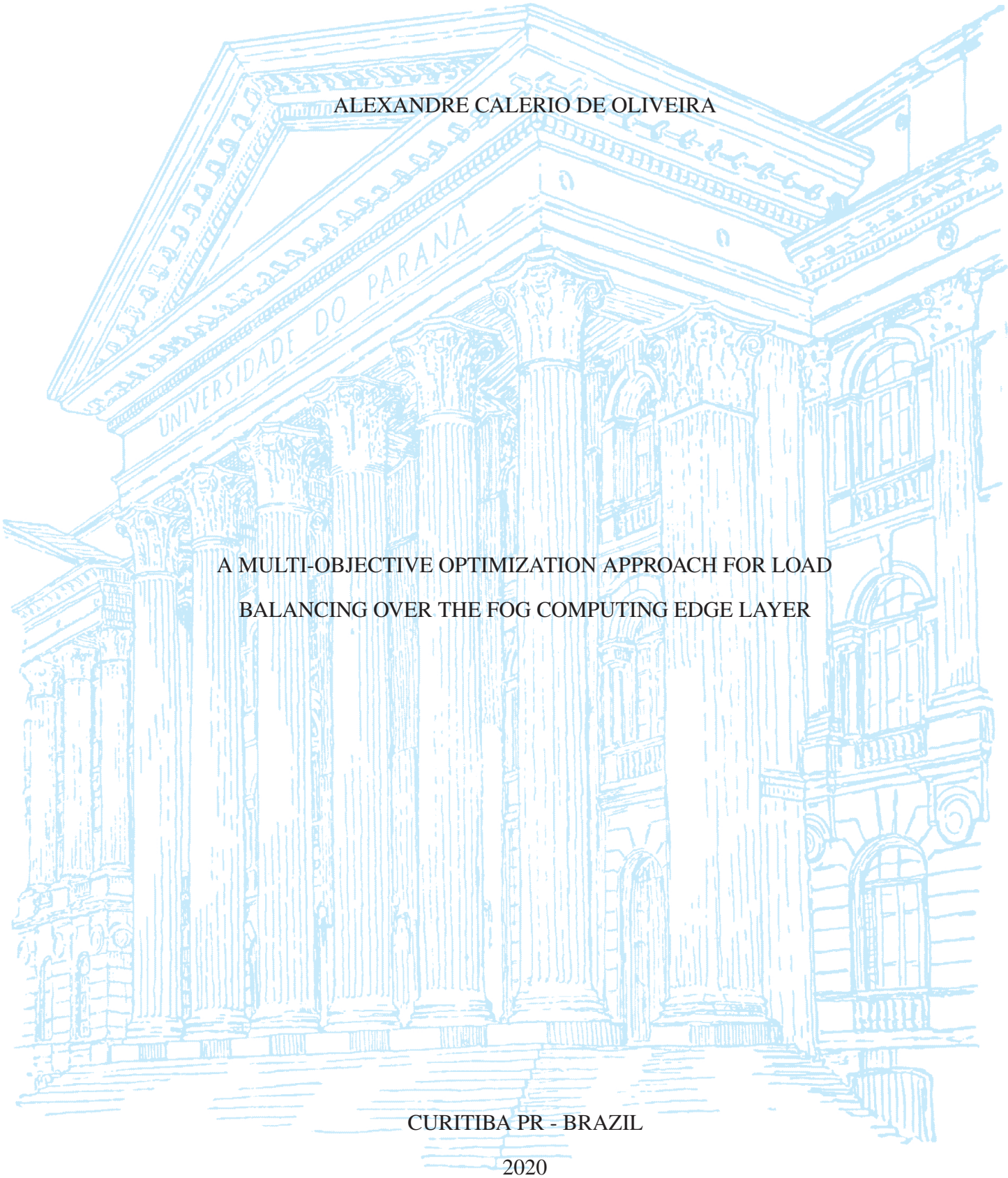
FEDERAL UNIVERSITY OF PARANÁ

ALEXANDRE CALERIO DE OLIVEIRA



A MULTI-OBJECTIVE OPTIMIZATION APPROACH FOR LOAD BALANCING OVER THE FOG COMPUTING EDGE LAYER

CURITIBA PR - BRAZIL

2020

ALEXANDRE CALERIO DE OLIVEIRA

A MULTI-OBJECTIVE OPTIMIZATION APPROACH FOR LOAD BALANCING OVER THE FOG COMPUTING EDGE LAYER

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Field: *Computer Science*.

Advisor: Eduardo Todt.

CURITIBA PR - BRAZIL

2020

CATALOGAÇÃO NA FONTE – SIBI/UFPR

O48m Oliveira, Alexandre Calerio de

A multi-objective optimization approach for load balancing over the fog computing edge layer [recurso eletrônico]/ Alexandre Calerio de Oliveira, 2020.

Dissertação (Mestrado) - Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.
Field: *Computer Science*.
Advisor: Eduardo Todt

1. Internet das coisas. 2. Computação em nuvem. I. Todt, Eduardo. II. Universidade Federal do Paraná. III. Título.

CDD 004.678

Bibliotecária: Vilma Machado CRB9/1563

SETOR DE CIENCIAS EXATAS
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO INFORMÁTICA - 40001016034P5

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **ALEXANDRE CALERIO DE OLIVEIRA** intitulada: **A Multi-objective Optimization Approach for Load Balancing Over the Fog Computing Edge Layer**, sob orientação do Prof. Dr. EDUARDO TODT, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua aprovação no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 17 de Março de 2020.

EDUARDO TODT
Presidente da Banca Examinadora (UNIVERSIDADE FEDERAL DO PARANÁ)

JOAO ALBERTO FABRO
Avaliador Externo (UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ)

LUIZ CARLOS PESSOA ALBINI
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

MICHELE NOGUEIRA LIMA
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

MINISTÉRIO DA EDUCAÇÃO - UNIVERSIDADE FEDERAL DO PARANÁ - Setor de Ciências Exatas - Coordenação de Pós-Graduação em Informática

ACKNOWLEDGEMENTS

Ao Prof. Eduardo Todt pela orientação, pelo apoio e pela motivação durante todo desenvolvimento deste trabalho.

Aos meus amigos e colegas, em especial aos do laboratório VRI, Leonardo, Felipe, Cides, Ivan, Eric, Ricardo, Daniel, Laura, Alessandra e Beatriz pela amizade e pelo companheirismo durante esses anos.

Agradeço à CAPES pelo suporte financeiro.

Por fim, um agradecimento especial à minha família e à Kelin pelo suporte e apoio durante todos esses anos, especialmente durante os momentos mais difíceis.

Muito obrigado!

RESUMO

A Computação em Névoa é um paradigma emergente proposto com o intuito de estender a nuvem até a borda da rede para suportar o aprimoramento da Internet das Coisas (IoT). Devido às características dinâmicas da interação Névoa-IoT, as taxas de requisições assim como as demandas e disponibilidades de recursos em diferentes *gateways* podem divergir significativamente, contribuindo para degradar o desempenho da névoa, sendo necessários mecanismos efetivos para evitar que alguns dispositivos fiquem sobrecarregados enquanto outros permaneçam subutilizados. Além disso, os dispositivos de borda da camada da névoa são geralmente descritos como restritos em recursos, exigindo abordagens computacionalmente leves para que sejam mantidos seus desempenhos sob alta demanda. Este trabalho apresenta uma abordagem multiobjetivo que visa solucionar o problema do balanceamento de carga na camada de borda da névoa, considerando também a garantia de atendimento ao critério da latência máxima esperada para a execução dos serviços disponíveis e a otimização de intensidade do sinal da rede sem fio dos dispositivos de IoT. Na solicitação de serviço feita por um dispositivo de IoT, o agrupamento dos dispositivos da borda da névoa ao alcance de transmissão formam uma partição, onde cada dispositivo de névoa envolvido sintetiza dados sobre seus recursos computacionais e a intensidade do sinal recebido em um único parâmetro, o qual é repassado a um dispositivo dedicado da névoa que decide qual dispositivo da partição é o mais adequado para atender a solicitação recebida. Em conjunto, um segundo processo é executado nos dispositivos computacionais de borda da névoa, onde as relações entre ofertas e demandas temporais por recursos computacionais são ajustadas de forma a maximizar a taxa de utilização de recursos nos dispositivos como também garantir que os serviços sejam executados dentro dos seus limites máximos de tempo esperados. Para avaliar a abordagem proposta, foi construído um simulador que contempla os requisitos de detalhamento da rede de borda e da arquitetura envolvidos, fornecendo ferramentas apropriadas para avaliar características relevantes da rede e dos dispositivos de borda bem como as dimensões que envolvem a abordagem de balanceamento de carga multiobjetivo. Os resultados das simulações demonstraram a abordagem utilizada como sendo eficaz, aumentando o nível médio de sinal na comunicação sem fio na borda da rede, garantindo que os serviços sejam executados abaixo de seus limites máximos de tempo bem como minimizando as discrepâncias na distribuição de cargas entre os dispositivos de borda da névoa, superando neste último quesito resultados encontrados na literatura.

Palavras-chave: Internet das Coisas. Computação em Nuvem. Computação em Névoa. Computação em Borda. Arquitetura de Névoa. Balanceamento de Carga. Aplicações Sensíveis à Latência. Aplicações de Tempo Real.

ABSTRACT

Fog Computing is an emerging paradigm, proposed to extend the cloud to the edge of the network as enhanced support to the Internet of Things (IoT). Due to the dynamic characteristics of the Fog-IoT layers interplay, task arrival rates and consequently resources demands at different gateways can be significantly different over the Fog edge layer, contributing to degrade the Fog performance, making necessary effective mechanisms to avoid some computing devices to become overloaded while others are underloaded. Furthermore, Fog edge computing devices are usually described as having constrained computational resources, demanding lightweight approaches to preserve their performances under high load demands. This work introduces a multi-objective approach that addresses the load balancing issue in the Fog edge layer, also considering the fulfillment assurance for the maximum expected execution time for the available services, as well as the wireless signal strength optimization in the IoT-Fog communication. Under a service request from an IoT device, the group of Fog edge devices that are in its wireless communication range forms a partition, where each one of them synthesizes data about its computational resources and the received signal strength into a single parameter to be relayed to a dedicated Fog device, that is responsible to decide which partition's device is the most feasible to respond to the incoming request. Jointly, a second process runs in the Fog edge computational devices with the purpose to maximize their computational resources usage rate by fitting the temporal resources demands to the device's resources availabilities, as well as monitoring and ensuring that services are performed within their maximum expected latency. To evaluate the proposed approach, a simulator was built, which implements the details of the applicable network edge as well as the architecture requirements of the Fog system involved. It provides appropriate tools to evaluate relevant network and device features, as well as to evaluate the load balancing multi-objective optimization approach performance. The simulation results showed that the approach is effective, improving the average signal strength for the wireless communication at the edge of the network, ensuring that the services are performed below their maximum latency limits, as well as minimizing discrepancies in load distribution among the Fog edge devices, and in this last item, surpassing results found in the literature.

Keywords: Internet of Things. Cloud Computing. Fog Computing. Edge Computing. Fog Architecture. Load Balancing. Latency-sensitive Applications. Real-time Applications.

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

BF	Best Fit
BFD	Best Fit Decreasing
DCU	Data Collecting Unity
DRR	Dynamic Resource Reallocation
FD	Fog Device
FEC	Fog Edge Controller
FED	Fog Edge Device
FF	First Fit
FFD	First Fit Decreasing
IaaS	Infrastructure as a Service
IIoT	Industrial Internet of Things
IoBT	Internet of Battlefield Things
IoT	Internet of Things
IoTD	Internet of Things Device
LQI	Link Quality Indicator
M2M	Machine-to-Machine
MANET	Mobile Ad Hoc Network
MQTT	Message Queueing Telemetry Transport
PaaS	Platform as a Service
QoS	Quality of Service
RSSI	Received Signal Strength Indicator
SaaS	Software as a Service
SRA	Static Resource Allocation
VM	Virtual Machine
XaaS	Anything as a Service
WSAN	Wireless Sensors and Actuators Network

CONTENTS

1 INTRODUCTION

In the last years, significant technological advances in sensors, wireless communication, and microprocessors have enabled the design of small-size low-power and low-cost devices that can be networked or connected to the Internet empowering the emerging paradigm of the Internet of Things (IoT). The IoT paradigm aims to make everyday objects connected to the Internet affording a novel type of interaction among humans, environments, and machines embracing all kinds of activities and phenomena being those, healthcare, urban life, computing, business, industry, energy, transport and logistics, sport, agriculture, and many others [4, 5]. It enables the implementation of the Smart Cities concept [3, 6], the efficient management of infrastructures, and provides a variety of services to enhance the quality of life and the utilization of resources [7]. The evolutionary rate of the IoT concept and its ever-growing scenario applications create increasing demands for processing capabilities, which is translated in a need for deploying a large number of devices, in order to serve tasks cognitively for strict computational purposes, improving the Quality of Service (QoS), latency, and efficiency [8].

The IoT traditional approaches usually interplay with the Cloud Computing paradigm to fulfill its computational, storage and analytics needs [9]. Clouds have been the prominent and dominant infrastructures to develop, debug, deploy, and deliver pioneering business and IT applications as it offers a *pay-as-you-go* computing model, enabling convenient on-demand network use of a shared pool of configurable computing resources and provides ubiquitous access to the content and services [10, 11]. However, Cloud Computing cannot fulfill all IoT needs due to its own drawbacks. The fundamental limitation lies in the connectivity over the Internet between the cloud and the end devices, what is impracticable for a large set of cloud-based applications such as the latency-sensitive ones [12]. Those applications and services usually suffer unacceptable round-trip latency when data is exchanged between end devices and the cloud data center through multiple gateways. Some examples of latency-sensitive applications include connected vehicles [13], fire detection and firefighting [14], smart grid [13], and content delivery [15]. Furthermore, the massive data generated by sensors when transmitted to the cloud may overload the network, also, the transmission of sensitive data to a distant location may become undesirable due to security reasons.

To fulfill cloud's shortcomings, the Fog Computing paradigm extends the cloud to the edge of the network, making it suitable to enable specific IoT domains which require mobility support, geo-distribution, location awareness and low latency [16]. The fog platform is hierarchical, where the fog is the middle layer of a three-layered general architecture, lying in between the Cloud and the IoT layers. It is expected from the fog architecture to have as many internal layers as the applications and services demand [3], however, its first internal layer is designed for machine-to-machine (M2M) interaction, to collect, process the data, and issues control commands to the actuators, if required [16], many times interplaying to the IoT level through 802.14.5 networks [3, 17].

Fog Computing has been shown to be an important IoT service and application enabler, however, to accomplish the IoT full potential, it is still necessary to address several challenges and develop conceptual and technological solutions to tackle them. Resource management methods, messaging architectures and smart task profiling for different scenarios, are some examples of the still open issues for Fog Computing [8]. There are many studies in Fog Computing that present several functional and architectural approaches to tackle the efficient resources management, latency, and load-balancing issues.

Kapsalis *et al.* [8] have proposed a division in the fog layer containing distinct nodes types where different resources demands are allocated accordingly to the nodes' characteristics. Xu *et al.* [18] have proposed the clusterization of services in different nodes categories and an additional architectural layer, the service layer, which offloads and migrates the services to the appropriated nodes according to the time urgency and the resources amount matching. In another study [19], the authors use a similar approach but applied to VM (virtual machines) scheduling, leveraging the VM live migration technique. Jia *et al.* [20] have targeted the load balance issue among small-scale computer clusters close to the edge of the network, named cloudlets, by introducing a model to capture the response times of offloaded tasks, and by their proposed optimization algorithm, find an optimal redirection of tasks between cloudlets such that the maximum of the average response times of tasks at cloudlets is minimized. Shi *et al.* [21] have targeted the load balance issue into the Industrial Internet of Things (IIoT) context by proposing to integrate the Fog Computing to the cloud-based IIoT, so building a cloud-fog integrated IIoT (CF-IIoT) network to achieve ultra-low services response latency, employing in their solution a genetic algorithm to optimize the load balancing problem of the distributed cloud-fog network. Wang *et al.* [22] have proposed a strategy to reduce latency and enhance reliability of tasks in the context of the Internet of Battlefield Things (IoBT) through a "combat cloud-fog" network architecture, where a distributed approach through a generalized diffusion algorithm is used to promote the load balance on the edge of the fog environment, achieving tasks low latency.

As diverse as the IoT use cases and the Fog Computing employment, are the proposed solutions to tackle diverse inherent issues, which may comprise architectures, algorithms, functional enhancements and many other approaches. It is possible to notice that many proposed approaches have in common to preserve essential characteristics of the Fog Computing paradigm, as low latency, and in many studies the computational load distribution concept is involved as part of the solution. Also, the presented solutions are usually focused on the Fog inner-layer or in the Fog-Cloud layer relationship being uncertain if they also can be feasible to be applied to the Fog edge nodes if they act as computing devices. Furthermore, to the best of our knowledge, an approach for the Fog Computing environment that also considers the IoT-Fog connection links as an optimization dimension has not been proposed yet.

This work introduces a multi-objective approach to the load balancing issue targeting the Fog edge layer computing devices, which also considers as optimization parameters the running services latency assurance, and the IoT-Fog wireless link quality as well.

In particular, the contributions of this work are as follows:

1. An architectural generic model based in Fog edge partitions which are formed due to the IoT device wireless range. The only Fog devices which are capable to interact with a given IoT device are those that are within its wireless communication range. The model takes those in-range devices as a Fog partition where, on a multi-parameter selection process, the most feasible fog device is chosen to support the requesting IoT device. Furthermore, the Fog edge device entity is modeled in terms of modular functional structures, which makes it suitable to be easily reformulated, expanded, improved as it is proper to be adaptable to simulator environments and real-world deployments as well.

2. A lightweight multi-objective load balancing approach. The presented method aims to minimize discrepancies in load distribution among the Fog edge computing devices, as well as optimizing the IoT-Fog wireless link and ensuring that the requested services are executed within latency limits. It is done in two steps, where the first step considers the link optimization and computational resources allocation, and in the second step, it promotes the matching between the temporal availability and the demand for computational resources in the device.

3. A discrete event simulator, modeled under the architectural viewpoint of Fog edge partitions. Most of the simulators for Fog environments were built as extensions of Cloud simulators, keeping their focus on the Cloud-Fog relationship where some details of the IoT-Fog interplay are still missing. Besides inherent utility to the validation of this work, a specific simulator modeled under the IoT-Fog layers interplay may be a valuable tool to simulate and evaluate mechanisms on the edge of the network in Fog environments.

The remainder of this work is structured as follows:

- Chapter 2 presents fundamentals aspects and characteristics related to the Fog Computing paradigm and its interplay between Cloud Computing and the Internet of Things.
- Chapter 3 presents some related studies that comprise challenges, methodologies, architectures and functional approaches that were relevant to build the architectural and functional models presented in this work.
- Chapter 4 presents an overall Fog edge system modeling, and the load balancing approach with multi-objective optimization to the Fog edge layer, its structure, components, and operation.
- Chapter 5 presents the discrete event simulator developed to evaluate the multi-objective load balance technique presented in this work, with details of its design and implementation.
- Chapter 6 presents the evaluation of the multi-objective load balancing method together with an analysis of the basic characteristics that permeate the Fog edge layer that are relevant to the presented method.
- Chapter 7 concludes this work and presents some perspectives for future works.

2 RELATED CONCEPTS AND TECHNOLOGIES

This chapter presents fundamental aspects and characteristics related to the Fog Computing paradigm. The following sections address basics on the Internet of Things and Cloud Computing, followed by an overview of the Fog Computing paradigm and its interplay between Cloud Computing and the Internet of Things. Also, it is presented the Fog internal architecture by the functional and deployment viewpoints. The chapter finalizes with a section dedicated to a discussion about the presented concepts and their relation to this work.

2.1 THE INTERNET OF THINGS

Despite the diversity of research on Internet of Things (IoT), its definition still remains fuzzy [1]. Common characteristics lead the IoT concept to be understood as a multidisciplinary paradigm in which objects, able to sensing and eventually interacting with the physical world, can be networked together and connected to the Internet to enhance functionalities, provide new services, and increase the efficiency of processes and systems. Such objects or simply *"things"*, in most cases are constituted by standalone or embedded sensors and actuators characterized by resource constrained computational devices (i.e., low memory, reduced battery capacity, and limited processing power) [23], linked through machine-to-machine (M2M) connections, mostly using a wide range of technologies in wireless links [24], which are logically gathered together around specific application domains [25].

The flexibility of the IoT concept allows the paradigm to embrace all kinds of activities and phenomena, provides a variety of services to enhance the quality of life and the utilization of resources [4, 5, 6, 25], that is achieved through the employment of many technological areas, ranging from enabling technologies and components to several mechanisms to effectively integrate these low-level components. The IEEE Internet of Things Initiative[1] provides an overview on the technological and social aspects related to the IoT, which can be visualized in Figure 2.1. Each listed technological aspect embraces many areas which have been intensively studied with the intent to overcome several challenges in order to achieve the IoT full potential when addressing specific services and applications for determined business interests.

Despite the technological and architectural heterogeneity involving the IoT ecosystems, there are some basic processes that are presented in most the cases. Taking as reference Figure 2.1, at the *Enabling Technologies and System Architecture* level it is promoted the integration between the physical and cyber worlds through the IoT devices which are responsible for acquiring, processing, storing and transmitting or receiving data. When the generated data meets the *Software Architecture* level is where it meets the Big Data [26] methods making the IoT data meaningful to the *Services and Application* level through IoT data aggregation and analytics.

Depending on the required services and applications, the Big Data's processes may demand large amounts of storage and processing power from the host system to perform its tasks on data aggregation and analytics. Such requirements may be fulfilled by the Cloud Computing paradigm which offers networking and elastic computational resources, which are accessible through a variety of *pay-as-you-go* service models [10], making it suitable to interplay with many of the IoT ecosystems. However, for some applications, especially for latency-sensitive ones which require real-time processing and response, Cloud Computing may fail to fulfill its needs due to its own architectural characteristics [13]. As response to those drawbacks, the

[1] https://iot.ieee.org

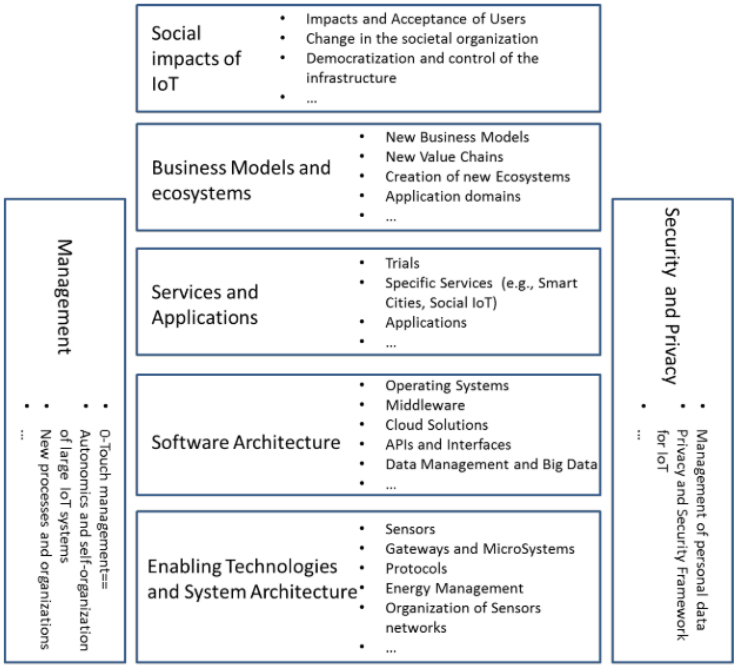


Figure 2.1: Technological and social aspects related to IoT. Source [1]

Fog Computing paradigm was proposed to extend the cloud to the edge of the network bringing real-time and local computation, making it suitable to attend some IoT specific domains [16].

Cloud Computing and Fog Computing paradigms are discussed in the following sections.

2.2 CLOUD COMPUTING

Along the last decade, Cloud Computing has been a paradigm that has gained prominence in both business and academia. Cloud data centers are large pools of highly accessible virtualized resources that can be dynamically reconfigured for a scalable workload. This reconfigurability is beneficial for clouds services that are offered with a pay-as-you-go cost model promoting ubiquitous, on-demand network access to shared computing resources [10, 11]. The academic interest in Cloud Computing paradigm lies mainly in the objectives of optimizing platforms, concepts, services, access, energy consumption, resource allocation, etc, where several studies are produced annually. In the business area it is a powerful enabler for a technological cost-effective model which allows users to conveniently access remote computing resources and data management services being only charged for the amount of resources they use. The cloud-based services may be offered in diverse formats and usually are based on a certain type of level or hierarchy in XaaS format (Anything as a Service) being the main types based in the concepts of infrastructure, platform, and software as services (IaaS, PaaS, SaaS). According to NIST (National Institute of Standards and Technology) [11], those cloud services categories must accomplish the following characteristics:

IaaS (Infrastructure as a Service)
"The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls)."

PaaS (Platform as a Service)
"The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment."

SaaS (Software as a Service)
"The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure (typically done on a pay-per-use or charge-per-use basis). The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings."

Cloud services can be utilized for distinct use cases for a variety of end users. Figure 2.2 illustrates the relationship among IaaS, PaaS, and SaaS service models with the underlying cloud infrastructure, and illustrates what portion of the application stack is managed by cloud providers and which portion is managed by the service users.

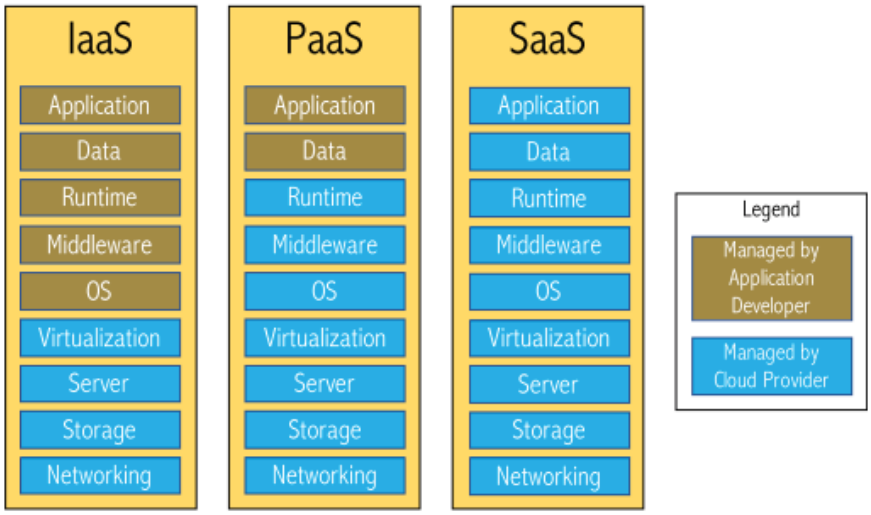


Figure 2.2: The Iaas, PaaS and SaaS cloud service models and their classifications relative to which portion of the application stack that is managed by cloud providers. Source [2]

Cloud Computing aggregates a range of technologies which are used to implement the concepts of virtualization, resource allocation, bandwidth utilization, distributed computing, web computing, load balancing, networking, and software applications. Those technologies provide dynamic scalability and ubiquitous computing for different categories of data and applications making the paradigm suitable to interplay with the IoT paradigm, fulfilling its needs for processing, aggregation, storage and analytics of massive data amount generated by large areas populated by geo-spreaded sensors. However, for some specific use cases, for instance, such ones involving the IIoT (Industrial Internet of Things) concept which require time-sensitive applications and real-time analytics, the Cloud Computing support fails to provide the appropriate low latency needed [21, 27]. Furthermore, the massive data generated by sensors may easily overload the network, and the transmission of sensitive data to a distant location may become undesirable due to security reasons.

2.3 FOG COMPUTING

Fog Computing is an emerging paradigm, suitable to serve the particular needs of IoT networks enabling applications that support faster real-time processing of time-sensitive data by the deployment of computational devices at the edge of the network [16]. In contrast to the cloud, fog platforms have been described as dense computational architectures at the network's edge. The main characteristics of fog-based platforms reportedly include low latency, faster response, context awareness, and wireless communication, enabling services that provide real-time analytics and improved security.

Despite similarities, Fog Computing is often (and erroneously) called Edge Computing, but there are key differences. Fog works with the cloud, whereas edge is defined by the exclusion of cloud. Fog is hierarchical, where edge tends to be limited to a small number of layers [3]. In the IoT nomenclature, *edge* refers to the local network where sensors and IoT devices are located, that is, the immediate first hop from the IoT devices (not the IoT nodes themselves), such as the WiFi access points or gateways [2]. If the computation is done on IoT devices themselves, this computing paradigm is referred to as *Mist Computing* [28]. The *OpenFog Consortium* [3] defines Fog Computing as: *"a horizontal system-level architecture that distributes computing, storage, control and networking functions closer to the users along a cloud-to-thing continuum"*. By definition, fog comprises the middle-layer of a three-layered architecture, as shown in Figure 2.3.

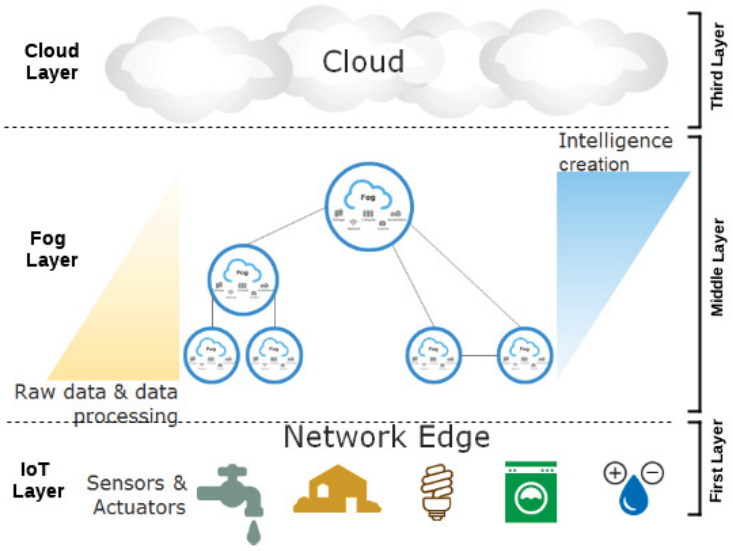


Figure 2.3: Fog Computing as an intermediate layer in a three-tier architecture. Adapted from [3]

Figure 2.3 shows sensors and actuators representing the IoT layer, the fog instances standing at the architecture's middle layer while the cloud represents the third layer. Lower levels in the architecture are usually permeated by raw data that need basic processing while the data insights grow toward the upper levels. That condition reflects the number and computational resources capability of the devices regarding their position on the fog's levels. As ascending into the hierarchy from edge to cloud the devices are present in reduced numbers, however, their computational capability grows, in order to produce even more accurate and meaningful insights from the ascending data at the lower levels.

2.3.1 The Fog Layer Internal Architecture

In most fog deployments, the fog layer may be formed by several internal device layers. The number of internal fog layers is usually determined by the scenario requirements, which may include, for instance, the amount and type of computational work required by each layer, number of IoT devices, computational capabilities of the devices at each layer, network latency between fog devices and latency between sensing, processing information, and actuation, reliability or fog device's availability [3]. One of the many scenarios that may evidence a Fog internal multi-layer architecture is shown in Figure 2.4 where a smart city scenario is presented. Each building, neighborhoods, and regions are connected to provide an infrastructure that may be optimized for service delivery. It is also shown that fog devices may have the ability to communicate vertically and horizontally through the architecture, being linked to form a mesh network to provide load balancing, resilience, fault tolerance, data sharing, and minimization of cloud communication.

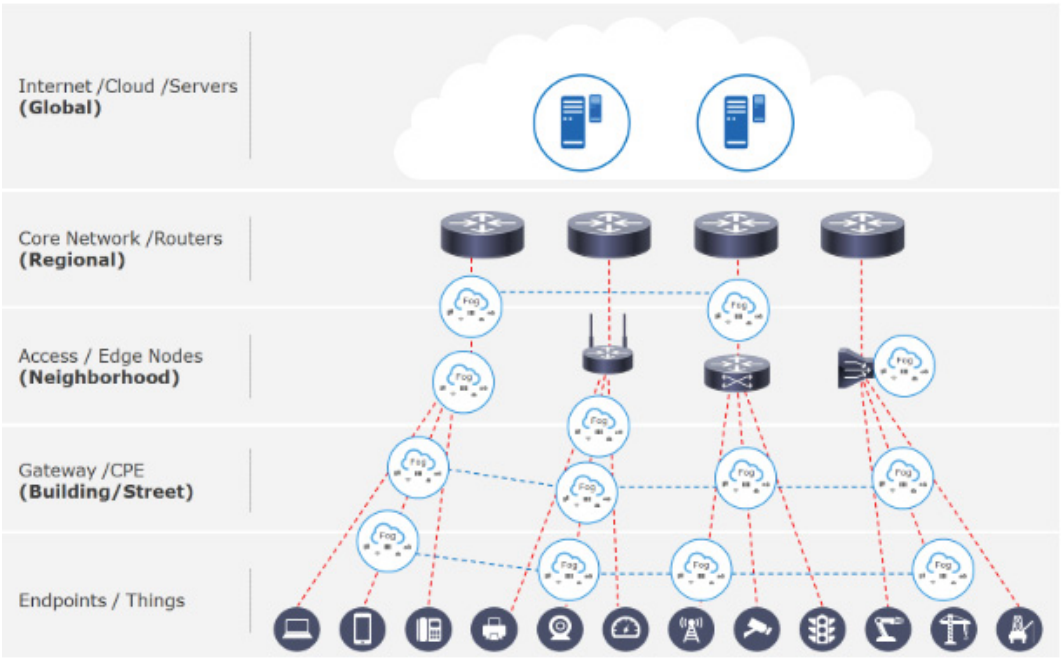


Figure 2.4: Example of fog internal multi-layer architecture for the smart city scenario. Source [3]

Real-world Fog deployments may have more or fewer levels. Different application use cases may use a fog hierarchy differently. For the smart city example, there may be Fog devices in a region, neighborhood, street corner, and building level. For a smart factory scenario, the hierarchy may be divided by assembly lines, manufacturing cells, and machines. For use cases like agriculture, connected cars, and remote weather stations, the related applications and services may have only the fog edge devices involved which may provide only basic services, as connectivity and protocol translation. These use cases may employ the cloud for entire data processing operations due to their constrained environments, where the deployment of the fog infrastructure may not be feasible or economically viable.

The fog layer, regarding the deployment viewpoint, may have their devices classified according to the position on which they stand into the architecture. Figure 2.5 illustrates three distinct devices' categories: the *cloud level devices* in the upstand of the fog architecture, the *edge level devices* which are in direct communication to the sensors and actuators, and the *intermediate level devices* which stand in between edge and cloud devices.

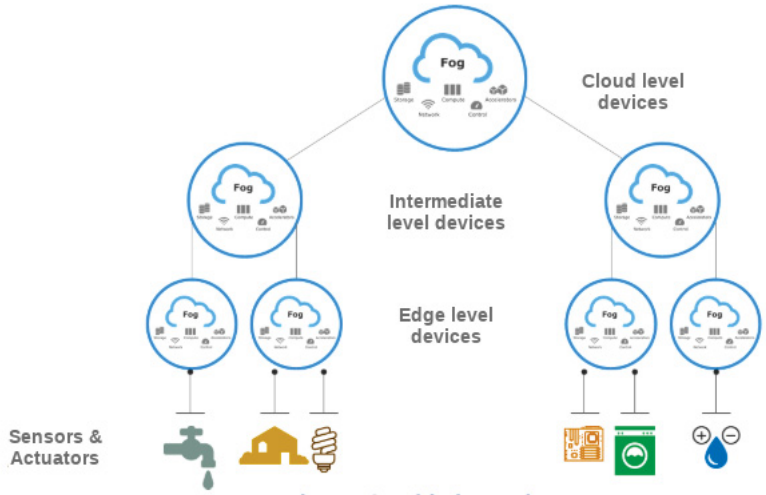


Figure 2.5: Functional viewpoint of the fog internal architecture and devices classification according to their respective positions into the architecture. Adapted from [3]

Generally, the basic functional attributions to these fog computing devices categories includes, but they are not restricted to the following functions:

Cloud level devices
Typically focused on aggregating data and turning the data into knowledge.

Intermediate level devices
Provide communication to the cloud level nodes as support to the fog edge nodes which may include tasks offloading, data aggregation and depending on the application may assume control and data insights tasks.

Edge level devices
These devices are the entry point to the fog and they interplay to the IoT nodes through sensor data acquisition/collection, actuators control, network protocol translation, multi-technology wireless communication, authentication, orchestration, and mobile support.

The fog internal layers, like the functional boundaries between the fog devices, are fluid. The fog devices can be physically deployed in multiple combinations based on the architecture of the domain-specific solutions. Depending on the scenario and application, the fog devices may assume more than one class of functional responsibility. Regardless of the scenario, the interplay to the IoT devices and consequently the related functional responsibilities are strictly necessary, what makes the fog edge layer the only mandatory fog layer for all the cases.

2.4 SUMMARY AND DISCUSSION

This chapter presented some concepts and technologies related to the Fog Computing paradigm. The first section provides an overview of the Internet of Things concept and establishes its relation to the Cloud Computing and Fog Computing paradigms. Next, some basics on the Cloud Computing paradigm is exposed through its main characteristics and service models that makes it suitable to interplay with, and to support some Internet of Things domains. Finally, the Fog Computing paradigm is discussed, passing through its basic characteristics and its role in the Cloud-Fog-IoT architecture, addressing its internal architecture, providing the smart city concept as a use case example, followed by its functional and devices deployment viewpoints, emphasizing the importance of the Fog edge layer, on where this work takes place in the Fog architecture.

The Fog Computing paradigm exists within the union of two other paradigms, the Internet of Things and Cloud Computing, comprising a three-layered general architecture intended to be an enabler to the IoT applications that require real-time interactions, location awareness, and geo-distributed processing. By definition, the fog is an extension of the cloud, that makes natural extending to the edge of the network cloud's virtual computing environments as its involved services models, being feasible the adoption of the concepts of IaaS, PaaS, SaaS for the Fog Computing layer. To provide those services models, the cloud data centers usually adopt the virtual machines environments with well-determined amounts of computational and networking resources for running the hosted applications. Thus, although there is no formal definition for modeling computational resources in Fog Computing devices, it seems feasible to use a model where discrete units of processing capacity, storage, and volatile memory are considered, analogous to a virtual machine.

On the Fog-IoT interplay, the flexibility of the IoT concept and its applications' requirements many times may reflect architectural approaches in the fog layer to tackle some IoT-specific domain problems. Due to this characteristic, widely present in the literature, it is often unclear whether an architectural solution may be feasible to be applied to another IoT-domain problem, or even for an agnostic analysis of the characteristics and behaviors of an IoT-Fog layers interplay. Regardless of the application scenario, the IoT devices tend to request data or send data to the fog layer periodically where the fog devices have the responsibility to process or to forward the IoT demands to the proper computing nodes.

This behavior leads to a general model of the Fog layer, in which the IoT layer is perceived as a set of local and temporal requests for computational resources, being the Fog layer responsible for meeting these demands, either by locally processing or offloading it to another instance, and the place where domain-specific architectural or algorithmic solutions are built over.

The next chapter addresses some studies that comprise challenges, methodologies, architectures and functional approaches that are relevant to this work.

3 RELATED WORK

This chapter presents works related to the development of this work, some of which served as a basis for the elaboration of the presented Fog border model. The following works address issues related to the properties and optimizations of wireless communication for the IoT devices, as well as methodologies for optimizing the main functional properties of the Fog Computing paradigm, emphasizing the minimization of computational loads discrepancies among Fog devices and the execution's latency reduction for tasks performed in Fog environments.

Bonomi *et al.* [16] have first presented the Fog Computing paradigm, extending the Cloud Computing paradigm to the edge of the network, as an appropriate platform for a number of critical Internet of Things services and applications. As characteristics that define the Fog Computing the authors mention, low latency and location awareness, wide-spread geographical distribution, mobility, very large number of nodes, predominant role of wireless access, a strong presence of streaming and real-time applications, and heterogeneity. As examples of motivating Internet of Things scenarios are mentioned Connected Vehicle, Smart Grid, Smart Cities, and Wireless Sensors and Actuators Networks, presenting inherent benefits to these scenarios when assisted by a Fog environment. The authors conclude by opening an invitation for collaborative works regarding correlated subjects to the Fog Computing paradigm, such as architectures for massive infrastructures for computing, storage, and networking devices, orchestration and resource management of the Fog nodes, innovative services and applications to be supported by the Fog.

Kapsalis *et al.* [8] presented an architecture in four layers for a fog-enabled platform that is responsible for allocating and managing the computational resources needed to host application components. The fog stratum has two device categories, the *stable edge devices* and the *mobile edge devices*. The stable edge devices include an architectural module called *Fog Broker* which is responsible for enriching the messages received from the lower layers by incorporating additional information, and they are also responsible for task management and allocation so performing the workload balance, while the mobile edge devices are responsible for communicating with the devices in the lower layer. The Workload Balancer incorporates a score based procedure that allows the Fog Broker to sort the available hosts and choose the most suitable one for each component based on its current utilization, latency, battery, and the required resources. For each task, the balancer algorithm creates a list of eligible hosts and excludes the unfeasible ones (not enough RAM or battery). Subsequently, it applies the score function to each host and selects one of them to allocate the task for execution. All the messages exchanged between the fog nodes utilize a distributed communication model (Pub/Sub) and standardized messages in order to create a cooperative fog layer, where multiple devices and networks can be involved. Under the test criteria and different scenarios, the simulation results showed that the proposed architectural approach enables fog networks to manage efficiently when handling time-critical tasks.

Xu *et al.* in [18] proposed a system framework for fog computing and the load balance analysis for various types of computing nodes and sequentially, a corresponding resource allocation method in the fog environment, designed through static resource allocation and dynamic service migration intended to achieve the load balance for the fog computing systems. The framework consists of four layers: the IoT application layer, the service layer, the fog layer, and the cloud layer. The IoT application layer contains a large number of service requirements that need to be responded to by selecting appropriate computing nodes according to the time

urgency and the resource amount. The fog layer is divided into two parts, one consisting of the edge computing nodes, and another, containing the intermediate computing nodes. The services with the highest time urgency and less computation density can be calculated by the edge computing nodes in the fog layer, while the less urgent tasks could be executed in the intermediate fog computing nodes, and the cloud layer is appropriate for hosting the latency-tolerant tasks which demand large amounts of computational resources. The proposed method consists of four steps: First, the fog services are classified as several sets based on the resource requirements. Second, the nodes are classified according to their capacity to host the services by detecting the spare space of computational resources units of all the computing nodes. In the third step, it is promoted the static resource allocation, where subsets of services, on which each subset contains services with a similar request start time, are allocated into the computing nodes identified as proper to host these services, so initializing the resource allocation process. The fourth and last step consists of the dynamic resource allocation, which monitors nodes' spare space of computational resources and services demands in running time promoting a matching by allocating, relocating or migrating services among nodes in order to achieve a consistent resource utilization. For the tests, the authors used four different scales of datasets, and the proposed method was tested against four other allocation algorithms, FF (First Fit), BF (Best Fit), FFD (First Fit Decreasing) and BFD (Best Fit Decreasing). Evaluations were performed considering different metrics: the employed computing nodes, resource utilization, and load balance variance, where the proposed method achieved superior results on all tested metrics against the tested allocation algorithms.

Xu *et al.* proposed a Virtual Machine (VM) scheduling method for load balancing in fog-cloud computing [19]. The authors propose a heuristic VM scheduling method designed through VM placement and dynamic VM scheduling by leveraging the VM live migration technique, where each VM represents an application provided by the fog instance. During the VMs placement moment, the proposed algorithm takes into account the instant computing node's resources usage, the required VM instance type, the requested amount of VM instances, the execution start time, and the duration time of the requested service, respectively. Also, the algorithm leverages the idea of Best Fit Decreasing (BFD) to realize the process of computing node matching for the application's VMs. After the first VMs' allocation, a global VM placement and scheduling method take place to achieve load balancing. At the scheduling period, the execution finish time for each type of VM is traced, and the computing node's spare space are released by the completed fog service. Then, at the finish time of each application VM, the spare resources spaces of the computing node are updated and the nodes are sorted in the decreasing order of spare space. To improve load balancing for each type of computing node, the application VMs in the computing nodes with the lower resource usage could be migrated to the computing nodes with higher resource usage. After VM scheduling, some computing nodes with more spare space can be set as vacant and shut down further or set to sleeping mode to save energy consumption. For the tests, the authors used four different scales of datasets, three different capacity nodes types, and the proposed method was tested against two other allocation algorithms, FFD (First Fit Decreasing) and BFD (Best Fit Decreasing). Evaluations were performed considering different metrics: number of services for each type of computing node, average resource utilization, and average load variance, where the proposed method achieved superior results on all tested metrics against the tested allocation algorithms.

Srinivasan and Levis [29] presented an investigation over the Received Signal Strength Indicator (RSSI) as an estimator of the link quality. The RSSI is compared against another parameter, the Link Quality Indicator (LQI), which is implemented on the CC2420 radio, based on the IEEE802.15.4 standard. The adopted methodology for the tests was to measure RSSI

and LQI against the Packet Reception Rate (PRR) parameter to compare them under the same background in order to determine the RSSI and LQI correlation. Under the performed tests, the authors conclude that the RSSI has very small variance compared to LQI for any link over time, suggesting that the RSSI measured for a single packet provides a good estimate of the average RSSI over many packets in that link. Also, the results indicate that RSSI is highly correlated with PRR (except when it is close to the receiver sensitivity) what makes the RSSI a link quality indicator suitable to be used as an inexpensive and agile link estimator over LQI.

Almeida *et al.* [30] proposed and evaluated a novel LoRa MAC protocol as an alternative to the LoRaWAN protocol, which is promoted by the LoRa Alliance, capable of operating in scenarios where multiple Sink stations are considered. The platform is intended for environmental data gathering in Smart Cities scenario addressing the heterogeneity of IoT devices through the employment of multi-technology communications, LoRa (based on IEEE 802.15.4) and WiFi (IEEE 802.11 family), and serves as an infrastructure for a network with opportunistic communication capabilities. To understand the network topology, for each of the three tested scenarios, an RSSI (Received Signal Strength Indicator) signal assessment between each DCU (Data Collecting Unity) and each Sink station was performed. Tests were made in order to better understand the impact of the signal strength on the packet reception for the different multi-sink evaluation tests, using different types of data packets to have distinct acquisition rates. Results show that a stronger signal affects positively the number of packets successfully transmitted, as well as reduces the number of attempts needed for each DCU to gain channel access.

3.1 SUMMARY AND DISCUSSION

Since Fog Computing was proposed by Bonomi *et al.* in 2012 [16], there has been a growing interest in research in this area. According to Mouradian *et al.* [31], sixty-eight relevant works were published and analyzed between 2013 and 2017, presenting solutions and proposals involving architectures and algorithms, where most of these works target specific use cases, and the main addressed characteristic is the services' latency, where computational load balancing and resource management techniques are often used to achieve the fog services' latency reduction. To better understand some Fog Computing inherent characteristics and their functional implications for the fog system, some use cases agnostic works were analyzed.

In [19], a heuristic algorithm for allocation, reallocation, and scheduling of VMs employing live migrations to achieve fog devices load balancing and services latency improvements was proposed and evaluated. Despite the good obtained results, the VMs migration is usually cited in the literature as a costly process in terms of computational and network resources usage, thus, remains unclear if this technique can be successfully employed in the lower layers of the fog architecture, generally reported as permeated by low computing capacity devices. In [18] it was proposed a system framework for resource allocation and load balance analysis in fog computing environments for three types of nodes, which represent Fog's edge computing nodes, intermediate computing nodes and VMs instances in the Cloud. Although the Fog edge layer is represented, the nodes' location awareness was not considered, so remaining unclear how the nodes' geo-spreading at the edge of the network influences the presented approach. In [8], a fog-enabled platform is proposed, that has the ability to forward the previously classified tasks as non-critical to the Cloud, and promote workload balance in the Fog layer for the time-critical tasks, using a scoring function to select the most viable computing nodes, which considers network latency, battery level, and device's resource usage as parameters. The authors use the task's execution time as the main evaluation criterion, remaining unclear how the workload is distributed among the

individual Fog devices, and also, the devices geo-spreading was not considered, thus, its influence is unknown.

In most works, the geographic dispersion factor for Fog and IoT devices is not considered, however, this characteristic can impact the computational workload distribution in Fog systems that comprise large areas. The devices' geographical dispersion effect can be analyzed through the edge wireless network, where the links range impose limits to devices groups to communicate with each other. In this direction, in [30] is presented an approach for multi-technology wireless networks optimization, contemplating a multi-sink station scenario, which may be considered compatible with Fog edge networks, and in [29] the RSSI is evaluated as a valid wireless link quality indicator, what may be a useful parameter for the wireless Fog edge network analysis.

The next chapter presents generic modeling of the fog edge layer for large areas, as well as an approach to multi-objective load balancing, where it considers the minimization of the variation of the workload distribution between the fog edge devices, together with the improvement of the wireless edge link, and the assurance of compliance with the maximum computation time of the services requested by the IoT layer.

4 FOG EDGE LAYER LOAD BALANCING

This chapter presents a multi-objective optimization approach for a general Fog Computing architecture, that addresses the load balance issue through a computational resource allocation method applied to the fog edge devices. The following sections present the problem statement, the system components modeling for the presented load balance issue, followed by its requirements, and the proposed load balancing method approach, with its structure and its operation.

4.1 SYSTEM MODELING AND PROBLEM STATEMENT

Depending on the application scenario, there is a subset from the total set of services provided by the fog layer that is necessarily or preferably deployed at the edge devices of the network, such as real-time data processing, data pre-processing, data caching, data collection and aggregation, protocol translation, computation offloading, etc [31, 32]. In the Internet of Things (IoT) and Fog Computing interplay, each fog edge device may be considered an IoT device entry point to the Fog which is attached to one wireless multi-technology gateway or the FED is the gateway device itself [30]. In the context of this work, the terms *Fog Edge Device (FED)*, *Gateway (GW)* or *Access Point (AP)* refer to the same device category, otherwise it will be specified.

Due to the dynamic characteristics of the fog system (geo spreading, fixed and mobile devices, services requirements and so on), task arrival rates and, consequently, resource demands at different gateways can be significantly different. Mechanisms in fog computing that aim to manage computational resources have been a promising area of research due to its contribution to one of the main characteristics of the paradigm that is to reduce the response time of the system for real-time and latency-sensitive applications [3, 16, 20, 21, 31, 33, 34].

An instance of the load balance in the fog edge layer problem targeted in this work is represented in Figure 4.1, where a hypothetical network scenario is shown. The IoT layer is formed by thousands of IoT devices (IoTDs) that are randomly dispersed in the lower level over an area of 800 by 800 meters. In the upper level there are FEDs, comprised of sixteen computing devices and one Fog Edge Controller (FEC), represented in green color. In this scenario, the active IoTD (represented in red) is establishing wireless communication with the fog layer.

Due to the distance between the active IoTD and the FEDs, the IoTD wireless signal strength is attenuated along the path making the IoTD be in wireless range for just a few FEDs (also represented in red color) creating a logical cluster over the fog edge layer, also referenced as fog edge *partition* in this work. Once the partition receives a service request from the IoTD, each FED belonging to the requesting IoTD partition synthesizes some of its internal state information and send it to the FEC, which has the responsibility to choose the appropriate FED to serve the IoTD service's request.

The criteria adopted for choosing the most feasible fog edge device to serve the IoT demand may consider more than one optimization objective, and also may take advantage of some inherent characteristics from the fog edge layer which may be useful to contribute to the overall performance on the edge network.

The first criterion considers an estimative of the computational resources usage rate in the computing Fog device. The constant monitoring of this parameter may provide an indication of the viability of the device to accept, reject, or offload incoming tasks, with the aim of maintaining the rate of occupation of the device as constant as possible over time, avoiding overload or idleness.

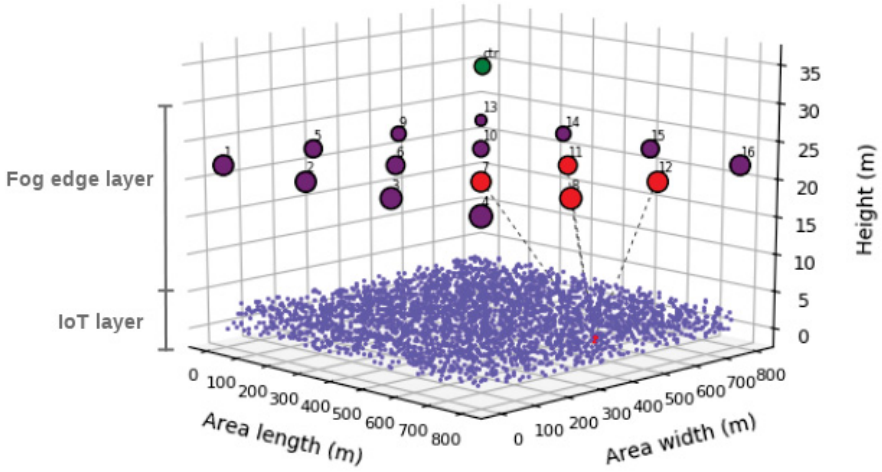


Figure 4.1: Load balancing instance over one fog edge partition

The success of the strategy to optimize the use of computational resources should contribute to the load balancing of individual computing devices and, consequently, to the load balancing of the entire Fog layer [18, 19], as discussed in the following sections.

Another criterion that is considered refers to the maintenance of a basic characteristic of the Fog Computing paradigm regarding the execution of tasks on Fog computational devices. It is expected especially for latency-sensitive applications to choose the edge computing devices as a priority to host the required services [35, 36], which demands constant performance estimations to avoid choosing a host device that does not comply to the required latency expectations.

Also, the proposed method also aims to optimize wireless communication at the edge of the network, taking advantage of the fact that the FEDs are directly connected to the wireless hardware interfaces, so they are the only fog devices that may have the ability to directly monitor the edge wireless links. Due to the high correlation between the Received Signal Strength Indicator (RSSI) and the Packet Reception Rate (PRR) parameters in wireless sensor networks [29] the RSSI monitoring may be considered a generic method, and a computationally inexpensive approach for improving the link quality on the edge of the network [30, 37]. By minimizing packet retransmissions between the IoT layer and the fog layer, the RSSI monitoring may contribute to reducing congestion at the transmission medium, and by the IoTD perspective, the packet delivery rate improvement reduces response time while saving energetic resources.

The combination of efficient computational resource management, service execution time monitoring, and wireless signal strength optimization requires a multi-objective approach to evaluate the referred parameters for the computation device choosing process. The proposed method consists of finding metrics that allow a compromise solution to satisfy the three dimensions of the multi-objective optimization problem over a generic fog edge architecture, in which load balancing is considered the main objective.

The next section presents the system components and definitions that are necessary for the construction of the Fog Edge generic conceptual model presented in Section 4.2, and the proposed method approach is exposed in detail with its structure and its operation on Section 4.3.

4.1.1 System Components and Definitions

Fog Edge Computing Device

A *Fog Edge Computing Device* (or simply *Fog Edge Device*) is one of the elements belonging to a set of D fog devices at the edge layer, responsible for serving IoT devices running or offloading its required services, where each device is uniquely identified by an integer value $d \in \{1, ..., D\}$. Each FED_d is specified by the tuple

$$FED_d = (\ AS_d, ID_d, CR_d, RM_d(\cdot), SM_d(\cdot), IM_d(\cdot)\ ) \tag{4.1}$$

where

AS_d - Denotes the subset of services that is present in FED_d from a set of *S Available Services* in the fog instance. Each service SE is uniquely identified in the fog instance by an integer s.

$$AS_d = \{SE_s \mid s \in \{1, ..., S\}\} \tag{4.2}$$

ID_d - Denotes the set of *IoT Devices* being serviced by the FED_d. Each IoTD is specified by its unique integer identifier, its unique requesting service (SE_s) and its wireless *Signal Strength* (SS_e) which is sensed by the access point.

$$ID_d = \{(e,\ SE_s,\ SS_e) \mid e \in \{1, ..., E\},\ s \in \{1, ..., S\}\} \tag{4.3}$$

CR_d - Denotes a set of discrete amount of available *Computational Resources* on the current FED_d, and it is specified by a tuple composed by CPU (CPU_d), volatile memory (RAM_d) and non volatile memory (STO_d).

$$CR_d = (CPU_d,\ RAM_d,\ STO_d) \tag{4.4}$$

$RM_d(\cdot)$ is the utility function that performs periodical *Resource Monitoring*. It keeps updating the used and spare computational resources, also providing information on the computational resources variances in the current device. This information is useful, in the appropriate moment, to evaluate in advance the likely effect of the incoming demands.

$SM_d(\cdot)$ is the utility function that performs *Service Monitoring*, evaluating individual service performance, in terms of computational resource demands and response time. This information starts with the initial configuration and is improved step-by-step over time, according to the services launching. It is used by the device as a parameter to calculate in advance the impact of the incoming service requesting, assisting in the decision to launch or not the requested service, when it is necessary.

$IM_d(\cdot)$ is the utility function that performs the *IoTD Monitoring* for services execution requests evaluating the device instant capabilities, so deciding if the incoming service request should be computed on the device or if the request should be offloaded.

Fog Edge Partition

A fog edge partition W is a logical cluster formed by a subset of FED which have a common IoT node ID_e in wireless range communication.

Fog Edge Controller

Fog Edge Controller (FEC) is one of the devices belonging to the fog edge that has the strict responsibility to serve the FED when it receives for the first time a service request from a given ID_e to choose the most feasible FED among the fog edge partition. In order to fulfill this task, FEC needs full network connectivity to all FEDs.

$$\forall\{\ FED_d \mid d \in D\ \}\ \exists\ \{\ (FED_1, FEC), (FED_2, FEC), ..., (FED_d, FEC)\ \} \tag{4.5}$$

Fog Services

A deployed *Service* SE_d in a fog device FED_d is defined as any instance from a set of AS_d available services which is generated by IoT applications that runs in the fog devices and consumes computation time and computational physical resources aiming to manage, aggregate, store, forward, transform, compute, analyze the data delegated to it from any ID_d. It is identified by an uniquely integer value s, and has a predefined *Threshold* latency value (TH_s).

$$SE_s = \{\ (\ s, TH_s\ ) \mid s \in S\ \} \tag{4.6}$$

IoT Devices

IoT Devices (IoTDs) are the nodes that populate, in large numbers, the IoT layer. Each IoTD is specified by a unique integer e and requires one specific service SE_s from the fog edge layer. The service request is done for the first time after a given *Activation Time* (AT) and after that, it sends data periodically to its allocated FED in a determined *Communication Interval* (CI).

$$IoTD_e = \{\ (\ e, SE_s, AT_e, CI_e\ ) \mid e \in E,\ s \in S\ \} \tag{4.7}$$

Cloud Instance

Cloud Instance makes up the third layer of the Fog Computing paradigm. However, in the context of this work, the Cloud Instance does not provide computation to run services, but its only function is to absorb the workload excess that cannot be computed at the edge of the Fog layer.

Fog Device Computational Resources

The computational resources requirements of the IoT services for the fog devices are various, as fog applications have different demands of computing power, volatile and permanent storage, being also diverse the capacity of the fog devices to provide these computational resources. The fog device computational resources are quantified as a finite number of resource units, and each resource unit contains various physical resources. The available physical computational resources in FED_d complies to the Definition 4.4.

IoT Devices Signal Strength

Every FED_d is an IoT device entry point to the Fog and it is attached to one wireless multi-technology gateway or FED_d is the gateway device itself. Due to the geographic dispersion characteristic of the fog gateways, each $IoTD_e$ can be in range of more than one FED_d often sensed with distinct wireless signal strength (SS_e) [30] which is aggregated to the ID_d as mentioned in *Definition 4.3*. The SS_e is treated by FED as technology agnostic and is quantified by a dimensionless normalized value (see Section 4.3.1).

Load Balancing

According to [18] and [19]: "*The load balancing value for each type of computing nodes is measured by the variance of the resource utilization which is dynamically changed according to the load on the computing nodes*". In the context of this work, the load balancing of a device FED_d is measured as a function of time in reference to a logical cluster, defined above as *Fog Edge Partition*, thus it is relative to the variance of the computational resources utilization among the members of the partition W on where FED_d belongs.

4.2 FOG EDGE LOAD BALANCE MODEL

Due to heterogeneity and the variety of services execution time expectations, the available resources on the fog computing devices may not be fully utilized [18]. A load balance technique aims to minimize load distribution discrepancies, under certain constraints, also avoiding situations as overloading or idleness among the fog nodes.

The following load balance model may be referenced to the devices highlighted in red color in the Figure 4.1 which represent one load balancing problem instance. It addresses a determined partition from the edge elements of the fog layer and contributes as a *local solution* while the *global solution* comprehends the entire fog edge layer, which is obtained by the intersection of subsequent partitions due to the inherent geo-spread and gateway redundancy characteristics of the system.

Let $A_d^s(t)$ be a binary function that defines if the fog node FED_d can provide the service SE_s at the instant t in order to admit the ID_m demand. $A_d^s(t)$ can be defined as

$$A_d^s(t) = \begin{cases} 1, & \text{if } FED_d \text{ deploys the service } SE_s \\ 0, & \text{otherwise} \end{cases} \tag{4.8}$$

The computational *resources utilization* by the node FED_d at time t to run S services where each service SE_s demands rd_s from the CR_d available resources can be evaluated by

$$ru_d(t) = \frac{1}{CR_d} \sum_{\forall s \in S} (A_d^s(t) \cdot rd_s) \tag{4.9}$$

The mean expected computational resource utilization in a *partition* W at the instant t is determined by the resource utilization of all its belonging devices as follow

$$RU_w(t) = \frac{1}{|W|} \sum_{\forall w \in W} ru_w(t) \tag{4.10}$$

The individual load balance lb_w for a given FED_d that belongs to the partition W is closely relevant to the resource utilization of the entire partition at the time instant t. According to the load balancing definition in Section 4.1.1, it can be expressed as

$$lb_w(t) = (\ ru_w(t) - RU_w(t)\ )^2 \tag{4.11}$$

Then, the average variance value for all computing nodes belonging to the partition W at time instant t can be expressed as

$$LB_w(t) = \frac{1}{|W|} \sum_{\forall w \in W} lb_w(t) \tag{4.12}$$

Considering a time interval $\Delta T = (T_{i+1} - T_i)$ as a sampling time for the load balancing measurement, the load balancing variance for the partition W can be evaluated by

$$LB_w = \frac{1}{\Delta T} \int_{T_i}^{T_{i+1}} LB_w(t)\ dt \tag{4.13}$$

Based on the above definitions and observations, the load balance optimization problem on the partition W is achieved through

$$\min LB_w,\ \forall\ w \in W\ \text{ s.t. }\ \Delta T = (T_{i+1} - T_i) \tag{4.14}$$

and it is subjected to the following constraints:

$$\text{Partition size: }\ 1 \leq |W| \leq |FED| \tag{4.15}$$

$$\text{Partition resources: }\ \textstyle\sum_{\forall s} rd_s(t) \leq \sum_{\forall w} CR_w(t) \tag{4.16}$$

$$\text{Services' sampling rate: }\ \forall\ SE_s \subset FED_d,\ \Delta T \leq \min_{s \in S} (TH_s) \tag{4.17}$$

On the load balancing process, the main objective is achieved through Equation 4.14 which consists in minimizing partition resources usage variance with respect to time. By the fog edge layer perspective, there is a request arrival rate distributed through overlapping fog partitions, where each partition load distribution temporal solution contributes for the general solution. Constraint 4.15 is quite intuitive and makes reference to the partition size, where it is composed by at least one FED and may contain the entire fog edge layer. Constraint 4.16 guarantees that the FEDs should provide enough computational resources to deploy the services properly while Constraint 4.17 imposes a maximum evaluation time, what is necessary to properly evaluate variations on FEDs resource usage.

4.3 METHOD APPROACH

The proposed method has the main objective to minimize the computational load variance for the fog edge layer devices, which is relevant to the resource utilization of each computing device and the average resource utilization among the fog edge partition. The secondary objective is to provide priority to the FED with improved IoTD link quality and as the third objective, to guarantee the fulfillment of the expected max latency for the requested service. The involved processes consists in the allocation and reallocation of the FEDs loads which are required by

the running services, and the related tasks are divided into two steps. First, the *Static Resources Allocation* (SRA) aims to attach IoTDs to the FEDs under optimal resources availability and edge network link conditions and the *Dynamic Resources Reallocation* (DRR) that aims to find an optimal temporal set of running services instances to improve the FED resources usage.

The Static Resource Allocation and Dynamic Resource Reallocation processes are discussed in detail in the following sections.

4.3.1 Static Resource Allocation

Over time, several IoTDs start to become active and then begin transmitting their service requests to the Fog layer. For these unserved IoTDs, the proper FEDs should be identified to host their services. In this step, each involved FED assimilates its information about the IoT link signal strength and its available computational resources into a single parameter, the *Request Acceptance Coefficient* (RAC). Next, RAC is transmitted to the FEC that identifies the involved fog edge partition, and the participating FED with the appropriate RAC is selected.

The Request Acceptance Coefficient is determined in function of a tuple of normalized estimators, $RAC = (R, L)$, where R denotes a representation of the node's computational resource, and L denotes the wireless link quality scored by the IoTD signal strength. The R and L estimators and the Request Acceptance Coefficient RAC are determined as follows.

Resource Estimator

Based on the fog edge load balancing model in Section 4.2, Equation 4.11 denotes the load balancing for a single FED and indicates the resource utilization $ru(t)$ as its single parameter that may be completely controlled by the own FED that contributes to its resources variance. The purpose of Resource Estimator R in the SRA process is to provide a normalized representation of the state of the resources in use on the device, and it is implemented by the utility function RM_d as it was defined in Section 4.1.1.

The computational resources utilization in a given FED by its running services can be represented as in Figure 4.2. This example shows a hypothetical scenario where the FED starts to accept IoT requests and deploy its requested services. There are six different running services $\{SE_1, ..., SE_6\}$ where each of them has its first launching time which occurs after the IoT request acceptance, and it is periodically executed using determined amounts of FED's resources for a determined interval of time. The FED instant load is delimited by the red curve while the average FED loads between time instants are indicated above the time instants ticks marks.

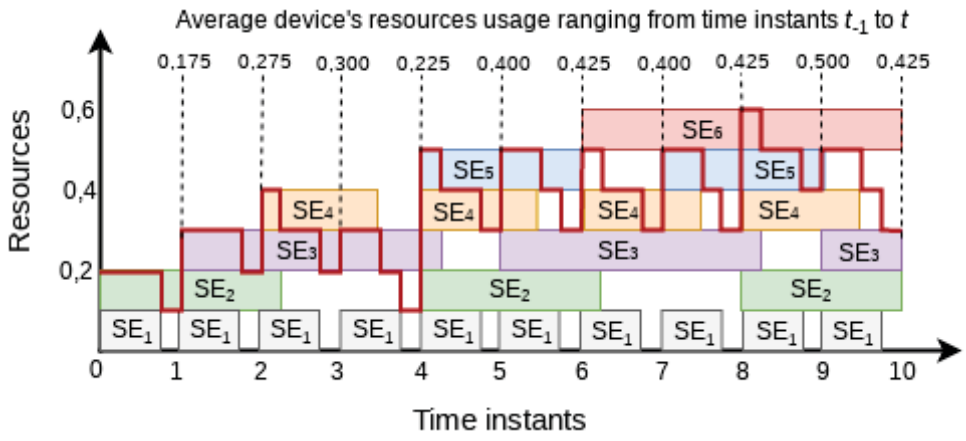


Figure 4.2: Example of FED's resource usage by running services over time. Distinct services are shown consuming similar resources amount only for graphical simplification purposes.

It is possible to notice, in this part of the SRA process, how the FED computational load values fluctuate, making inconsistent to assume a reliable device's load profile based on the instant device load value, or by its average load value for a short period of time. Also, the new services admittance rate and their load impact on the FED along the time when combined with the already running services make mapping a reliable device load profile a challenging task. In this case, the adoption of a heuristic approach may be an option to deliver a feasible solution to the problem under low computational costs.

Let ΔT be the sampling time as defined in the Equation 4.17, t' the interval of time that precedes t in the length of the highest run-time limit TH for the devices set of services and ru_d the instant devices computational resources being used, and CR_d the discrete amount of computational resources on the device. Then, the resources estimator R for the time instant t assumes the values,

$$R = \begin{cases} 0 & \text{if the } FED \text{ is overloaded} \\ \dfrac{\Delta T}{CR_d \cdot t'} \displaystyle\sum_{\forall\ (t' \leq\ t_i \leq\ t)} ru_d(t_i) & \text{otherwise} \end{cases} \tag{4.18}$$

Thus, the estimator R represents the device's normalized average resources use, which is calculated based on the immediately previous time interval which has the length of the highest run-time limit from the set of already launched services. Also, R may be evaluated to 0 if there are no spare computational resources on the device.

Link Quality Estimator

Following [38], the link quality indicator (LQI) value is represented by an integer ranging from 0 to 255 where the values should be uniformly distributed between these two limits which are associated with the lowest and highest quality compliant signals detectable by the receiver but, instead of having to process each received packed, the link quality may be associated with RSSI and correlated to PPR as demonstrated in [29, 30].

For the L estimator determination, the lower scale limit $SS_0 = 0$ is determined on the RSSI threshold value in dBm where the relation between RSSI and PRR is verified, and the upper scale limit $SS_m = 255$ is determined as the maximum RSSI value for a given receiver. Let SS_i be the normalized instant RSSI measured value, so the link quality L assume the value,

$$L = \frac{SS_m - SS_i}{SS_m} \tag{4.19}$$

Thus, the estimator L represents a normalized and inversely proportional RSSI measurement.

Request Acceptance Coefficient

The *Request Acceptance Coefficient RAC* syntetizes the estimators of FED's computational resources and the IoTD link quality, which were previously calculated, and to provide a perspective of the FED's displacement from a theoretical optimal working condition. This indicator is used by the fog edge controller device to select which FED belonging to the partition is the most appropriate to accept the IoTD service request. Being the estimators R and L conditioned to positive values into a minimization problem, the theoretically optimal estimator tuple is $E_0 = (0, 0)$. This tuple is used as a referential point in a two-dimensional space, making feasible the evaluation and comparison of the FEDs through their respective *Euclidean Distance* modulus

in the selection process. Each FED occupies a position into the $\mathbb{R}^2$ space and their respective distances relative to the origin $(0, 0)$ represents how far each FED is from the optimal solution. The Euclidean distance between two points p_1 and p_2 in the $\mathbb{R}^2$ space may be evaluated through $d_{1,2} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. Applying the equation to the given problem, where each FED_i is represented as a point in the coordinates (R, L) and all the distances d_i are calculated in reference to E_0, leads to $d_i = \sqrt{R_i^2 + L_i^2}$ or simply $d_i = L$ if Equation 4.18 evaluates R as 0 due to insufficient available resources on the FED. In this particularly case, where the R dimension is nonexistent, it is important to flag that the FED is available only as a *gateway*, which considers imposing to RAC to assume $RAC = -L$.
Based on these observations, the RAC indicator may assume the values:

$$RAC = \begin{cases} -L, & \text{if } R \text{ is } 0 \\ \sqrt{R^2 + L^2}, & \text{otherwise} \end{cases} \tag{4.20}$$

Thus, the Request Acceptance Coefficient RAC represents how far a given FED situates from the optimal solution or assumes $-L$ if the device's resources are insufficient to launch the requested service, signalizing that the FED can only be set as a gateway.

The Static Resource Allocation Process

The SRA selection process is managed by the Fog Edge Controller device (FEC), a distinct device among the Fog Edge layer that has network connectivity to all the computing devices in the fog edge, as defined in Equation 4.5. After the FEDs receive a service request from unserved IoT devices, they assimilate information about the computational demands from the requested service, their current resource availability and the wireless link quality, proceeding to the RAC calculation and further transmitting it to the FEC. Then, the FEC gather all received requests in a certain window of time $(t_i - t_{i-1})$, separate them based on their respective fog edge partitions and select the most feasible FED based on the RAC parameter. Next, the selection is informed to the respective FED that adds the IoTD to its client's list.

According to Equation 4.20, the FEC selection algorithm may fall into two relevant situations:

1. At least one RAC parameter in the partition is positive. In this case, at least one FED can be selected as a computing node.

2. Each RAC parameter in the partition is negative. This situation indicates that all FEDs in the partition are overloaded, so the FEC selects the FED with the better wireless link to act as a gateway.

After FED receives the FEC message that the request must be accepted, the IoT device *id* is added to the device's client list. If the returned RAC parameter is negative, the IoT device *id* is also added to a forwarding list, indicating that all further processing requests sent by that IoT device must be offloaded properly, otherwise, all processing requests from that IoT device will be locally processed. Next, the DRR process takes place to reallocate the required IoT tasks, fitting them according to their temporal demands to the device's resources.

The SRA process which is performed in the FEC is described by Algorithm 1, and the DRR process is presented in the next section.

Algorithm 1 - The SRA selection process:

1: // Gather requests up to the sampling time trigger
2: $R \leftarrow \{\ \}$ // set of incoming services requests
3: **while** $(t_i - t_{i-1}) < \Delta t$ **do**
4: $R \leftarrow R + ServiceRequest(IoTD_i, FED_i, RAC_i)$
5: **end while**
6: // Separate requests into partitions by the IoT ids
7: $P \leftarrow \{\ \}$ // requests to be grouped as partitions
8: **for all** $r \in R$ **do**
9: $P_i \leftarrow \{r_i, ..., r_j \mid r_i.IoTD_n = ... = r_j.IoTD_n\}$
10: **end for**
11: // Search into the n found partitions for the
12: // most feasible RAC in each one of them
13: **for** $i = 1$ **to** n **do**
14: **if** $\forall\ (\ r \in P_i,\ r.RAC < 0\ )$ **then**
15: $a \leftarrow\ max\ (\ r.RAC\ )$
16: **else**
17: $a \leftarrow\ min(\ r.RAC\ )$
18: **end if**
19: // Informs the selected FED
20: $FED_a \leftarrow AcceptRequest(\ IoTD_a, RAC_a\ )$
21: **end for**

4.3.2 Dynamic Resources Reallocation

The SRA, previously described, has as the task to provide the first allocations for the requesting IoTD, while the DRR process is intended to work with the already allocated devices and reduce the entropy on the FED's temporal computational resource's usage.

Considering the FEDs as computing nodes, and due to the characteristic of the FEDs to be entry points to the Fog layer, in a densely populated and active IoT layer, there are plenty of heterogeneous demands for computational resources. The DRR process takes advantage of the high availability of temporal resources demands from the connected IoTDs by dynamically selecting the demands in arrival time to be executed or offloaded.

DRR's decision process is implemented by the IM_d (IoTD Monitoring) utility function, as defined in Section 4.1.1, and considers as its input a representation of the node's computational resources, provided by the R estimator and a representation of the requested service i latency behavior, provided by the S_i estimator. The R and S_i estimators are determined as follows.

Resource Estimator

The DRR process uses the same estimator R employed by the SRA process, which is provided by the utility function RM_d. It was previously defined in Section 4.3.1 and it is evaluated by Equation 4.18.

Services Latency Estimator

The purpose of the Service Latency Estimator S is to obtain information about the performance of a given service that is available on the computing device. It is provided by the utility function SM_d, defined in Section 4.1.1, which implements similar heuristic as RM_d for keeping individual statistics updates for each device's running service.

The estimator S_i provides a normalized latency profile for the service i according to the TH_i, i.e., the max expected latency for the execution of the service i, and it is determined as follows. Let ΔT be the sampling time as defined in the Equation 4.17, t' the interval of time that precedes t in the length of the highest run-time limit TH for the device's set of services, lat_{d_i} the latency history for the service i on the device d, and TH_i the max expected latency for the service i. Then, the service i latency estimator S_i for the time instant t is evaluated as,

$$S_i = \frac{\Delta T}{TH_i \cdot t'} \sum_{\forall\ (t' \leq\ t_i \leq\ t)} lat_{d_i}(t_i) \tag{4.21}$$

Thus, the estimator S_i represents the device's normalized average latency for the service i, which is calculated based on the immediately previous time interval which has the length of the highest run-time limit from the set of already launched services. Also, S_i may be evaluated to 0 if there is no run-time history for the service i on the device.

The Dynamic Resource Reallocation Process

The DRR process runs on the Fog Edge computational nodes and it is effective when the FED approaches the established max computational load limit, or when the service to be launched approaches to its expected max latency limit.

When the FED receives an IoTD's processing request, the DRR process checks for the device's R and S_i parameters. If the device's conditions are favorable to the service execution, the request is added to the execution queue, otherwise, the request is added to the offloading queue. The DRR process is performed as described by Algorithm 2 as follows.

Algorithm 2 - The DRR process:

```
1:  // The DRR process is triggered by a data processing request
2:  if FED ← ProcessingRequest(IoTD_i, SE_i, data) then
3:      R ← FED.RM.Evaluate(STO, RAM, CPU)
4:      S_i ← FED.SM.Evaluate(SE_i)
5:      // Test for R and S_i limits
6:      if ( R ≥ R_lim ) or ( S_i ≥ S_lim ) then
7:          Offload(IoTD_i, SE_i, data)
8:      else
9:          Execute(IoTD_i, SE_i, data)
10:     end if
11: end if
```

Due to the conditions evaluated by the R and S_i estimators, the DRR process can become active during the IoTD allocations, which are performed by the SRA process, however, as soon as no new service request is received from the IoTDs, only the DRR process remains active.

4.4 SUMMARY AND DISCUSSION

This chapter has presented a multi-objective approach to target the computational load balance issue for the Fog Computing architecture in its edge layer. It started from the problem discussion to the conceptual system's modeling and its involved components. For the Fog edge system modeling, it was considered a wide geographical area permeated by a large number of IoT devices and predominance of wireless communication. As a consequence of the devices' limited wireless range, only a group of Fog edge devices is reached by a given IoT device, which was defined as Fog edge *partition*, and the most feasible Fog edge device to serve the IoT device should be an integrant from the partition formed by it, that is, within its wireless range. Based on this principle, the system's components for all the layers of the general Fog architecture as well as the functional parts of the system were defined.

Then, followed the theoretical formulation of the Fog edge load balancing approach, where load balancing and the computational resources definitions from Xu *et al.* [18, 19] were considered, as previously presented in Section 4.1.1, where the load balancing is correlated to the device's load variance, and the computational resources are defined as a finite number of resource units, where each resource unit contains various physical resources, leading to a set of constraints that the system should comply.

Next, the proposed multi-objective optimization load balancing method approach was presented, consisting of two functional parts. The first is the Static Resource Allocation (SRA) process, which is intended to optimizing the RSSI from the IoT-Fog wireless communication, also considering the computing device resource's availability on the process of choosing the most feasible Fog edge device to serve the requesting IoT devices on their service's demands. The second part is performed by the Dynamic Resource Reallocation (DRR) process, which considers keeping the executed services latency under control while optimizing the Fog edge devices resource usage by dynamically fitting the temporal devices' resources demands to the resources availabilities.

The next chapter presents the evaluation environment, consisting in a discrete event simulator that was built to evaluate the presented Fog edge model and the multi-objective load balancing method as well.

5 EVALUATION ENVIRONMENT

This chapter introduces the discrete event simulator developed to evaluate and validate the multi-objective load balancing model presented in this work. The details of the represented architecture, its components, that provide the foundation for running the simulator, and the details of its design and implementation are discussed in the following sections.

5.1 IOT AND FOG COMPUTING SIMULATION

With the emergence of the IoT paradigm and recently, Fog Computing paradigm and their interplay, there has also arisen the need to create a resource performance evaluation platform that reflects the models these concepts represent. An IoT system is typically composed of several physical devices, spread through different layers (cloud, fog, edge), working together to fulfill tasks belonging to a certain problem domain. Those tasks are usually delegated to devices according to specific models in order to optimize the synergy among them and providing to the end-users services as sensing, controlling, visualization, data analysis, etc. The modeling of a general IoT environment to build effective and smart services can be quite difficult, due to the heterogeneous possible scenarios, however, prototyping using a large number of hardware nodes may not be practical due to time and costs constraints involved, thus, IoT simulators are necessary for both quantitative and qualitative aspects. Among simulators present in the literature for IoT environments, SimIoT [39] and iFogSim [40] stand out.

The SimIoT simulator was developed as a hybrid implementation of the SimIC [41], which is intended to model inter-cloud facilities and collaboration schemes. The IoT layer is not implemented in SimIoT, data is generated from physical devices like sensors that automatically submit information processing requests to the cloud. It can be used to evaluate jobs in cloud-based data processing systems, based on data submitted by the sensors or users of the IoT services.

The iFogSim simulator provides a full-stack environment that also supports fog computing and it achieves this by extending the simulator CloudSim [42]. In iFogSim, the IoT layer is not independently implemented, the Fog edge devices are created with their associated sensors and actuators directly connected in [43]. The simulator can be used mainly for conducting performance evaluations on control loop latency, network bandwidth, and energy utilization of various service deployment approaches, such as cloud-only versus fog computing.

Among the available simulators, iFogSim has been rated to be the most complete option for simulating Fog computing environments. However, so far it does not present some implementations that are essential for the validation of the multi-objective optimization approach presented in this work, which focuses on the fog edge layer and in determined characteristics of its interaction with the IoT layer. For instance, the independence of the IoT layer devices and the effects of their interactions with the fog edge devices through wireless communication due to their geolocations are important components for the formation of device clusters, defined as edge partitions, which is one of the elements of the *SRA* process, as described in Section 4.3.1.

Due to the difficulty in finding an adequate simulator that represents a Fog environment with a sufficient detailed edge layer implementation, a proper discrete event simulator was developed using the SimPy (Simulation in Python) tool [44] as the core, being focused on the architectural demands and in the system's elements as they were defined in Chapter 4, to evaluate the presented multi-objective optimization method. Details of the referred discrete event simulator, as well as its experimental setup are presented in the following sections.

5.2 ARCHITECTURE AND FUNCTIONAL MODULES

The simulator's general architecture follows the typical three-layered architecture for Fog systems, as illustrated in Figure 5.1. The communication between layers is done bidirectionally in the two existing networks. The wireless network is the edge network, accessible to the IoT devices and Fog edge devices, interconnecting these two layers, while the wired network promotes the interconnection of all the Fog devices, as well as their connection to the Cloud.

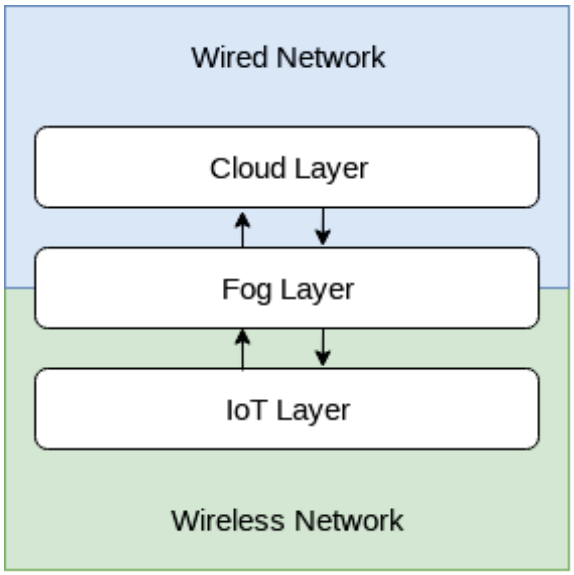


Figure 5.1: Discrete event simulator's three-layered general architecture.

The functional responsibilities of each of the three layers from the simulator's general architecture are described as follows.

IoT Layer The IoT layer is composed of thousands of devices, which are spread out geographically according to an uniform distribution. Each IoT device, after a random initial period of inactivity, starts to request a service from the Fog environment at regular intervals of time.

Fog Layer The Fog layer is composed by two types of devices: A Fog Controller, responsible for the decision-making process on the IoT clients assignments to other Fog devices category, and dozens of Fog edge devices, which have the function of serve the IoT devices demands for their requested services, acting as computing nodes or to forward the IoT requests for services to another layer, assuming the role of gateways.

Cloud Layer The Cloud layer is composed of only one cloud instance. As this work focuses on the Fog edge layer, the Cloud layer does not perform computation, it only gathers data and generates statistics of the IoT services which are forwarded from the Fog layer to the cloud.

The simulator software implements the three layers of the general architecture, as well as the two present networks, through several functional modules. The main functional modules, as well as their relations, are represented through a simplified class diagram that describes the simulator's architecture core, illustrated in Figure 5.2. The description of each functional module is presented as follows.

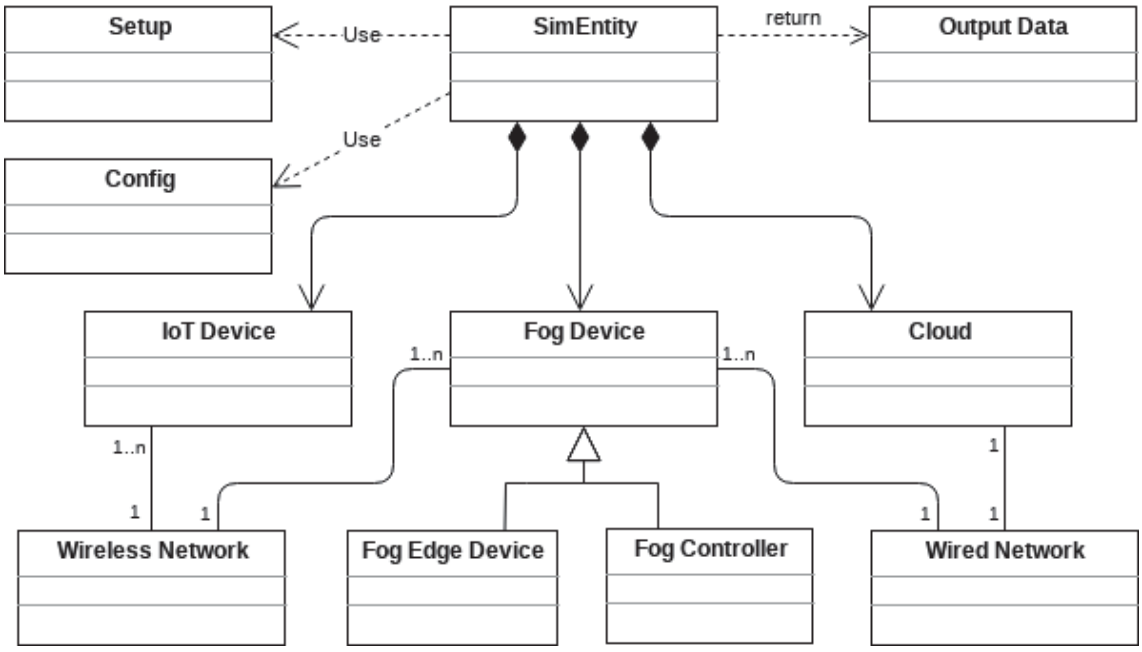


Figure 5.2: Discrete event simulator's architecture core.

SimEntity
This is the simulator's main functional module, which builds the entire system. It gathers the simulation's input settings from Setup and Config modules, creates a SimPy discrete event clock instance that synchronizes all the functional modules, creates instances for the elements of the general architecture layers, as well as for the networks, and interconnects all the system components. At the simulation's end, it also gathers relevant execution data from all the systems modules and redirects them to the Output module.

Setup
The Setup module gathers basic execution settings of the simulator and for the system components, such as the simulation total time and time step, the quantity, positioning and dispersion model for the Fog and IoT devices, networks addressing, and diverse other settings. More details are presented in Section 5.5.

Config
The Config module allows the user to pass parameters to the simulator when it is executed. These parameters include basically a setup file to be loaded, the simulation output data file, displaying runtime useful information, and to choose the graphics to be generated. More details are presented in Section 5.5.

Output
The Output module is responsible for managing and displaying the information generated by the simulation during or after its execution, including display data in terminal or data stored in file format. More details are presented in Section 5.5.

IoT Device
This module represents the IoT layer, which is densely populated by independent instances representing the IoT devices, that exchange data with the Fog Device module through the Wireless Network module.

Fog Device
This module represents the Fog layer and it is a generalization of the device's types that are present in this layer, that is, the Fog Controller device and the Fog edge devices.

Fog Controller
The Fog Controller module represents a device belonging to the Fog which does not perform services' computation. It has a network connection to all Fog edge devices and it is dedicated to the process of IoT clients allocation for the Fog edge devices.

Fog Edge Device
The Fog Edge Device module represents a category of Fog devices that communicate to the IoT layer through the wireless network and provide computation for the requested services. Also, these devices may perform the gateway function by offloading the IoT service's requests to the Cloud layer through the wired network.

Cloud
This module represents the Cloud layer. It does not perform computation, it is dedicated to gathering simulation data and generates statistics for the IoT services which are forwarded from the Fog layer.

Wired Network
This module interconnects the instances that represent the Fog devices, as well as connects them to the Cloud instance. It is implemented using the Store feature of the SimPy library, where it is possible to put and retrieve any Python type of data. Each Fog or IoT device connected to the network has an addressable data slot, where the transmitter stores the data and the receiver retrieves, using the production and consumption model.

Wireless Network
This module connects the instances that represent the IoT devices with the instances that represent the Fog edge devices. Like the Wired Network module, it is also implemented using the Store feature from the Simpy library. However, the Wireless Network module is able to determine the Euclidean distance between transmitter and receiver and jointly to the transmitter's power, it calculates the propagation path loss in the open air for each network data packet, making this information available to the receiver.

5.3 INTER-LAYER COMMUNICATION

Data tuples are the fundamental unit of communication between the simulator entities. They are implemented using the Python list data type, which may encapsulate diverse values, even from different data types, so building messages that are exchanged between devices' instances. A message is characterized by its type, the source, the destination, and payload field, where the payload field encapsulates a set of relevant information regarding the message type.

To implement the communication of the simulators processes between instances of different devices, six different types of messages were defined:

Service Request (SERVREQ) An unserved device requires a connection to another device so that it can provide processing of its generated data, which corresponds to a particular Fog service.

Service Accept (SERVACC) This message type is generated as a response to the *Service Request* message, answering to the device's request. In this way, a connection between the devices is established.

Service Refuse (SERVREF) This message type may be generated as a response to the *Service Request* message, declining the device's request, being in this case optional (the request also may be declined by timeout). In addition, if this message is received from a device whose connection has already been established, it has a disconnect effect.

Service Data (SERVDAT) This type of message characterizes a data processing request to a host whose connection has already been established.

Ping Request (PINGREQ) This is a control message, useful to determine if a given device is reachable in the network.

Ping Response (PINGRES) Response to the *Ping Request* message.

The communication dynamics between layers and devices necessary for the simulation of the multi-objective method of load balancing in the Fog edge layer, presented in Section 4.3 is described by the sequence diagram shown in Figure 5.3.

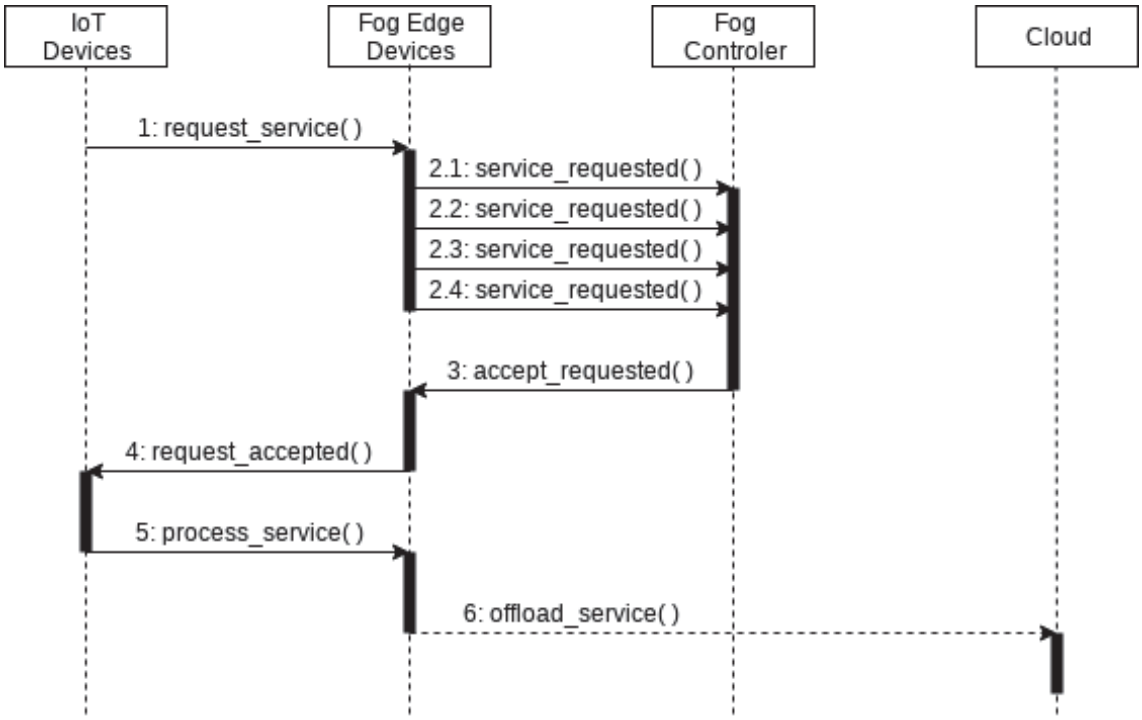


Figure 5.3: Inter-layer device's communication sequence diagram.

The entire communication process takes place in six different moments, as described below.

1. After an initial random waiting time, the IoT device starts up and searches for a connection to the Fog, then broadcasting a service request through the wireless network.

2. All Fog edge devices that receive the request from the IoT device calculate the *Request Acceptance Coefficient*, as described in Section 4.3.1, and send it to the Fog Controller Device.

3. The Fog Controller Device gathers all received requests from diverse Fog edge devices to the same IoT device, then it chooses the most feasible Fog edge device and returns a message informing it to accept the IoT device request.

4. The chosen Fog edge device inserts the IoT device id in its client's list and then informs the IoT device that its service request has been accepted.

5. The IoT device stores the Fog edge host id, and periodically send by unicast the data intended to be processed.

6. The Fog edge device receives the data to be processed from the IoT device and then checks if its id is present in the offloading list. If yes, it relays the data processing request to the Cloud, otherwise, it proceeds to the received data computation.

After moment 4, where the connection between the IoT and Fog devices is established, the IoT device sends data to be processed to the Fog host at regular intervals of time, thus the procedures referring to moments 5 and 6 are repeated until the simulation's end.

5.4 THE FOG EDGE DEVICE

The Fog edge device was defined as having one or more network interfaces or it is directly attached to them, being able to assume the functions of gateway or computing node. The operating mode decision process is launched for each data processing request received from an IoT device that is already being served by the Fog edge device. This process uses three internal structures: The IoT client's list, which is populated by the SRA process, the offloading table which is populated by the DRR process and contains tuples with client and computing device ids, and the device processing table, which contains useful information for each running task, such as the id and timestamp for the task being executed and the device's computational resources in use.

Figure 5.4 describes the sequence of events in the process of deciding the operational mode of the Fog edge device. Upon receiving a data processing request from an IoT device, the list of clients is checked, and if the IoT device id is also present in the offloading list, the request is redirected to the corresponding computing device, otherwise, the task is inserted into the process table and the requested service is launched to start processing the received data.

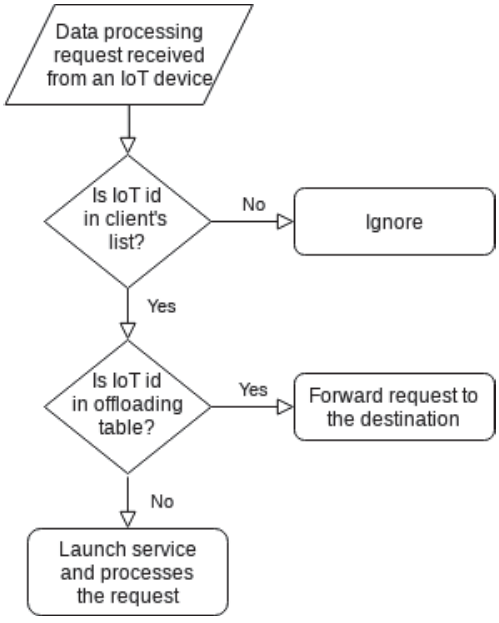


Figure 5.4: Simplified flowchart for the decision process on enabling gateway mode or computing mode for a data processing request in the Fog edge devices.

The request forwarding process in the gateway mode generates a new data processing request message. The message's source and destination are replaced by the current device's id and the corresponding destination which is present in the forwarding table, respectively, and the new message is inserted to the network output buffer from which the destination device is connected to. The information about the original requesting IoT device remains available, however, it is inserted into the new message's payload field.

When the data processing request is computed locally, a new task is created, and the message fields referring to the request timestamp, source IoT device id, and requested service id are inserted into the device's processing table, along with the data regarding to the amount of storage, RAM and CPU that are required to perform the data processing. On the task's processing starting, the device's available CPU capacity is divided equally among all tasks present in the table, that the simulator updates on each sampling period. If a given task requires more than one sampling cycle to be completed, the amounts of required storage and RAM remain reserved, while the amount of CPU that was used to compute the task is decreased and becomes available on the device for the next cycle. When the task's computation is finished, the amount of storage and RAM which have been reserved are released, and the CPU time used on the task's processing is computed as a sample for the service's latency estimative.

5.5 SIMULATOR INPUT AND OUTPUT

The simulator was built to describe the main architectural and functional characteristics of the Fog edge model proposed in this work, as well as to produce relevant simulation data to the analysis of its dynamics. However, it is flexible and can be customizable, where various properties and functional blocks can be modified or added to describe a variety of scenarios and functional entities. As described in Section 5.2, the simulator has two modules for entering parameters and settings and a module that formats the relevant information from the simulator output. Details of the implemented customizations are shown as follows.

The Setup Module

The Setup module is responsible for configuring the simulator's operation, as well as the properties of the layers and devices. The definitions are inserted into a file that is read by the main module that configures the entire system at the simulation's starting time. The following options can be customized in the Setup module:

General Total simulation time, simulator step time, time scale (clock time or abstract units).

Architectural Number of layers, range of devices' ids by layer, number of IoT devices, number of Fog devices, number of Cloud instances, IoT devices geo-dispersion statistic distribution, Fog devices geo-dispersion model and statistic distribution, coverage space for the IoT and the Fog devices ([x,y,z] coordinate limits).

Network Number of networks, type (wireless or wired network), device id ranges by network.

Services Number of different services, resources demands by service (CPU, RAM, and storage), resources demands and execution time threshold attributions by service (fixed values or by a statistic distribution).

IoT device Device id, service assignment, wake-up time, geo-position, network interfaces, Tx power and Rx sensitivity (for the wireless network interface).

Fog device Type (edge or controller), device id, geo-position, network interfaces, Tx power and Rx sensitivity (for the wireless network interface), computational resources capacity (CPU, RAM, and storage, which are fixed values or they are assigned by a statistic distribution), resources usage limit, flags to enable optimizations on wireless signal, resources, and service's latency.

The Config Module

The Config module is intended to parse the command line for loading optional simulators control parameters and to select the relevant simulator's output data. The three main groups of customizable options are:

Configuration management It is possible to run the simulation by passing a configuration file as input, overlapping completely or partially the settings present on the Setup module. The settings that can be independently loaded are, the IoT device settings, the Fog device settings, and the available services' settings.

Live simulation info The simulator is capable of producing and displaying runtime information regarding the evolution of the states of several functional entities of the simulator. The list of live data info that can be displayed on the standard output is presented in the Output module.

Graphs generation Part of the information which is generated by the simulator is described in the form of graphs, where twelve different types were implemented and can be independently selected. The list of graphs that can be generated is presented in the Output module.

The Output Module

The Output module gathers and formats the relevant information from the simulation for display or analysis where most can be selected or manipulated by the Config module. The options for each group of information are as follows.

Live info The following information can be displayed to the standard output in simulation runtime or as a post-simulation summary: The RAC value and its parameters, the exchanged messages on all networks, the IoT clients by each Fog device, Fog devices statistics (number of samples, standard deviation, and variance on the RSSI, the device's resources usage and for the services' execution latency), status for Fog devices' tasks processing table, IoT devices' partition lists and available Fog devices' services.

Graphs The following information can be displayed in the form of graphs: Nodes locations, the clients of a given Fog edge device, the partition formed by a given IoT device, the IoT layer resources demands by area, the load sharing between Fog and Cloud layers, the individual load profile for a Fog device, the average load by individual device in the Fog layer, the number of IoT clients by Fog device, the RSSI on individual Fog device, the RSSI for the entire Fog layer, the services' latency, IoT to Fog layers service's requests rates.

In addition, the Output module generates logs for all the main actions, warnings and error information, as well as saves to files all the configuration used for the simulation and all the data generated by it for later analysis.

5.6 SUMMARY AND DISCUSSION

This chapter has presented a discrete event simulator, that has been built under the necessary requisites to evaluate the Fog edge model and the multi-objective load balancing approach presented in this work.

It has been initiated with a brief analysis of simulators widely used in IoT scenarios, followed by the presentation of the general architecture, typical for a Fog Computing environment, formed by three main layers: IoT, Fog and Cloud layers. These layers are connected through two networks, a wireless network that connects the IoT and Fog layers and a wired network that interconnects the elements of the Fog layer and also connects them to the Cloud layer.

Then, the procedures for the communication between layers and the functional elements of the simulator, which represent the physical devices, were presented. Different types of messages that are formed by tuples of data, being them the basic communication units of the simulator were defined, with the objective to promote relevant data exchanging. The information exchange process is essential for the dynamics of the simulator, and consequently, for the functioning of the previously introduced multi-objective load balancing approach, being presented through a communication diagram jointly to a brief description of its stages.

Next, the Fog edge device was introduced, showing its two operational modes, acting as a gateway or as a computing node, and the descriptions of the internal structures that support the decision making for the device's operating modes, as their roles in the involved processes. Also, it was described how the device simulates the task's processing in the computing mode and the sampling determination for the latency of the device's available services.

Following, more details are presented on the simulator modules that allow to configure parameters of the functional entities and customize the simulator output data. The simulation general parameters, the details of the general architecture for specific use cases and the parameters for the functional entities are configured in the Setup module, while the settings for optional parameters on the simulation control and the selection of the simulation output information are adjusted via a set of options that are passed to the simulator on its execution being captured by the Config module, and the formatting of the various types of information and data which are produced by the simulation is on the responsibility of the Output module.

One of the main motivations for implementing the *Live info* option of the Output module was as a debugging tool to be used during the simulator's implementation process. Through it is possible to monitor, for each step of the simulation, the relevant changes in the internal states of the computing devices and in this way, independently audit the behavior of the simulator, as well as the correctness of the information that is being generated during the evolution of the simulation.

The next chapter presents the evaluation and analysis of the presented Fog edge model jointly to the evaluation of the multi-objective load balancing approach presented in this work.

6 SIMULATION RESULTS AND ANALYSIS

This chapter presents the evaluation of the presented multi-objective load balancing method, which was fully performed using the discrete event simulator presented in Chapter 5. In addition, the basic characteristics that permeate the presented three-layered general Fog model on its edge due to the IoT-Fog layers interaction are analyzed. Initially, the experimental setup and simulation's conditions are presented, as well as the IoT layer modeling and how it is sensed by the Fog edge layer, followed by an analysis of the effectiveness of the *SRA* process on allocating IoT devices. Next, the results in the optimization of IoT-Fog connection links, the individual Fog edge devices load distribution and the fulfillment of the service execution threshold time are analyzed, ending with an overview of the load distribution between the Fog and Cloud layers and the load balancing over the entire Fog layer.

6.1 SIMULATION SETUP AND EVALUATION CONDITIONS

The multi-objective load balancing method presented in this work was evaluated under high-load conditions, where the demands from the IoT layer overcomes the fog edge layer resource capacities. To achieve this condition, the number of IoT devices, the number of fog edge devices, the devices' coverage area, the services load demand, and the fog edge nodes' computational capacity were properly set. Simulator's main parameters setup configuration were defined as shown in Table 6.1, and any change on these values will be appropriately specified.

Table 6.1: Simulator's main parameters setup configuration

Parameter	**Value**	**Note**
Simulation area	$64 \times 10^4 m^2$	800m × 800m
Simulation Time	1000 or 4000	Abstract units (simulation cycles)
IoT Devices wake-up delay	[1, 750] or [1, 3200]	Abstract units (simulation cycles)
IoT Devices Z Positioning	[0, 3]	Height in the Z axis, in meters
Fog Edge Devices Z Positioning	[20, 25]	Height in the Z axis, in meters
IoT Devices	3200	Geo-dispersion: random
Fog Edge Devices	16	Geo-dispersion: random or grid
Fog Edge Controllers	1	Geo-dispersion: centered
Cloud Instances	1	Ubiquitous (to the Fog layer)
Number of Services	16	Heterogeneous in res. demands
Requirements by Service	[15, 45]	Computational resources units
Fog Edge Device Capacity	[300, 600]	Computational resources units
Wireless Transmission Power	0 dBm	For all Iot and Fog edge devices
Wireless Reception Sensitivity	-90 dBm	For all Iot and Fog edge devices

The simulation considers a three-dimensional space where devices are spread over an area of 800 by 800 meters. Also, IoT devices can vary in height from 0 to 3 meters, being geographically dispersed in their three coordinates according to a random uniform distribution.

By simulating a large number of IoT devices, where each one of them requests one out of sixteen different services (in terms of max processing time and computational resources usage) at regular time intervals to the Fog edge devices, the IoT layer can be perceived as a set of localized and temporal demands for computational resources. This effect can be observed in Figure 6.1,

where the IoT layer temporal demands for resources on every $25m^2$ (5 by 5 meters) are quantified from the simulated area. It is possible to notice a spatial unbalance, where some area units require much more computational resources per unit of time than others. This unbalance is one of the factors that need to be properly compensated by the SRA and DRR algorithms at the Fog edge layer to minimize discrepant computational load distributions among its computational devices.

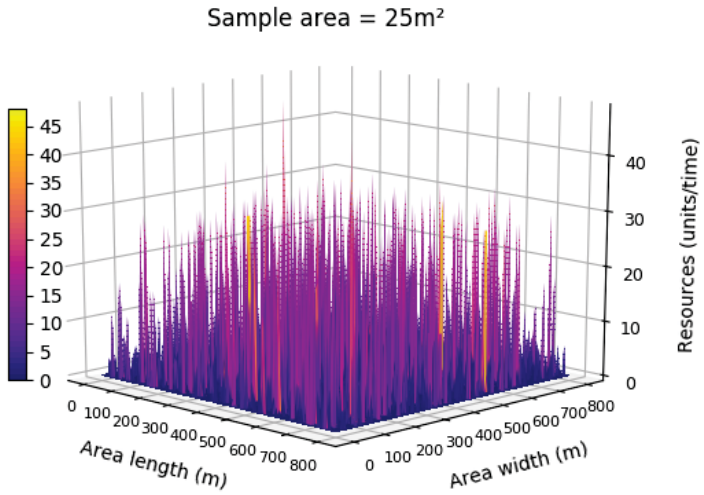
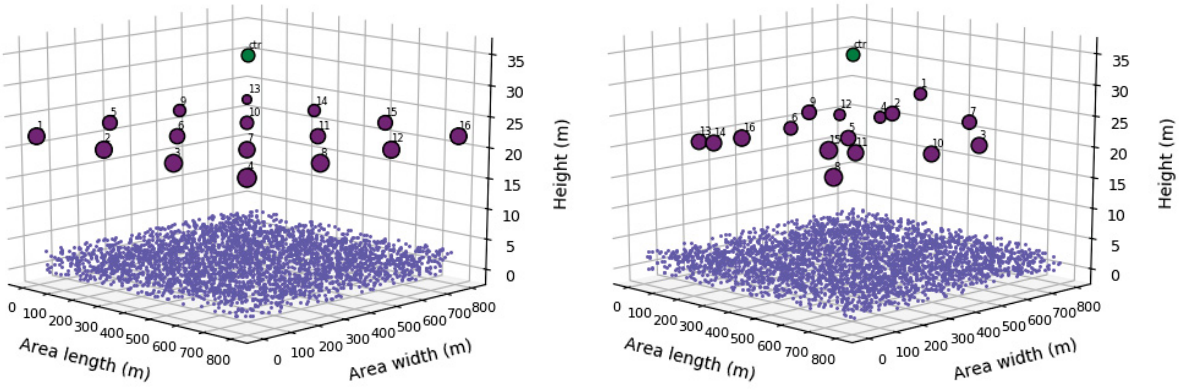


Figure 6.1: IoT Layer - Temporal computational resources demands by area.

In addition to the unbalance of load requirements caused by IoT layer heterogeneity and devices' geo-dispersion, other factors may contribute to the unbalance between requests and the supply offer of computational resources at the system's edge, such as computational capacity heterogeneity and the action's range of the Fog edge devices. As a result, four different scenarios were chosen to evaluate the effectiveness of the multi-objective optimization method, where the entropy of the Fog edge system is changed at different levels. In this way, the Fog edge devices are arranged at regular distances in a grid format or they are scattered randomly, as well as their computational resources are changed, having equal or random capacities, which are set within the limits shown in Table 6.1.

The geo-dispersion formats for the system devices are shown in Figure 6.2. The IoT layer devices are represented in blue color, the Fog edge controller device is represented in green, while the Fog edge devices, represented in purple are grid-dispersed in Figure 6.2(a) and it is shown in Figure 6.2(b) one of the possible random scattering scenarios.



(a) Fog Edge devices in a grid formation.

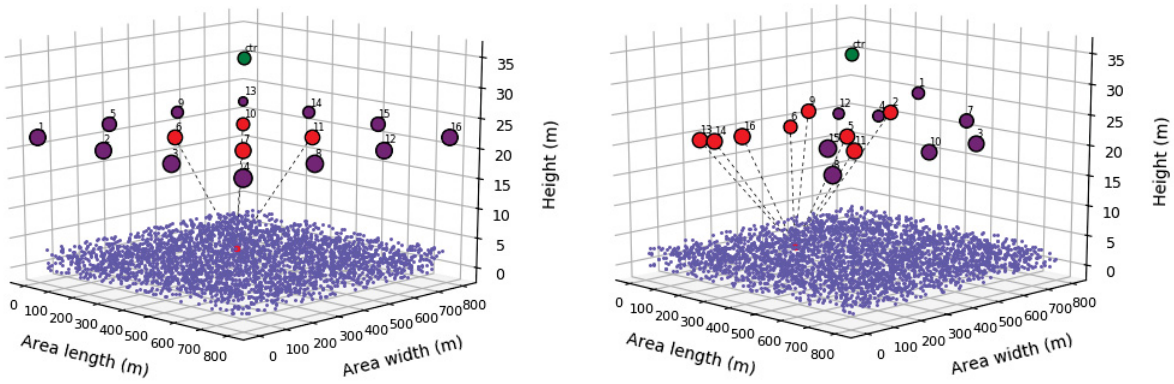
(b) Fog Edge devices scattered randomly.

Figure 6.2: Scatter patterns of the system devices used to vary the entropy between the supply and the demand for computational resources at the network's edge.

6.1.1 Interaction Between IoT and Fog Layers

One of the characteristics of electromagnetic waves in wireless communication is the signal attenuation along its way, a phenomenon known as *path loss*. Given the signal path loss and the distances between IoT devices and Fog edge devices, only device groups of both layers are able to communicate with each other. The group of Fog edge devices reached by an IoT device was previously defined as *partition* and it is one of the criteria considered by the SRA process.

Figure 6.3 shows the effect of creating partitions according to the spreading models adopted for the Fog edge layer. Devices marked in red are on the communication process, and evidences an inclination to form larger partitions when the Fog edge devices are randomly scattered.



(a) Fog edge devices in grid formation.

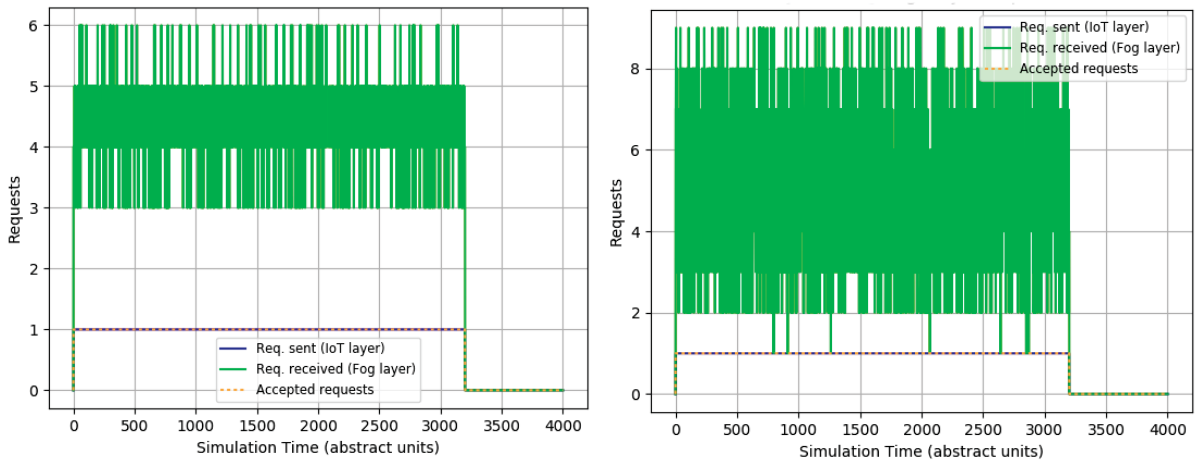
(b) Fog edge devices scattered randomly.

Figure 6.3: Trends of Fog edge partition formation due to the device's geo-dispersion.

IoT devices can have their activation times set in relation to the total simulation time. When the IoT device's activation time is reached, the device wakes up and makes its first request to the Fog edge layer, and from the moment it is allocated to a Fog edge device, it transmits data at regular time intervals, keeping a client-server relationship. This activation time is adjusted for each IoT device within the range of 1 time unit to 75% of the total simulation time, as reported in Table 6.1. Thus, it is possible to evaluate the SRA process for the IoT devices allocation, both SRA and DRR processes simultaneously while the activation time limit is not reached, as well as to evaluate the DRR process in standalone after the activation time limit.

Another possibility that can be explored through the ability to adjust the activation time of IoT devices is to trace an overall profile of the effect of partitioning on the Fog edge layer. In Figure 6.4, the total simulation time has been set to 4000 time units, and each one of the 3200 IoT devices was configured to have distinct activation times. The green curves denote the Fog edge devices requests receptions, the blue curves represent the requests made by IoT devices, and the orange dashed curves represent the IoT devices allocations, performed by the SRA process.

Figure 6.4(a) shows that due to the Fog edge layer in a grid layout, the formed partitions have a lower average number of devices, however, concurrency for the SRA process remains present as each request is received by at least three Fog edge devices. Figure 6.4(b) shows the results for the simulation keeping the same set of IoT devices but with the Fog edge devices arranged randomly. It can be noticed in comparison to Figure 6.4(a) that there has been an increase in the average number of devices per formed partition, however, there are cases where there is no concurrency in the SRA process as some IoT devices are within the reach of only one Fog edge device. Through these results, it is possible to confirm the partitioning trends regarding the arrangement of Fog edge layer devices that were presented in Figure 6.3, where the partition formed by only one IoT device was taken into account.

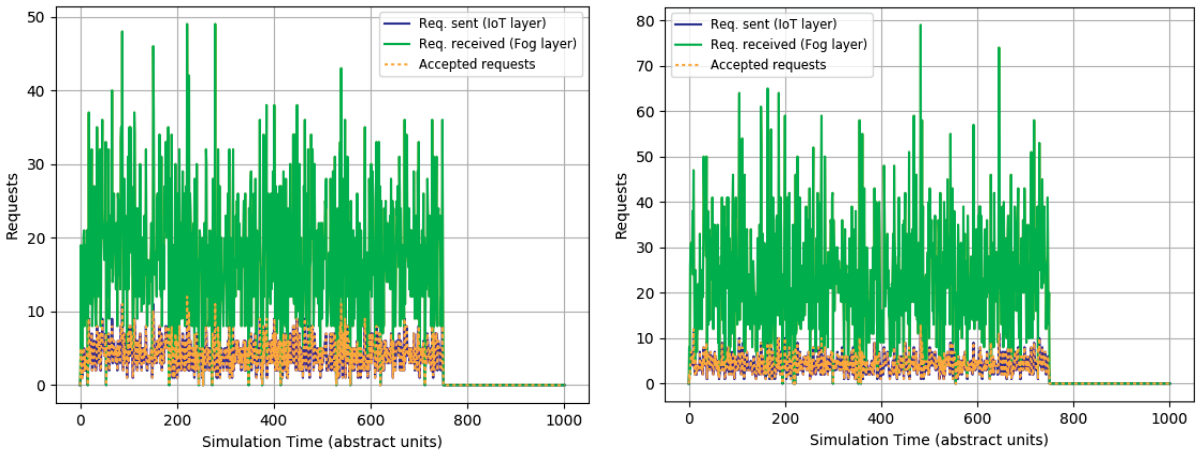


(a) Fog edge devices in grid formation.

(b) Fog edge devices scattered randomly.

Figure 6.4: IoT layer service requests and Fog edge layer allocations during the SRA process, conditioned on one request per simulation time.

Under regular simulation conditions, IoT devices activation times are set randomly, so, within the stipulated simulation interval of 1000 time units, multiple IoT layer devices can be activated simultaneously. This concurrency behavior may cause overload on the Controller device that is responsible for managing IoT device assignments to Fog edge devices in the SRA process. As it acts as a centralized agent, the activating of multiple IoT devices may cause the creation of many partitions simultaneously at the Fog edge where many devices are shared among them. This condition is analyzed in Figure 6.5. In Figure 6.5(a), the Fog edge devices are arranged in a grid format and in Figure 6.5(b) they are randomly scattered. The green curves denote the requests received by the Fog edge layer, and through the blue curves, it is possible to track variations in the number of IoT devices activation and their requests made to the Fog edge layer, and the dashed curves in orange show the IoT devices being allocated by the SRA process along the time. In both cases, it is possible to notice the blue and the dashed orange curves are overlapping, indicating that the Fog edge controller can handle multiple requisitions simultaneously smoothly in the SRA process.



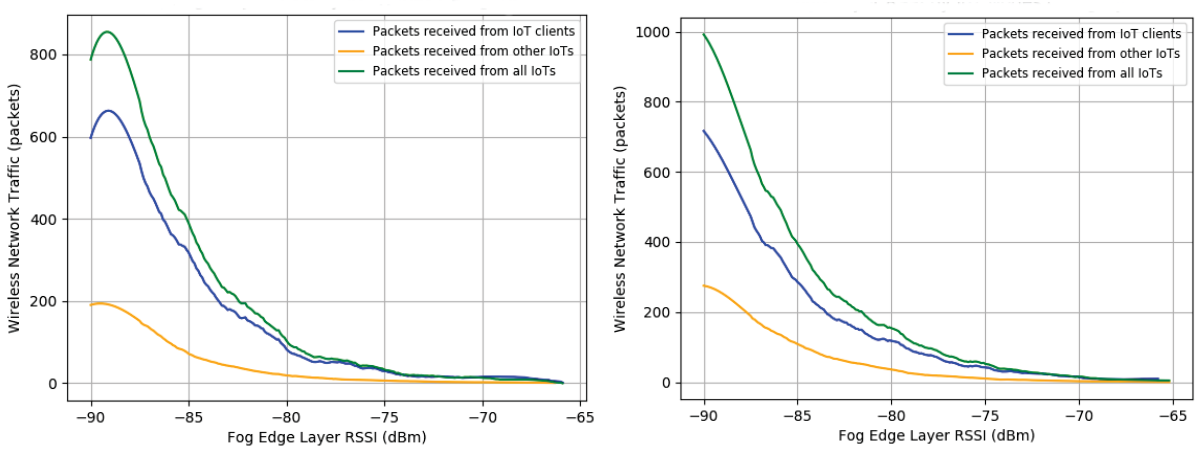
(a) Fog edge devices in grid formation.

(b) Fog Edge devices scattered randomly.

Figure 6.5: IoT layer service requests and Fog edge layer allocations during the SRA process, with multiple requests per simulation time.

In the simulated model, the transmitters are limited to the power of 0dBm while the receivers have a typical sensitivity of -90dBm, and the path loss is calculated based on a free path, i.e., without attenuation due to intermediate obstacles.

Due to these characteristics, there will be scenarios where the communication between the IoT and the Fog layers will occur with good signal strength to scenarios where the signal has fainted until it reaches the out of range threshold. Figure 6.6 shows an overview of network edge communication, demonstrating how the Fog edge layer senses the IoT layer through messages exchanges, named in the figures legends as *packets*. In Figure 6.6(a) the Fog edge devices are arranged in a grid format and in Figure 6.6(b) they are randomly scattered. The green curves represent all Fog's received packets from the IoT layer on the course of the simulation time, the blue curves represent the packet's communication to the allocated IoT devices while the orange curves represent the packet's communication to the unallocated IoT devices.



(a) Fog Edge devices in grid formation.

(b) Fog Edge devices scattered randomly.

Figure 6.6: Perception of IoT layer by Fog edge layer through RSSI measurements.

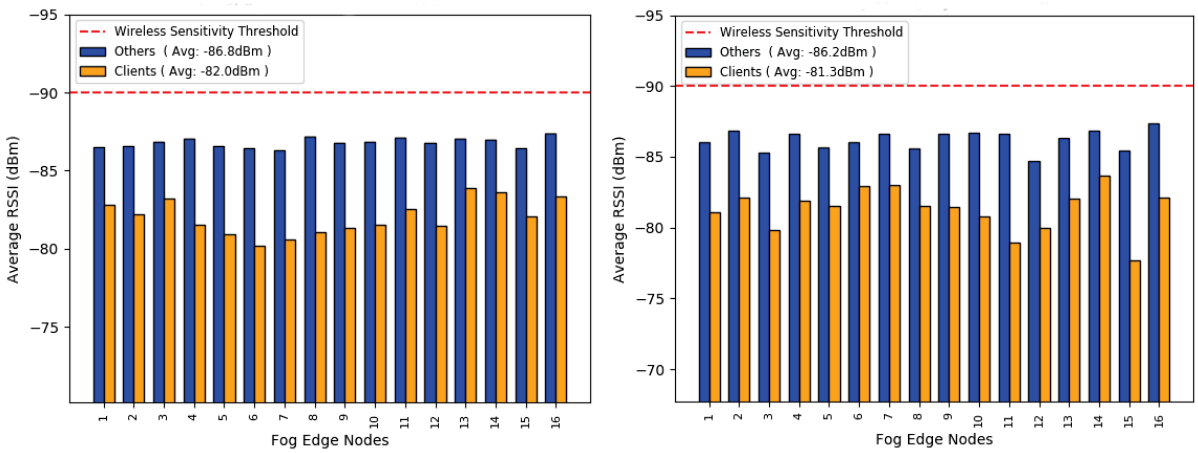
By comparing the two figures above, in addition to the noticeable increase in network traffic due to the dispersion model of the Fog edge devices, it is also possible to notice that most part of the IoT-Fog communication happens with the signal strength near to the receiver's sensibility threshold, indicating that in most cases, as regards to the signal optimization, the SRA process will not count with significant RSSI differences on each round of the selection process, leading it to select IoT devices even with weak signal, having this trend confirmed by the dynamics indicated by the blue curves.

6.2 WIRELESS LINK QUALITY OPTIMIZATION

The wireless link quality optimization happens exclusively in the SRA process during the admission of unallocated IoT devices, where RSSI and available computational resources are relevant. The evaluation included six scenarios, where IoT device service request concurrency, computational resources heterogeneity, and Fog edge devices dispersion model were considered.

Figure 6.7 shows the RSSI measurements of Fog Edge devices for two scenarios. The IoT devices concurrency factor was eliminated, i.e., at each simulation time step, only one IoT device makes a service request to the Fog layer. Also, all the Fog edge devices heterogeneity was eliminated, as they all have the same computational capability. In this case, the entropy of the Fog edge layer relies only upon the Fog edge layer devices dispersion model, wherein Figure 6.7(a) shows measurements for the devices dispersed in grid format while in Figure 6.7(b)

for the devices randomly scattered. For each Fog edge device, the average RSSIs of the connected IoT devices are represented by the orange bars, while the in-range, but not connected IoT devices are represented by the blue bars, and the average RSSIs for the entire Fog edge layer for both devices categories are shown in the figures captions.

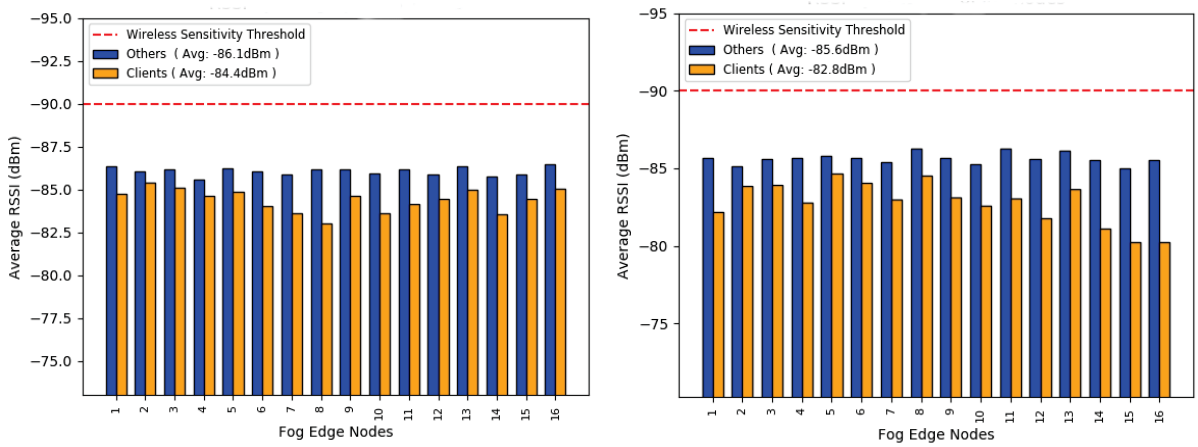


(a) Fog Edge devices in grid formation.

(b) Fog Edge devices scattered randomly.

Figure 6.7: Individual average RSSI for devices with homogeneous computational capabilities in the Fog edge layer and no services request concurrency.

Next, IoT devices concurrency was considered, where several devices make requests for services at the same simulation time step, however, the homogeneity of the Fog edge devices computational resources were maintained. The obtained results are shown in Figure 6.8, wherein Figure 6.8(a) the Fog edge devices were scattered in a grid format and in Figure 6.8(b) they were scattered randomly.

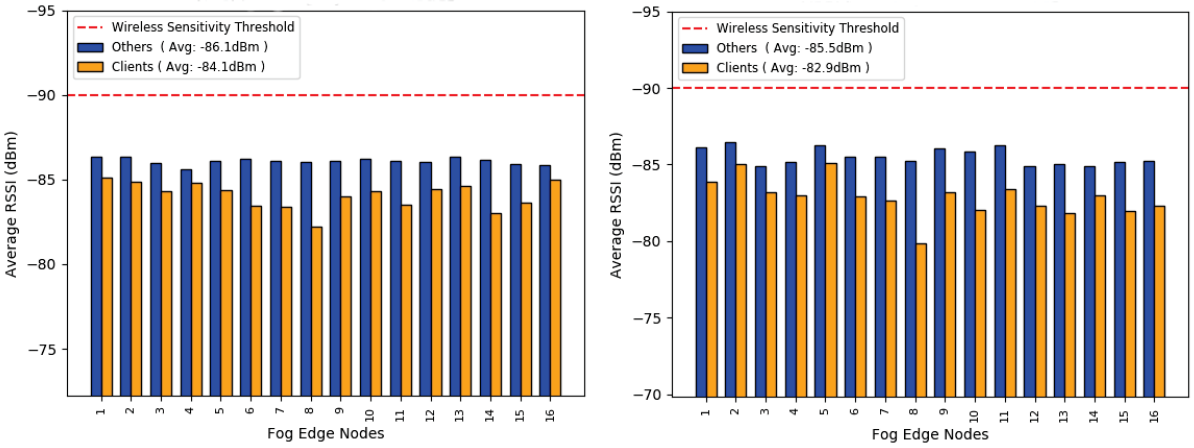


(a) Fog Edge devices in grid formation.

(b) Fog Edge devices scattered randomly.

Figure 6.8: Individual average RSSI for devices with homogeneous computational capabilities in the Fog edge layer with services request concurrency.

Finally, Fog edge devices computational resources heterogeneity was considered, also maintaining the concurrency of the IoT devices by its simultaneous services requests. Figure 6.9 shows the obtained results, wherein Figure 6.9(a) the Fog edge devices were scattered in a grid format and in Figure 6.9(b) they were scattered randomly.



(a) Fog Edge devices in grid formation.

(b) Fog Edge devices scattered randomly.

Figure 6.9: Individual average RSSI for devices with heterogeneous computational capabilities in the Fog edge layer with services request concurrency.

Results show that the considered factors on the composition of the testing scenarios had some influence on the efficiency of the SRA process, where the formation of many partitions simultaneously due to the concurrency of the IoT devices during the service request process may be considered the main factor that interferes with the effectiveness of the optimization process, as can be noticed by comparing the RSSI gain in Figure 6.7 with figures 6.8 and 6.9.

Another noticed influencing factor on the RSSI optimization refers to the size of the formed partitions, where the Fog edge devices configurations which are randomly scattered, that tend to form larger partitions, have slight improvements in the RSSI level for the entire edge layer in comparison to the Fog edge devices which are scattered in a grid format. This trend is also noticeable for the RSSI level of the individual devices, where those who assume central positions in the simulated area, which are easily identified in configurations with a grid format, in most cases they show better RSSI optimizations if they are compared to the others in the edge layer.

The heterogeneity of Fog edge devices computational resources also evidenced to be an influential factor in the efficiency of RSSI optimization, where the overall RSSI gain for the Fog edge layer formed by homogeneous devices, shown in Figure 6.8, was superior to that formed by heterogeneous devices, shown in Figure 6.9, for any case of devices dispersion.

Despite the differences, the achieved results from the RSSI analysis showed that the SRA process was able to improve the average wireless signal strength in the communication between all devices in the IoT-Fog layers for all analyzed cases.

6.3 FOG EDGE DEVICES WORKLOAD DISTRIBUTION

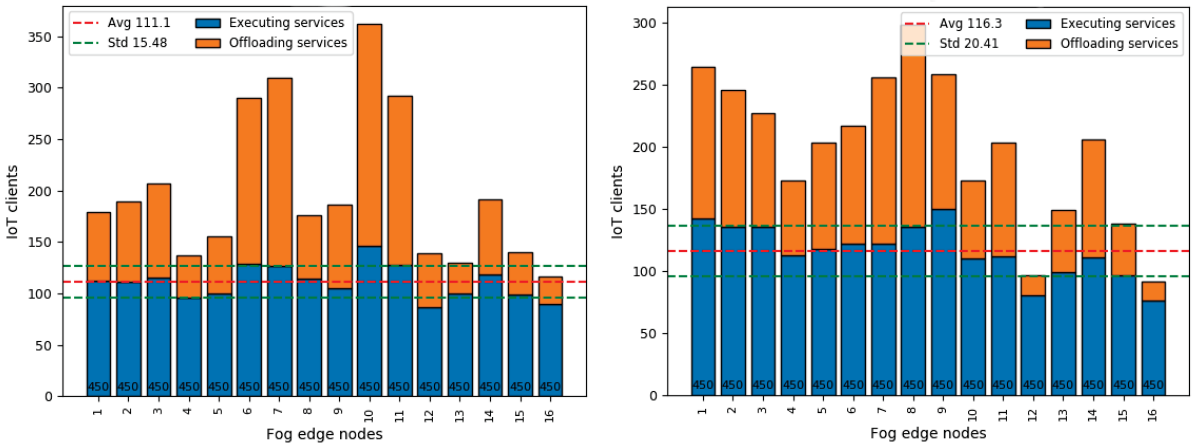
The workload distribution, necessary to achieve the load balancing over the Fog edge layer, is performed by both SRA and DRR processes with respect to a pre-established device's computational load threshold. The SRA process has as its optimization dimensions the RSSI and the computational resources, acting on the IoT to Fog devices allocation, while the DRR process has as its optimization dimensions the maximum expected latency for the available services and also the computational resources, acting on the already allocated IoT devices by dynamically fitting services resources demands to the device's resources availability.

In this section, the effects on the Fog edge devices resulting from the SRA and DRR processes are evaluated. First, it is presented the IoT to Fog devices performed allocations, next it is presented the load distribution for the individual Fog edge devices, and finally, it is presented the achieved device's load variance optimization according to different device's load profiles.

6.3.1 IoT Clients by Fog device

For being the entry points for the IoT devices to the Fog layer, the Fog edge devices must manage all IoT layer connections, either providing processing or acting as gateways. Therefore, the SRA process continues to allocate IoT devices even after the device's computing resources have reached out to its threshold. In this case, only the RSSI is considered to promote the IoT devices allocation, and the Fog edge device will perform the gateway function for processing some of the IoT requests at higher levels of architecture.

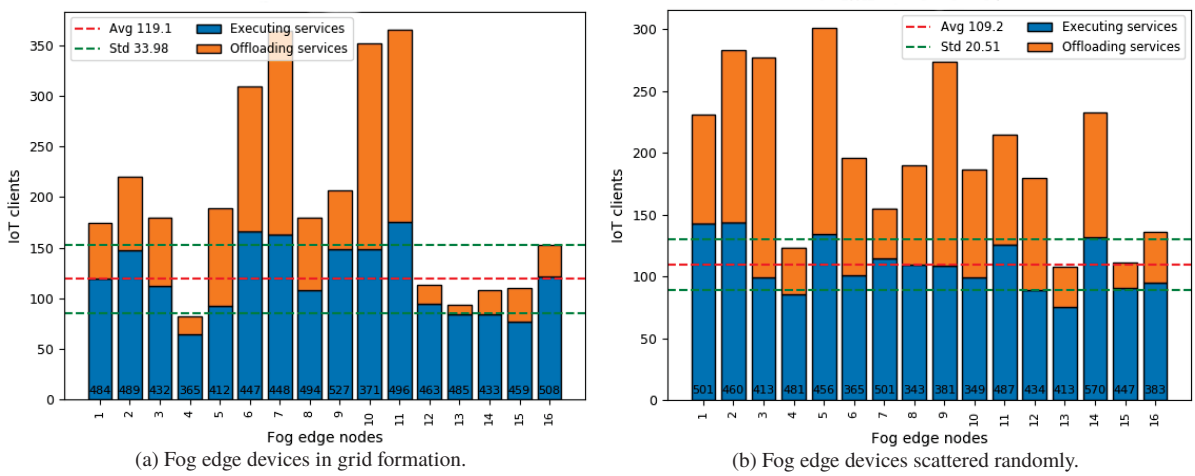
Figures 6.10 and 6.11 show the distributions of the IoT devices between Fog edge devices in different scenarios. The spatial distribution of Fog edge devices is considered, being it in grid formation or randomly scattered, as well as the capacity variation of Fog edge devices, as homogeneous and heterogeneous. The bars represent the number of IoT nodes allocated to each Fog edge device, named clients, while the blue part represents those that have their services being locally executed, and the orange part, those that have their services being offloaded. The capacity in resources units of each device is indicated at the bottom of the bars, and the figures legends show the average and the standard deviation for the blue bars for the entire edge layer.



(a) Fog edge devices in grid formation.

(b) Fog edge devices scattered randomly.

Figure 6.10: IoT clients distribution over the Fog edge devices with homogeneous resources.



(a) Fog edge devices in grid formation.

(b) Fog edge devices scattered randomly.

Figure 6.11: IoT clients distribution over the Fog edge devices with heterogeneous resources.

In all analyzed cases, the wide disparity in the number of IoT clients served by each Fog edge device is visually noticeable. In figures 6.10(a) and 6.11(a), where the positions of Fog edge devices are known due to the grid-shaped dispersion, it is possible to notice that the client's admission by the SRA process is strongly influenced by the relative positions between

Fog edge devices and IoT devices. This trend is less evident when figures 6.10(b) and 6.11(b) are analyzed, where the devices on the edge of Fog are randomly dispersed, tending to form larger partitions, allowing the SRA to have more possibilities in its admittance decision process.

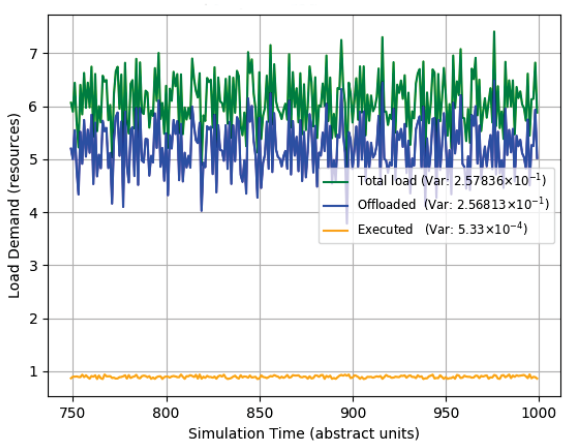
Regarding the number of IoT devices processed by each Fog edge device, in the four analyzed cases, the means and standard deviations did not show a direct correlation with the geographical layout or with the individual variation of the computational capacity of Fog edge devices, suggesting that the association of both characteristics is jointly influential for the matching of the IoT-Fog layers resources demands and availabilities.

The wide variation in IoT clients being attended by each Fog edge device can cause different load profiles, interfering on the individual device's load balancing. Some load profiles for individual Fog edge devices are discussed in the next section.

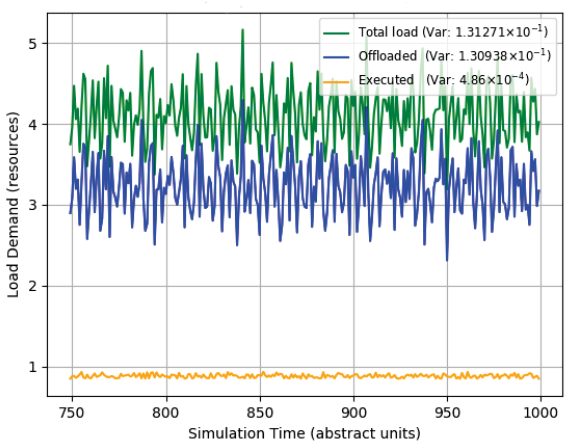
6.3.2 Fog Edge Devices Load Profiles

The total number of IoT clients allocated to different Fog edge devices reflects the load demands of what each device is subjected to. As the DRR process takes advantage of the high availability of temporal demands for computational load, it is not always possible to fit a demand for computational resources to the Fog edge device's availabilities.

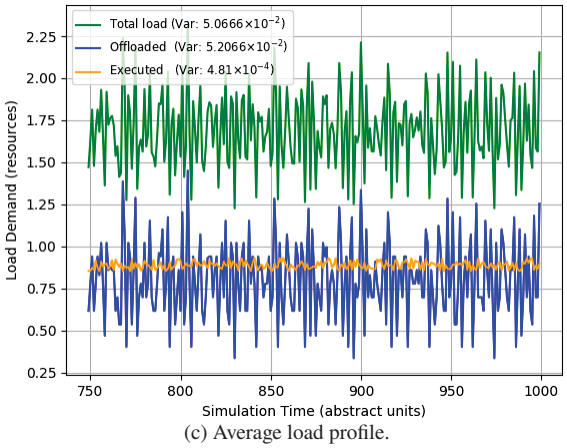
Figure 6.12 shows four possible load scenarios for Fog edge devices referring to the load they are subjected to, which are obtained after the end of the wakeup time of all devices in the IoT layer, where only the DRR process is operational. The green curves represent the normalized total load requests received from IoT devices, the blue curves represent the load being offloaded and the orange curves represent the load being computed by the Fog edge devices.



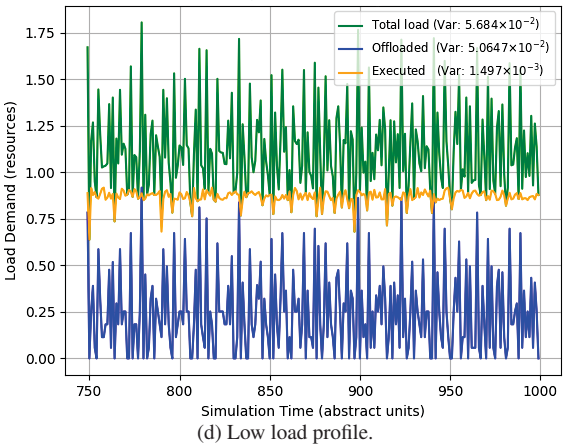
(a) Very high load profile.



(b) High load profile.



(c) Average load profile.



(d) Low load profile.

Figure 6.12: Fog edge devices' load profile variance according to the load demands they are subject to.

The analyzed scenarios range from a very large load request rate, with an average requested load above six times the device's capacity, to a low load request rate, with an average load very close to the device's capacity. In all cases, the DRR process was able to drastically reduce the load variance on the computational device, so contributing to the individual load balancing, while it maintains the device's total used load below its limit. However, in the specific case where the device's load was classified as low, represented by Figure 6.12(d), the variance reduction performance was inferior to the other cases, caused due to the unavailability of the IoT devices load demands at certain time instants in the analyzed period.

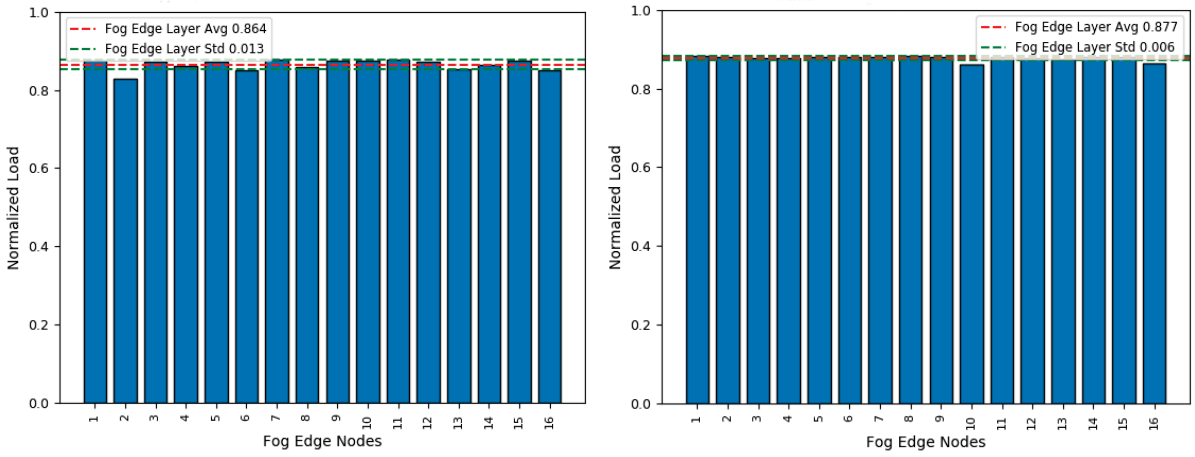
It is also possible to notice that the load requests from the IoT devices for the presented Fog edge model, which are represented by the green curves in Figure 6.12, do not present an evident pattern, which makes the use of a deterministic algorithm for predicting the required computational load and manage the required resources in the Fog devices a challenging task.

The influence of different performances on reducing the computational load variance on individual devices for the load balancing of the Fog edge layer is discussed in the next section.

6.3.3 Individual Load by Fog Edge Device

Due to the stochastic nature of the geographic distribution, as well as the resource demands of IoT layer devices, Fog edge devices can assume different load profiles, as previously analyzed, leading to temporarily affecting the device's individual load and consequently affecting the load balancing of the entire Fog edge layer.

To analyze the condition of the individual Fog edge devices load, four scenarios were established. The devices were arranged in grid format or randomly distributed across the Fog layer, maintaining their homogeneous computational capacities, shown in Figure 6.13, and adopting heterogeneous devices, illustrated in Figure 6.14. Devices' load reference was set at 0.85 of their capacities, and the measurements took place at the end of the simulation time, being it calculated as defined by Equation 4.18. Also, averages and standard deviations were measured to determine mean resources' occupancy and load balancing for the entire Fog edge layer.



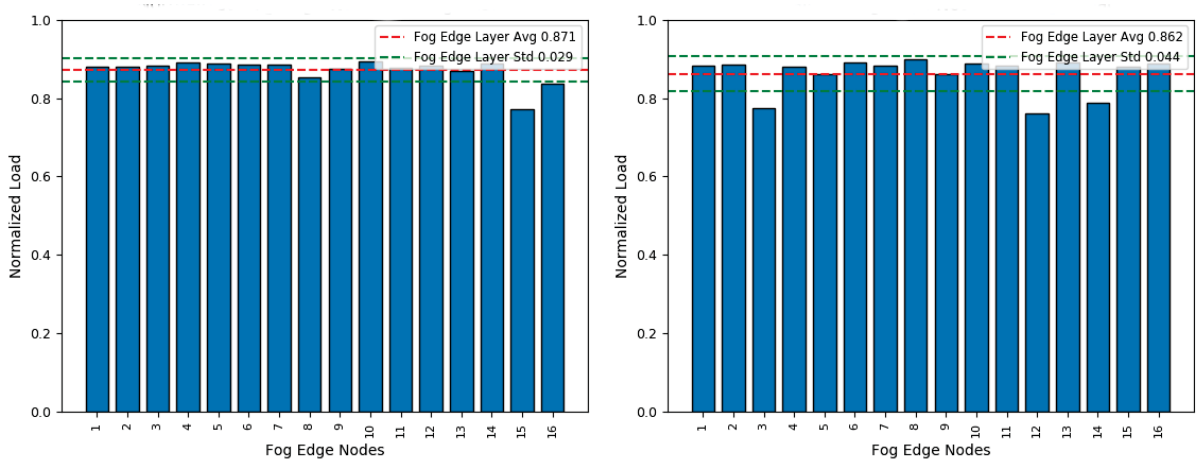
(a) Fog Edge devices in grid formation.

(b) Fog Edge devices scattered randomly.

Figure 6.13: Individual mean load for Fog edge devices with homogeneous resources.

The minimum and maximum layer load variations, measured by the layer standard deviations, occurred when the Fog edge devices have homogeneous and heterogeneous computational resources respectively, and in both cases, they are geographically dispersed at random.

This indicates that both the devices geographic positioning and the heterogeneity jointly influence the process of load distribution among the devices on the current Fog edge layer model.



(a) Fog Edge devices in grid formation.

(b) Fog Edge devices scattered randomly.

Figure 6.14: Individual mean load for Fog edge devices with heterogeneous resources.

In all analyzed cases, fluctuations were observed around the reference load value for the individual devices, as it was expected. These fluctuations show a greater discrepancy in the case illustrated by Figure 6.14(b), indicating that the devices may assume quite varied load profiles, corresponding to the analysis in Section 6.3.2, however, the DRR process was able to rearrange the load demands in order to maintain an average load occupation on the individual devices, within their sampling period, tending to the reference value and avoiding overloads, thus reducing load variances and contributing to the load balancing of the entire Fog edge layer.

6.4 SERVICES LATENCY FULFILLMENT

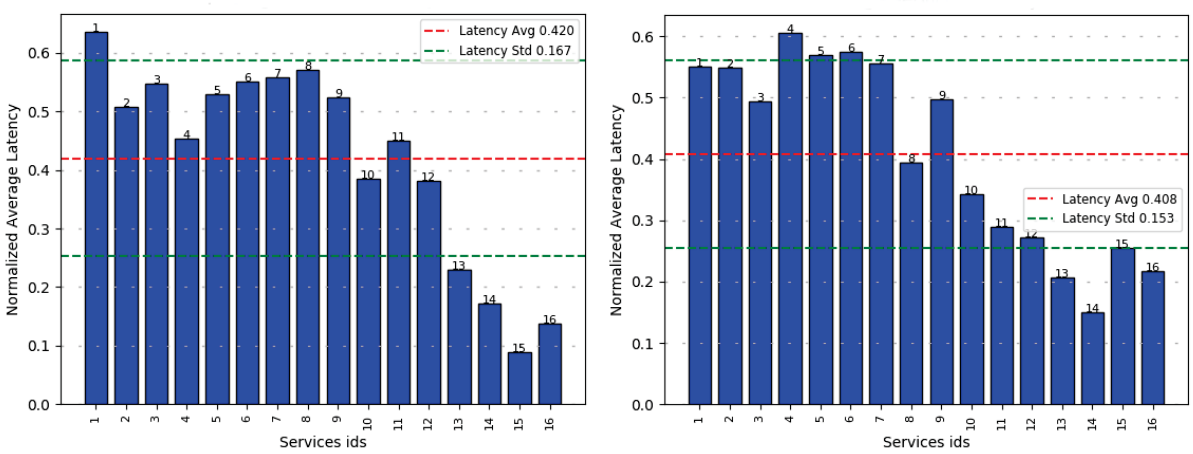
The services required by the set of IoT devices that are allocated to the Fog edge devices can be locally computed or forwarded to another computing instance. In addition to the computational resources monitoring, the DRR process also constantly monitors the device's running services, in order to decide if the incoming service demand is feasible to be executed immediately. However, this process does not set priorities for different service requirements, it only ensures that the maximum execution time requirement is met.

Each service may require variable and random amounts of computational resources, according to the limits described in Table 6.1, however, for the sixteen available services on the Fog layer, the maximum execution time limits were established in the interval of one to sixteen simulator's sampling time, which was defined according to Equation 4.17 in Section 4.2.

For the evaluation of the latency fulfillment to the available services in the Fog layer, the averages for each service at the end of the simulation were considered, for all service executions performed on all computational devices in the layer. Their latencies behavior was evaluated keeping the same criteria of the previous analyzes, considering the relative Fog edge devices' geo-dispersion, where they were arranged in a grid format or randomly dispersed, and the homogeneous and heterogeneous computational devices capacities, as illustrated by Figures 6.15 and 6.16. The means and standard deviations were measured for each case, and the maximum latency for each service is indicated at the top of their respective bars.

In all cases, the average latency on the services' execution fluctuated around 0.4 of their relative limits, while there was a fluctuation around 0.6 for the services with the shortest

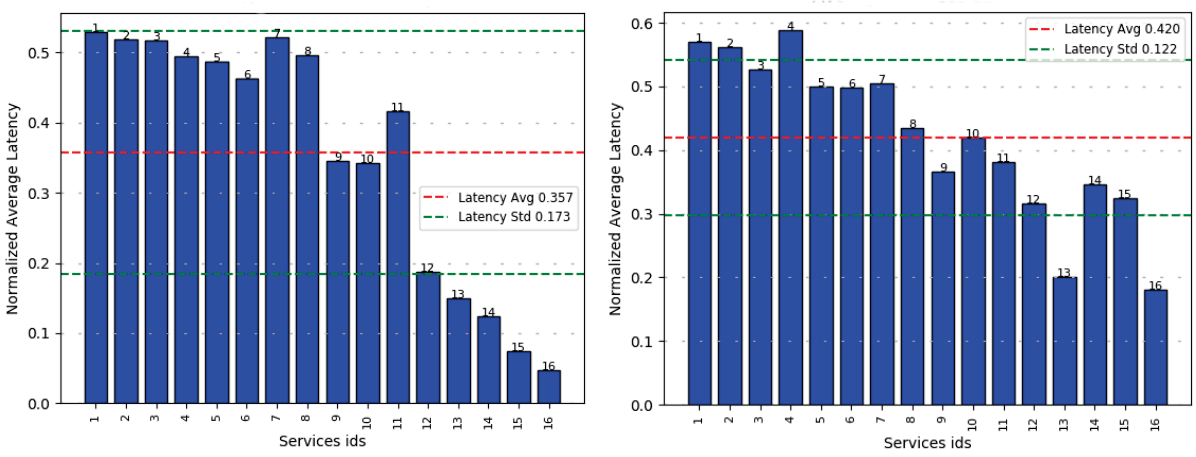
maximum latency, and the standard deviations remained around 0.15, which is a significant value when they are compared to the averages. This happens due to the tendency of the most urgent services to have higher relative latencies if compared to the less urgent services, being it a consequence of the absence of different priorities for the services or the implementation of task scheduling approaches, according to their maximum allowed latencies.



(a) Fog Edge devices in grid formation.

(b) Fog Edge devices scattered randomly.

Figure 6.15: Individual services' mean latency for the Fog edge layer with homogeneous devices.



(a) Fog Edge devices in grid formation.

(b) Fog Edge devices scattered randomly.

Figure 6.16: Individual services' mean latency for the Fog edge layer with heterogeneous devices.

Some fluctuations around the average latency values are expected, due to the stochastic nature of the edge system, however, patterns have not been recognized on which allow to associate the average services' execution latencies with the geo-dispersion or heterogeneity characteristics of the computational devices. Thus, the average latency for the service's execution may be considered an intrinsic characteristic of the computational device, where its internal mechanisms are responsible for the services' latencies control.

In the evaluated approach, the DRR process was able to keep the service's maximum latencies under control, and in no case, it was noticed the extrapolation of the maximum expected latency for the services available on the Fog edge layer in the computing devices, proving it to be effective for the control of the maximum expected services latencies' fulfillment.

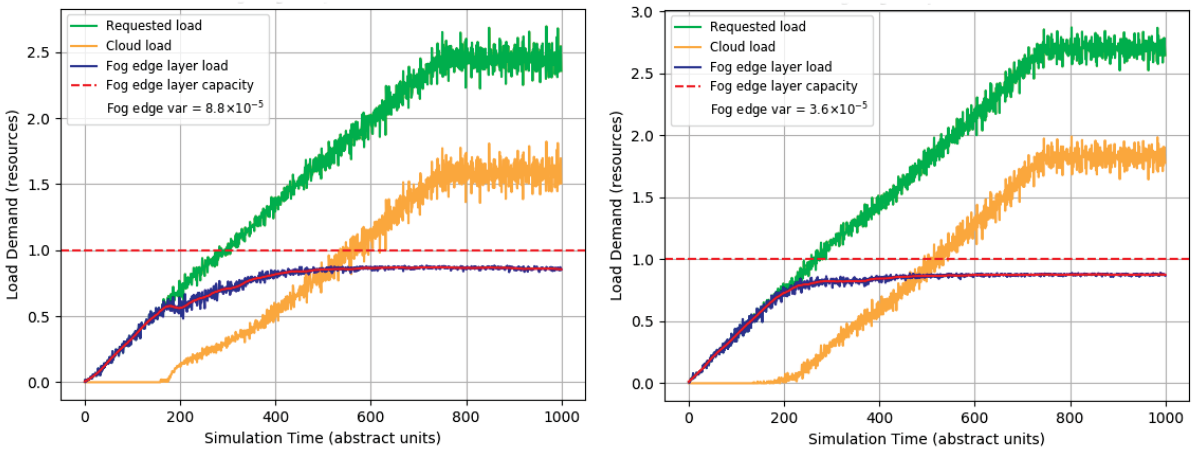
6.5 FOG EDGE LAYER LOAD BALANCING

The main purpose of the multi-objective load balancing approach is to reduce discrepancies in the computational load distribution among the computational devices in the Fog edge layer, which is achieved by maximizing the devices' resource usage while reducing the computational loads' variance. In this direction, previous analyses have shown that fog edge devices can assume different load profiles and that these load profiles influence the devices' average individual computing load, and consequently, the load balance of the entire Fog edge layer.

To accomplish a more effective analysis and evaluation of the presented Fog edge model and the load balancing approach, measurements were conduced on the evolution of the requested IoT layer load, on the load which is absorbed by the Fog layer and on the load which is offloaded to the Cloud instance, where the entire simulation period is considered.

In this evaluation, the relative positions of the computational devices were considered, where they were dispersed in a grid format and randomly scattered, as well as holding homogeneous computational capacities, illustrated in Figure 6.17 and also having heterogeneous computational capacities, shown in Figure 6.18. The green curves represent the IoT layer load demand, the blue curves denote the load assigned to the Fog edge devices and the orange curves indicate the load that has been offloaded to the Cloud instance. The Fog edge devices' normalized load limits are indicated by the dashed red lines, and their fitted load values are represented by the red curves. The Fog edge load reference value was set to 0.85 of the devices' capacity, and the load variances for the entire simulation period were calculated and they are indicated in the figures legends.

In the four illustrated cases by Figures 6.17 and 6.18, some characteristics of the simulated model are evident. For instance, the IoT devices wake-up time was attributed through a uniform distribution between the simulation times 1 and 750, producing in this interval an upward fitted line of demands for computational resources, and after time 750, where all the IoT devices are operational, the curve stabilizes horizontally. The vertical variations on this curve indicate different amounts of IoT devices becoming active, as well as different instant computational resources demands due to the characteristics of the different requested services.



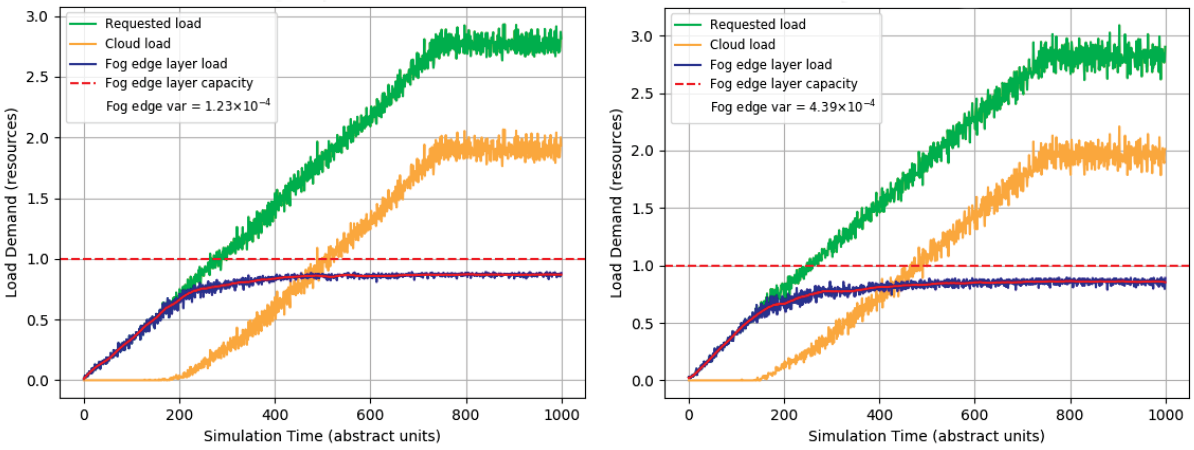
(a) Fog Edge devices in grid formation.

(b) Fog Edge devices scattered randomly.

Figure 6.17: IoT layer resources' requirements over the simulation period and the load sharing between the Cloud instance and the Fog edge layer with homogeneous computational devices.

Another characteristic that shows evident in all cases refers to the offloading process starting before the computational load of the Fog layer reaches the load value of 0.85, which was established as a reference load value. This behavior can be attributed to the fact that some devices can assume profiles of an average or high load before other devices and then they initiate

the offloading process, as well as the possibility of high rates of transient requests from the IoT layer. The transient load request effect can be clearly noticed in Figure 6.17(a) on simulation time 200, where it is shown a sudden increase in the offloading rate due to a short pike period on the Fog edge layer load demands. Such events may occur for all the simulation period, however, it is probable to happen before the IoT requests' stabilization, that is, for the present simulation settings, it is up to the simulation time 750.



(a) Fog Edge devices in grid formation.

(b) Fog Edge devices scattered randomly.

Figure 6.18: IoT layer resources' requirements over the simulation period and the load sharing between the Cloud instance and the Fog edge layer with heterogeneous computational devices.

For the load balancing analysis on the Fog edge layer, the variances measurements for all considered cases showed better results in cases where the devices have homogeneous computational capacities, reaching variances in the order of 10^{-5} in comparison to cases where the devices have heterogeneous computational capacities, which reached variances in the order of 10^{-4}. There was also noticed the influence of the computational devices dispersion patterns in the edge layer, wherein the cases where the devices were randomly dispersed, tending to form larger partitions, obtained a better decrease in the layer load variances if compared to the cases where the devices were arranged in the form of a grid.

Each of the four analyzed cases on the load balancing in the edge layer, which considers the full simulation period, is respectively in accordance with the trends pointed out by the edge layer load analysis in Section 6.3.3, indicating that a measurement performed during a limited time interval can provide a valid sampling for the computing devices load state for an extended period of time. This principle is used by the SRA process when deciding for the admission of a new IoT client, and by the DRR process when deciding the device's operational mode when a service data processing request arrives.

From the results obtained, it is possible to conclude that the multi-objective optimization approach is effective to maximize the utilization of the resources of the Fog edge computing devices, while promoting the minimization of the computational load variation for the entire layer and, thus, improves its load balancing.

6.6 SUMMARY AND DISCUSSION

This chapter has presented the evaluation of the multi-objective load balancing approach for Fog edge devices, which was done jointly to an analysis of the functional characteristics that permeate the relationship between layers at the edge of the network and their influence on the parameters' optimization which are addressed by the presented approach.

The following are discussions about the results obtained, divided into three parts. First, the simulation conditions and the analysis of the fog edge model, followed by the evaluation of the individual optimization objectives, and ending with the evaluation of the Fog Computing edge load balancing approach, which is the main contribution of this work.

Simulation Conditions and Fog Edge Model Analysis

The simulated scenario configuration considers the characteristics of the Fog Computing paradigm where the computing environment covers wide areas with a large number of devices and there is a predominance of wireless communication. The proposed method focuses on the edge layer devices only, and it is expected that the demands of the IoT layer exceed the available Fog edge resources, causing a possible overload condition. To achieve the expected scenario and condition, the simulator was configured according to the parameters shown in Table 6.1.

An analysis of the spatial and temporal demands of the IoT layer was made, where Figure 6.1 shows peak demands in determined areas. As load balancing depends on the matching between spatial and temporal demands of the IoT layer and the availability of the Fog layer, four relevant scenarios were chosen for the evaluation and validation of the proposed load balancing method, where the entropy of the IoT-Fog layers relationship is varied according to the geographical positions of the Fog edge devices, being them arranged equidistantly in grid format or randomly spread, as illustrated in Figure 6.2, as well as their computational capacities were considered, where the devices may have homogeneous or heterogeneous capacities.

From the definition of the dispersion models for the devices in the Fog edge layer, the IoT-Fog layers relationship analysis was carried out based on the wireless communication range. For some IoT devices chosen at random, Figure 6.3 indicates that the configuration where the devices are arranged randomly tends to form larger partitions if compared to the devices arranged in a grid format, which is relevant to the proposed load balancing approach. This trend is confirmed by the simulation shown in Figure 6.4, where the size of the partitions formed by each IoT device was measured. In the case where the Fog edge devices were dispersed in a grid format, partitions were formed containing between 3 and 6 devices while when they were randomly dispersed, most partitions have presented between 2 and 9 devices.

Under normal simulation conditions, several IoT devices tend to transmit simultaneously. As shown in the communication diagram illustrated by Figure 5.3, part of the process needs communication and evaluation from a specific device, the Fog Edge Controller. Thus, during the Static Resource Allocation (SRA) process, intense network traffic is generated both on the wireless network and on the wired network. The evaluation of this condition is shown in Figure 6.5, where the influence on the average sizes of the formed partitions can be seen. In the configuration where the Fog edge devices are in grid formation, there is a lower average on the number of messages transmitted on the wireless network, as well as less intense peaks when compared to the configuration where the devices are randomly arranged. However, in both cases, the IoT device admission process happens within the simulator's sampling period, demonstrating that the presented centralized approach, using the Fog Edge Controller as an external device, do not add additional delays to the system.

The simulated edge scenario, which comprises a wide geographical area, makes the Fog edge devices to receive transmissions from the IoT devices with RSSIs up to their sensitivity limits. This is relevant to the SRA process, as it considers the RSSI and the availability of resources on the device jointly. So, an evaluation of the perception of the IoT layer by the Fog edge layer was made, considering the RSSIs of all messages received during the entire simulation. Figure 6.6 shows the influences of the dispersion model, where randomly dispersed devices, which tend to form larger partitions, have more intense network traffic if compared to

the grid-shaped dispersion model. In both cases, it was noticed that the average of the RSSIs for the IoT devices admitted as clients demonstrate superior intensities than the average RSSIs of all devices that are within the communication range, however, most of the communication occurs close to the devices' sensitivity limit, indicating that RSSI optimization for the individual Fog edge devices happens smoothly, but without showing major discrepancies.

Evaluation of Optimization Objectives

The evaluation of the simulation results for the proposed approach was carried out for the three target objectives, which are: The improvement of RSSI for the inter-layer wireless communication, the compliance with the maximum latency of the available services, and the minimization of the load variance on computing devices, as follows.

In the RSSI optimization evaluation, the situations where there was no concurrency for the received requests from the IoT devices, shown in Figure 6.7, demonstrated a superior increase in the average RSSI for the devices admitted as clients, with an improvement of approximately 4.8dBm if compared to the situations where there were concurrent requests, shown in Figures 6.8 and 6.9, which evidenced an average improvement of approximately 2.3dBm, indicating that this may be a subject for improvements in the SRA algorithm to be addressed in future work. The cases where the heterogeneity and the relative positions of the computational devices were considered, relative to Figures 6.8 and 6.9, the positioning was the predominant factor, where the cases with the devices positioned at random showed better optimization if compared to devices displaced in grid-shaped positioning. However, for all tested cases and for all Fog edge devices, there was an improvement in the mean RSSI for IoT devices admitted as clients if compared to the average RSSI of the remaining devices within the wireless communication range, proving the SRA process as valid and effective on the RSSI improvement.

On the services' maximum expected latency fulfillment evaluation, the scenarios with variations of computational resources heterogeneity and devices' geo-spreading models were considered, as seen in figures 6.15 and 6.16. However, these characteristics variations showed no evident patterns which allow the average services' execution latencies to be associated with. In all cases, the average latencies fluctuated around 0.4 of their relative limits, while there was a fluctuation around 0.6 for the services with the shortest maximum latency, and no latency over-limit was noticed, indicating that the Dynamic Resource Reallocation (DRR) process was effective on the control of the maximum expected services latencies' fulfillment. Also, high relative values were measured for the standard deviations of the services latencies in all evaluated cases, however, this behavior is expected, as all services have the same execution priority. An optimization for the priority services latencies through scheduling techniques improving the presented approach may be considered for future work.

The evaluation of the workload distribution between the devices was carried out in some stages, in which characteristics that influence the load assignment to the devices were analyzed.

First, in Section 6.3.1, the total number of IoT clients admitted by each Fog edge device was analyzed, as well as the number of requests being computed or offloaded by each device. In this evaluation, the geographic distribution pattern of the devices proved to be the most influential factor in the admission of the total number of IoT clients, as evidenced in Figures 6.10 and 6.11. It is also possible to observe that devices can assume a diversified relationship between the total number of clients and the number of IoT requests being computed, and in some cases, the computing device may not fully use its resources, resulting in a device with a low load profile. This relationship is relevant to the DRR process since its algorithm fits the IoT devices' demands to the computational device temporal availabilities.

Then, different device load profiles were analyzed, as shown in Figure 6.12. It was noticed, as shown in Figure 6.12(d), that the computational device in a low load profile condition can go through periods of idleness, then negatively impacting the variance reduction on the device's load, which is relevant to the Fog edge layer load balancing.

In the analysis of the workload distribution among the computational devices, the different device load profiles were noticed due to the discrepancies in the measured load for different computing devices belonging to the same edge layer in all four analyzed cases, as shown in Figures 6.13 and 6.14. Despite the discrepancies measured on the individual devices, they have demonstrated a low impact on load balancing when the entire Fog edge layer is considered.

The average values measured for each device fluctuated around the reference load value that was stipulated at 0.85, as expected due to the inconsistencies of the instantaneous values of the loads on the devices, as shown in Figure 4.2, therefore the DRR algorithm makes its decisions based on an estimate of short-term loads, producing more reliable but not fully accurate estimates.

Despite the fluctuations measured for individual devices, the load variance for the entire Fog edge layer was reduced to 3.6×10^{-5} in the best case, shown in Figure 6.13(b), and it was reduced to 1.9×10^{-3} in the worst case, shown in Figure 6.14(b). However, this is a short-term analysis, taken at the end of the simulation period, that shows the load profiles that individual devices assume and provides a trend for the load balancing for the Fog edge layer. Therefore, a more complete analysis was also performed, considering the behavior of the load variations in the entire Fog edge layer for the entire simulation time. Thus, it was possible to dynamically evaluate the effectiveness of the proposed load balancing method, observing the behavior of the SRA and DRR processes in all stages of the simulated period.

Fog Computing Edge Load Balancing Proposed Approach Evaluation

The main objective of the presented approach is to improve the load balance in the computational devices on the edge of the Fog Computing environment. In this sense, the analysis and evaluation presented in Section 6.5 considered the entire simulation period, as well as the load requests made by the IoT layer, and the requests that were forwarded to the Cloud. The four scenarios previously used were also considered, where the geographical spread and the heterogeneity of the computational devices were varied. These scenarios proved to be relevant since smaller load variances were obtained in cases with homogeneous devices, as well as an improvement in the layer load balancing was noticed when the devices are randomly dispersed.

Despite using different cases and scenarios as well as different Fog layer models, works [18] and [19] present approaches for the load balancing in Fog environments that employ the same definition of load balancing adopted in this work, that is, the objective is to reduce the computational load variance on the devices belonging to the Fog layer. Also, it was adopted the same computational resource modeling, defined as a finite number of resource units, where each resource unit contains various physical resources. In both works, the authors achieved variances in load balancing within the order of 10^{-2}, while the method presented in this work achieved variances in the order of 10^{-4} for the worst cases, when the devices present heterogeneous computational capabilities, as shown in Figure 6.17, and in the order of 10^{-5} for the best cases, when the devices present homogeneous computational capabilities, as evidenced in Figure 6.18.

The obtained results demonstrate that the proposed method was successfully able to considerably decrease the variance in the loads attributed to Fog edge layer devices, which is relevant to load balancing while keeping their occupation very close to the reference value. Also, secondary optimization objectives were achieved. The mean RSSI for each Fog edge device was increased and the latencies for all the available services were kept under their thresholds.

The next chapter concludes this work and presents perspectives for future works.

7 CONCLUSION

Fog Computing is been considered an IoT enabler on supporting faster real-time processing of time-sensitive data by the deployment of geo-spreaded computational devices at the edge of the network, and unlike the Cloud computing which employs a physically centralized structure, Fog computing employs a distributed architecture requiring specific approaches to perform load management among its computing devices.

The main contribution of this work consists of a multi-objective optimization approach to the load balance in the Fog edge layer. The proposed approach was implemented through two algorithms, where the first promotes a static allocation of the IoT devices to the most feasible Fog edge device, considering the wireless signal strength and an estimative of the available computational resources on the Fog edge devices, and a second algorithm that promotes the reallocation of computational resources on the device, considering the IoT devices' temporal load demands and ensuring that the latencies of the requested services are maintained within their expected limits. To perform the evaluation, an edge model was described considering a large number of IoT nodes spread over a wide geographic area, where the limited range of wireless links between the edge devices plays an important role in the formation of clusters in the Fog layer, which are part of the IoT to Fog devices allocation process. Also, a simulator was implemented, describing the edge model under study, with the necessary details and providing the tools for the evaluation of both the edge model and the proposed multi-objective optimization approach.

The evaluation of the Fog edge model showed that in scenarios of a wide geographic region and a large number of devices, the IoT-Fog layers relationship is characterized by demands and offers of computational resources that have geographical and temporal variants, where the main task of the load management mechanism is to decrease the entropy between layers, promoting a matching between load's offers and demands as a function of time. Also, the multi-objective approach to load balancing applied to the presented Fog edge model proved to be effective in all its optimization dimensions. For all relevant cases that were analyzed, the RSSIs averages were optimized, and all services were performed with latencies below their stipulated limits, for all the computing devices. As for the load balancing in the Fog edge layer, the variances of the required loads could be reduced by up to four orders of magnitude.

With the evaluation of the Fog edge environment for the employed model and the obtained results, some future works are planned to improve the evaluation environment as well as to expand the functionalities and to improve the characteristics of the Fog system, as follows.

- Evaluate the presented approach considering the offloading of the IoT requests not only to the Cloud layer but also to computing devices from Fog intermediate layers.
- Evaluate the presented approach considering the devices and model properties sampled from different statistical distributions, as a normal distribution for instance.
- Evaluate the presented approach considering the adoption of a distributed approach instead of using a centralized controller element for the Static Resource Allocation process.
- Improve load distribution among Fog edge layer devices, considering the exchange of IoT clients between computing devices along the Dynamic Resources Reallocation process.
- Add another Fog Computing paradigm characteristic by implementing mechanisms that allow the mobility of IoT devices.
- Implement mechanisms that allow not only to control the latencies of the services but also to optimize their execution times, such as, the employment of advanced task scheduling techniques.

REFERENCES

[1] Roberto Minerva, Abyi Biru, and Domenico Rotondi. Towards a definition of the Internet of Things (IoT). Technical Report Rev.1, IEEE IoT Initiative, May 2015.

[2] Ashkan Yousefpour, Caleb Fung, Tam Nguyen, Krishna Kadiyala, Fatemeh Jalali, Amirreza Niakanlahiji, Jian Kong, and Jason P. Jue. All One Needs to Know about Fog Computing and Related Edge Computing Paradigms: A Complete Survey. *arXiv*, September 2018.

[3] OpenFog Architecture Workgroup. OpenFog Reference Architecture for Fog Computing. Technical Report OPFRA001.020817, OpenFog Consortium, Fremont, CA, USA, February 2017.

[4] Shancang Li, Li Da Xu, and Shanshan Zhao. The internet of things: a survey. *Information Systems Frontiers*, 17(2):243–259, April 2015.

[5] Gubbia Jayavardhana, Rajkumar Buyyab, Slaven Marusic, and Marimuthu Palaniswami. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660, February 2013.

[6] Andrea Zanella, Nicola Bui, Angelo Castellani, Lorenzo Vangelista, and Michele Zorzi. Internet of things for smart cities. *IEEE Internet of Things Journal*, 1(1):22–32, 2014.

[7] Angelo Cenedese, Andrea Zanella, Lorenzo Vangelista, and Michele Zorzi. Padova smart city: An urban internet of things experimentation. *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, pages 1–6, 2014.

[8] Andreas Kapsalis, Panagiotis Kasnesis, Iakovos S. Venieris, Dimitra I. Kaklamani, and Charalampos Z. Patrikakis. A Cooperative Fog Approach for Effective Workload Balancing. *IEEE Cloud Computing*, 4(2):36–45, March 2017.

[9] Charith Perera, C H I Harold Liu, S. Jayawardena, and Min Chen. A Survey on Internet of Things From Industrial Market Perspective. *IEEE Access*, 2:1660–1679, 2014.

[10] J. Caceres L. M. Vaquero, L. Rodero-Merino and M. Lindner. A break in the clouds: Towards a cloud definition. *SIGCOMM Computer Communication Review*, 39(1):50–55, January 2009.

[11] Peter Mell and Timothy Grance. The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology. Technical report, National Institute of Standards and Technology, Gaithersburg, MD, September 2011.

[12] Lei Jiao, Roy Friedman, and Xiaoming Fu. Cloud-based computation offloading for mobile devices: State of the art, challenges and opportunities. In *2013 Future Network and Mobile Summit*, pages 1–11. IEEE, July 2013.

[13] 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)*, pages 117–122. IEEE, nov 2014.

[14] Sami Yangui, Pradeep Ravindran, Ons Bibani, Roch H. Glitho, Nejib Ben Hadj-Alouane, Monique J. Morrow, and Paul A. Polakos. A platform as-a-service for hybrid cloud/fog environments. In *2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, volume 2016-Augus, pages 1–7. IEEE, June 2016.

[15] Xiaoqing Zhu, Douglas S Chan, Hao Hu, Mythili S. Prabhu, Elango Ganesan, and Flavio Bonomi. Improving video performance with edge servers in the fog computing architecture. In *Intel Technology Journal*, volume 19, pages 202–224. Intel, March 2015.

[16] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. 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, pages 13–16, New York, NY, USA, 2012. ACM.

[17] Mounib Khanafer, Mouhcine Guennoun, and Hussein T. Mouftah. A Survey of Beacon-Enabled IEEE 802.15.4 MAC Protocols in Wireless Sensor Networks. *IEEE Communications Surveys & Tutorials*, 16(2):856–876, 2014.

[18] Xiaolong Xu, Shucun Fu, Qing Cai, Wei Tian, Wenjie Liu, Wanchun Dou, Xingming Sun, and Alex X. Liu. Dynamic Resource Allocation for Load Balancing in Fog Environment. *Wireless Communications and Mobile Computing*, 2018, 2018.

[19] Xiaolong Xu, Qingxiang Liu, Lianyong Qi, Yuan Yuan, Wanchun Dou, and Alex X. Liu. A Heuristic Virtual Machine Scheduling Method for Load Balancing in Fog-Cloud Computing. *2018 IEEE 4th International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing, (HPSC) and IEEE International Conference on Intelligent Data and Security (IDS)*, pages 83–88, 2018.

[20] Mike Jia, Weifa Liang, Zichuan Xu, Meitian Huang, and Yu Ma. QoS-Aware Cloudlet Load Balancing in Wireless Metropolitan Area Networks. *IEEE Transactions on Cloud Computing*, 1(1):1–12, 2018.

[21] Chenhua Shi, Zhiyuan Ren, Kun Yang, Chen Chen, Hailin Zhang, Yao Xiao, and Xiangwang Hou. Ultra-low latency cloud-fog computing for industrial Internet of Things. In *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE, April 2018.

[22] Yiming Wang, Zhiyuan Ren, Hailin Zhang, Xiangwang Hou, and Yao Xiao. "Combat cloud-fog" network architecture for internet of battlefield things and load balancing technology. *Proceedings - 2018 IEEE International Conference on Smart Internet of Things, SmartIoT 2018*, pages 263–268, 2018.

[23] Arslan Musaddiq, Yousaf Bin Zikria, Oliver Hahm, Heejung Yu, Ali Kashif Bashir, and Sung Won Kim. A Survey on Resource Management in IoT Operating Systems. *IEEE Access*, 6:8459–8482, 2018.

[24] Farzad Samie, Lars Bauer, and Jörg Henkel. IoT Technologies for Embedded Computing: A Survey. In *Proceedings of the Eleventh IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis - CODES '16*, pages 1–10, Pittsburgh, PA, USA, October 2016. ACM Press.

[25] Pethuru Raj and Anupama C. Raman. *The Internet of Things: Enabling Technologies, Platforms, and Use Cases*. CRC Press, 2017.

[26] Shabnam Shadroo and Amir Masoud Rahmani. Systematic survey of big data and data mining in internet of things. *Computer Networks*, 139:19–47, July 2018.

[27] Djabir Abdeldjalil Chekired, Lyes Khoukhi, and Hussein T Mouftah. Industrial IoT Data Scheduling Based on Hierarchical Fog Computing: A Key for Enabling Smart Factory. *IEEE Transactions on Industrial Informatics*, 14(10):4590–4602, October 2018.

[28] Alex Davies. Cisco pushes IoT analytics to the extreme edge with mist computing. `https://rethinkresearch.biz/articles/cisco-pushes-iot-analytics-extreme-edge-mist-computing-2`, 2014. [Online] Feb 2019.

[29] Kannan Srinivasan and Philip Levis. RSSI is Under Appreciated. *Proceedings of the 3th ACM Workshop on Embedded Networked Sensors (EmNets '06)*, pages 1–5, 2006.

[30] Rodrigo Almeida, Ruben Oliveira, Daniela Sousa, Miguel Luis, Carlos Senna, and Susana Sargento. A Multi-Technology Opportunistic Platform for Environmental Data Gathering on Smart Cities. In *2017 IEEE Globecom Workshops (GC Wkshps)*, pages 1–7. IEEE, December 2017.

[31] Carla Mouradian, Diala Naboulsi, Sami Yangui, Roch H. Glitho, Monique J. Morrow, and Paul A. Polakos. A Comprehensive Survey on Fog Computing: State-of-the-Art and Research Challenges. *IEEE Communications Surveys and Tutorials*, 20(1):416–464, 2018.

[32] Wei Yu, Fan Liang, Xiaofei He, William Grant Hatcher, Chao Lu, Jie Lin, and Xinyu Yang. A Survey on the Edge Computing for the Internet of Things. *IEEE Access*, 6(c):6900–6919, 2018.

[33] Swati Agarwal, Shashank Yadav, and Arun Kumar Yadav. An Efficient Architecture and Algorithm for Resource Provisioning in Fog Computing. *International Journal of Information Engineering and Electronic Business*, 8(1):48–61, 2016.

[34] Mohit Taneja and Alan Davy. Resource Aware Placement of Data Analytics Platform in Fog Computing. *Procedia Computer Science*, 97:153–156, 2016.

[35] Xiaolong Xu, Xuan Zhao, Feng Ruan, Jie Zhang, Wei Tian, Wanchun Dou, and Alex X. Liu. Data Placement for Privacy-Aware Applications over Big Data in Hybrid Clouds. *Security and Communication Networks*, 2017:1–15, 2017.

[36] Xiaolong Xu, Wanchun Dou, Xuyun Zhang, Chunhua Hu, and Jinjun Chen. A traffic hotline discovery method over cloud of things using big taxi GPS data. *Software: Practice and Experience*, 47(3):361–377, March 2017.

[37] Jerry Zhao and Ramesh Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, pages 1–13, New York, NY, USA, 2003. ACM.

[38] IEEE Standards. *IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs)*. IEEE, 2006.

[39] Stelios Sotiriadis, Nik Bessis, Eleana Asimakopoulou, and Navonil Mustafee. Towards simulating the internet of things. *Proceedings - 2014 IEEE 28th International Conference on Advanced Information Networking and Applications Workshops, IEEE WAINA 2014*, pages 444–448, 2014.

[40] Gupta Harshit, Dastjerdi Amir Vahid, Ghosh Soumya K, and Buyya Rajkumar. iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. *Software - Practice and Experience*, 47:1275–1296, 2017.

[41] Stelios Sotiriadis, Nik Bessis, Nick Antonopoulos, and Ashiq Anjum. SimIC: Designing a new Inter-Cloud simulation platform for integrating large-scale resource management. *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, pages 90–97, 2013.

[42] R. N. Calheiros, R. Ranjan, C. De Rose A. Beloglazov, and R. Buyya. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software - Practice and Experience*, 41(1):23–50, 2011.

[43] Redowan Mahmud and Rajkumar Buyya. Modeling and Simulation of Fog and Edge Computing Environments Using iFogSim Toolkit. In *Fog and Edge Computing*, chapter 17, pages 433–465. Wiley STM, 2019.

[44] Klaus G. Müller and Tony Vignaux. SimPy - Discrete event simulation for Python (version 3.0.9). `https://simpy.readthedocs.io/en/latest`, 2016. Online; accessed 12-March-2019.