Universidade Estadual de Campinas
Instituto de Computação

Rodrigo Augusto Cardoso da Silva

The Fog Node Location Problem

O Problema de Localização de Nós Névoa

CAMPINAS
2022

Rodrigo Augusto Cardoso da Silva

The Fog Node Location Problem

O Problema de Localização de Nós Névoa

Tese apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Thesis presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Supervisor/Orientador: Prof. Dr. Nelson Luis Saldanha da Fonseca

Este exemplar corresponde à versão final da Tese defendida por Rodrigo Augusto Cardoso da Silva e orientada pelo Prof. Dr. Nelson Luis Saldanha da Fonseca.

CAMPINAS
2022

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

Si38f

Silva, Rodrigo Augusto Cardoso da, 1990-
The fog node location problem / Rodrigo Augusto Cardoso da Silva. – Campinas, SP : [s.n.], 2022.

Orientador: Nelson Luis Saldanha da Fonseca.
Tese (doutorado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Computação em névoa. 2. Computação em nuvem. 3. Localização de instalações (Pesquisa operacional). 4. Aeronaves não tripuladas. 5. Otimização multiobjetivo. I. Fonseca, Nelson Luis Saldanha da, 1961-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: O problema de localização de nós névoa
Palavras-chave em inglês:
Fog computing
Cloud computing
Facility location (Operational research)
Drone aircraft
Multiobjective optimization
Área de concentração: Ciência da Computação
Titulação: Doutor em Ciência da Computação
Banca examinadora:
Nelson Luis Saldanha da Fonseca [Orientador]
Edmundo Roberto Mauro Madeira
Jó Ueyama
Luiz Fernando Bittencourt
Maycon Leone Maciel Peixoto
Data de defesa: 02-06-2022
Programa de Pós-Graduação: Ciência da Computação

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0002-9874-2605
- Currículo Lattes do autor: http://lattes.cnpq.br/8593993681646906

Universidade Estadual de Campinas
Instituto de Computação

Rodrigo Augusto Cardoso da Silva

The Fog Node Location Problem

O Problema de Localização de Nós Névoa

Banca Examinadora:

- Prof. Dr. Nelson Luis Saldanha da Fonseca
  Universidade Estadual de Campinas
- Prof. Dr. Edmundo Roberto Mauro Madeira
  Universidade Estadual de Campinas
- Prof. Dr. Jó Ueyama
  Universidade de São Paulo
- Prof. Dr. Luiz Fernando Bittencourt
  Universidade Estadual de Campinas
- Prof. Dr. Maycon Leone Maciel Peixoto
  Universidade Federal da Bahia

A ata da defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria do Programa da Unidade.

Campinas, 02 de junho de 2022

Dedication

For my mother Elizabeth.

"I could tell you my adventures
— beginning from this morning,"
said Alice a little timidly:
"but it's no use going back to yesterday,
because I was a different person then."
(Lewis Carroll, Alice in Wonderland, 1865)

Acknowledgements

Earning my Ph.D. degree was the most complex challenge I have ever had in my life (so far!). I am deeply thankful for the many people and institutions that supported me during these years. First, I would like to thank professor Nelson Fonseca. His academic guidance made me a better researcher and person. I am grateful for my years working with him, and the lessons taught me. I also want to thank professor Raouf Boutaba for his supervision at the University of Waterloo in Canada. I am grateful to the members of my Ph.D. defense committee, Prof. Dr. Edmundo Roberto Mauro Madeira, Prof. Dr. Jó Ueyama, Prof. Dr. Luiz Fernando Bittencourt, and Prof. Dr. Maycon Leone Maciel Peixoto.

I am genuinely grateful to my parents, Elizabeth and Jairo, for supporting me from childhood until now. I would have never been in a university without them. I am also very grateful to my sister Priscila who provided different types of support in all these years. I am truly grateful to my best friend, Jéssyca Nobre; without her wise and funny words, I would undoubtedly have resigned from this Ph.D. years ago.

I want to thank many friends and colleagues for their friendship, support, and help during this Ph.D.: Atílio Gomes, Carlos Astudillo, Cyndi Lauper of Kindlys Youngs, Daniel Lago, Diego Oliveira, Diogo Gonçalves, Fábio Usberti, Fabíola Oliveira, Fernanda Brito, Filipy Borghi, Guilherme Russi, Helder Oliveira, Joahannes Bruno, John Hofstetter, José Carrilho, Judy Guevara, Karen Fletcher, Leandro Villas, Luciano Chaves, Marcela Porto, Marcelo Minetto, Matheus Zago, Mauro Mulati, Milena Andreotti, Pedro Libório, Rodrigo Vignoli, Ruben Cardoso, Sindo Dias, Takeo Akabane, Thaís Ussami, Tiago Fonseca, Vanderlei Busnardo, and Vanessa Maike. You all are part of this journey in some way. Special thanks to all friends from the Computer Networks Laboratory (LRC) and the Institute of Computing (IC), and all IC staff.

I would also like to acknowledge the financial support provided for this thesis. This study was financed in part by The Brazilian National Council for Scientific and Technological Development (CNPq), grant 140464/20182. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior Brasil (CAPES) - Finance Code 001. This study was financed in part by the São Paulo Research Foundation (FAPESP), grant 2015/24494-8. This study was financed in part by Google through the Latin America Research Awards program in 2019 and 2020, and the Government of Canada through the Emerging Leaders in the Americas Program.

Resumo

A computação em névoa é um paradigma no qual recursos computacionais estão próximos dos usuários finais, complementando a computação em nuvem e permitindo a execução de cargas de trabalho com latência reduzida. A computação em névoa permite o desenvolvimento de novas aplicações com requisitos de baixa latência e também pode melhorar a execução de aplicações típicas da nuvem. A infraestrutura de uma névoa é composta por nós névoa, que são dispositivos com recursos de processamento, comunicação, e armazenamento posicionados no contínuo entre os usuários finais e a nuvem. Um dos primeiros passos na criação de uma infraestrutura de computação em névoa é a determinação da localização de nós névoa. Essa decisão é fundamental pois os usuários são móveis e, consequentemente, os nós névoa precisam estar posicionados em diferentes regiões geográficas a fim de suportar os requisitos de latência das aplicações. Além disso, a localização de nós névoa, assim como a configuração de hardware dos nós, precisa considerar as demandas variáveis de usuários no tempo e espaço.

Esta tese propõe soluções para a localização de nós névoa considerando diferentes aspectos de uma infraestrutura de computação em névoa. Primeiro, uma solução para reduzir as despesas de capital da infraestrutura é proposta. Segundo, a localização de nós névoa é decidida de forma a reduzir o consumo de energia dos dispositivos de usuários. Terceiro, soluções com nós névoa montados em veículos aéreos não tripulados (VANTs) são estudadas. Finalmente, um mecanismo de alocação de recursos para ambientes de névoa e nuvem é proposto. Todas soluções visam prover a melhor infraestrutura para usuários executando cargas de trabalho com requisitos de baixa latência. As diferentes soluções podem ser aplicadas individualmente ou combinadas. Nesta tese, o problema de localização de nós névoa é formulado com modelos de programação linear, e diferentes algoritmos heurísticos são propostos a fim de lidar com cenários representando áreas metropolitanas. Todas avaliações foram obtidas através de simulações.

A avaliação das soluções desta tese foi feita através de simulações de áreas metropolitanas habitadas por milhões de pessoas. Nós névoa são caracterizados de acordo com sua localização e capacidade de processamento. VANTs com operações limitadas por baterias também são simulados como nós névoa. Os resultados mostram que, apesar da dificuldade de lidar com demandas variáveis, diferentes soluções são possíveis para reduzir a subutilização de recursos, como reduzir ligeiramente a aceitação de usuários a fim de reduzir os custos de implantação, ou empregar VANTs para processar picos de demandas. Os algoritmos propostos são escaláveis. Esta tese amplia o conhecimento do problema de localização de nós névoa e trabalhos futuros podem dar continuidade às soluções aqui propostas.

Abstract

Fog computing is a paradigm in which resources are close to the end-users, complementing cloud computing and allowing the execution of workloads with reduced latency. Fog computing enables the deployment of new applications with low-latency requirements and can improve the execution of typical cloud applications. Fog computing relies on fog nodes, facilities with processing, networking, and storage resources placed in the continuum between end-users and the cloud. An early step in the design of a fog computing infrastructure is the location of fog nodes. This decision is crucial because end-users are mobile and, consequently, fog nodes must be deployed in different geographical regions to meet the latency requirements of applications. Moreover, users' demands are variable in time. Therefore, the location of fog nodes as well as their hardware configuration must take into account the variable demands of end-users in time and space.

This thesis proposes solutions to the location of fog nodes considering different aspects of a fog computing infrastructure. First, a solution to reduce the capital expenditure of the infrastructure is proposed. Second, the location of fog nodes is decided so that the end-user devices can reduce their energy consumption. Third, solutions with mobile fog nodes mounted on unmanned aerial vehicles (UAVs) are investigated. Finally, a resource allocation mechanism for fog-cloud infrastructures is proposed. All solutions aim at providing the best infrastructure for end-users running workloads with low-latency requirements. Different solutions can be individually applied or combined. In this thesis, the fog node location problem is formulated as linear programming models, and different heuristic algorithms are proposed to deal with scenarios representing metropolitan areas. All evaluations were made using simulations.

The evaluation of solutions in this thesis was made using simulations of metropolitan areas inhabited by millions of people. Fog nodes are characterized by their location and processing capacity. UAVs with operations limited by batteries are also simulated as fog nodes. Results show that, although dealing with variable demands is challenging, different solutions are possible to reduce underutilization of resources, such as slightly reducing the acceptance of requests to obtain large savings with the deployment costs, or employing UAVs to process peaks of demands. The proposed algorithms were shown to be scalable. The work in this thesis pushes the boundaries of the knowledge of the fog node location problem and can be adapted for future work.

List of Figures

List of Tables

List of Algorithms

Acronyms

3GPP 3rd Generation Partnership Project
AP Access point
API Application programming interface
BS Base station
CAPEX Capital expenditure
CDR Call detail record
D2D Device-to-device
DAG Directed acyclic graph
DC Data center
DIFS Distributed coordination function
EC Edge computing
EDTA Energy and Demand Trade-off Algorithm
ETSI European Telecommunications Standards Institute
GPR Gaussian process regression
GPRFCA Gaussian Process Regression for Fog-Cloud Allocation
GPS Global positioning system
GS Ground station
HAP High-altitude platform
IaaS Infrastructure as a service
ILP Integer linear programming
IoT Internet of Things
IEEE Institute of Electrical and Electronics Engineers
LAP Low-altitude platform
LoS Line-of-sight
LTE Long term evolution
MACC Mobile ad hoc cloud computing
MC Mobile computing
MCC Mobile cloud computing
MDPI Multidisciplinary Digital Publishing Institute
MEC Multi-access edge computing

MILP Mixed-Integer linear programming

NAS Network-attached storage

NASA National Aeronautics and Space Administration

NFV Network function virtualization

NGA National Geospatial-Intelligence Agency

NIST National Institute of Standards and Technology

OpenFog RA OpenFog reference architecture

PaaS Platform as a service

PUE Power usage effectiveness

QoS Quality of service

QoT Quality of transmission

SaaS Software as a service

SIFS Short inter-frame space

SMS Short message service

STUFog Spatio-Temporal UAV Fog Node Location

SUL Sequential UAV Fog Node Location

UAV Unmanned aerial vehicle

UFL UAV fog node location

VM Virtual machine

VTOL Vertical take-off and landing

Contents

Part I

Introduction and State-of-the-art

Chapter 1

Introduction

The number of devices connected to the Internet has already surpassed the global population [44], and 1.5 mobile devices per inhabitant are predicted for 2022 [80]. Such growth also increases the volume of mobile data traffic, expected to be 77.5 exabytes per month by 2022, a seven-fold increase in the period from 2017 to 2022 [80]. Data generated by the Internet of Things (IoT) devices have commonly been processed in the cloud [29], which provides computing and storage capabilities for resource-limited IoT devices. Cloud computing relies on large data centers that host computing, networking, and storage resources accessed on-demand through the Internet. However, they are typically located in remote areas [29], which makes a variety of applications with strict latency requirements unfeasible. One solution to alleviate this limitation is the employment of fog computing, an architecture to provide computing, storage, and networking capabilities anywhere along the continuum between the cloud and the end-users [79].

Fog computing was designed to support delay-sensitive applications as well as mobility by providing computing, networking, and storage capabilities at the edge of the network [79]. Fog computing fills the gap in service provisioning for latency-sensitive applications not considered by cloud computing. The fog allows the reduction of delays to only a few milliseconds. Moreover, fog is a distributed architecture, not centralized as are clouds. Fog computing was designed to complement cloud computing, but not to replace it.

Fog nodes are the basic units for fog computing and can be a network device with processing capabilities, dedicated servers, or computational servers to coordinate underlying devices [70]. Previous work [79, 70, 98, 59] has discussed the role of fog nodes in the architecture and their connection to other network elements, but have not discussed the impact of the deployment of fog nodes on different geographical locations. The location of the fog nodes consists in deciding where they should be placed given a set of potential locations and the devices available for deployment. The solution to the problem is crucial for fog providers. Indeed, the location decision affects both users and the provider. Wrong decisions can jeopardize user access: if the delay in accessing the fog impacts the application, user expectations and needs will not be fulfilled. Moreover, the deployment of servers influences the costs of fog providers: reckless decisions can guarantee users' satisfaction, but at a high deployment cost.

One challenge in deciding the location of fog nodes is the variability of end-user demands. There are both time and space variabilities due to the mobility of users [43], which

tends to concentrate a large number of users on certain areas during short periods. This can cause overdimensioning of the fog infrastructure, making processing resources idle for the most time. This thesis aims at dealing with such an issue in different ways. First, it evaluates how fixed fog nodes should be positioned so that underutilization is minimized. Then, it evaluates the employment of mobile fog nodes dispatched to different locations to cover peaks of demands.

The employment of fog nodes mounted on vehicles allows resource mobility, complementing the fixed fog infrastructure. Mobile fog nodes can be dispatched on demand to augment the resources of fixed fog nodes or go to locations without any terrestrial fog node. Unmanned aerial vehicles (UAVs) are highly suitable to host fog nodes due to their small size, flexibility to fly and access remote locations, and remote or autonomous operation. Moreover, UAVs can be equipped with processing and networking capabilities, allowing them to process user workload at the edge. However, some issues need to be addressed when using fog nodes mounted on UAVs. UAVs have limited access to energy supplies, which considerably limits their operational time. Moreover, obtaining good-quality wireless links can be challenging due to a lack of physical stability during flights. A significant part of this thesis is devoted to studying the location of fog nodes mounted on UAVs.

This thesis proposes solutions for variations of the fog node location problem. All solutions considered variable demands in time and space, and actual data of cellular networks were used in the evaluation. All results were obtained by simulations. There are solutions for infrastructures with only fixed servers, with servers on UAVs, and a solution with both types of servers. The fog node location is an optimization problem, and this thesis modeled it using linear programming formulations. Algorithms were also proposed to deal with large-scale inputs representing metropolitan areas. This thesis also proposes a solution for the allocation of resources in fog-cloud infrastructures. Results obtained in this thesis show that the proposed solutions can deal with the variability of demands, producing results for scenarios with many locations and fog nodes. Moreover, this thesis shows that the deployment of UAVs as fog nodes is possible with the proper location decision.

The remainder of this chapter provides further details of this thesis and is organized as follows. Section 1.1 states the objectives of the thesis. Section 1.2 summarizes the main contributions of this work. Section 1.3 lists the publications resulting from the Ph.D. Finally, Section 1.4 presents the outline of the remaining chapters.

1.1 Objectives

This thesis attempts to provide answers to the fog node location problem, given by the following question: *"How should fog nodes be located to process end-user demands that are variable in time and space?"*. In order to answer this question, this thesis studied the problem of selecting which locations should be used for the deployment of fog nodes from a set of candidate locations. This problem is known as the fog node location problem, an optimization problem that is a variation of the facility location problem. The main

goal is to process the maximum workload possible, but other criteria were adopted in multi-criterial formulations. This thesis proposes solutions to variations of the fog node location problem with different inputs.

Other research questions were elaborated to study this problem from different perspectives. The first perspective is that of the reducing the capital expenditure (CAPEX), aimed at answering the following question *"How should fog nodes be located to process end-user demands variable in time and space to reduce the cost of the fog infrastructure?"*. This is the first question because costs and variable demands are inherent aspects of the fog node location problem. The first solution serves as basis to investigate the remaining questions. This study is presented in Chapter 4.

The second perspective of the fog node location problem focuses on the energy consumption of end-user devices, attempting to answer the following question: *"How should fog nodes be located to process end-user demands variable in time and space to reduce the energy spent by end-user devices?"*. The answer to this question addresses an important need of end-users, which is the battery limitation of mobile devices. An infrastructure designed to reduce such an energy consumption can extend the time users remain connected to the infrastructure, which is beneficial for both users and providers. This study is presented in Chapter 5.

After studying these questions, this thesis focuses on the employment of a mobile infrastructure to further explore the variability of demands in time and space, starting with an important question: *"Are UAVs worth adopting to replace fixed nodes in a fog infrastructure?"*. By answering this question, this thesis evaluates to what extent fixed fog nodes can be replaced by aerial nodes, considering the advantages and limitations of each type of node. This study is presented in Chapter 6.

Results from the previous investigation shows that UAVs have a great potential to be used as fog nodes. Therefore, this topic was further explored considering different types of UAVs: rotary-wing and fixed-wing. For that, two questions are formulated: *"What should be the location and operation period of fog nodes mounted on rotary-wing UAVs to maximize the number of end-users served while reducing the delay between ground nodes and UAVs?"* and *"How should fixed-wing UAVs be positioned to provide a fog computing infrastructure to deal efficiently with variable demands in relation to time and space, as well as maximize the number of processed requests?"*. Both questions were studied to understand the role of different types of UAVs as fog nodes. Different solutions address the differences between these types of UAV. These studies are presented in Chapters 7 and 8.

Finally, this thesis also includes a proposal of a resource allocation mechanism to be used in any infrastructure with fog and cloud, whether fixed or aerial nodes were considered. Applications of end-users can be organized as sets of dependent tasks and, in this context, an important question should be answered: *"Which tasks of an application should be processed by the fog and which ones should be processed by the cloud?"*. The answer to this question is a resource allocation mechanism that considers the processing and networking requirement of applications to reduce blockage of user requests. This study is presented in Chapter 9.

1.2 Contributions

This thesis modeled different variations of the fog node location problem using either mixed-integer linear programming (MILP) or integer linear programming models (ILP), and it also proposed algorithms to deal with large-scale instances. The following list summarizes these contributions:

- A multi-objective MILP model for the fog node location problem with fixed fog nodes aimed at optimizing the acceptance ratio of requests, infrastructure cost, and fog node usage (Chapter 4);
- A multi-objective MILP model and the Energy and Demand Trade-off algorithm for the fog node location problem with fixed fog nodes aimed at optimizing the acceptance ratio of requests, the energy spent by end-user devices, and the infrastructure cost (Chapter 5);
- A multi-objective MILP model and the UAV Fog Node Location algorithm for the fog node location problem with fixed and mobile fog nodes aimed at optimizing the acceptance ratio of requests, and infrastructure cost (Chapter 6);
- A multi-objective ILP model and the Sequential UAV Fog Node Location algorithm for the fog node location problem with mobile fog nodes mounted on rotary-wing UAVs aimed at optimizing the acceptance ratio of requests, infrastructure cost, and latency (Chapter 7);
- A multi-objective ILP model and the Spatio-Temporal UAV Fog Node Location algorithm for the fog node location problem with mobile fog nodes mounted on fixed-wing UAVs aimed at optimizing the acceptance ratio of requests, infrastructure cost, and latency (Chapter 8);
- The Gaussian Process Regression for Fog-Cloud Allocation mechanism for the allocation of tasks either in the fog or in the cloud, aimed at optimizing the acceptance ratio of requests and fog node usage (Chapter 9).

1.3 Publications

Results of the research conducted in this thesis were published in international journals and conferences. Moreover, during the Ph.D., the student collaborated with other researchers, which led to publications that are not a direct result of this thesis. The list of publications is presented below:

- **R. A. C. da Silva** and N. L. S. da Fonseca, "Resource Allocation Mechanism for a Fog-Cloud Infrastructure,", 2018 IEEE International Conference on Communications (ICC), 2018, pp. 1-6, doi: 10.1109/ICC.2018.8422237. Status: published.
- **R. A. C. da Silva** and N. L. S. da Fonseca, "On the Location of Fog Nodes in Fog-Cloud Infrastructures", Sensors 2019, 19, 2445, doi: 10.3390/s19112445. Status: published.

- **R. A. C. da Silva** and N. L. S. da Fonseca, "Location of Fog Nodes for Reduction of Energy Consumption of End-User Devices,", IEEE Transactions on Green Communications and Networking, vol. 4, no. 2, pp. 593-605, June 2020, doi: 10.1109/TGCN.2020.2986753. Status: published.
- **R. A. C. da Silva**, N. L. S. da Fonseca and R. Boutaba, "Evaluation of the Employment of UAVs as Fog Nodes," in IEEE Wireless Communications, vol. 28, no. 5, pp. 20-27, October 2021, doi: 10.1109/MWC.101.2100018. Status: published.
- **R. A. C. da Silva** and N. L. S. da Fonseca, "Design of fog computing infrastructures with rotary-wing UAVs", 2022 IEEE Global Communications Conference (GLOBECOM), 2022. Status: under review.
- **R. A. C. da Silva** and N. L. S. da Fonseca, "Design of aerial fog computing with fixed-wing UAVs", IEEE Wireless Communications, 2022. Status: under review.

Other publications during the Ph.D. course:

- Daniel G. Lago, **Rodrigo A.C. da Silva**, Edmundo R.M. Madeira, Nelson L.S. da Fonseca, and Deep Medhi, "SinergyCloud: A simulator for evaluation of energy consumption in data centers and hybrid clouds", Simulation Modelling Practice and Theory, Volume 110, 2021, 10.1016/j.simpat.2021.102329. Status: published.
- Ana Isabel Montoya-Munoz, **Rodrigo A.C. da Silva**, Oscar M. Caicedo Rendon, and Nelson L. S. da Fonseca, "Reliability Provisioning for Fog Nodes in Smart Farming IoT-Fog-Cloud Continuum", Computers and Electronics in Agriculture, 2022. Status: under review.

This thesis includes materials from the first-authored papers listed above. Most papers were published by or submitted to the Institute of Electrical and Electronics Engineers (IEEE), and one paper was published by the Multidisciplinary Digital Publishing Institute (MDPI). The IEEE has a policy on the reuse of published papers on a thesis stating that "The IEEE does not require individuals working on a thesis to obtain a formal reuse license". For the MDPI, the copyright is retained by the authors while the paper is published under a Creative Commons CC BY 4.0 license. Images and portions of the text of the first-authored papers listed above were used in Chapters 1–10.

1.4 Thesis outline

This thesis is organized into five parts containing a total of ten chapters; such organization is illustrated in Figure 1.1. Part I is introductory and reviews the main concepts to understand this thesis. Part II presents solutions to the fog node location problem with only terrestrial infrastructure. Part III introduces solutions to the fog node location problem with UAVs used as fog nodes. Part IV presents a resource allocation mechanism made for fog-cloud infrastructures. Finally, Part V concludes this thesis.

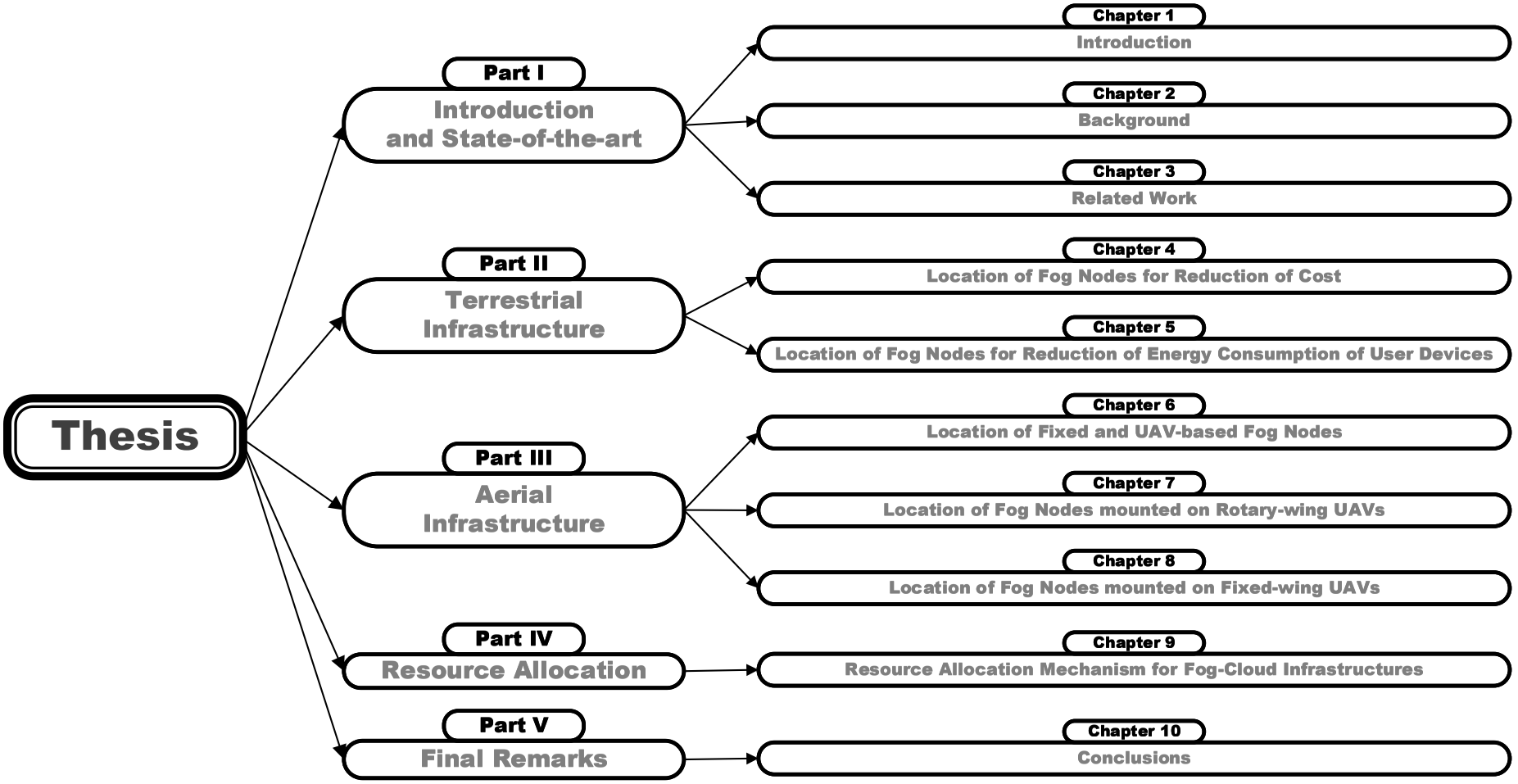


Figure 1.1: Organization of this thesis.

Part I contains Chapters 1–3. The current chapter (Chapter 1) introduces this thesis. Chapter 2 explains the concepts needed to understand the work developed in this thesis, while Chapter 3 reviews different papers from the literature related to the proposals in Chapters 4–9.

Part II comprises two chapters. First, Chapter 4 describes a solution to the fog node location problem in which reducing the infrastructure cost is the main goal. Second, Chapter 5 models a fog node location solution to reduce the energy spent by end-user devices.

Results obtained in Part II identified a high resource underutilization with only fixed fog nodes. Part III attempts to explore this gap by using mobile fog nodes mounted on UAVs, and it has three chapters. Chapter 6 presents a solution to the fog node location problem with both fixed and aerial nodes. The work in Chapters 7 and 8 solves the location problem assuming the ground infrastructure already exists and only UAVs need to be located. Chapter 7 employs only rotary-wing UAVs and Chapter 8 only fixed-wing UAVs.

Part IV has only one chapter (Chapter 9) that presents a resource allocation mechanism for fog-cloud infrastructures. Finally, Part V has one chapter dedicated to concluding this thesis. Chapter 10 presents the main conclusions and limitations of this thesis and suggests future directions.

Chapter 2

Background

This chapter reviews the main concepts used in this thesis and is organized as follows. Section 2.1 presents the definition of cloud computing and its main characteristics. Section 2.2 introduces the concept of fog computing, presents its differences to other concepts, defines fog node, and discusses typical applications and architectures. Section 2.3 presents an overview of the facility location problem. Section 2.4 discusses the optimization of multi-objective problems. Finally, Section 2.5 reviews the main concepts of UAVs, such as their classification, operation, energy consumption, and wireless channel modeling.

2.1 Cloud computing

The most accepted definition of cloud computing is the one given by the National Institute of Standards and Technology (NIST): "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." [72]. The role of the cloud provider is to offer access to computing, network, and storage services in a transparent way to the users. Cloud users neither need to know where the resources are located nor underlying details (such as hardware, operating system management, and cooling infrastructure): they just need to be able to obtain access to resources or release them in an on-demand manner, paying for what they use.

According to NIST, cloud computing provides three main service models: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) [72]. Under SaaS, users rent applications hosted in the cloud, without the need of managing physical resources or operating systems. Under PaaS, users deploy their applications using programming languages, libraries, and services hosted on the cloud. Under IaaS, users rent an operating system with a set of built-in software, and they are free to deploy arbitrary software and applications.

Cloud computing has four deployment models: private cloud, community cloud, public cloud, and hybrid cloud. A private cloud is used by a single organization and is typically maintained by the same organization. Community clouds are used by multiple consumers that share common needs and can be owned and managed by organizations in the com-

munity or by a third party. Public clouds offer services for the general public and are maintained by a cloud provider. A hybrid cloud is an infrastructure composed of two or more clouds with different deployment models, and an inter-operation between different clouds.

The physical resources of a cloud are kept in data centers (DCs), which are large facilities that host computing, networking, and storage devices. The access to services hosted in public data centers is remote, via the Internet. Typical devices in data centers include servers for processing various workloads, switches, and routers to provide intra-DC and inter-DC connectivity, network-attached storage (NAS) for keeping operating system images and users' data, and cooling infrastructure to maintain the adequate temperature. Servers inside data centers are typically connected by wired networks and their arrangement follows a particular network topology, such as the one suggested by Cisco [1] or the Fat-tree topology [5].

A cloud consists of one or multiple data centers that can be distributed in different regions of the globe [30]. By placing data centers in different cities or countries, cloud providers take advantage of low energy prices, renewable energy sources, reduced operational costs (e.g. staff), local regulations, and appropriate climate to improve cooling. Furthermore, placing a data center near the end-users reduces the delay to access services, which improves users' response time. However, there are applications with strict latency requirements that cannot be processed in the cloud.

2.2 Fog Computing

This section introduces fog computing and is organized as follows. Subsection 2.2.1 defines fog computing. Subsection 2.2.2 discusses similar paradigms and their differences to fog computing. Subsection 2.2.3 reviews the concept of fog nodes. Subsection 2.2.4 discusses the main characteristics of fog applications, and Subsection 2.2.5 presents fog architectures.

2.2.1 Definition and Characteristics

The term "fog computing" was coined by Cisco [16], which defined fog computing as a "highly virtualized platform that provides compute, storage, and networking services between end devices and traditional Cloud Computing Data Centers". In 2015, the OpenFog Consortium, a consortium of technology companies and academic institutions, was founded to define an interoperable architecture for fog computing known as the OpenFog Reference Architecture (OpenFog RA) [79]. The OpenFog RA defined fog computing as "a horizontal, system-level architecture that distributes computing, storage, control and networking functions closer to the users along a cloud-to-thing continuum.". The OpenFog Consortium merged with the Industrial Internet Consortium in 2019.

Fog computing was designed to address the weaknesses of cloud computing, such as the high latency to access cloud data centers and the support for mobile users. Fog and cloud computing are not opposing concepts, but cooperative technologies to support a broad range of applications. Bonomi et al. [16] summarised the necessary characteristics

a fog computing infrastructure must have, detailed in the next paragraphs: low latency, geographical distribution, a large number of nodes, mobility, predominant wireless access, strong use of streaming and real-time applications, and heterogeneity.

The main reason for using fog computing is low latency. Fog computing was proposed envisaging a plethora of applications that have strict latency requirements and cannot rely on centralized data centers. To reduce the latency, compute, storage, and networking resources should be close to end-users, avoiding transmission of data to the core network. Different from the centralized cloud, fog computing is usually deployed in a geographically distributed manner to minimize the response time to end-users. The response time required can vary from a few milliseconds to some seconds depending on the application [4]. There should be fog resources in various locations to support users in different regions.

Fog computing was conceived to support mobile users. Mobility is a challenge in the design of fog computing infrastructures, given that fog resources are typically fixed. Therefore, a large number of distributed nodes is essential to cope with end-users visiting different regions over time. Moreover, the communication between end-users and the fog is predominantly wireless, usually with WiFi or cellular networks, facilitating the connection to multiple fog nodes.

Bonomi et al. [16] assumed a strong use of streaming and real time applications in the fog, typical applications with low-latency requirements. Nonetheless, other applications can benefit from the fog, such as health monitoring, surveillance, content caching, and vehicle navigation [4].

Fog nodes can be heterogeneous, i.e, they can have different resource configurations. Fog nodes should also be interoperable, able to seamlessly connect to users and the cloud. To make it feasible, it is recommended that fog computing should operate using open hardware and software standards to allow the coexistence of different vendors' solutions [79]. Finally, fog computing deployments are typically hierarchical, with different levels of fog and cloud nodes. In complex scenarios, for example, there can be a layer of fog nodes directly connecting to end-users, another layer for interconnection of fog nodes, and cloud layers with public and private clouds. Layers can have different roles, with, for example, a fog node in charge of only one application, and more generalist fog nodes supporting a variety of applications.

2.2.2 Related Concepts

Although "fog computing" was coined in the last decade, the processing of workloads at the network edge was investigated before. This subsection discusses concepts related to fog computing and their main differences to fog. These concepts are cloud computing, mobile computing (MC), mobile cloud computing (MCC), mobile ad hoc cloud computing (MACC), edge computing (EC), multi-access edge computing (MEC), cloudlet computing, and mist computing. A complete review of these concepts is given in [104].

The first technology related to fog computing is cloud computing, already discussed in Section 2.1. The hardware used in the fog can be different from that used in the cloud since fog computing may favor smaller devices to facilitate deployment. Security and privacy mechanisms for fog computing can be quite different from those for clouds due

to the different geo-distribution, opportunities of attacks, and technologies involved. The cloud is usually connected to the Internet and constantly available, whereas fog nodes are not necessarily connected to the Internet or operational all the time. Low latency is the main advantage of fog computing compared to the cloud, but not the only one. For example, fog nodes can be deployed in remote areas or directly used to gather data from sensors due to the physical proximity, which cannot be achieved using only the cloud.

MC refers to the computation performed in mobile devices such as laptops, tablets, or smartphones. MC is usually accomplished using wireless protocols for transmitting data, such as Bluetooth, WiFi, and ZigBee. Fog and cloud computing usually assume dedicated devices for processing, different from those used in MC. Nonetheless, some authors [91, 92] considered MC resources part of the fog infrastructures. This thesis assumes that fog computing is made available using dedicated hardware, and MC is evaluated as an alternative to the fog in Chapter 5.

MCC is defined as an infrastructure in which data storage and processing are not performed by the mobile device. According to NIST, MCC exploits the synergy between mobile devices, Internet of Things (IoT) devices, and cloud computing [76]. Fog and cloud computing enable MCC by offering possibilities of offloading workload from end-user devices. This thesis considers MCC enabled by fog computing.

MACC is a solution to interconnect mobile devices when no external infrastructure is available. In MACC, devices form a temporary network with dynamic topology and can use their capabilities to offer networking, storage, and computing services [104]. MACC is different from fog computing since the former is more decentralized and not organized into layers.

Edge computing is the most similar concept compared to fog computing, and, consequently, some authors use these terms interchangeably. Edge computing is a way to implement MCC in which devices do not belong to end-users, but they are dedicated devices at a few hops away from them. The authors of [89] defined "edge" as "any computing and network resources along the path between data sources and cloud data center", and edge computing consists of enabling computation at such edge instead of the end devices or the cloud. Edge computing is a key enabler of fog computing, with fog being a broad concept that also defines the infrastructure, the layers, and the interplay with the cloud.

Multi-access edge computing, formerly known as mobile edge computing, employs edge computing for cellular networks. The European Telecommunications Standards Institute (ETSI) is in charge of defining the MEC architecture [42], which brings computing resources close to the access network using virtualization. Some authors [98] assumed that MEC employs small virtualized data centers deployed at base stations, accessed via various network interfaces such as WiFi and 5G. MEC deployments are expected to employ devices with network function virtualization (NFV) capabilities [56].

Cloudlets were introduced in 2009 [87] and are defined as small data centers typically at a one-hop distance from end-users, virtualized, and continuously connected to the Internet. Cloudlets are not necessarily integrated into a cellular network and focus on reducing the latency and supporting mobile users. Cloudlets are a direct implementation of cloud data centers at the edge. Fog computing is different from cloudlet in various manners. For example, virtualization is not mandatory for fog nodes. Moreover, fog

Table 2.1: Comparison of fog computing related technologies.

Technology	Users	Devices	Architecture	Latency	Geographical distribution	Standardized
Fog computing	General	Different devices	Distributed and hierarchical	Low	Yes	Yes
Cloud computing	General	Data centers	Centralized and hierarchical	High	Possible	Yes
MC	Mobile	Mobile devices	Distributed	Medium	No	Yes
MCC	Mobile	Edge devices or data centers	Centralized or distributed	High	No	No
MACC	Mobile	Mobile	Distributed	Medium	No	No
EC	General	Edge devices	Distributed	Low	Yes	Yes
MEC	Mobile	Small data centers	Distributed and hierarchical	Low	Yes	Yes
Cloudlet	Mobile	Small data centers	Distributed	Low	Yes	No
Mist computing	General	IoT devices	Distributed	Medium	Yes	No

computing does not require the employment of small data centers; in some deployments, network devices with processing capabilities can be fog nodes.

The final paradigm discussed in this subsection is mist computing. This paradigm assumes that IoT devices provide computing, networking, and storage capabilities. Different from MACC, devices in mist computing may not employ ad-hoc links and do not need to be mobile. Mist computing can be part of fog computing, but fog is a more complex architecture.

Table 2.1 summarizes the characteristics of the technologies reviewed in this subsection, classifying them according to the type of user, the devices used, architecture, latency, whether computing nodes are geographically distributed, and if the technology is standardized. Several concepts were proposed to allow processing at or near the edge to complement the cloud. Most of these concepts aim at reducing the latency to users by performing processing, networking, and storage somewhere close to the end-user devices. The presented concepts share similarities and, in some cases, it is hard to distinguish between them. For example, some authors do not distinguish between fog and edge computing, or between fog nodes and cloudlets. To standardize the text of this thesis, the term fog computing is used. One of the reasons for this decision is the organization in layers, a characteristic of fog computing. For example, Parts II and IV of the thesis considered fog and cloud layers, and Part III assumed two fog layers (one fixed and terrestrial, and another mobile and aerial). Moreover, we did not assume processing made only by edge data centers or virtualized cloudlets, but also by UAVs. Solutions for the fog node location in this thesis can be directly used or easily adapted for cloudlets, MEC, and EC. Moreover, the proposal in this thesis can serve as a basis for solutions that integrate MC, MACC, or mist computing with fog computing.

2.2.3 Fog nodes

In cloud computing, data centers host processing, networking, and storage devices; fog nodes are the equivalent entities for data centers in fog computing. Cloud data centers follow a standardization [1]. On the other hand, fog nodes have very different definitions, ranging from user devices to micro data centers. This subsection discusses existent defini-

tions of fog nodes and states the one adopted in this thesis. A comprehensive discussion on the fog nodes definition is given in the paper [70].

Cisco stated that fog nodes are heterogeneous and deployed in various environments [16]. The OpenFog architecture proposed a wide definition that all computing, networking, storage, and acceleration devices are fog nodes [79]. Marín-Tordera et al. [70] reviewed various papers on fog computing and proposed a broad yet concise definition of fog nodes: "Fog nodes are distributed fog computing entities enabling the deployment of fog services, and formed by at least one or more physical devices with processing and sensing capabilities (e.g., computer, mobile phone, smart edge device, car, temperature sensors, etc.) All physical devices of a fog node are connected by different network technologies (wired and wireless) and aggregated and abstracted to be viewed as one single logical entity, that is the fog node, able to seamlessly execute distributed services, as it were on a single device." [70].

Fog nodes are typically assumed to be small clouds, with devices acting as either producers/consumers of data or computational-rich devices for processing. In [103], the fog node is a single server installed on a bus to augment the capacity of roadside units in its trajectory, while the work in [102] considered the opposite, i.e., the fog nodes are structures at the roadside units. In 5G deployments, fog nodes are typically facilities empowered with processing devices connected to the base stations [98]. Several authors considered fog nodes as small data centers [7, 51, 14, 84]. Some approaches also considered fog nodes as other devices, such as IoT devices [68], and UAVs [35, 74]. Thus, fog nodes can be single devices or facilities to host processing devices, they can be fixed or mobile, powerful or resource-constrained.

The work in this thesis adopts the fog node definition given by the authors of [70] and employs two types of fog nodes, fixed and mobile. Fixed fog nodes are facilities equipped with dedicated servers distributed over a geographical region. They have continuous access to energy supplies and can run 24 hours a day. In terms of connectivity, fog nodes connect to users using either WiFi or cellular interfaces, and they are connected to the cloud via the Internet using wired interfaces, such as Ethernet and optical networks. This thesis considered fixed fog nodes directly connected to base stations in a cellular network; nonetheless, the solutions can be applied to other scenarios such as fog nodes installed on access points. Fixed fog nodes in this work are heterogeneous since they host a variable number of processing servers. Solutions in this thesis do not consider particular aspects of virtualization; they can be applied to both virtualized and non-virtualized fog nodes.

Part III of this thesis introduces the mobile fog nodes mounted on UAVs. In this case, the UAV hosts an onboard computer in charge of processing users' workload. These fog nodes are mobile and can fly to different regions to meet the needs of mobile users. Different from the fixed nodes, the operation time of mobile fog nodes is limited to the UAV battery. UAV fog nodes employ only wireless interfaces and process data in the same manner a fixed node does. In this thesis, UAVs communicate with base stations on the ground to receive the data to be processed. Section 2.5 provides more details about UAVs.

2.2.4 Applications

Many applications can benefit from fog computing. Ahmed et al. [4] studied 30 of these applications and the requirements that should be guaranteed by a fog platform. This subsection discusses applications that are typically processed by the fog, their characteristics, and the assumptions made in this thesis.

The work in [4] lists seven reasons for using fog computing, discussed in the following: latency reduction, bandwidth optimization, computational offloading, privacy and security, service management, monitoring edge devices, energy efficiency, cost reduction, and caching. The main reason to use fog computing is to reduce latency in comparison to the cloud. Applications can be partially deployed on the fog to reduce latency, leaving components with non-critical delays hosted in the cloud. Fog computing can also be employed for pre-processing of data, reducing the traffic sent to the cloud, especially in monitoring applications that generate heavy streams of data, such as video surveillance [79]. Computational offloading is another use case for fog, especially for supporting resource-constrained devices that cannot perform heavy computation. For users who do not want to send sensitive data to public clouds, private fogs can guarantee privacy and security, for example in healthcare applications.

Fog computing can be used to coordinate underlying devices, such as IoT sensors and actuators. Monitoring is an important application of fog computing since the short distance between fog nodes and devices allows direct communication between them. Fog nodes that do not have continuous access to the Internet can process information gathered from sensors to further upload data to the cloud. Quick decisions can be made by fog nodes such as triggering an alarm in case of fire detection or calling emergency services if critical information is sent by health monitoring devices. Fog nodes can process workload sent by battery-constrained devices, allowing a long operation of IoT devices. In terms of costs, paying for deploying a fog node can reduce the cost of renting resources in the cloud and; in some cases, the owners can rent their fog node when resources are idle. Finally, using fog nodes for caching commonly accessed content is a way to reduce the traffic to the Internet as well as reduce delay to the users. By using fog nodes for caching, cached content can be tailored to the specific content consumed by users in the geographical region where they are located [100].

The access network varies according to the application, and is dominated by wireless technologies. A wide range of protocols can be used, ranging from short-distance protocols, like Bluetooth and Zigbee, to 5G and LoRaWAN for long-distance communications. Most fog applications in the literature used WiFi and cellular interfaces [35, 74, 14, 51, 68, 92, 91, 36, 71, 98, 103]. The decision of the network interface depends on the end-user devices as well as the available bandwidth and amount of data to be transmitted. Tolerable delay values vary as a function of the application: a few milliseconds for real-time applications, video broadcasting, and surveillance; dozens of milliseconds for some games and caching; some seconds for image processing, patient monitoring, and telemedicine. Applications can offload from a few kilobytes to some terabytes. The choice of the access network, frequency of data transmission, and network interface, depends on the requirements of each fog scenario.

In this thesis, applications in the fog are characterized by their processing and latency requirements. In all formulations of the fog node location problem, fixed fog nodes have a variable number of servers to deal with the variable processing requirements. In the work in Chapters 4–6, fog applications have strict latency requirements and cloud applications have flexible latency requirements; specific values of delays are not considered. In Chapters 5–8, the number of bytes to be transmitted to the fog or the minimum data rate was part of the formulation. The work in this thesis considered Wifi and cellular network interfaces. Caching was not considered, but the solutions can be adapted to deal with such an application. This thesis does not consider the privacy and security aspects of fog computing applications.

2.2.5 Architectures

A fog computing architecture comprises the definition of fog nodes, protocols, location of fog nodes, hardware, layers, among other aspects. The main effort in this direction has been the one by the Industrial Internet Consortium, which is in charge of defining an open and interoperable architecture for fog computing, known as OpenFog Reference Architecture (OpenFog RA) [79], that organizes fog nodes in hierarchical layers. The OpenFog RA aims at defining a platform driven by eight principles: security; scalability; openness; autonomy; programmability; reliability, availability, and serviceability; agility; and hierarchy. Fog deployments based on the OpenFog RA should be based on hierarchical layers of interoperable fog and cloud nodes with well-defined interfaces for hardware and software.

Most published work in the literature assumes a common three-tier architecture with devices, fog, and cloud layers [35, 74, 51, 68, 92, 91, 36, 71, 98, 103]. Some papers proposed novel fog architectures [71, 98, 59]. The paper [71] proposed the Fog-to-cloud architecture, with fog and cloud nodes organized in layers, and an administrative component in charge of finding the best nodes for end-users to meet the quality of service (QoS) requirements. The authors presented a medical emergency use case to illustrate how that coordinated management of resources can speed up the processing of applications. The TelcoFog architecture [98] also considers fog layers, but is tailored to the scenario of cellular networks and recommends that fog nodes powered with computational and storage resources should always be deployed next to cellular base stations. The proposal in [59] is a user-participatory architecture in which fog nodes are installed and owned by end-users and typically located at their homes. If the provider infrastructure is not sufficient to deal with current demands, user can lease their fog nodes to make the infrastructure scalable.

Proposals of fog architectures [71, 79, 98, 59] recommend how different components interconnect, but they do not specify the physical location of the fog nodes. In this thesis, we consider a typical architecture with user devices, fog, and cloud layers. Solutions proposed can be applied in different fog architectures since the fog node location problem is a common problem independent of the architecture. Although we consider parameters that are specific to one architecture, such as energy consumption in WiFi or channel model in cellular networks, the work in this thesis can be adapted to different fog architectures by changing the input to the formulation.

2.3 Facility Location Problem

The facility location is a traditional problem in operational research whose studies trace back to Evangelista Torricelli and Bonaventura Francesco Cavalieri in the seventeenth century [18]. Its input is a set of candidate locations to deploy a facility, and the output is the subset of locations chosen for deployment. This problem is an optimization problem aiming at minimizing cost while satisfying some demands given by a set of constraints.

The facility location problem has many variations [36]. The simplest approach to this problem is the single facility location problem, in which a single facility must be deployed in one of the candidate locations. Many real-world problems can be formulated as a single facility model, such as deciding on the location of a new hospital, fire station, or library in a metropolitan area [36]. Conversely, the multi-facility location problem consists of locating at least two facilities over a set of candidate locations. In some cases, this problem can be optimally solved for N facilities by N executions of a single facility solution, but this is not true in some cases. For example, when part of the customers can be served by multiple facilities. Facility location problems can be single or multi-objective, and optimizing an objective can degrade the value of another objective.

Facility location problems can usually be written as linear programming formulations. Location problems are typically NP-hard, and inputs representing real-world instances are usually large-scale, which requires approximate solutions and polynomial heuristic algorithms. The algorithms implemented by solvers can help obtain solutions for small to medium-scale instances of the problem.

In this thesis, facility location formulations are modeled as multi-facility formulations. Fog nodes are the facilities, and application requests are the customers. The number of fog nodes is variable, and application requests vary in time and space. Mathematical formulations are multi-criteria, and the objectives are typically the maximization of users served and cost reduction. Results in this thesis were obtained using the solver Gurobi, but its employment was not possible for some large inputs, motivating the proposal of heuristic solutions in many chapters. Related work to facility location in computer networks is presented in Chapter 3.

2.4 Multi-objective Optimization

Solutions to formulations with multiple criteria lead to a set of possible solutions known as a Pareto Front. However, a single solution must be selected as the final solution to a problem, which is made utilizing a technique such as scalarization or goal programming [21]. The choice of the technique depends on the problem, especially when the objectives have different priorities. Under scalarization, for example, multiple objectives are combined into a single expression by the assignment of weights to each objective function; thus favoring the evaluation of different trade-offs. In the ε-constraints method, one objective is selected as the main objective function, and the remaining ones become constraints limited to given target values. Goal programming receives as input goal values for each objective and attempts to jointly optimize them. Multi-level programming is used when objectives can be hierarchically ordered. In this case, a sequential optimization of the

objectives is established: the main objective function is optimized first, and the second one only considers elements of the Pareto front that optimize the first objective. The same procedure is used for each subsequent objective until a final solution is obtained after the evaluation of all the objectives.

In this thesis, multi-level programming is adopted in the proposed solutions due to the hierarchy of the objectives. In the fog node location problem, serving end-users is crucial: it is no use saving in the deployment cost if users are not served by the infrastructure. After optimizing the service of end-users, other criteria can be considered. Multi-level programming can be adapted to allow degradation in one of the objective functions. In this case, a specified level of degradation of the value of the objective function is tolerated if it can lead to a improvement of the values of the other objective functions. Each chapter explains how the multi-level programming was applied (the number of objectives, the hierarchy between objectives, and if degradation was allowed).

2.5 Unmanned Aerial Vehicles

Part III of this thesis proposes solutions to the location of fog nodes mounted on Unmanned Aerial Vehicles (UAVs). This section first introduces UAVs and how they work in Subsection 2.5.1, then it reviews the operation of rotary-wing and fixed-wing UAVs in Subsections 2.5.2. Finally, Subsections 2.5.3 and 2.5.4 present the energy consumption and wireless channel models for UAVs, respectively.

2.5.1 Overview and UAV types

UAVs are aircraft or robots that can fly without an onboard pilot, controlled remotely or autonomously, in any direction [85]. Most literature defined the acronym UAV as Unmanned Aerial Vehicle, but some authors prefer the term "Unoccupied Aerial Vehicles" or "Uncrewed Aerial Vehicles" [55]. In general, the words UAV and drone are interchangeable; this thesis will mostly use the acronym UAV. For a long time, UAVs were only used in military applications; for example, the possible origin of UAVs was in the nineteenth century, when Venice, Italy, was attacked by unmanned hot-air balloons with bombs [32]. It was only in the mid-2000s that technological advances allowed a broader deployment of UAVs in other domains, such as disaster relief, search and rescue, and inspection [85]. A major interest in UAVs took place after Amazon announced its intentions to use drones for delivery in 2013. Since then, technology has allowed equipping UAVs with a variety of sensors, and networking and processing capabilities, which have enabled the use of UAVs in many other domains, such as recreation, aerial photography, base stations, and fog computing.

UAVs can be classified according to the altitude they fly or the type of drone wing [85]. In terms of altitude, UAVs can be either HAPs or LAPs. HAPs fly at high altitudes, typically above 17 km from the ground, while LAPs fly a few meters to hundreds of meters to the ground. HAPs typically have long endurance, being used for long-term missions, such as providing connectivity to large areas. They are heavier than LAPs and can carry more payloads. On the other hand, LAPs are light aircraft that provide quick

mobility, but their operation typically lasts from some minutes to a few hours. Therefore, they are more suitable for short operations. LAPs are cheaper than HAPs and can be used by both big companies and end-users, subject to local regulations. In this thesis, UAVs are employed to deal with the demands of end-users for short periods, complementing the terrestrial infrastructure. Therefore, this thesis considered only LAPs.

Concerning the type of wing, a UAV can be classified as fixed-wing, rotary-wing, airship, or balloon. Fixed-wing UAVs resemble modern airplanes, and their main characteristics are continuous motion and the inability to hover. Fixed-wing UAVs can be small LAPs or even big HAPs that require airport runways to take off and land. Rotary-wing UAVs are LAPs that have one or multiple rotor-based wings, similar to helicopters. Their main advantage is the ability to hover and move in any direction, making them able to access limited areas. Airships are mostly HAPs, and they are large quasi-stationary aircraft used for long-term operations. Finally, balloons can be either LAPs or HAPs; they are mostly stationary aircraft, but lighter than airships. The solutions in this thesis employed both rotary-wing and fixed-wing UAVs; a more detailed discussion on the characteristics of these UAVs is made in the next subsection.

All UAVs can be equipped with network capabilities, which enables them to connect to different devices [85]. UAVs can be users of the ground infrastructure, requesting services from base stations (BSs), but they can also play roles of the BS itself, a relay, or a processing node. Flying at high altitudes, UAVs have line-of-sight (LoS) with more users on the ground. However, these long links may not be appropriate to guarantee low latency. A review of papers that employed UAVs as edge elements of the network is given in Section 3.2, and more discussion on the channel model for UAVs is introduced in Subsection 2.5.4.

2.5.2 Rotary-wing and Fixed-wing UAV Operation

Aerial network infrastructures can combine different types of aircraft. Previous work mainly considered fixed-wing and rotary-wing UAVs. This subsection reviews the main advantages and limitations of these types of UAVs, the ones studied in this thesis.

The aerodynamics of fixed-wing and rotary-wing UAVs differ in how they take off, land, and fly. Fixed-wing UAVs are similar to commercial airplanes: the engine generates thrust to move the aircraft forward, and the wings are shaped to generate lift. Fixed-wing UAVs are efficient for longer trajectories since, once in the air, they do not need high power to keep flying. However, they usually require a runway to take off and land, which is a complex ground infrastructure. On the other hand, rotary-wing UAVs are vertical take-off and landing (VTOL) vehicles that generate thrust and lift by rotating rotor blades. Both single-rotor and multi-rotor versions are available. Single-rotor UAVs have a design similar to that of a helicopter; they can endure long trips but have less stability to hover. Multi-rotor UAVs are more suitable for hovering and have better maneuverability, thus facilitating access to limited spaces. However, the greater the number of blades required, the greater is the energy consumption which reduces the operational time. Most of the existing research on UAVs in wireless networks has considered multi-rotor UAVs.

Fixed-wing and rotary-wing UAVs flying at low altitudes are typically powered by bat-

teries and have limited autonomy [39]. The operation with the lowest energy consumption for fixed-wing UAVs [106] is a straight flight maintaining the same height at a constant speed (straight-and-level flight). Lower speeds lead to lower energy consumption, but fixed-wing UAV flights require a certain minimum speed to generate lift. If the UAV needs to cover a small area, a circular trajectory can be adopted. The larger the radius, the lower is the energy consumption. Rotary-wing UAVs also consume more energy at high speeds [105], but they can hover, reducing energy consumption. However, the energy consumed to keep a rotary-wing UAV hovering is usually greater than that to maintain a fixed-wing one flying in a circular trajectory with a large radius.

Both types of UAVs can carry onboard networking, processing, and storage devices. The network between UAVs and devices on the ground is largely composed of line-of-sight links due to the altitude at which the UAVs fly [35, 106, 105]. The length of these links affects the channel quality with longer links leading to greater path loss [65]. If a UAV needs to provide a limited area with wireless coverage or processing, a rotary-wing UAV can be more efficient, since it can hover and maintain a stable wireless channel. Fixed-wing UAVs, however, will have a variable wireless channel. If a user on the ground sends a request to the UAV, they may need to wait for the fixed-wing UAV to come closer, even though the request may be blocked if the networking requirements are not met. Fixed-wing UAVs can also be affected by the Doppler shift, although this phenomenon can usually be compensated in flights at low altitudes and speeds [106].

Fog nodes can be mounted on both fixed-wing and rotary-wing UAVs. Fixed-wing UAVs are usually more energy-efficient than rotary-wing ones, but the latter can provide wireless links with a higher quality of transmission. Rotary-wing UAVs have been broadly considered as part of the infrastructure in previous work [111, 99, 105]. Nonetheless, with adequate planning, fixed-wing UAVs can efficiently serve as fog nodes. Fog providers can benefit from their low energy consumption to keep them operational longer, serving more users. This thesis presents solutions that consider both rotary-wing and fixed-wing UAVs, attempting to take advantage of their best characteristics.

2.5.3 Energy Consumption

The limited energy of UAVs is an important issue since the battery has a very limited capacity and flying is an energy-consuming operation, particularly for LAPs [107]. Realistically modeling the energy consumption is crucial in the evaluation of a solution since it determines the maximum time a UAV remains operational. If an evaluation considers a UAV could stay powered on for longer than it should, an implementation with real hardware can make wrong decisions, preventing UAV from serving end-users or even causing disruption of service in the middle of flights. This section discusses known energy consumption models from the literature and also discusses the employment of tethered UAVs.

LAPs are usually powered by onboard batteries that, once depleted, need to be recharged to allow further operation. The main source of energy consumption in UAVs is the propulsion to make them fly; other sources of consumption are onboard processing, wireless communications, and the use of various sensors (photography, temperature,

among others). The component related to propulsion depends on various factors, such as UAV type, altitude, air density, payload, and wind. Although it is almost impossible to model all possible sources of energy consumption, there are good models of energy consumption for UAVs, either based on real measurements [3] or derived from an analysis of aerodynamics [106, 105]. This thesis employed three different energy consumption models, two for rotary-wing UAVs, and one for fixed-wing UAVs. These models are detailed in the following.

Abeywickrama et. al [3] measured the energy consumption of an Intel AeroReady to Fly Drone, a rotary-wing UAV, and derived energy consumption models for various operations. The energy consumption in hovering $P(h)$ (watts) depended on the height h (in meters) and was calculated as $P(h) = 13.0397h + 196.8490$. The power for flying horizontally was calculated as 245.2815 W. The energy spent in flying vertically up ($E^U(h)$) and down ($E^D(h)$) are given in Joules and depend on the vertical distance traveled h; such consumption values were calculated as $E^U(h) = -16.9396H^2 + 216.6944H - 157.9473$ and $E^D(h) = 4.6817H^2 - 11.9708H + 135.3118$. When the drone was on the ground, the average power was 8.2637 W. The main advantages of this model [3] are the simple use and ability to model different altitudes. However, it has some disadvantages: it modeled a specific UAV, and it represents the conditions (climate, geography, air density, among others) of the place and the date when the measurements were taken. Despite the disadvantages, this model can be really useful to evaluate rotary-wing UAVs, and it was employed in the evaluation of Chapters 6 and 7.

Zeng, Xu, and Zhang [105] proposed another model for rotary-wing UAVs. Different from [3], they evaluated the aerodynamics of the aircraft, analyzing which forces are applied to the UAV, and calculating the power required to maintain the flight. The power $P(V)$ in a straight-and-level flight is given by Equation (2.1) as a function of V, the UAV speed, and is divided into three parts. The first one is the blade profile, P_0 is a constant representing the blade profile power, and U_{tip} is the tip speed of the rotor blade. The second one is the induced power, where P_{ind} is a constant representing induced power in hovering status, and v_0 is the mean rotor induced velocity. The third part is the parasite power, where d_0 is the fuselage drag ratio, ρ is the air density, s is the rotor solidity, and A is the rotor disc area. Constants P_0 and P_{ind} are calculated using A, ρ, and the UAV weight. If the UAV is hovering ($V = 0$), the power can be simplified to $P_0 + P_{\text{ind}}$. The main advantage of this model is the ability to model any UAV under arbitrary conditions. However, obtaining the values of all constants is not straightforward. This model was employed in the evaluation of Chapter 8.

$$P(V) = P_0 \left(1 + \frac{3V^2}{U_{\text{tip}}^2}\right) + P_{\text{ind}} \left(\sqrt{1 + \frac{V^4}{4v_0^4}} - \frac{V^2}{2v_0^2}\right)^{1/2} + \frac{1}{2} d_0 \rho s A V^3 \tag{2.1}$$

Zeng and Zhang [106] also derived an energy consumption model for fixed-wing UAVs in a similar manner they obtained the one for rotary-wing UAVs [105]. The energy for a straight-and-level flight is given by Equation (2.2), where V is the UAV speed, and constants c_1 and c_2 are related to the parasite and induced power, respectively. c_1 and c_2 are calculated using many parameters, such as the zero-lift drag coefficient, wing area, aspect

ratio of the wing, air density, and UAV weight. If the fixed-wing UAV is performing a circular trajectory with radius r, the power can be calculated with Equation (2.3), where g is the gravitational acceleration. This model shares the same advantages and disadvantages as the previous one: it models arbitrary UAVs and conditions, but calculating c_1 and c_2 is not simple. This model was employed in the evaluation of Chapter 8.

$$P(V) = c_1 V^3 + \frac{c_2}{V} \tag{2.2}$$

$$P(V, r) = \left(c_1 + \frac{c_2}{g^2 r^2} \right) V^3 + \frac{c_2}{V} \tag{2.3}$$

One solution to overcome the autonomy limitations of battery-powered UAVs is the employment of tethered UAVs [60]. In such deployments, the UAVs are connected to ground stations (GSs) by a tether that provides both energy supply and connectivity. GSs can be installed either on the ground or rooftops; in urban scenarios, rooftops are preferred to avoid the obstructions imposed by tall buildings. Both fixed-wing and rotary-wing UAVs can be tethered. However, the tether imposes significant limitations. For example, it limits the distance between the UAV and the GS to the length of the tether, considerably reducing mobility, and more human intervention is required. Nonetheless, tethered UAVs can be quite advantageous if UAVs are to stay in the same spot for long periods. This thesis presents evaluations of both tethered and battery-powered UAVs.

2.5.4 Channel Model

UAV communications are predominantly wireless due to aircraft mobility. Even tethered UAVs [60] connect to distant users using their wireless interface instead of the tether. A variety of technologies can be employed, such as Bluetooth, Wifi, and Long Term Evolution (LTE); the choice of protocol depends on the application. For example, in a farm aerial photograph mission with a nearby operator, WiFi is a good alternative. If a drone flies autonomously and needs to report its position periodically, cellular technology can be the most efficient protocol. This thesis considered LTE for the communication between UAVs and BSs. This subsection discusses some of the challenges of wireless communications for UAVs, followed by an overview of channel models, specifically presenting the one adopted in the evaluation in Chapters 7 and 8.

UAVs have some advantages over ground devices related to wireless communications. The high altitude helps to provide LoS links, which guarantees a higher signal strength than, for example, refracted or reflected signals. UAVs can quickly change their position to improve the links with terrestrial nodes. Moreover, UAVs can optimize their 3D position, while ground nodes are limited to a two-dimensional space. On the other hand, UAVs face problems that ground users do not have. Cellular antennas are prepared to serve terrestrial devices and, therefore, they are usually down-tilted [85]. Therefore, UAVs are usually served by the side lobes of antennas and, consequently, they may have a better connection to a distant BS than the one directly under it. The flying altitude can be another issue, since long wireless links suffer more from path loss, which incurs low data rates, preventing using a UAV as a fog node.

The design of a fog infrastructure must address the characteristics of wireless channels and, for that purpose, a channel model is adopted. The channel model estimates the strength of wireless signals, modeling diverse phenomena with equations. Channel models can be theoretical, based on the analysis of physics, or empirical, based on real measurements. The main phenomenon that affects the quality of wireless links is the free-space path loss, which models the signal attenuation as a function of the distance between transmitter and receiver. Other phenomena are fading, in which there is a variation of the attenuation of a signal, and the Doppler effect [58]. The work developed in this thesis with UAVs does not aim at proposing new channel models nor deeply evaluating the physical layer, but it simulates the aerial channel to evaluate the effectiveness of the proposed solutions. Therefore, this thesis adopts the channel model proposed by the 3rd Generation Partnership Project (3GPP), described in the following. The channel model was modeled by the path loss, as in previous approaches [52, 109, 106, 105].

The 3GPP conducted a study on the LTE support for aerial vehicles using real UAVs and measured the wireless channel under different conditions. This study is described in the Technical Report 36.777 [2]. LoS probability, path loss, and fast fading models were derived for LTE communications between UAVs and antennas for both urban and rural scenarios. This thesis models the wireless channel using the path loss model proposed by 3GPP. The path loss depends on three factors: line-of-sight (LoS), the height of the UAV, and type of BS (urban micro, urban macro, or rural macro). This thesis considered users communicating with the nearest cellular BS, and UAVs communicating with a terrestrial BS to receive the workload for processing. Urban macro base stations are considered to have 25 m high antennas, as in the 3GPP study. The LoS probability (P_{LOS}) is given by Equation (2.6) and depends on the values of d_1 and $p1$, calculated by Equations (2.4) and (2.5), respectively, where h is the height of the UAV in meters in relation to the ground, and d_{2D} is the horizontal distance in meters between the UAV and the BS. The path loss PL in dB is calculated using Equation 2.7, where d_{3D} is the distance in meters between the UAV and the antenna, and f_c is the channel bandwidth in Gigahertz.

$$d_1 = \max\left(460\log_{10}(h) - 700, 18\right) \tag{2.4}$$

$$p_1 = 4300\log_{10}(h) - 3800 \tag{2.5}$$

$$P_{LOS} = \begin{cases} 1, & d_{2D} \leq d_1 \\ \frac{d_1}{d_{2D}} + \exp\left(\frac{-d_{2D}}{p_1}\right)\left(1 - \frac{d_1}{d_{2D}}\right), & d_{2D} > d_1 \end{cases} \tag{2.6}$$

$$PL = \begin{cases} 28.0 + 22\log_{10}(d_{3D}) + 20\log_{10}(f_c), & \text{if LoS} \\ -17.5 + (46 - 7\log_{10}(h))\log_{10}(d_{3D}) & \\ +20\log_{10}\left(\frac{40\pi f_c}{3}\right), & \text{otherwise} \end{cases} \tag{2.7}$$

Chapter 3

Related Work

This chapter reviews previous work related to this thesis and is organized as follows. Section 3.1 reviews solutions for the facility location problem in computer networks, Section 3.2 reviews papers that employed UAVs at the network edge, Section 3.3 reviews previous work on energy efficiency and offloading of workload by end-user devices, and Section 3.4 reviews papers on resource allocation in fog computing. This chapter also reviews data sets (Section 3.5) and software tools (Section 3.6) used in this thesis.

3.1 Facility Location Problem in Computer Networks

The facility location is a common problem in computer networks in which facilities can be single devices or even large data centers. This subsection reviews previous work on the location of cloud data centers [62, 63, 27], cloudlets [54, 34], WiFi hotspots [77], and edge servers [112, 67]. At the end, this subsection compares the reviewed work to the proposals in this thesis.

Some papers have addressed the cloud data center location [62, 63, 27]. Larumbe and Sansò presented solutions [62, 63] to select the location of a data center in a backbone network. The work in [62] and [63] employed a MILP formulation and a scalable tabu search algorithm, respectively, to decide on the location of data centers to minimize delay, energy consumption, costs, and the emission of greenhouse gases. On the other hand, the solution proposed by Covas, Silva and Dias [27] considered a multiple criteria decision that quantified the social, economic, and environmental impact of the candidate location for the data center. Their proposal employed the method ELECTRIC TRI to classify all criteria; the solution was validated with a local provider. Solutions for cloud data center location cannot be directly applied for the fog node location since cloud data centers are centralized, while fog nodes are distributed; thus, the decision must consider other aspects and criteria.

The placement of cloudlets has been explored in previous papers [54, 34]. Jia et al. [54] determined the location of cloudlets to reduce the delay of user tasks. Fan and Ansari [34] included the cloudlet cost in the decision. Using an optimization model, they showed that their solution can reduce deployment cost as long as additional delays are acceptable. These papers [54, 34] ignored the diversity of latency requirements of

applications. Oliveira and Viana [77] presented a solution for the WiFi hotspot location that attempts to maximize the offloaded traffic by mobile users limited by the number of devices available for this deployment. The solution first identifies points of interest, i.e., locations commonly visited by users, to serve as candidate locations. It employed a graph that connects mobile users and points of interest. Results showed that the number of hotspots could be reduced by the use of the solution.

Zhao et al. [112] proposed a solution for selecting the position of edge servers to provide real-time data processing for IoT. They proposed a metric that quantifies the performance gain of candidate locations in terms of data collection and message dissemination. Their solution assigns edge servers to locations according to the proposed metric until all users have been covered by an edge server. Results showed that the algorithm reduced the number of servers deployed and allowed high data rates between servers and end devices. Lähderanta et al. [67] surveyed various edge server location solutions to propose an algorithm customizable for different networks. Their solution considered different hierarchical fog layers, and the evaluation showed that the number of layers depends on the time and space variation of workload.

Table 3.1 compares the papers reviewed in this subsection and compares them to the work developed in this thesis. The table classifies proposals according to the type of facility, the type of solution, criteria adopted, whether time variable workload was considered, and whether a mobile facility was considered. Chapters 4–8 present different solutions to the fog node location problem. These solutions differ from previous work in different ways. Firstly, they were designed specifically for fog computing and, consequently, different aspects were modeled. Secondly, solutions in this thesis decide on the location of fog nodes considering variable demands in time and space. Third, not only the location of nodes was decided, but also their capacity. Moreover, each solution to the fog node location presents different features compared to previous work. The work in Chapters 4–5 consider different classes of services in terms of latency. Chapter 5 considers the energy consumption of end-user devices in the location decision. The work in Chapters 6–8 introduces the use of mobile fog nodes mounted on UAVs in the location problem; in Chapter 6 the location of both fixed nodes and UAVs is studied, and Chapters 7–8 consider only UAVs. Therefore, this thesis expands the state-of-the-art of the facility location in computer networks in different manners.

3.2 UAVs as Networking Elements

As networking elements, UAVs can play different roles, such as base stations (BSs), processing nodes, or even users of the infrastructure. As BSs, UAVs provide connectivity to places without terrestrial infrastructure, such as remote areas or those affected by disasters. As processing nodes, UAVs offer computing capabilities with very low latency. As users, they access the terrestrial infrastructure while performing various tasks.

In the work in [20, 48, 66, 106, 105], UAVs are users of the infrastructure. The authors of [20] optimized the trajectory of UAVs to take advantage of ground BS positions and minimize the completion time of offloaded tasks, constrained by the UAV departure and

Table 3.1: Comparison of previous work related to the facility location problem.

Proposal	**Type of facility**	**Solution**	**Criteria**	**Time-variable workload**	**Mobile facility**
[62]	Data center	MILP	Delay, deployment costs, and energy consumption	No	No
[63]	Data center	Tabu search	Delay, CO_2 emissions, and deployment costs	No	No
[27]	Data center	ELECTRE TRI	Social, economic, and environmental aspects	No	No
[54]	Cloudlet	Heuristic	Response time	No	No
[34]	Cloudlet	MILP	Deployment cost and delay	No	No
[77]	Wi-Fi hotspot	Greedy algorithm	Offload ratio	No	No
[112]	Edge server	Heuristic	Deployment cost	No	No
[67]	Edge server	Coordinate descent	Distance to APs	Yes	No
Chapter 4	Fog node	MILP	Workload acceptance, cost, and fog use	Yes	No
Chapter 5	Fog node	MILP and heuristic	Workload acceptance, energy consumption, and fog use	Yes	No
Chapter 6	Fog node	MILP and heuristic	Workload acceptance and cost	Yes	Yes
Chapter 7	Fog node	MILP and heuristic	Workload acceptance, cost, and delay	Yes	Yes
Chapter 8	Fog node	MILP and heuristic	Workload acceptance, cost, and delay	Yes	Yes

arrival positions. In [48], the authors considered UAVs gathering data from IoT sensors, which were then totally or partially processed by the UAV and sent to a terrestrial fog node. The authors proposed an energy-efficient scheduler of processing resources and data transmission that considered the UAV trajectory. In [66], UAVs were in charge of collecting data from sensors, and a distant cloud provided UAVs with processing. A theoretical model of the system was introduced, and an evaluation demonstrated its stability and reliability under different traffic patterns. The authors of [106] optimized the trajectory of a fixed-wing UAV to improve the data transmission to the ground. An algorithm based on sequential optimizations of the position and speed of the UAV provided reasonable data rates and energy efficiency to end-users. The same authors considered rotary-wing UAVs in [105], optimizing the trajectory to reduce energy consumption and the completion time. These proposals [66, 106, 105] did not consider processing capabilities at the UAV.

In the context of cellular networks, previous authors have envisioned the employment of UAVs as aerial base stations [23, 75, 9]. The SkyRAN architecture [23] aimed at providing cellular access to ground users by equipping UAVs with LTE antennas. SkyRAN controls the 3D position of UAVs so that connectivity to end-users is improved. The SkyCore architecture [75] brought routing functions to aerial BSs while reducing latency to end-users. Both of these architectures [23, 75] tested real prototypes, demonstrating the feasibility of using UAV-based BSs. UAVs can also serve as relay nodes between small satellites and mobile users to provide 5G coverage [9].

UAVs equipped with onboard computers can serve as mobile fog nodes, processing tasks of end-users at very low latency [74, 52, 53, 109, 64, 99, 114, 111]. The UAVFog architecture [74] introduced the use of fog nodes mounted on UAVs to support diverse IoT applications, allowing on-demand deployment of fog nodes. Other authors explored the optimization of diverse operations performed by UAV processing workload at the edge [52, 53, 109, 64, 99, 114]. Li et al. [64] focused on reducing the energy spent by the UAV. An

approximate solution optimizes the trajectory and resource allocation, estimating ground user positions to decide on the trajectory. On the other hand, other approaches used a UAV to provide opportunities for offloading, thus reducing the energy spent by end-user devices [52, 53, 109]. The authors of [52] attempted to reduce the energy consumed by end-user devices that offload tasks to UAVs while optimizing the UAV trajectory. Their offload scheme significantly reduced the energy consumed by mobile devices in comparison with scenarios with processing only in the device. The authors of [53] optimized the UAV trajectory and resource allocation in an attempt to reduce the energy spent by end-user devices and UAVs. Zhang et al. [109] considered a scenario with workload offloaded to either an aerial or a terrestrial fog node and proposed a game theory-based solution to achieve a trade-off between latency and energy consumption. The authors of [111] proposed algorithms to optimize the UAV trajectory, task schedule, and resource allocation to reduce the energy consumed by the UAVs while ensuring fairness among end-users.

In [113], UAVs provided both processing and energy supply for end devices. The authors proposed an algorithm to optimize the UAV trajectory, CPU frequencies, user offloading time, and transmission power. Results have shown that UAVs are efficient for extending the operational time of end-user devices. The authors of [35] attempted to reduce the energy consumed by the onboard computer of a fog node UAV, proposing a solution based on reinforcement learning to reduce the number of active CPUs. Results showed that reducing active CPUs caused a slight increase in the latency but allowed a significant reduction in the energy spent in computation.

The scheme proposed in [99] dispatched UAVs to hover at specific areas to process user tasks within a given deadline. The authors demonstrated that the employment of UAVs increased the number of processed tasks when compared with deployments using only ground nodes. Furthermore, the solution led to latency fairness and avoided the underutilization of resources. Zhou et al. [114] considered network supporting virtual reality applications by employing UAVs with processing and caching capabilities and proposed an algorithm to minimize latency by optimizing the 3D location of aerial nodes.

Table 3.2 compares the papers reviewed in this subsection and compares them to the work developed in this thesis. The table classifies proposals according to the research problem studied, the type of solution, criteria adopted, the role performed by the UAV, the type of UAV, and whether the battery was considered. Most papers considered rotary-wing UAVs and, although some of them modeled the energy spent by the UAV, only the work in [109] considered the battery limitation of the UAV. Nonetheless, no previous work considered UAVs as fog nodes and proposed solutions to the fog node location problem. The work in Chapters 6–8 proposes different solutions for the fog node location problem with aerial fog nodes, either considering aerial and ground nodes (Chapter 6) or only UAVs (Chapters 7–8). The battery is considered, limiting the time a UAV remains operational. Moreover, variable requests in time and space are considered so that the infrastructure is adapted to deal with peak demands.

Table 3.2: Comparison of previous work on UAVs.

Proposal	Research problem	Solution	Criteria	UAV role	Type of UAV	UAV Battery
[20]	Trajectory optimization	Heuristic algorithm	Completion time	User	Rotary-wing	No
[48]	Resource and Transmission Scheduling	Linear programming	Energy consumption	User	Fixed-wing	No
[66]	Transmission control	Open Jackson network	Stability and reliability	User	Rotary-wing	No
[106]	Trajectory optimization	Sequential Optimization	Energy efficiency	User	Fixed-wing	No
[105]	Trajectory optimization	Successive Convex Approximation	Energy efficiency	User	Rotary-wing	No
[23]	Architecture	Architecture and testbed	Thropughput	Base station	Rotary-wing	No
[75]	Architecture	Architecture and testbed	Latency	Base station	Rotary-wing	No
[9]	Architecture	Architecture	Transmission	Base station	LAPs and HAPs	No
[74]	Architecture	Architecture	Delay	Edge Server	—	No
[52]	Trajectory optimization	Successive convex approximation	Energy consumption	Edge Server	Rotary-wing and fixed-wing	No
[53]	Trajectory optimization	Heuristic algorithm	Energy consumption	Edge Server	Rotary-wing	No
[109]	Task offloading	Game theory algorithm	Energy consumption and latency	Edge Server	Rotary-wing	Yes
[64]	Trajectory Optimization and Resource Allocation	Dinkelbach and successive convex approximation	Energy consumption	Edge Server	Rotary-wing	No
[111]	Trajectory optimization	Successive Convex Approximation and Coordinate Descent	Energy consumption	Edge Server	Rotary-wing	No
[113]	Trajectory Optimization and Resource Allocation	Successive Convex Approximation	Energy consumption	Edge Server	Rotary-wing	No
[35]	Trajectory Optimization and Resource Allocation	Dinkelbach and successive convex approximation	Energy consumption	Edge Server	Rotary-wing	No
[99]	UAV Dispatch	Heuristic algorithm	Workload acceptance	Edge Server	Rotary-wing	No
[114]	UAV 3D Location	Iterative optimization	Latency	Edge Server	Rotary-wing	No
Chapter 6	Fog node location	MILP and heuristic	Workload acceptance and cost	Edge Server	Rotary-wing	Yes
Chapter 7	Fog node location	MILP and heuristic	Workload acceptance, cost, and delay	Edge Server	Rotary-wing	Yes
Chapter 8	Fog node location	MILP and heuristic	Workload acceptance, cost, and delay	Edge Server	Fixed-wing	Yes

3.3 Reduction of energy consumption via offloading

This section reviews previous solutions that considered the energy efficiency of end-user devices and the offloading of tasks to other devices [96, 38, 101, 108, 51]. Moreover, it compares such solutions to the work developed in Chapter 5.

The work in [96] analyzed the task completion time and the energy consumed by the end-user device involved in the offloading of tasks to neighboring devices using wifi and the offloading of tasks to the cloud using cellular networks. The authors concluded that small tasks should not be offloaded to other mobile devices nor to the cloud because of the energy consumed in transmission, whereas those that require the transmission of a large number of bytes should be offloaded. Fiandrino et. al. [38] analyzed task completion time and energy consumption for an Android application executed on a wearable device. They measured the energy consumption in mobile device processing and offloading involving the transmission of data using Bluetooth and wifi. The results indicated that processing at the end-user device was beneficial when data transmission implied a large energy consumption. These studies [96, 38] highlighted the importance of analyzing the energy consumed by end-user devices with different network interfaces.

Previous work [101, 108] defined the fog as a set of end devices cooperating by using device-to-device (D2D) communications, proposing solutions to achieve energy efficiency when offloading tasks. Yang et al. [101] introduced an algorithm for scheduling tasks among neighboring devices by employing the unused spectrum in D2D communications. Zhang et al. [108] proposed a task scheduling mechanism based on a fairness metric that accounts for the history of energy consumption and device priority. These two papers showed how to improve the energy efficiency of end-user devices; however, they only considered the devices layer. Jalali et al. [51] analyzed the energy consumption spent by network devices when the workload is processed in the cloud and compared it to the consumption required when nearby fog nodes are used. They concluded that processing traffic-intensive applications in the fog instead of in the cloud can reduce the energy spent in large data centers.

The work in Chapter 5 differs from previous work in different ways. It investigates the employment of fog computing to reduce the energy consumption of battery-constrained devices carried by end-users, analyzing the energy spent spent in processing and transmission. It also examines an architecture with three layers, with offloading to either the fog or the cloud. Finally, the work in Chapter 5 aims at achieving energy efficiency with the proper location of fog nodes, which has not been considered in previous work.

3.4 Resource Allocation in Fog Computing

The allocation of resources needs to be defined to make a fog computing infrastructure operational. Such a decision consists on selecting the resources to be used for processing end-user requests. This subsection reviews previous papers [15, 83, 102, 51, 91, 92, 90, 86, 33, 94, 69, 57] on resource allocation in fog computing and compares them to the proposal in Chapter 9.

Resource allocation in fog and cloud computing shares similarities with that of hybrid

clouds. In hybrid clouds, a workload can be either processed in a private local cloud or in a public cloud data center. A private cloud typically offers limited resources but provides low latency and low operational cost whereas a public cloud offers virtually unlimited resources priced on demand with a high latency. Resource allocation in such a scenario aims at achieving a trade-off between cloud monetary costs and processing time [13]. For instance, the authors of [15] studied resource allocation in hybrid clouds considering requests modeled as directed acyclic graph (DAG) workflows in which there is one or more tasks with dependencies established between them. The authors proposed the Hybrid Cloud Optimized Cost scheduling algorithm for allocating resources aiming at minimizing monetary costs and makespan while respecting deadlines. The employment of their solution allowed adjusting some parameters to either favor reduction of costs or execution time. Similar to [15], Pham et al. [83] considered requests modeled by DAGs, but in a scenario with multiple fog and cloud nodes. Assuming that fog nodes can be used free of charge, the authors proposed a scheduling algorithm to balance makespan and the costs of execution in clouds. They showed that an availability of multiple cloud nodes speeds up the execution of tasks, yet increasing costs.

Assuming the feasibility of virtual machine (VM) migration between fog nodes, Yao et al. [102] proposed a heuristic algorithm to decide on the path used for a VM migration between distinct cloudlets to reduce the network cost. They evaluated a scenario of vehicular networks in which fog nodes are located at roadside units to which vehicles connect. They showed the employment of their algorithm optimized the use of links connecting cloudlets, reducing costs for the provider. In [51], the authors analyzed the energy consumption of a blogging application in a system composed of cloud data center and a raspberry mini computer as a fog node. They analyzed the impact of using resources either in the cloud or in the fog on the energy consumption of end-user devices, fog, and cloud nodes. The authors concluded that moving applications from the cloud to the fog is beneficial when they produce large amount of data at the end-user device.

Some papers [91, 92, 90] proposed resource allocation mechanisms for the F2C architecture reviewed in Subsection 2.2.5. In [91], a solution to support the quality of service (QoS) requirements of applications in fog-cloud scenarios was proposed. The solution attempted to reduce latency by favoring the allocation of resources in the fog instead of the cloud, considering an architecture composed of two fog layers and the cloud. The authors showed that employing multiple fog layers reduces the overall latency. The work in [92] employed a resource allocation mechanism based on the knapsack problem to minimize latency and reduce the energy consumption. The work in [90] studied protection in fog computing by analyzing two recovery strategies in case of resource failure: proactive and reactive protection. In the former, resources used for protection are pre-allocated, which tends to produce high delays due to the unavailability of resources. In the latter, protection resources are only allocated in case of failure, which does not guarantee that recovery will always be possible. Results showed that deciding between these strategies depends on the overall load, and both solutions helped to balance the workload in the fog, reducing the need for the cloud.

Sarkar et al. [86] analyzed the suitability of fog computing for IoT, evaluating the performance in scenarios with different amounts of data forwarded from fog to cloud nodes.

Table 3.3: Comparison of resource allocation mechanisms.

Proposal	Scenario	Criteria	Request	Prediction
[15]	Hybrid cloud	Cost and makespan	Workflow	No
[83]	Fog-cloud	Cost and makespan	Workflow	No
[102]	Fog	Network cost	Single task	No
[51]	Fog	Energy	Single task	No
[91]	Fog-cloud	Delay	Single task	No
[92]	Fog-cloud	Delay, resource utilization, and energy	Distributed Dataflow	No
[90]	Fog-cloud	Delay and protection	Single task	No
[86]	Fog-cloud	Energy	Single task	No
[33]	Fog-cloud	Delay and energy	Single task	No
[94]	Fog-cloud	Resource utilization	Distributed dataflow	No
[69]	Fog-cloud	Delay	Distributed dataflow	No
[57]	Fog-cloud	Delay, resource utilization, and energy	Distributed dataflow	No
Chapter 9	Fog-cloud	Energy, delay, and resource utilization	Distributed dataflow	Yes

The authors considered different aspects such as CO_2 emission, energy consumption, latency, and costs of fog nodes and cloud data centers. Results showed that moving workload to the fog reduced transmission and processing latency, helping to improve energy efficiency and greenhouse gas emissions. The authors of [33] proposed an optimization problem and an approximate solution to achieve a trade-off between energy consumption and delay. The authors showed that the interplay of cloud and fog can help to improve the overall delay and energy consumption.

Taneja and Davy [94] proposed an allocation algorithm to decide which tasks should be processed by fog and cloud nodes, considering the resource occupancy in the decision. Their solution matches tasks and resources by using a best-fit approach between the requested and available resources. The algorithm produced better latency compared to a cloud-only solution. The allocation policy in [69] focused on assuring the latency between dependent tasks, favoring the placement of tasks in distant nodes as long as the required latency is respected. Furthermore, the authors introduced a forwarding strategy to relocate tasks in an attempt to reduce the number of active nodes. Results showed an improvement in latency.

Finally, the authors of [57] proposed a fog computing architecture that employs a publish–subscribe protocol for the discovery of resources: physical devices announce their availability and users fetch a database to decide where to host their workload. The authors proposed a workload allocation mechanism that employs a score-based function to choose the most suitable resource for each task, considering utilization, latency, and battery state of devices. Results for mixed scenarios with cloud and fog showed that using fog computing is beneficial for latency and completion time.

Table 3.3 compares the papers reviewed in this subsection and compares them to the work developed in Chapter 9. The table classifies proposals according to the scenario adopted (hybrid cloud, fog, or fog-cloud), criteria adopted, type of request considered, and whether prediction of future requests was considered. The resource allocation mechanism in Chapter 9 considers a fog-cloud scenario and, different from previous work, it employs prediction of future tasks to avoid an overload of fog resources. Moreover, a complex application model (distributed dataflow [41]) is considered, and the proposed mechanism aims to improve the energy consumption as well as the delay.

3.5 Workload and Data Sets

Previous work evaluated scenarios in which all requests are submitted at the same time [11, 12, 61]. Differently, this thesis evaluated two variable characteristics of requests: space and time. Space is important in fog computing because fog nodes are geographically distributed and requests are sensitive to latency; therefore, serving distant requests is typically not possible and space constraints should be taken into account. Time is also important because a fog node can serve a limited amount of requests simultaneously, and requests are typically submitted at different times to the fog node. Therefore, fog nodes need to be provisioned considering such variability. In order to perform a realistic evaluation of the proposed solutions, this thesis employed data sets that model time and space variability. The remaining of this subsection presents such data sets, used in Chapters 4–8.

The main data set used in the evaluation is the one presented in [8]. The data set was collected by Telecom Italia and consists of Call Detail Records (CDRs) of mobile users in the metropolitan region of Milan, Italy. Although records were collected in 2013, this data set is largely used and represents demand patterns in metropolitan areas [67]. CDRs are records of transactions made by mobile users, representing calls, Short Message Service (SMS), and Internet accesses. The geographical area where CDRs were recorded was represented as a 100x100 grid, with each cell containing information about the SMSs and phone calls received and sent, as well as Internet accesses. CDRs were collected between November 1, 2013 and December 31, 2013, but they are not individually represented in the data set; instead they have been aggregated into 10-minute intervals. Discrete time was assumed in the formulations of this thesis due to this aggregation. The assumption does not represent a limitation: 10-minute length allows capturing users' mobility in a metropolitan area since users typically take more than 10 minutes to commute from one location to another.

The data set in [8] has CDR information about 10000 cells, but end-user requests are actually submitted to a base station (BS). In the work developed in this thesis, the coordinates of BSs are important. For instance, the work in Chapters 4–5 used the BSs as candidate locations for the deployment of fog nodes, and the distance between UAVs and BSs is used to calculate the wireless link capacity in Chapters 7–8. However, the data set in [8] had no information about the BSs, which required a second data set, the OpenCellId project [78]. The OpenCellId is a public database with location information of base stations worldwide collected by mobile users. The location of the base stations was obtained by filtering the existing base stations for the same period available in the Milan data set [8]. To estimate the workload received by each BS, the workload of each cell was mapped to the closest base station, as in [24]. In case of multiple base stations in a cell, the workload is equally balanced among the BSs. This operation resulted on a total of 1150 BSs. Each chapter explains how information from these data sets are used in the evaluation.

In Chapters 4 and 6, the input to the problem is the workload at each location, which can be represented by real or integer numbers. On the other hand, in Chapters 5, 7, and 8, the integer number of requests made at each location is needed. In the first case, the

value obtained from the Milan data set [8] was directly used in the evaluation. However, the second case required a different solution since the owners of the Milan data set [8] anonymized the total number of accesses by multiplying the real number by an unknown constant. To cope with this in Chapters 5, 7, and 8, the number of requests was normalized using a constant Z, which was multiplied by the value representing Internet access data in the data set to simulate the number of requests in each cell.

A third data set was required in Part III of this thesis. The OpenCellId project [78] provides information on the geographic coordinates (latitude and longitude), but it has no information on the altitude of these geographical locations; such information is important to calculate the 3D distance between UAVs and BSs. Thus, the altitude of BSs was obtained from the Shuttle Radar Topography Mission [37], a project conducted by the National Aeronautics and Space Administration (NASA) and the National Geospatial-Intelligence Agency (NGA) to map the elevation of most parts of the globe.

3.6 Software tools

The work developed in this thesis was evaluated using computational simulations. Code for evaluating solutions in Chapters 6–8 was developed in Python and in the Gurobi solver. Moreover, the solution in Chapter 9 employed simulations using Java and the iFogSim simulator [46]. This subsection presents a brief review of these tools.

Python is a popular interpreted programming language that can be used for many purposes. A significant advantage is its external libraries that introduce many new functionalities. Python was chosen to implement the fog node location algorithms and mathematical formulations due its ease of use and easy integration with the Gurobi solver. The Gurobi Optimizer is a commercial optimization solver developed by Gurobi Optimization, LLC. It can solve different types of models, such as integer linear programming (ILP), mixed-integer linear programming (MILP), quadratic programming, and mixed-integer quadratic programming [47]. Gurobi has application programming interfaces (APIs) for different languages, such as Python, C++, Java, .NET, C, and R. Gurobi employs different algorithms, such as simplex, parallel barrier with crossover, concurrent and sifting. Gurobi was chosen to obtain solutions for linear programming models due its easy use, good performance, and the availability of licenses for academic use.

Java is a multi-purpose object-oriented programming language that can run on multiple platforms. It is very popular and one of the most used programming language in the world. The evaluation of Chapter 9 employed Java because it is the language of the iFogSim simulator [46]. iFogSim was built using the core of the CloudSim simulator [19] and allows the simulation of scheduling algorithms and multiple layers of fog nodes. iFogSim comes with support to the simulation of sense-process-actuate applications, in which sensors produce data and actuators execute actions. Sensors and actuators are connected by intermediary processing tasks; such tasks are hosted by fog or cloud nodes, and algorithms to map tasks to these nodes can be implemented and evaluated in the simulator. iFogSim was used in Chapter 9 because it provided built-in functions that facilitated the implementation of a resource allocation mechanism.

Part II

Terrestrial Infrastructure

Chapter 4

Location of Fog Nodes for Reduction of Cost

4.1 Overview

A fog-cloud infrastructure is useful in the execution of applications consisting of multiple tasks with different latency requirements [45]. For example, the augmented reality application described in [97] is divided into four tasks; two of them should be processed in the fog due to their strict latency requirements, and the other two tasks can be processed in the cloud. These two tasks with flexible latency requirements can also be processed in the fog, reducing the delay experienced by end-users.

Previous papers [79, 70, 98, 59] have discussed the role of fog nodes in the architecture and their connection to other network elements, but have not discussed the impact of the creation of fog nodes on different physical locations. The problem of locating fog nodes consists in deciding where these nodes should be placed given a set of potential locations and the devices available for deployment. The solution to the problem is crucial for fog providers since it affects both users and the provider. If the delay in accessing the fog is too high, the execution of some applications will be infeasible, jeopardizing end-users. Moreover, reckless decisions can guarantee user satisfaction, but at a high deployment cost for fog providers.

This chapter presents a solution for the fog node location problem aimed at reducing the capital expenditure (CAPEX), and it considers only terrestrial fog nodes. The problem is formulated as a mixed-integer linear programming (MILP) model that considers various inherent aspects of a fog-cloud system. To evaluate different classes of service, the model considers two types of demands in terms of latency: strict (which can only be processed in a fog node) and flexible (which can be hosted either in the fog or in the cloud). By considering these two types of workload, the solution attempts to serve requests which are dependent on the fog while improving the latency experienced by flexible applications. The solution was designed as a multicriterial optimization problem, aiming at providing service of all demands at a reduced cost. The demand of workload to be processed varies with time and such variability is taken into account in the formulation. A multi-level programming approach was employed to select a solution from the Pareto front,

ordering the multiple objectives in a hierarchical manner. Solutions were obtained using the hierarchical order of the objectives, and alternative solutions allowing degradation in the objective functions were evaluated; this showed that reducing the quality of service in the service provisioning at a certain extent can lead to big savings in infrastructure costs.

The remainder of this chapter is organized as follows. Section 4.2 introduces the system model adopted. Section 4.3 presents the linear programming formulation of the problem and exemplifies the multicriterial solution. Section 4.4 presents the experimental setting and numerical evaluation of the proposed formulation. Lastly, Section 4.5 concludes this chapter.

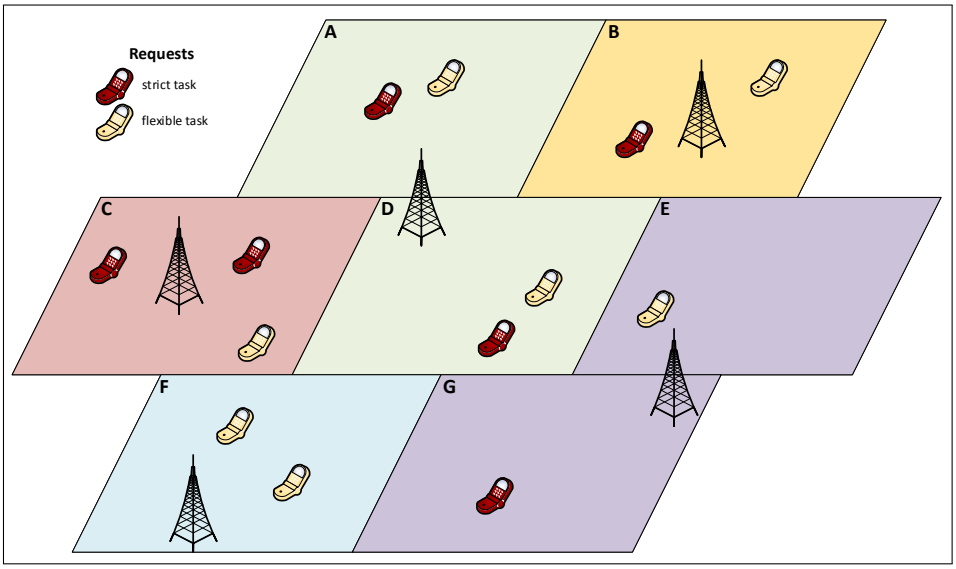
4.2 System Model

This section details the model for the system considered in this chapter as well as the fog node location problem. The system is composed of a cloud and various fog nodes, hierarchically organized in three layers: cloud, fog, and end-user devices. The cloud can be accessed by any device. The fog layer is formed by fog nodes, with each fog node having a limited area of coverage. A fog node is a small facility that hosts dedicated servers capable of processing end-user workload. Compared to the cloud, fog node resources are limited. End-user devices are in the lowest layer and can move between different areas. These devices run several types of applications with different latency requirements. A user can access either the closest fog node (as long as this fog node covers the user) or the cloud. The decision of where to process user workloads depends on the application requirements. In this chapter, the workloads are classified into two classes: fog (strict latency) and cloud (flexible latency) workloads. The former represents workloads which can only be hosted in a nearby fog node due to the latency requirements, while the latter can be processed in either the fog or the cloud.

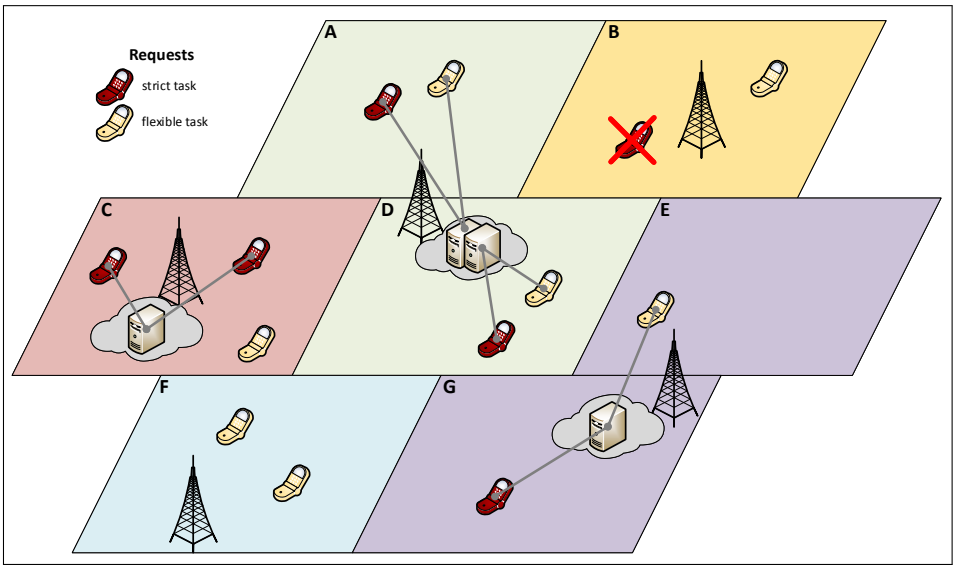
Supporting client applications (workload) requires making the decision about the location of the fog nodes. To make such a decision, the selection of potential locations for receiving dedicated servers is necessary. Moreover, the selection of fog node locations and capacity of each node should take into consideration the history of demands in these locations.

Each fog node is characterized by its location and the number of servers. The greater is the number of servers, the larger is the capacity of the fog node. To increase the total workload processed, strict latency workload should be first assigned for execution on fog servers. The remaining capacity of the fog nodes can then be used to process flexible latency workload. Executing flexible latency workload in the fog can reduce the latency for this type of load, thus enhancing user experience.

The system considered in this chapter assumes that both strict and flexible workloads vary over time. Without loss of generality, a discrete-time model has been adopted. Figure 4.1a illustrates the node location problem. This figure presents a segment of a city, divided into seven areas, identified by letters A–G, with end users served by five base stations (BSs). Regions A and D are served by the same BS,another BS processes the requests made in Regions E and G, and the remaining regions are each served by an individual



(a) Possible locations and available number of servers.



(b) Fog nodes decided and requests served by them.

Figure 4.1: Example of fog location decision making.

BS. A cellphone represents a request and the color associated with it identifies the type of request (strict or flexible). BSs are considered to be possible locations for hosting a fog node. Suppose that the provider can employ up to four servers, and each server can host two requests at the same time. One possible solution for this scenario is the one in Figure 4.1b. Three fog nodes have been created, one with two servers in the BS in Region D, and two nodes with a single server in Regions C and G. The fog node in D can serve both strict and flexible requests in its coverage area. The fog node in C serves strict requests in its area, as well as a flexible request. In Region B, strict requests are blocked, since no fog node is available.

This example provides a snapshot of user positions. However, end users can change their position dynamically, thus leading to different occupation of devices over time in each region. Consequently, the deployed infrastructure must be efficient for the service

over time, not only during a specific time interval.

4.3 Formulation

The fog node location problem is modeled as a multicriteria mixed-integer linear programming formulation. The goal is to process most of the strict workload in the fog nodes using the minimum number of servers possible to reduce the overall cost. Moreover, the unused capacity of fog nodes should be used for the processing of flexible latency workload to further reduce the latency of users with this type of workload. This section is divided into two parts: Subsection 4.3.1 presents the formulation of the optimization problem, and Subsection 4.3.2 illustrates the optimization with a numerical example.

4.3.1 Mathematical Model

The notation used in the model is presented in Table 4.1. The provider budget constraint is given by N, the maximum number of dedicated servers to be employed in the fog nodes, each of them with capacity R. $\mathcal{L}$ and $\mathcal{T}$ are the location and time interval sets, respectively. f_{lt} and c_{lt} are also part of the input and represent, respectively, the strict latency and flexible latency workload demands at location l and time t. The solution consists of α_l, the number of dedicated servers deployed at each location. Additionally, variables ff_{lt}, cf_{lt}, and α_{lt} indicate where each demand is processed (fog or cloud) for all locations and time periods.

The multi-objective formulation is given by Equations (4.1)–(4.10):

$$\text{maximize} \sum_{l \in \mathcal{L}} \sum_{t \in \mathcal{T}} (ff_{lt}) \tag{4.1}$$

$$\text{minimize} \sum_{l \in \mathcal{L}} \alpha_l \tag{4.2}$$

$$\text{maximize} \sum_{l \in \mathcal{L}} \sum_{t \in \mathcal{T}} (cf_{lt}) \tag{4.3}$$

$$\sum_{l \in \mathcal{L}} \alpha_l \leq N \tag{4.4}$$

$$ff_{lt} + cf_{lt} \leq \alpha_l \cdot R, l \in \mathcal{L}, t \in \mathcal{T} \tag{4.5}$$

$$ff_{lt} \leq f_{lt}, l \in \mathcal{L}, t \in \mathcal{T} \tag{4.6}$$

$$cf_{lt} + \alpha_{lt} = c_{lt}, l \in \mathcal{L}, t \in \mathcal{T} \tag{4.7}$$

$$cf_{lt} \geq 0, l \in \mathcal{L}, t \in \mathcal{T} \tag{4.8}$$

$$ff_{lt} \geq 0, l \in \mathcal{L}, t \in \mathcal{T} \tag{4.9}$$

$$\alpha_l \geq 0, l \in \mathcal{L} \tag{4.10}$$

Equations (4.1)–(4.3) are the objective functions. Equation (4.1) maximizes the pro-

Table 4.1: Notation used in the fog node location problem formulation.

Input	
Notation	**Description**
N	Maximum number of servers to be deployed
R	Capacity of a single server
L	Number of locations where a fog node can be deployed, $L \in \mathbb{N}^+$
$\mathcal{L}$	Set of all locations where a fog node can be deployed: $\mathcal{L} = \{1, 2, \ldots, L\}$
T	Total number of discrete time intervals, $T \in \mathbb{N}^+$
$\mathcal{T}$	Set of all discrete time intervals: $\mathcal{T} = \{1, 2, \ldots, T\}$
f_{lt}	Strict workload at location $l \in \mathcal{L}$ at time $t \in \mathcal{T}$
c_{lt}	Flexible workload at location $l \in \mathcal{L}$ at time $t \in \mathcal{T}$
Decision variables	
Notation	**Description**
α_l	The number of servers deployed at location $l \in \mathcal{L}$. If $\alpha_l = 0$, no fog node is created at location l
ff_{lt}	Strict workload originating at location $l \in \mathcal{L}$ at time $t \in \mathcal{T}$ and hosted by the local fog node
cf_{lt}	Flexible workload originating at location $l \in \mathcal{L}$ at time $t \in \mathcal{T}$ and hosted by the local fog node
cc_{lt}	Flexible workload originating at location $l \in \mathcal{L}$ at time $t \in \mathcal{T}$ and hosted by the cloud

cessing of strict workload on the fog nodes, i.e., it guarantees the maximum number of users for each time slot. To achieve this goal, the number of fog nodes at each location is determined using the minimum possible number of servers with Equation (4.2). Moreover, Equation (4.3) ensures that servers are deployed to locations where the remaining capacity can be used to boost the processing of flexible latency workload in the fog.

The constraints of the model are explained by the following. The constraint in Equation (4.4) limits the number of deployed servers to the total number of available devices N. The constraint in Equation (4.5) guarantees that the workload processed by each fog node (sum of strict and flexible workload) is never greater than its capacity (number of servers multiplied by the capacity of a single server). The constraint in Equation (4.6) limits the strict latency workload processed at a fog node to the demand at that location. The constraint in Equation (4.7) guarantees that all flexible latency demand is met, whether at a local fog node or in the cloud. Finally, the constraints in Equations (4.8)–(4.10) set the minimum values for the decision variables.

In the fog node location problem, the service of end users is essential. Once this is achieved, the provider costs should be reduced and the usage of the remaining servers optimized. As a consequence of this order of priorities, the problem is appropriate to be solved using multi-level programming explained in Subsection 2.4. Equation (4.1) is the main objective, followed by the objectives given by Equations (4.2) and (4.3). Other multicriterial methods could have been employed for the solution, but they do not take into consideration the hierarchy between the objectives, either favoring a single objective or a trade-off, which does not make them adequate for the problem in this

chapter. Nonetheless, to evaluate multiple solutions, degradation in some of the objectives is evaluated.

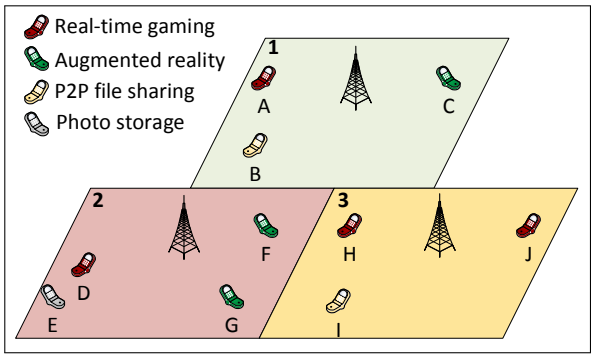
4.3.2 Numerical Example

To numerically illustrate the proposed MILP model, consider the example displayed in Figures 4.2a and 4.2b, which shows a snapshot of end-users' position at time slot 1 and 2, respectively, for a small region of a city. There are three locations (1, 2, and 3) served by base stations; such BSs are candidates for the deployment of fog nodes. Eleven users, identified by letters A–K, execute four different applications in their smartphone. Users A, D, H, and J play a real-time game, while Users C, F, and G execute an augmented reality application, both applications require a fog node due to the low latency constraints. The remaining users execute applications which can be either processed in the fog or in the cloud due to their flexible latency requirements: Users B, I, and K share files in a P2P network while User E takes photos and then processes and stores them externally. Users that share files can take advantage of the fog by sharing files between them without the delay imposed by the cloud. For User E, the presence of a fog node allows the image processing in the fog, which reduces the transmission of large raw files to the cloud. Although the execution of flexible latency applications can be boosted with a fog node, their processing can be realized by the cloud. Additionally, this example presents mobility: from time slot 1 to 2, User A goes from Location 1 to 2; User E from 2 to 3; Users D, G, and I leave the displayed area; and the User K arrives in Location 3 only at the second time slot.

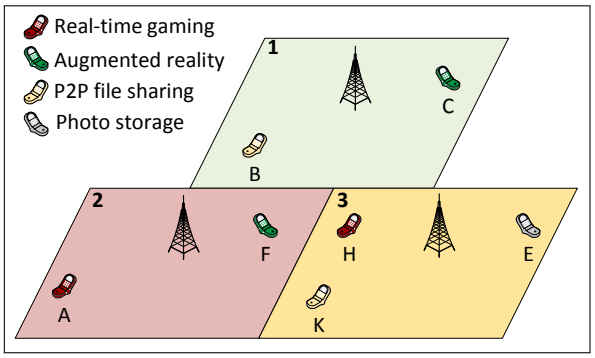
The presented scenario can be mapped into the input of the fog node location problem. There are three base stations in Figure 4.2, thus $L = 3$ and $\mathcal{L} = \{1, 2, 3\}$, and only two time slots are considered, so that $T = 2$ and $\mathcal{T} = \{1, 2\}$. Suppose that each fog server can host up to three requests at the same time ($R = 2$). Considering the requests displayed in Figures 4.2a and 4.2b, the strict and flexible workloads assume the following values: $f_{11} = 2$, $f_{21} = 3$, $f_{31} = 2$, $f_{12} = 1$, $f_{22} = 2$, $f_{32} = 1$, $c_{11} = 1$, $c_{21} = 1$, $c_{31} = 1$, $c_{12} = 1$, $c_{22} = 0$, and $c_{32} = 2$. All these values are used as input to the problem. The values of N are varied to exemplify the priority of each objective in the multi-level programming approach.

The main goal of the formulation is to serve all strict workload (the objective function in Equation (4.1)). To illustrate that, consider $N = 1$, i.e., only one fog node with a single server can be deployed. In this case, a fog node is created at Location 2 ($\alpha_1 = \alpha_3 = 0$ and $\alpha_2 = 1$) since it produces 5 for Equation (4.1). If $\alpha_1 = 1$ or $\alpha_3 = 1$, the produced values (3 in both cases) would not be optimal. The solution for $N = 1$ is displayed in Figures 4.2c–4.2d.

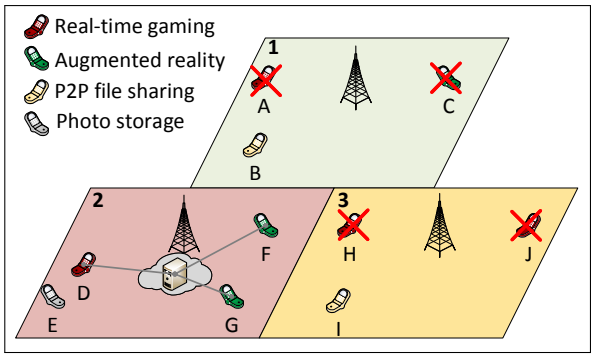
The effect of the objective function in Equation (4.2) is noticed for $N = 4$. In this case, all locations can be covered by fog nodes with a single server ($\alpha_1 = \alpha_2 = \alpha_3 = 1$), case in which no strict application is blocked and the value obtained for Equation (4.1) is 11. The addition of the fourth server in any fog node does not increase the value of Equation (4.1), thus the objective function in Equation (4.2) limits the employed servers to 3 to avoid extra costs with the infrastructure deployment. The scenario described in



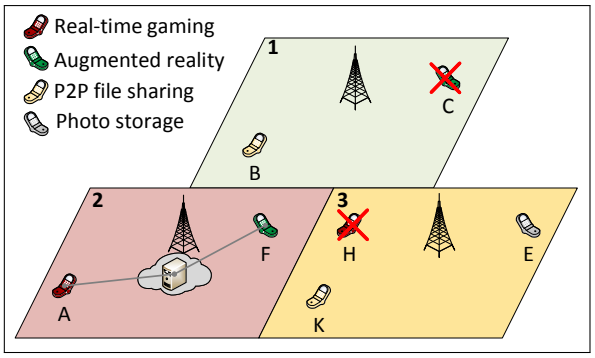
(a) Input at the first time slot.



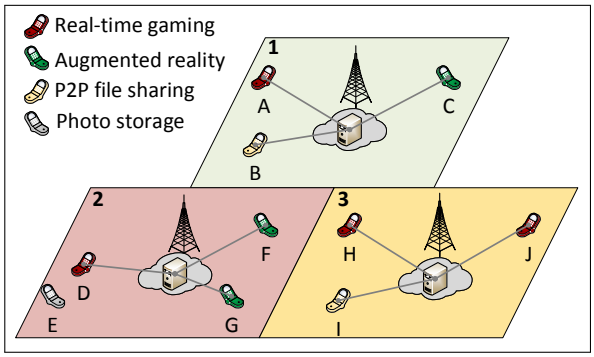
(b) Input at the second time slot.



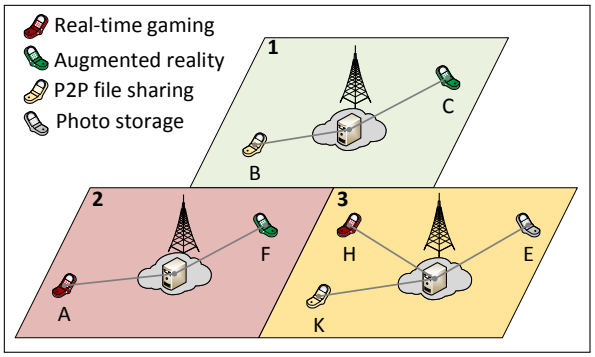
(c) Solution for $N = 1$ at the first time slot.



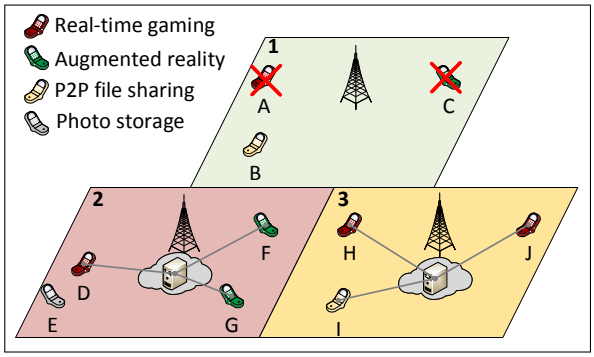
(d) Solution for $N = 1$ at the second time slot.



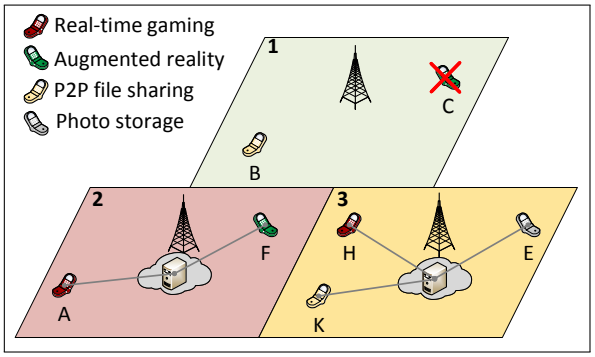
(e) Solution for $N = 4$ at the first time slot.



(f) Solution for $N = 4$ at the second time slot.



(g) Solution for $N = 2$ at the first time slot.



(h) Solution for $N = 2$ at the second time slot.

Figure 4.2: Numerical example of fog location decision making.

this paragraph is illustrated in Figures 4.2e–4.2f.

Finally, a practical example of the effect of the objective function in Equation (4.3) happens for $N = 2$ (Figures 4.2g–4.2h). As discussed earlier, the most demanded fog node is the one in Location 2 ($\alpha_2 = 1$), thus, when there is an extra server available, the decision is which of the other locations should host a fog node, $\alpha_1 = 1$ or $\alpha_3 = 1$. Either option produces the same value for Equations (4.1) and (4.2): 8 and 2, respectively. Therefore, the objective function in Equation (4.3) is evaluated. If $\alpha_1 = 1$, then Equation (4.3) assumes the value 2, while $\alpha_3 = 1$ leads to the value 3. Thus, the fog node is deployed in Location 3, allowing Users E, I, and K to use the fog instead of the cloud, improving the latency of the delivered service.

4.4 Performance Evaluation

In order to evaluate the location of fog nodes in a fog-cloud infrastructure, this section presents an evaluation of the MILP model using actual data of workloads. The MILP model was coded using the Gurobi Optimizer solver and two datasets were used to model the workload. This section is structured as follows. Subsection 4.4.1 explains how the workload was modeled, Subsection 4.4.2 describes how alternative trade-offs were obtained using objective function degradation, and Subsection 4.4.3 discusses the numerical results.

4.4.1 Workload

The set of locations $\mathcal{L}$ represents the 1150 BSs from the database in [78], and the values of variables f_{lt} and c_{lt} (fog/strict and cloud/flexible workload demands) were taken from the dataset in [8]. The process to obtain such values is explained in Section 3.5. The input to the problem consisted of N, R, $\mathcal{L}$, $\mathcal{T}$, f_{lt}, and c_{lt}. The capacity R of a server was fixed, and N was varied to evaluate solutions obtained under different budget constraints. The number of locations in $\mathcal{L}$ was determined using the OpenCellId dataset. T was also varied to evaluate the solution under different lengths of planning intervals, from 1 h to 24 h. The proportion of fog and cloud requests was varied using three scenarios, namely $P25$, $P50$, and $P75$. In $P25$, 25 % of the workload for an antenna was strict and 75 % flexible. In the $P50$ scenario, the proportion was 50 % for each type of request, and, in $P75$, the workload is 75 % strict and 25 % flexible. Table 4.2 summarizes the input values and the adopted scenarios.

4.4.2 Multi-objective solutions allowing degradation

The MILP model presented in Section 4.3 was coded using the multi-level programming approach; this solution was identified by OPT. Employing only OPT leads to a single solution for the problem. However, a fog provider can accept decreasing performance for one of the objectives if there is an advantageous trade-off for the multiple objectives. Various solutions were evaluated that allowed degradation in some of the objective functions.

Other solutions differ from OPT by allowing degradation of either the objective function in Equation (4.1) or the objective function in Equation (4.2). Solutions that allow

Table 4.2: Adopted values of input and scenarios.

Parameter	**Values**
N	1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048
R	1000
$\mathcal{L}$	$\mathcal{L} = \{1, 2, ..., L\}$, $L = 1150$
$\mathcal{T}$	$\mathcal{T} = \{1, 2, ..., T\}$, each $t \in \mathcal{T}$ represents a ten minute interval. T varies to represent 1 h, 3 h, 6 h, 12 h, and 24 h intervals
f_{lt} and c_{lt}, $l \in \mathcal{L}$, $t \in \mathcal{T}$	Aggregated workload of cells for each base station
Proportion between strict and flexible workloads	P25: 25 % of strict and 75 % of flexible latency workload P50: 50 % of strict and 50 % of flexible latency workload P75: 75 % of strict and 25 % of flexible latency workload

Table 4.3: Solutions evaluated in this chapter as well as objective function affected and level of degradation allowed.

	Objective degraded	**Level of degradation**
OPT	—	—
$STR5$	Equation (4.1)	5 %
$STR10$	Equation (4.1)	10 %
$STR15$	Equation (4.1)	15 %
$STR20$	Equation (4.1)	20 %
$SER5$	Equation (4.2)	5 %
$SER10$	Equation (4.2)	10 %
$SER15$	Equation (4.2)	15 %
$SER20$	Equation (4.2)	20 %
$SER25$	Equation (4.2)	25 %
$SER30$	Equation (4.2)	30 %

degradation of the first objective function (Equation (4.1)) are identified by $STRX$, where X is the percentage value that can be degraded from the total served strict workload. By allowing degradation of the first objective function, these solutions can employ fewer servers, thus reducing deployment costs. Degradation of the second objective function (Equation (4.2)) was also evaluated. $SERX$ identifies the solutions that degrade the number of employed servers, i.e. they allow an increase in the number of servers in X % in relation to OPT to increase the amount of flexible workload processed in the fog. Since strict workloads are blocked if not served in the fog, applying $STRX$ has a great impact on end users, thus only up to 20 % of degradation was evaluated. Employing more servers, differently, does not prevent the execution of strict workloads, thus, up to 30 % of degradation was evaluated for $SERX$. Table 4.3 summarizes all the solutions evaluated.

4.4.3 Numerical results

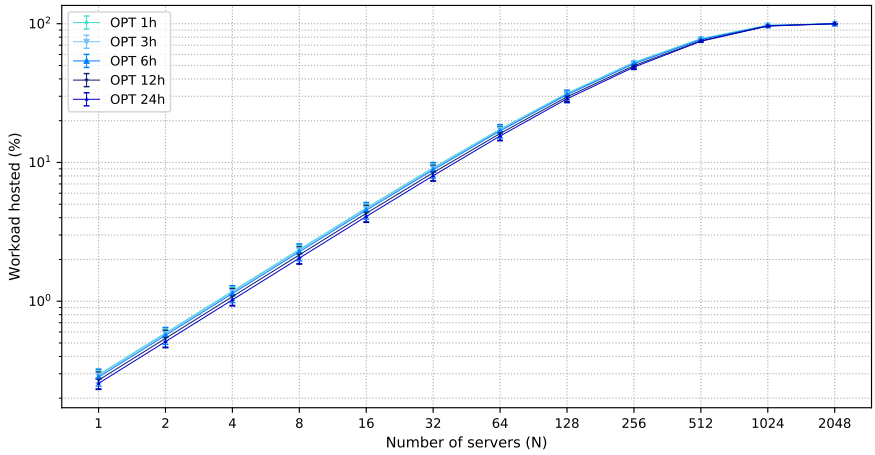
In this subsection, the performance of the proposed solution is assessed. This evaluation showed how the solution improves fog service, reducing costs and dealing with the two

types of workload. Furthermore, several scenarios with different traffic patterns and budget constraints were used to evaluate the efficiency of the solution. First, the results produced by OPT using different planning intervals are discussed. Then, the results obtained under degradation are presented. Finally, different scenarios of traffic patterns ($P25$, $P50$, and $P75$) were evaluated. Three metrics were considered: acceptance ratio of strict latency workload, acceptance ratio of flexible latency workload in the fog, and the number of deployed servers. The X-axis of all graphs represents the number of available servers for deployment. A 95 % confidence interval is used in the graphs. Graphs of the strict latency acceptance ratio are in a logarithmic scale.

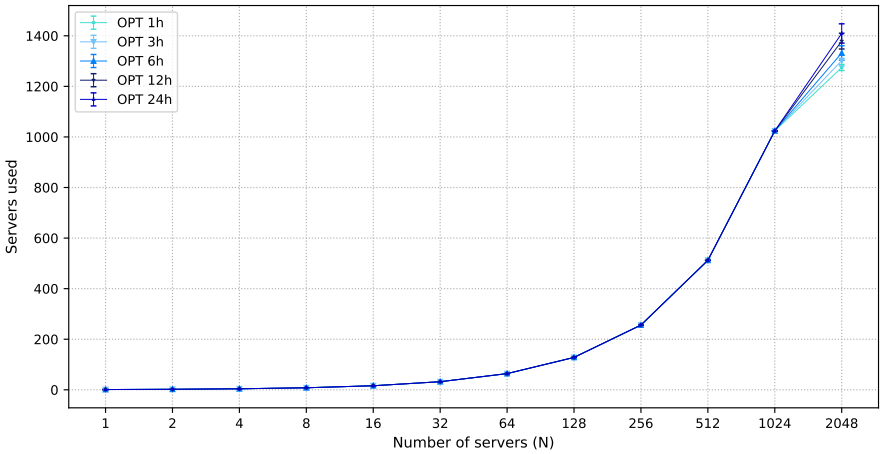
Figure 4.3 displays the resuls for OPT and the $P50$ scenario. The larger was the number of available servers, the larger was the number of servers utilized (Figure 4.3b). This is a result of the main goal of the solution to serve the maximum number of strict workloads possible, which led to more servers being used in the solution. For $1 \leq N \leq 1024$, the available servers could not cope with the entire strict latency workload since most of the available servers (N servers) were used. This caused the overlap of the curves for all planning intervals. For $N \geq 2048$, the available capacity was greater than the total demand, so that the entire demand was met (Figure 4.3a), requiring between 1480 and 1710 servers. The number of required servers varied according to the planning interval: short planning intervals may not contain periods during which a location is crowded. Consequently, for longer intervals, a large number of periods of peak demand was present for several locations, which required the deployment of a large number of servers. Results for $N > 2048$ were the same as those for $N = 2048$ since the multi-level programming approach optimized the entire served demand in Equation (4.1); hence a larger number of servers did not lead to any improvement in the strict latency workload service.

Figure 4.3c shows the ratio between flexible requests served in the fog and the total flexible workload. The extra capacity of fog nodes can be used to host the flexible workload, thus, when nearby 80 % of the strict workload is served ($N = 512$), more than 30 % of the flexible workload can be executed in the fog, which improved the latency of end users as well as allowed more flexibility in the energy management of the cloud data center. OPT maximizes flexible requests utilization of fog nodes (Equation (4.3)) only after satisfying the objective functions in Equations (4.1) and (4.2). As a result, no new fog servers were deployed to host only flexible workloads. Thus, for $N = 2048$, between 60 % and 80 % of the flexible workload was hosted in the fog and the remainder in the cloud. If the order of objective functions in Equations (4.2) and (4.3) were reversed in the multi-level optimization, flexible workload allocation would be prioritized in the fog, but at a higher server deployment cost than that was in the original order.

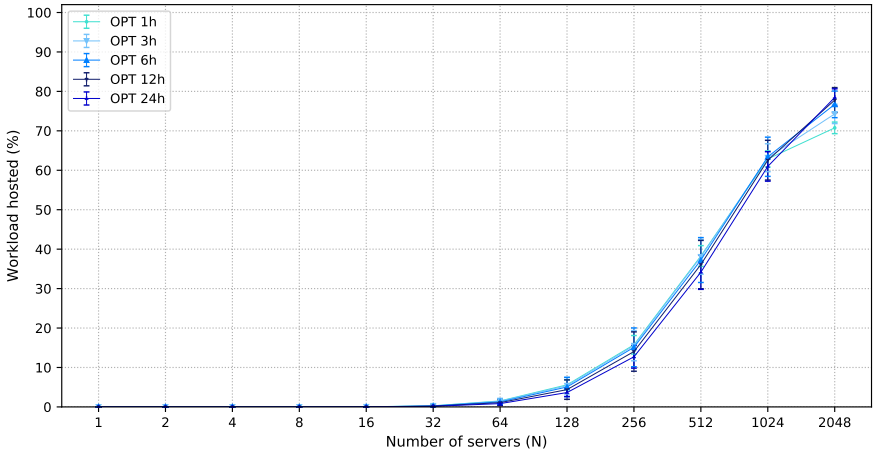
The order of the curves changes in the interval $1024 \leq N \leq 2048$ in Figure 4.3c due to the availability of a larger number of servers for $N = 2048$ and longer planning intervals. For $N \leq 1024$, the available resources did not meet the full demands of the strict workload. Conversely, when $N = 2048$, all strict workloads were processed, and powerful fog nodes tailored to the peak demands were deployed. Consequently, during periods with low strict demands, fog servers were used to host the flexible workload. Thus, solutions for larger planning intervals were capable of hosting more flexible workloads, explaining the difference in the order of the curves in Figure 4.3c for N between 1024 and 2048.



(a) Strict latency workload acceptance ratio.



(b) Average number of servers employed.



(c) Flexible latency workload acceptance ratio in the fog.

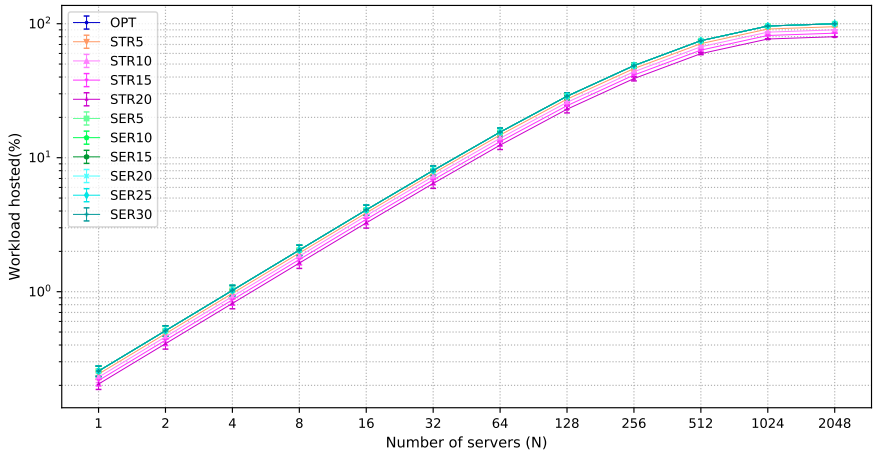
Figure 4.3: Results obtained for OPT under $P50$ scenario.

In the remainder of this subsection, results for 24 h planning intervals are shown. Using a large interval resulted in more variation of demands in the considered locations, which is useful when planning long-term infrastructures. A comparison between OPT and the solutions which allowed degradation in one of the objective functions is presented in Figure 4.4 for the $P50$ scenario. The acceptance ratio of strict latency workloads is shown in Figure 4.4a. The curves for OPT and all solutions that allowed degradation in the objective function in Equation (4.2) overlap since they were optimized after the objective function in Equation (4.1). Curves corresponding to $STRX$ are parallel to OPT in the log scale according to the allowed degradation, from 5 % to 20 %.

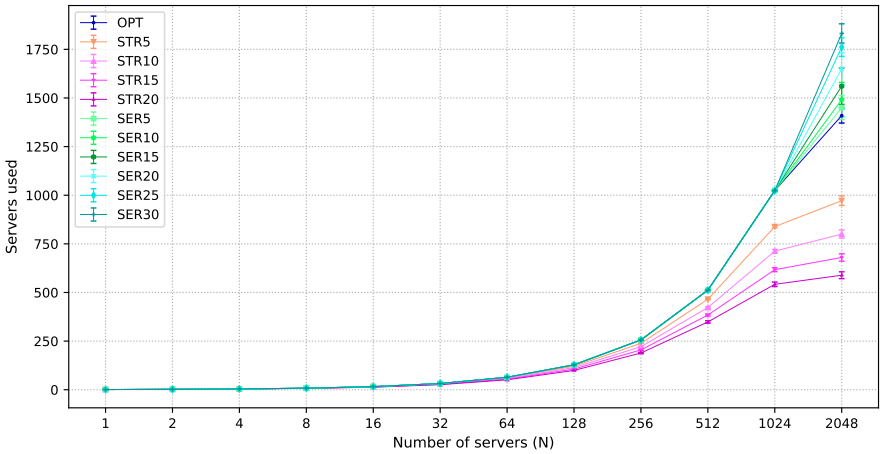
Figure 4.4b shows the number of employed servers as a function of N. $SERX$ deployed a larger number of servers than OPT and $STRX$. Since all servers were used for $1 \leq N \leq 1024$, differences in the values obtained by OPT and $SERX$ appear only for $N = 2048$, when there is more capacity than that required for the strict workloads. For $SERX$, extra servers are employed to host more flexible latency workloads in the fog, as shown in Figure 4.4c. Notice, however, that an increase in the number of servers less than 15 % resulted in a minimal increase in the flexible latency service in the fog. This is due to the distribution of demand across different locations. To explain this trend better, Figure 4.5 presents results for flexible latency workload in the fog for all values of the planning intervals considered (1 h, 3 h, 6 h, 12 h, and 24 h) and $N = 2048$. For 1 h planning, an increase in the number of servers led to an increased acceptance ratio of flexible workloads in the fog. However, results for longer intervals show that small gains were obtained for degradation smaller than 15 %. For small intervals, users are less mobile, which makes demands more uniform over all locations. Longer intervals, however, presented peak demand periods on a larger number of locations. Thus, given that servers cannot be moved from one fog node to another, serving the total flexible demand in the fog requires a large number of servers in many fog nodes, making the employment of $SERX$ effective only when high degradation is allowed.

One important effect of $STR5$ is noticed for $N = 2048$, where it reduced more than 400 servers in the solution in relation to OPT, which accounts to about 30 % of savings in server costs (Figure 4.4b). This is due to the fact that the removal of one or two servers from each fog node does not lead to great blockage. Serving strict workloads is the main goal of optimization, hence most servers process mainly this type of workload. However, to fully process the demand, a fog node may have servers that remain idle or process only a small number of strict workloads. For example, during an interval facing peak demand, a fog node may need five servers to process all the strict demand, while most of the time only three or four servers would be sufficient. Thus, even if degradation in the objective function in Equation (4.1) is small, high infrastructure costs can be avoided if the fog node capacity is not tailored to the peak demands in the fog area. If the blocking of a small number of requests is acceptable, $STRX$ becomes a viable solution.

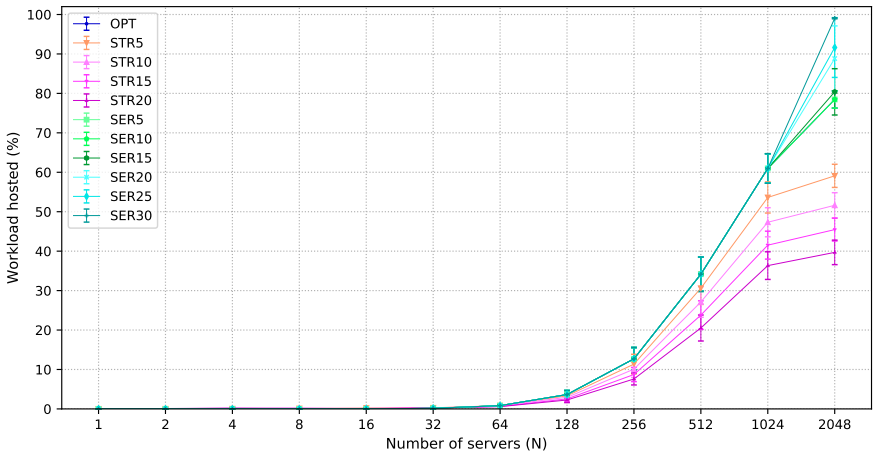
Results for the acceptance of strict latency workload in the $P25$ and $P75$ scenarios, yet for OPT, $STRX$, and $SERX$ solutions and 24 h planning intervals, are displayed in Figure 4.6. Results followed the same pattern of those in the $P50$ scenario. All solutions resulted in greater acceptance of strict latency workload under $P25$ than for those of $P50$, and less than those of $P75$. The former is explained by the reduction of the strict



(a) Strict latency workload acceptance ratio.



(b) Average number of servers employed.



(c) Flexible latency workload acceptance ratio in the fog.

Figure 4.4: Results obtained for all solutions under $P50$ scenario.

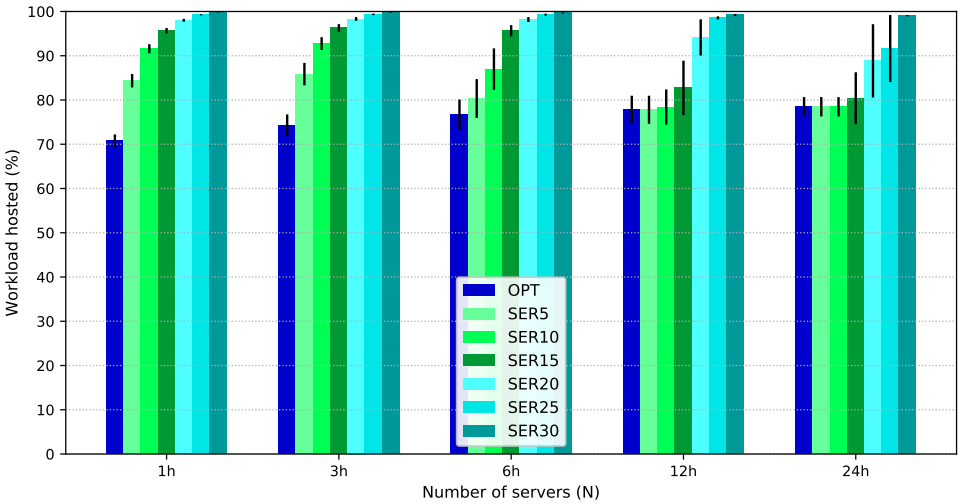


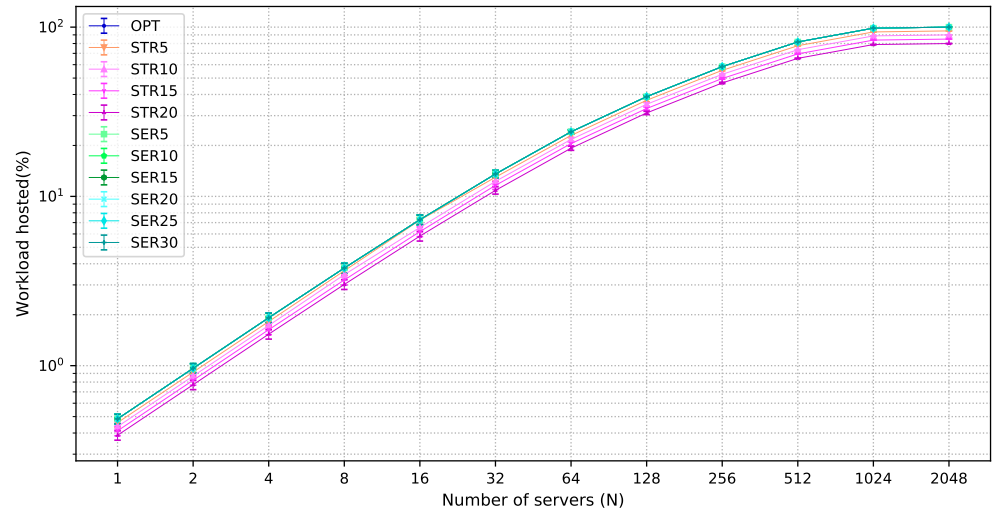
Figure 4.5: Flexible latency workload acceptance ratio in the fog for various planning intervals, $N = 2048$ and $P50$.

workload, making the available servers sufficient for dealing with a larger part of the strict demand. The opposite situation happens when there are more strict workloads, when the strict demands are harder to serve.

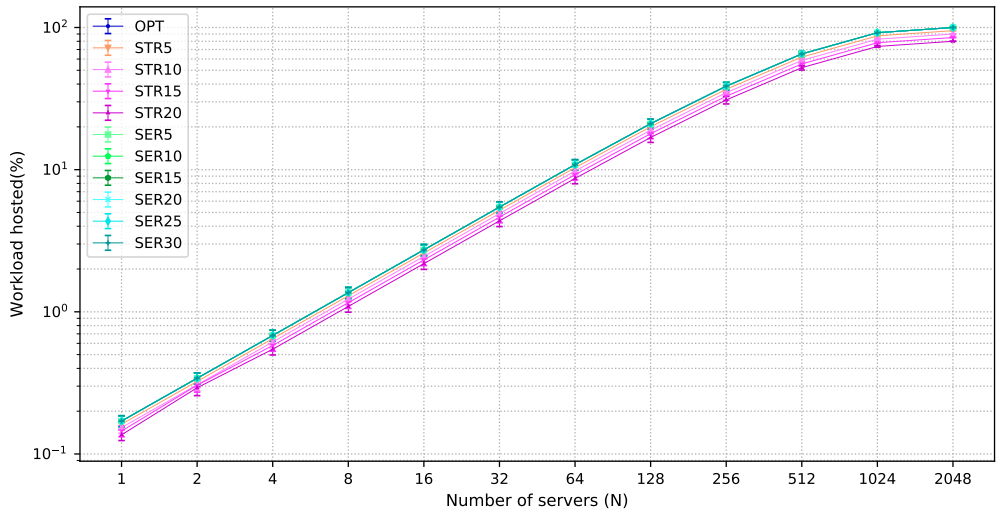
The acceptance of flexible latency workloads in the fog and the number of employed servers are shown in Figures 4.7 and 4.8, respectively, for both $P25$ and $P75$ scenarios. In the $P25$, there are less strict workloads. Accordingly, the total number of employed servers was reduced (Figure 4.8a), which also reduced the capacity available for hosting flexible workloads (Figure 4.7a). For $P75$, there was much more strict workload, which required about 1700 servers (Figure 4.8b). The reduced demand for flexible latency workloads (Figure 4.7b) allowed almost 100 % processing of this demand in the fog nodes for OPT, and the employment of $SERX$ under these circumstances led to few gains. Finally, applying $STR5$ instead of OPT led to savings about 30 % for both the $P25$ and $P75$, as shown for $P50$. When the strict workload demand is high ($P75$), the absolute number of servers is higher, thus $STR5$ could reduce costs considerably with the infrastructure.

All results in this subsection were obtained using the Gurobi Optimizer solver. The execution time depends on the input size, mainly affected by N and the planning interval length. Scenarios with the largest inputs, high N and 24 h intervals, took less than 350 s, which is less than 1 % of the planning interval length. Therefore, the proposed solution is feasible and, in the case of changes of demands, the location of fog nodes can be quickly recalculated.

This subsection has presented an evaluation of the results produced by the multicriterial optimization formulation employing multi-level programming proposed. Solutions considered hierarchical objectives with and without the allowance of degradation in one of the objective functions. The deployment of a fog infrastructure requires an analysis of all locations. Moreover, variability of end-user demands caused different regions to have demand peaks at different times, which is an important aspect to be accounted for in the location decision. Given the priority of the multiple objectives, OPT represents the

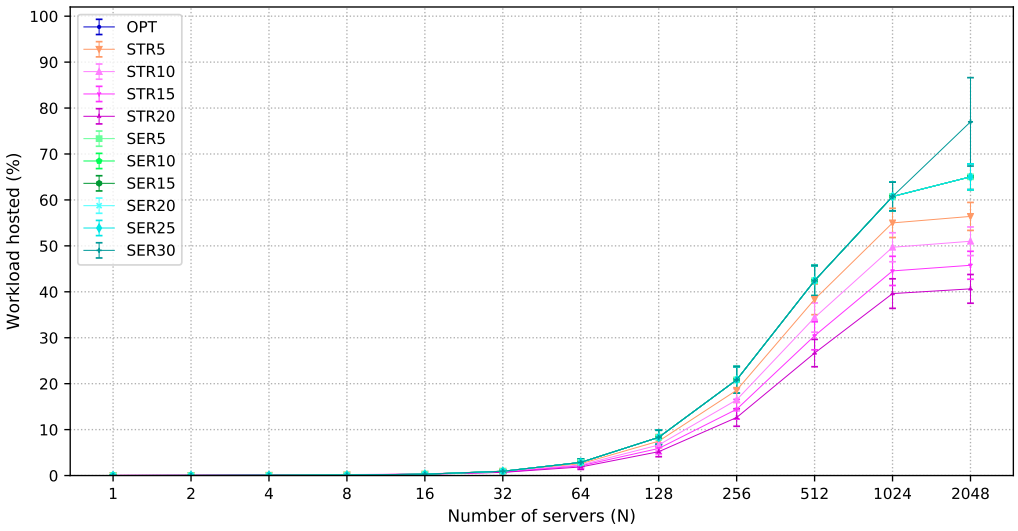


(a) $P25$.

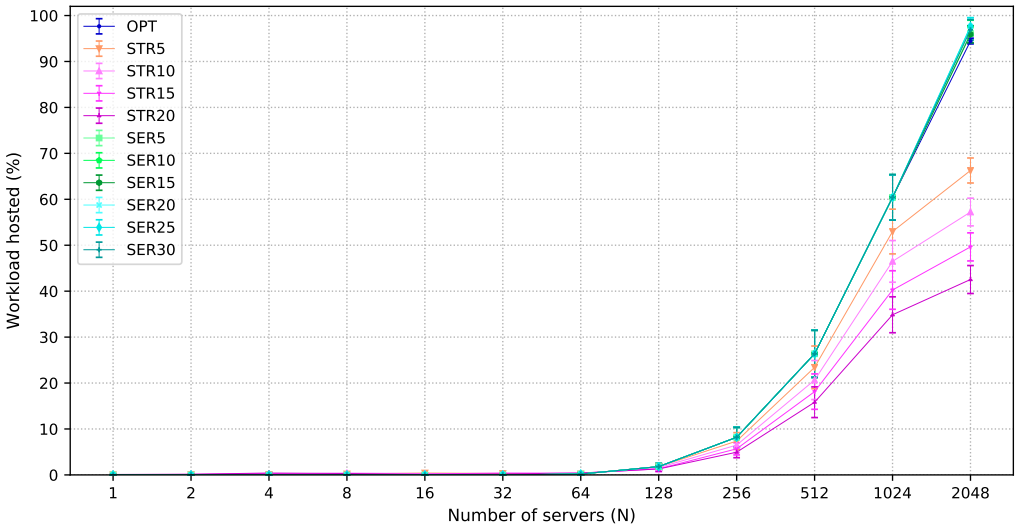


(b) $P75$.

Figure 4.6: Results for strict latency workload acceptance under $P25$ and $P75$ scenarios.

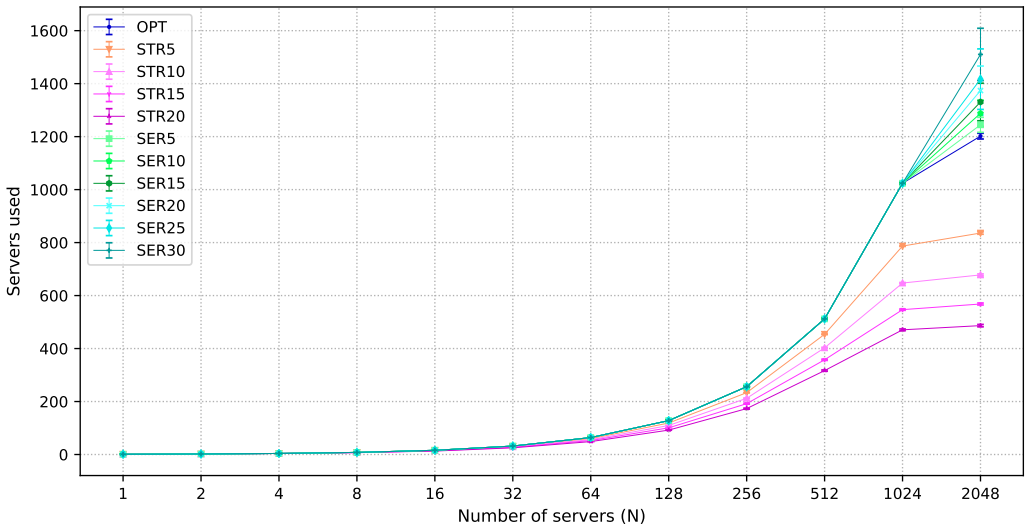


(a) $P25$.

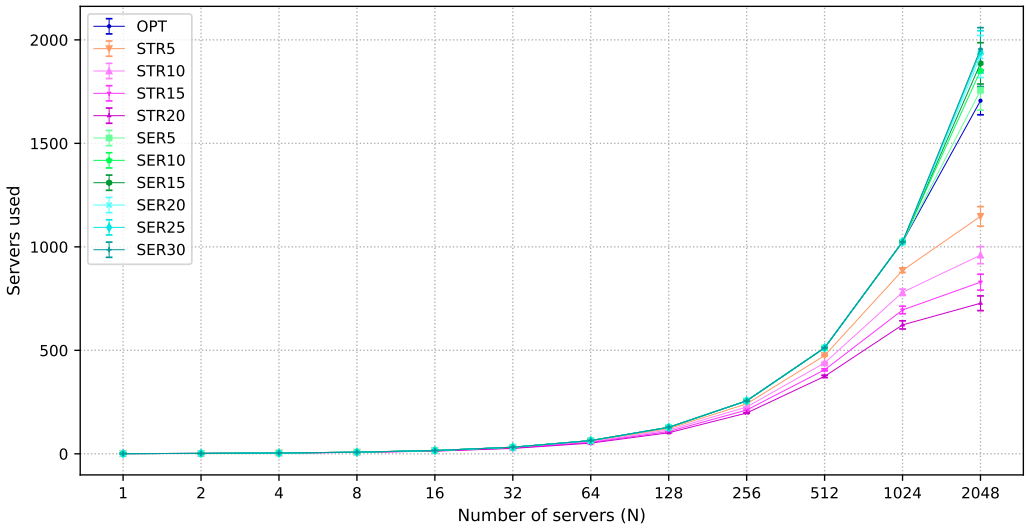


(b) $P75$.

Figure 4.7: Results for flexible latency workload acceptance ratio under $P25$ and $P75$ scenarios.



(a) $P25$.



(b) $P75$.

Figure 4.8: Results for the average number of servers employed under $P25$ and $P75$ scenarios.

ideal solution. However, results for the other solutions produced interesting results: if the provider accepts the blockage of some users, the employment of $STR5$ leads to large savings with physical servers in the infrastructure, which led to a potentially useful trade-off between service and deployment costs for the provider. The employment of $SERX$, on the other hand, is seldom useful due to the variability of demands. All results could be obtained in a reasonable time using the proposed formulation.

4.5 Conclusions

This chapter has studied the problem of locating fog node facilities in a fog-cloud scenario. The purpose is to decide on the locations where fog nodes should be deployed and the computing capacity of each node. This decision should improve the services delivered to end-users, guaranteeing that most users who depend on the fog are served, and improving the network deployed to mitigate provider costs.

The problem was solved using a multicriterial MILP model. Two types of workload were defined to simulate different applications in a fog-cloud system with the evaluation using real data of user mobility. A multi-level programming approach was employed to obtain the final solution, in which the objectives were sequentially optimized.

The proposed MILP model was also evaluated when degradation of some of the objective functions is allowed. The results show that, due to the distribution of demands in relation to time and space, infrastructure costs can be reduced if the provider is willing to accept the blockage of a limited number of users: allowing a 5 % degradation in the strict latency workload service leads to about 30 % savings in the number of servers for the infrastructure deployed. Furthermore, a substantial number of servers is needed to increase the processing of flexible demands in the fog, which significantly raises the deployment costs: allowing an increase of less than 15 % in the number of servers has little effect on the service of flexible workload demands in the fog. Results obtained with the proposed MILP model can be quickly obtained, thus the solution can be recalculated when there are changes in the network.

Chapter 5

Location of Fog Nodes for Reduction of Energy Consumption of User Devices

5.1 Overview

Deploying a fog infrastructure calls for adequate dimensioning to cope with the quality of service requirements of diverse applications. Besides supporting latency-sensitive and mobile applications, the fog should also promote energy savings for mobile devices, offering opportunities for workload offloading so that the duration of mobile devices operations can increase. Moreover, by allowing users to stay connected longer, service providers can potentially increase their profit.

Previous work focused on solutions to manage the energy consumption of processing and networking devices [31, 6]. The work in [96, 95] has also shown that mobile devices hosting heavy computational tasks involving several megabytes can reduce their energy consumption by offloading tasks to other devices. One possible approach for achieving such savings is offloading user workload to a fog node, and the proper determination of the location of fog nodes is essential to achieve such a goal. However, previous approaches to the location of computing facilities close to the end-users [112, 54, 34, 77] have not addressed the variable demands resulting from the mobility of users in conjunction with energy savings for mobile devices.

This chapter investigates the problem of how to locate fog nodes while taking into consideration the energy assumed by mobile end-user devices, i.e., having decisions on the location of fog nodes in a way which can reduce the energy consumption of mobile devices. The proposed fog node location problem is formulated as a multicriteria mixed-integer linear programming (MILP) that aims at maximizing the user acceptance ratio as well as minimizing the energy consumed by their devices. Time varying demands resulting from mobility patterns collected in a metropolitan area were used as input to model workload demands. A heuristic algorithm, designated the Energy and Demand Trade-off Algorithm (EDTA) has been introduced to solve large-scale formulations. The energy savings obtained by the EDTA, as well as the acceptance ratio of requests are very close to optimal values, as shown by extensive evaluations in scenarios with different applications and different provider budgets. Furthermore, the EDTA executes much faster

and can be executed for large metropolitan areas inhabited by millions of mobile users, while the solution of large areas cannot be obtained by the MILP model in a reasonable time.

The contribution of this chapter is two-fold. First, it models the fog node location problem using a MILP formulation that considers the energy consumed by mobile users and requests with different computing and latency requirements, representing various real applications. Second, it proposes a heuristic algorithm that produces results very close to optimum for all evaluated scenarios, yet requiring only a short execution time. The proposed solutions have been extensively evaluated and discussed. The use of fog nodes equipped with physical servers and wifi communication is a promising solution to create a device energy-aware infrastructure for the support of mobile users who do not have continuous access to energy supplies.

The remainder of this chapter is organized as follows. Section 5.2 introduces the system model adopted. Section 5.3 presents the linear programming formulation of the problem. Section 5.4 introduces the Energy and Demand Trade-off Algorithm. Section 5.5 presents the experimental setting and numerical evaluation of the proposed formulation and the EDTA. Lastly, Section 5.6 concludes this chapter.

5.2 System Model

This section details the classes of applications and the hierarchical architecture of the cloud, fog, and end-user devices layers considered in this chapter. Moreover, it presents the fog node location problem. The cloud data center hosts physical servers to process user workloads, and it can be accessed by any end-user on the Internet. Fog computing is provided by fog nodes which host physical servers and have wifi access. Nodes are located close to end-users at the network edge and provide processing for end-users in their coverage area. The lowest layers comprise mobile devices at the network edge. Such devices can access both the cloud and the fog. Access to the fog always uses wifi; the access to the cloud can be made using either cellular or wifi interfaces. If the user is in the coverage area of a fog node, the cloud is accessed using wifi, otherwise the cellular network is used. End-users use their mobile devices to request services from the infrastructure. Each request corresponds to an application class, and different classes have different processing and latency requirements.

Servers on the fog nodes and in the cloud provide end-users with processing capacity, storage, and networking for a variety of applications. Different applications have different latency and processing requirements, and these requirements determine where an application should be processed: on the mobile device hosting the application, in the fog, or in the cloud. In this chapter, four classes of application based on their latency and processing requirements were considered: fog-device, fog-only, fog-cloud, and flexible. Each class incorporates various applications with diverse requirements. Since fog-device application requires low latency and processing demands, it is processed either on the end-user device or in the fog nodes. This class includes services such as a program reading information from a sensor, processing it, and generating actions to be performed by an actuator. Fog-

Table 5.1: Characteristics of application classes.

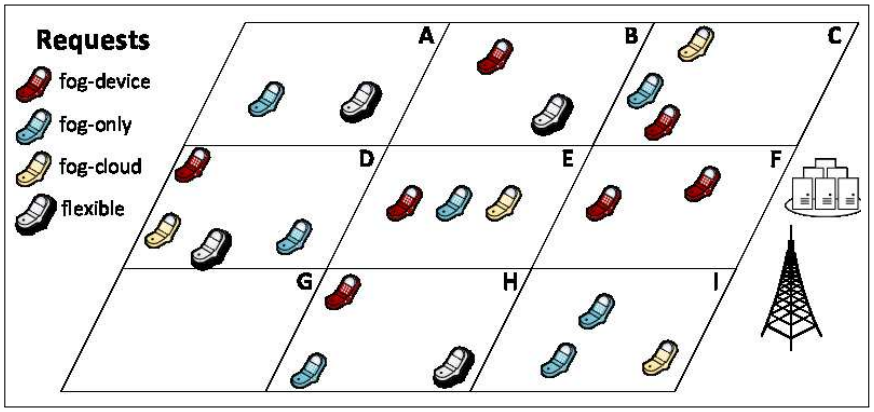
Class	Low latency	Processing in the device	Possible layers		
			Device	Fog	Cloud
fog-device	Yes	Yes	Yes	Yes	No
fog-only	Yes	No	No	Yes	No
fog-cloud	No	No	No	Yes	Yes
flexible	No	Yes	Yes	Yes	Yes

only applications are also sensitive to latency, but their heavy processing demands prevent execution on end-user devices and require more processing capacity. An example of this class is an application executing on a resource-limited smartphone that needs continuous processing of information gathered from sensors. Fog-cloud applications can be processed either in the cloud or in the fog but they cannot be executed on the end-user device. They include uses such as an online game that requires heavy computation without real-time constraints. Finally, flexible applications have no strict latency requirements and can be processed on the end-user device, in the fog, or in the cloud. Table 5.1 summarizes the requirements of different classes of application.

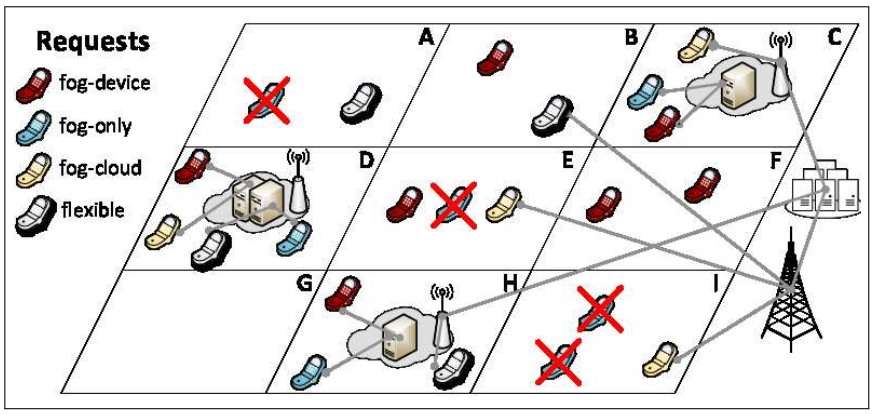
Fog-only applications have high priority for processing in the fog, and are blocked when there are not enough resources to process them on fog nodes. This results from the fact that these applications cannot be processed on the mobile device or in the cloud. Fog-device applications benefit from fog nodes since the offloading of workload leads to energy savings for end-user devices. Fog-cloud applications are always offloaded, but their execution on the fog node reduces the latency experienced by end-users, as well as the energy consumed by the device. Flexible requests are the easiest to handle since they can be executed on any layer. Execution in the fog reduces the energy consumption of end-user devices, making it advantageous for all classes of applications.

Each layer of the architecture has a different computational capacity. End-user devices process only requests made by their users. The capacity of the cloud is virtually unlimited, so that it can process all requests sent to it. Fog nodes, on the other hand, are resource-limited; their capacity is defined by the number of servers they have. One early step in the deployment of a fog-cloud infrastructure is a decision on the location of the fog nodes and their capacity. The decision made will determine whether or not end-user needs will be fulfilled. This chapter proposes a formulation for locating fog nodes so that the energy consumed by mobile devices can be reduced. While fog and cloud servers have continuous availability of energy, mobile users are constrained by their battery capacity. Thus, selecting an adequate location will result in an infrastructure that can help users stay connected to the network longer, allowing them to use more services offered by the application providers.

Energy is consumed during the data transmission, processing, as well as the idle state. The energy of data transmission depends on the number of bytes transmitted and on the network radio interface (wifi or cellular). The energy consumed in processing will depend on the power of the device and the number of cycles required to process the workload. The energy in the idle state is that consumed by the mobile device while waiting for the processing of a workload that was offloaded to either the fog or the cloud. The energy



(a) Example input.



(b) Example output.

Figure 5.1: Example of fog location decision.

consumed during idle state is constant and its value is less than that of the busy state (during processing). Before using a fog node, the mobile device checks for the fog node availability, i.e., if that node can process new workloads. If it can, data are sent to the fog for processing; otherwise, the device either sends the requested processing to the cloud or the application is blocked. The message sent to check the fog availability involves only a few bytes, and the energy consumed in this process is not accounted in this chapter.

The fog node location problem consists in deciding where fog nodes should be deployed as well as the number of physical servers at each location. The fog node location problem is exemplified in Figure 5.1. Figure 5.1a shows several devices hosting different applications. Nine regions, identified from A to I, are presented as possible locations for deploying fog nodes. If a fog node is placed in a given region, only devices in that region will be able to access it. A distant data center is represented on the right, and it can be reached via the Internet. A cellular base station can be accessed by all devices. Suppose the service provider has four servers available for deployment with each server capable of concurrently processing two requests, Figure 5.1b presents one possible decision, with fog nodes in regions C, D and H. Fog node D has two dedicated servers and is able to host up to four requests, while the other nodes can host two requests each. In such a deployment, all requests in region D will be offloaded to the fog, with only local wireless transmissions being made by the mobile devices. In regions C and H, however, only two of the three

requests will be served by the fog. Requests not served by the fog still can access the wifi access point provided by the local fog nodes. In the remaining regions, fog-device requests are processed on the end-user device, while fog-only requests are blocked, and fog-cloud requests will be executed in the cloud via the cellular network. In region B, a flexible request will use the cloud, while in region A the demand will be processed locally.

This example illustrates one possible solution to the problem, but it does not consider mobility. An end-user device can change its location over time. Therefore, the decision about the location of fog nodes must also consider demands varying over time.

5.3 Formulation

The fog node location problem is modeled by using multicriterial mixed-integer linear programming. The model optimizes the number of fog-only requests served, the global energy consumed by end-user devices, and the processing of all application classes in the fog. The notation used in this section is presented in Table 5.2. Discrete time is considered with the set $\mathcal{T}$ containing all time intervals and $\mathcal{L}$ being the set of all locations.

Different from the previous chapter, this model considers individual requests instead of the total workload. This is necessary to account the energy consumption of all end-user devices. Each request $r \in \mathcal{R}$ is generated at a specific location $l \in \mathcal{L}$ and in a specific time interval $t \in \mathcal{T}$. $\mathcal{X}_{rlt}$ indicates the location and time of each request. $\mathcal{D}_r$ and $\mathcal{C}_r$ indicate whether a request can be processed on the end-user device or in the cloud, respectively, depending on the class of application, as described in Table 5.1. $\mathcal{E}_{DEVr}$, $\mathcal{E}_{IDLEr}$, $\mathcal{E}_{WIFIr}$, and $\mathcal{E}_{CELr}$ specify the energy consumed by the end-user device hosting the request r. If request r is processed on the end-user device, $\mathcal{E}_{DEVr}$ is the energy required for that processing. Otherwise, $\mathcal{E}_{WIFIr}$ and $\mathcal{E}_{CELr}$ quantify the energy required by the mobile device for the transmission of all bytes of the request r using wifi and cellular interfaces, respectively. The energy consumed in the idle state while the device awaits for the processing of the request r offloaded workload is indicated by $\mathcal{E}_{IDLEr}$. The solution is given by variables γ_l, α_l, d_r, f_r, c_r, and w_r. γ_l indicates a fog deployment at location l, and α_l defines the number of dedicated servers at l. Each request is mapped onto a single layer, identified by the specific binary variables: d_r (device), f_r (fog), c_r (cloud accessed by cellular network), and w_r (cloud accessed by wifi). If the request r is processed in the cloud, $c_r = 1$ indicates that the end-user device uses the cellular interface to reach the cloud, whereas $w_r = 1$ indicates it accesses a wifi available at a fog node access point.

The formulation for the fog location problem is given by Equations (5.1)–(5.13):

$$\text{maximize} \sum_{r \in \mathcal{R}} (1 - \mathcal{D}_r)(1 - \mathcal{C}_r) f_r \tag{5.1}$$

$$\text{minimize} \sum_{r \in \mathcal{R}} \Big[\mathcal{E}_{DEVr} d_r + \mathcal{E}_{IDLEr}(f_r + c_r + w_r) + \mathcal{E}_{CELr} c_r + \mathcal{E}_{WIFIr}(f_r + w_r) \Big] \tag{5.2}$$

$$\text{maximize} \sum_{r \in \mathcal{R}} f_r \tag{5.3}$$

Table 5.2: Notation used in the fog node location problem formulation.

Input parameters	
Notation	**Description**
Common parameters	
N	Maximum number of servers to be deployed
K	Capacity of a single server
$\mathcal{L}$	Set of locations where a fog node can be placed: $\mathcal{L} = \{1, 2, \ldots, L\}$
$\mathcal{T}$	Set of discrete time intervals: $\mathcal{T} = \{1, 2, \ldots, T\}$
Requests	
$\mathcal{R}$	Set of requests: $\mathcal{R} = \{1, 2, \ldots, R\}$
$\mathcal{D}_r, r \in \mathcal{R}$	Binary variable, indicates whether request r can be processed on the mobile device
$\mathcal{C}_r, r \in \mathcal{R}$	Binary variable, indicates whether request r can be processed in the cloud
$\mathcal{X}_{rlt}, r \in \mathcal{R}, l \in \mathcal{L}, t \in \mathcal{T}$	Binary variable, $\mathcal{X}_{rlt} = 1$ if the request r is made in time slot t when the device hosting r is at location l, otherwise, $\mathcal{X}_{rlt} = 0$
Energy consumption	
$\mathcal{E}_{DEVr}, r \in \mathcal{R}$	Energy consumed in the processing of request r on the mobile device
$\mathcal{E}_{IDLEr}, r \in \mathcal{R}$	Energy consumed by the device while it is in the idle state awaiting processing of request r either in the fog or in the cloud
$\mathcal{E}_{WIFIr}, r \in \mathcal{R}$	Energy consumed by the device hosting r for data transmission using a wifi interface
$\mathcal{E}_{CELr}, r \in \mathcal{R}$	Energy consumed by the device hosting r for data transmission using a cellular interface
Decision variables	
Notation	**Description**
$\gamma_l, l \in \mathcal{L}$	Binary variable, $\gamma_l = 1$ if a fog node is created at location l, otherwise $\gamma_l = 0$.
$\alpha_l, l \in \mathcal{L}$	The number of fog servers at location $l \in \mathcal{L}$ if $\gamma_l = 1$. If $\gamma_l = 0$, then $\alpha_l = 0$.
$d_r, r \in \mathcal{R}$	Binary variable indicating whether request r is processed on the mobile device ($d_r = 1$) or not ($d_r = 0$).
$f_r, r \in \mathcal{R}$	Binary variable indicating whether request r is processed in the fog ($f_r = 1$) or not ($f_r = 0$).
$c_r, r \in \mathcal{R}$	Binary variable indicating whether request r accesses the cloud using a cellular network interface ($c_r = 1$) or not ($c_r = 0$).
$w_r, r \in \mathcal{R}$	Binary variable indicating whether request r accesses the cloud using a wireless network interface ($w_r = 1$) or not ($w_r = 0$).

$$\sum_{l \in \mathcal{L}} \alpha_l \leq N \tag{5.4}$$

$$\gamma_l \leq \alpha_l, \forall l \in \mathcal{L} \tag{5.5}$$

$$\gamma_l \geq \frac{\alpha_l}{N}, \forall l \in \mathcal{L} \tag{5.6}$$

$$d_r + f_r + c_r + w_r \begin{cases} \leq 1 & \forall r \in \mathcal{R} \mid \mathcal{D}_r + \mathcal{C}_r = 0 \\ = 1 & \forall r \in \mathcal{R} \mid \mathcal{D}_r + \mathcal{C}_r \geq 1 \end{cases} \tag{5.7}$$

$$c_r + w_r \leq \mathcal{C}_r, \forall r \in \mathcal{R} \tag{5.8}$$

$$d_r \leq \mathcal{D}_r, \forall r \in \mathcal{R} \tag{5.9}$$

$$c_r + \gamma_l \leq 1, \forall (r, l, t) \in \mathcal{R} \times \mathcal{L} \times \mathcal{T} \mid \mathcal{X}_{rlt} = 1 \tag{5.10}$$

$$w_r - \gamma_l \leq 0, \forall (r, l, t) \in \mathcal{R} \times \mathcal{L} \times \mathcal{T} \mid \mathcal{X}_{rlt} = 1 \tag{5.11}$$

$$\sum_{r \in \mathcal{R} | \mathcal{X}_{rlt} = 1} f_r \leq \alpha_l \cdot K, \forall l \in \mathcal{L}, \forall t \in \mathcal{T} \tag{5.12}$$

$$d_r, f_r, c_r, w_r, \gamma_l \in \{0, 1\}, \alpha_l \geq 0, \forall r \in \mathcal{R}, \forall l \in \mathcal{L} \tag{5.13}$$

The multi-objective optimization model has three objective functions. Equation (5.1) defines the first objective function, which seeks to maximize the number of fog-only applications processed.

The second objective function (Equation (5.2)) minimizes the total energy consumption of the mobile end-user devices. Equation (5.2) represents the sum of the energy consumed by all requests in $\mathcal{R}$, and it is divided into four components. The first component is the energy consumed in the mobile device processing. The second component is that consumed in the idle state while the end-user device awaits a response. The third component is the energy consumption due to the use of the cellular network interface by requests processed in the cloud, and the final component is the energy consumption due to the use of the wifi interface by requests processed in the fog (f_r) or in the cloud (w_r).

Equation (5.3) defines the third objective, which is the maximization of the number of requests processed in the fog, regardless of their application class. By optimizing this objective, latency is improved for the end-users, since this workload can be processed in the fog rather than in the cloud.

The first constraint (Equation (5.4)) limits the total number of servers employed on all fog nodes to N. Constraints given by Equations (5.5) and (5.6) guarantee that if $\alpha_l \geq 1$, then $\gamma_l = 1$, otherwise $\gamma_l = 0$. Equation (5.7) guarantees that all requests are processed. For fog-only applications, the sum is less than or equal to 1 since such requests can be

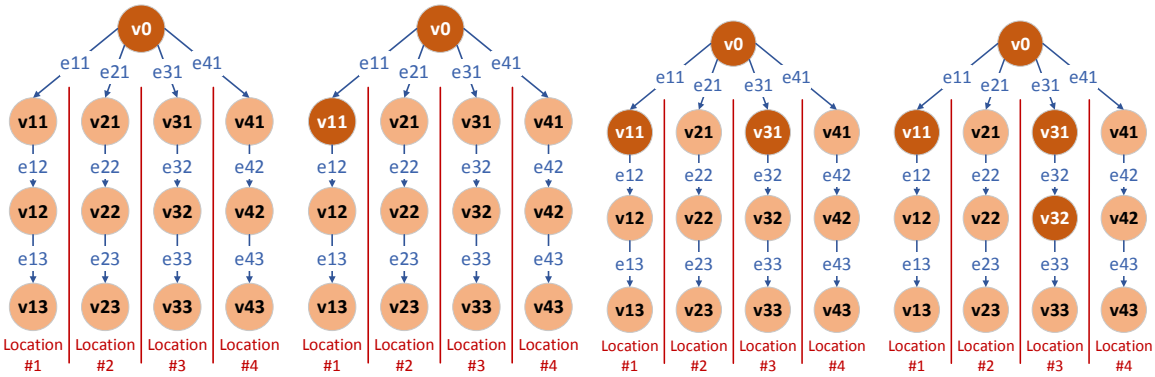
blocked. For the other requests, the sum is equal to one, guaranteeing that they will be executed. Constraints given by Equations (5.8) and (5.9) guarantee that requests executed in the cloud or on the mobile device are applications which can be processed on these layers. Since any request can be processed in the fog, a similar constraint for the fog is not necessary. Equation (5.10) assures that a request processed in the cloud uses a cellular network interface if there is no fog node in the area from which the request originates. In a similar fashion, Equation (5.11) assures the use of wifi for requests processed in the cloud from a location hosting a fog node. To avoid overloading the fog nodes, expression (5.12) limits the number of hosted requests processed on each fog node to the node capacity. Finally, Equation (5.13) establishes the domain of the decision variables.

The multi-level programming approach, reviewed in Subsection 2.4, was employed to solve the formulation. In the fog location problem, the service of fog-only applications is crucial, since they are blocked if no resources are available. After guaranteeing the execution of fog-only requests, energy consumption is analyzed, and finally the occupation of fog nodes can be improved. Thus, the multi-level programming approach employed Equation (5.1) as the main objective, Equation (5.2) as secondary one, and Equation (5.3) as the final one. The order of the objective functions in the multi-level programming approach is crucial for the solution: if the energy consumption were optimized first, fog-only applications might not necessarily be processed, since they involve greater energy consumption. In other words, although this chapter focuses on energy consumption, Equation (5.2), which optimizes that consumption, is not the main objective, since the blocking of users must always be avoided. Saving energy is useless if fog-only applications are blocked, thus Equation (5.1) must be considered to be the main objective. Equation (5.3) is the last objective.

5.4 Energy and Demand Trade-off Algorithm

The fog node location problem is a network design problem, typically solved off-line. In the real scenario considered in this thesis, hundreds to thousands of locations exist to serve millions of users. In an attempt to solve the formulation proposed for large scenarios, the formulation in the previous section was coded in the Gurobi Optimizer. However, the solver did not finish the execution of the model, even after long execution times, and it eventually crashed due to the high demand for main memory. Consequently, a heuristic algorithm has been proposed, denominated Energy and Demand Trade-off Algorithm (EDTA).

EDTA attempts to achieve a trade-off between the processing of fog-only requests and the energy consumed by end-user devices. These are the objectives represented by Equations (5.1) and (5.2), respectively. EDTA also favors the fog for the execution of all requests to achieve the third objective (Expression (5.3)). To achieve these goals, EDTA calculates the gain obtained by the inclusion of a new dedicated server at a candidate location (metric $\mathcal{M}$). This gain is a function of the proportion of fog-only applications accepted and of the proportion of energy saved as a result of the inclusion of a physical server in a fog node. Thus, based on this metric, servers are assigned to fog nodes in a



(a) Initial graph. (b) Selection of the first server. (c) Selection of the second server. (d) Selection of the third server.

Figure 5.2: Example of EDTA graph.

greedy manner. The EDTA operation employs an acyclic directed graph, with vertices representing the assignment of a new server to a fog node, and the weight of the edges is calculated by $\mathcal{M}$. The original graph represents all possible decisions, and a subgraph is obtained representing a single solution.

EDTA is described in Algorithm 5.1, and its operation is illustrated in Figure 5.2. The first step of the algorithm is the creation of the directed graph $G = \{V, E\}$. G is a tree with root v_0; the remaining $L \cdot N$ vertices are $v_{ij}, i \in \mathcal{L}, j \in \{1, \dots N\}$. There is an edge e_{i1} from v_0 to each of the vertices $v_{i1}, i \in \mathcal{L}$. Additionally, for all $v_{ij}, i \in \mathcal{L}, j \in \{2, \dots N\}$, there is an edge e_{ij} connecting $v_{i(j-1)}$ to v_{ij}. The graph resulting from $L = 4$ and $N = 3$ is illustrated in Figure 5.2a.

Each edge e_{ij} represents the decision to employ the j^{th} server at location i, and the gain associated with this decision is calculated by the metric $\mathcal{M}$. $\mathcal{M}$ is defined as $\mathcal{M} = \kappa\mathcal{F} + \lambda\mathcal{N}$, with κ and λ being parameters that define the importance of the fog-only service and energy consumption, respectively, $0 \geq \kappa \geq 1$, $0 \geq \lambda \geq 1$ and $\kappa + \lambda = 1$. Furthermore, $\mathcal{F}$ and $\mathcal{N}$ are the gains from deploying a new server in terms of fog-only service and energy consumption, respectively. Such gains are calculated as follows. The number of fog-only requests T^f at all locations over all time is calculated. The sum T^e of the energy consumption of the remaining requests (fog-device, fog-cloud, and flexible) is calculated for all requests considering the absence of fog nodes. Then, for each edge e_{ij}, the algorithm calculates T^f_{ij}, the number of fog-only requests served as a result of the employment of the j^{th} server at the fog node at location i. Similarly, T^e_{ij} is calculated as the energy saved by the employment of the j^{th} server at the fog node at location i. Finally, $\mathcal{F}$ and $\mathcal{N}$ are calculated for each edge as $\mathcal{F} = T^f_{ij}/T^f$ and $\mathcal{N} = T^e_{ij}/T^e$, allowing the calculation of $\mathcal{M}$ for all edges $e \in E$.

The rationale behind the computation of the $\mathcal{M}$ value is the trade-off between the two main objectives. Once all vertices and edges are added to the graph, the algorithm uses the edge weights to greedily add servers to the locations. The generation of the graph $G = (V, E)$ is the first step of Algorithm 5.1 (Line 1). Then the algorithm builds a subgraph G' to represent the solution, initialized in Lines 2 to 5. During this initialization, $\mathcal{M}$ is calculated for all edges in Line 4. This computation requires the calculation of

maximum energy consumption at the end-user device for all requests. Then the energy savings and the number of fog-only requests processed due to the inclusion of the new server are calculated. To realize this calculation, requests are assigned to each server, considering that a new server will first process fog-only and fog-device applications, and then fog-cloud and flexible requests. In this way, the third objective (Expression (5.3)) is considered in the EDTA.

Input: N,R,$\mathcal{L}$,$\mathcal{T}$,f_{lt}
Output: Array with the number of servers in each location
1 builds $G = \{V, E\}$;
2 $V' \leftarrow \{v_0\}$;
3 $E' \leftarrow \emptyset$;
4 calculates $\mathcal{M}$ for all $e \in E$;
5 $G' = \{V', E'\}$;
6 $n \leftarrow N$;
7 **while** $n > 0$ **do**
8 $possibleEdges \leftarrow \{(v_x, v_y) \in E | v_x \in V' \wedge v_y \notin V'\}$;
9 $(v_w, v_z) \leftarrow$ edge with minimum weight from $possibleEdges$;
10 **if** *weight of* $(v_w, v_z) > 0$ **then**
11 $E' \leftarrow E' \cup (v_w, v_z)$;
12 $V' \leftarrow V' \cup (v_z)$;
13 $n \leftarrow n - 1$;
14 **else**
15 break ;
16 **return** *List with the number of servers obtained from the leaves of* G';

Algorithm 5.1: EDTA.

The algorithm greedily assigns servers to fog nodes while resources are available and edges with a higher $\mathcal{M}$ value (Lines 7 to 15). In each iteration, all edges that connect vertices in G' to vertices in $G \setminus G'$ are obtained (Line 8). For example, in Figure 5.2a, these edges are v_{11}, v_{21}, v_{31}, and v_{41}. In each iteration, G' represents the current solution while the obtained edges represent the possible servers to be deployed. Then, the edge that has the highest $\mathcal{M}$ value is chosen (Lines 11 to 13). In Figures 5.2a and 5.2b, a server is assigned to the first fog node, decreasing the number of available servers N. If there is no edge leading to a vertex with a non-negative $\mathcal{M}$ value, the iteration stops (Line 15). By using v_o as the root of G', at the end of the execution, the leaves of G' represent the solution. Let $\{v_{x_1y_1}, v_{x_2y_2}, \ldots, v_{x_Uy_U}\}$ be the set of U leaves. The solution is to use y_i servers for the location x_i, $1 \leq i \leq U$. In Figure 5.2d, the leaves are v_{11} and v_{32}, resulting in two fog nodes, one in the first location with one server, and the other in the third location hosting 2 servers. A location not represented by a leaf indicates the absence of fog nodes at that location. The time complexity of EDTA is $\mathcal{O}(N \cdot L + N \cdot R)$. The creation of the graph (Line 1) analyzes the deployment of up to N servers over L locations, thus requiring $\mathcal{O}(L \cdot N)$. In Line 4, all $L \cdot N$ edges are visited. The inclusion of a server evaluates all requests it can serve, up to R requests, resulting in a complexity of $\mathcal{O}(N \cdot L + N \cdot R)$ in Line 4. The initialization of G' and n (Lines 2 to 6) is $\mathcal{O}(1)$. Then, the while loop is executed N times, with the most demanding operations performed in Lines 8 and 9, which is the selection of L edges with minimum cost. If the leaves of G' are kept in an array with L positions, the selection costs $\mathcal{O}(L)$ per iteration, or $\mathcal{O}(L \cdot N)$

for all iterations. By using such an array, the return value (Line 16) is easily obtained with $\mathcal{O}(L)$. Therefore, the most complex operations in EDTA are $\mathcal{O}(N \cdot L + N \cdot R)$.

5.5 Performance Evaluation

The MILP formulation presented in Section 5.3 was coded using the Gurobi Optimizer solver, and EDTA was coded in Python. The results obtained by these implementations are presented in this section. Subsection 5.5.1 presents the energy model employed for end-user devices. Subsection 5.5.2 describes the workload model, and Subsection 5.5.3 the application model. Finally, Subsection 5.5.4 discusses numerical results.

5.5.1 Energy model

The evaluation considers end-user devices as smartphones, and two operations accounted for their energy consumption: data transmission and processing. There are, however, other components involved in the energy consumption of a smartphone, such as the phone screen, use of Global Positioning System (GPS) sensors, and the operating system, but these components are not affected by the location of a fog node, and, consequently, their energy consumption is ignored in this evaluation.

The energy consumed in the transmission depends on the technology employed. In this work, two technologies are considered, IEEE 802.11g for Wi-Fi and Universal Mobile Telecommunication System (UMTS) for cellular networks. End-user devices are modeled as Samsung Galaxy S3 smartphone with an Exynos 4412 processor [96], and are able to process 14000 million instructions per second.

The energy consumed by using the wifi interface depends on the number of packets transmitted [40]. The wifi energy consumption in transmissions, E_W, is modeled as $E_W = P_W T_W$, where P_W is the power during transmission and T_W the data transfer time. The value of P_W used is a measured value described in [22]. T_W is calculated as $T_W = N(T_P + T_{ACK} + SIFS) + B + DIFS$, where N is the number of packets to be transmitted, T_P the individual packet transmission time, and T_{ACK} the transmission time for acknowledgments. Short Inter-frame Space (SIFS) and Distributed Coordination Function (DIFS) are intervals defined by the IEEE 802.11 standard. B is the backoff time to avoid contention.

The energy consumed for cellular communications is modeled as in [50]. The time T_C spent in communications is divided into two phases, the promotional phase, when the device listens to the channel before sending or receiving data, and the transmission phase, when bytes are sent or received. T_C is calculated as $T_C = T_{PR} + (D \times 8)/S$, where T_{PR} is the promotional time, D the number of bytes to be transmitted, and S the data rate. The energy consumed by a transmission is calculated as $E_C = P_{PR} T_{PR} + P_C T_C$, where P_{PR} is the power during the promotional phase. The value of P_C is based on measurement values reported in [22].

Finally, the energy consumed by processing and during the idle state is calculated as $E_D = P^* \times T_{request}$, where P^* is the power of the device and $T_{request}$ the time required

Table 5.3: Values adopted for energy model parameters.

Parameter	**Value**
$SIFS$	10 μs
B	1023 μs
$DIFS$	50 μs
T_{PR}	661.6 ms
S	2 Mbps
P_{PR}	659.4 mW
P_W	1264 mW
P_C	1543 mW
P_{busy}	2845 mW
P_{idle}	666 mW

to process the request. If the end-user device is processing the workload, $P^* = P_{busy}$, otherwise, if the device remains idle waiting for a fog or cloud response, $P^* = P_{idle}$.

The values employed for all parameters are shown in Table 5.3, and they are based on the measurement values in [40, 22, 50]. E_W and E_C are used as the input values of variables $\mathcal{E}_{WIFIr}$ and $\mathcal{E}_{CELr}$ in the formulation, and E_D is used for the input values $\mathcal{E}_{DEVr}$ and $\mathcal{E}_{IDLEr}$. The number of bytes to be transmitted and the processing time of tasks depend on the applications considered, as described in Subsection 5.5.3. Finally, fog and cloud servers are modeled as having Intel Core i7-7500U processors which can process 53840 million instructions per second.

5.5.2 Workload

Locations and request sets are taken from a data set [8] described in Section 3.5. In this chapter, the available locations ($\mathcal{L}$) are the cells where there is at least one BS, and users on the remaining cells were not considered due to the long distances that prevent the employment of wifi. This resulted in 895 locations. The work in this chapter considers individual request and, therefore a constant Z is used to obtain the number of requests in each cell, as explained in Section 3.5.

The evaluation considered different number of locations (L), $L = 100$ and $L = 895$. For $L = 895$, the whole metropolitan area, some scenarios accounted more than one million requests, which made it impossible for the MILP model to produce solutions in a reasonable time. On the other hand, for $L = 100$, about three hundred thousand requests are made during a 24-hour interval. Thus, results for $L = 100$ are presented for both MILP model and heuristic, but only results produced by the EDTA heuristic are presented for $L = 895$.

Other variables in the model had to be set: the capacity K of a server was fixed, and N was varied to simulate different budgets. The proportion of fog and cloud requests was varied with the scenarios adopted are described in Subsection 5.5.3. Table 5.4 presents the values used for input variables.

Table 5.4: Values adopted for the workload parameters.

Parameter	Values
N	1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048
K	25
Z	0.05
$\mathcal{L}$	$\mathcal{L} = \{1, 2, ..., L\}$, $L = 100$ and $L = 895$
$\mathcal{T}$	$\mathcal{T} = \{1, 2, ..., T\}$, each $t \in \mathcal{T}$ represents a ten minute time slot. $T = 144$ for a 24h planning interval

Table 5.5: Scenarios evaluated according to the proportion of application classes and request input sizes.

Scenario	Proportion of requests (%)				Request input sizes
	fog-device	fog-only	fog-cloud	flexible	
eq	25	25	25	25	100 Kb, 1 Mb, 10 Mb, 100 Mb
fd	70	10	10	10	
fo	10	70	10	10	
fc	10	10	70	10	
fl	10	10	10	70	

5.5.3 Application model

Four classes of application are considered, characterized by their latency and processing requirements, as introduced in Section 5.2. To simulate these applications, the required number of cycles and the number of bytes to be transmitted were modeled according to known applications [73].

Requests which cannot be executed on end-user devices have high processing demands. Consequently, it is considered that fog-only and fog-cloud applications require 44500 cycles per byte, whereas fog-device and flexible applications, though intensive in terms of processing, can be processed on the end-user device, so that they require only 8900 cycles per byte. As noted in previous studies [96, 73, 28], offloading of workloads requiring only a small number of cycles per byte is not energy-efficient. This happens because this offloading does not account for a significant amount of energy for processing on the device, therefore, low intensive applications are not considered in this chapter.

The amount of data transferred by each request to the fog or to the cloud is varied to evaluate different request input sizes. The values employed were 100 Kb, 1 Mb, 10 Mb and 100 Mb, a large range in order of magnitude to evaluate different traffic patterns. The proportion of each application class was also varied to assess the solutions under different scenarios. All traffic scenarios (eq, fo, fd, fc, and fl) and request input sizes (100 Kb, 1 Mb, 10 Mb, and 100 Mb) were evaluated. Table 5.5 summarizes the traffic scenarios.

5.5.4 Numerical results

In this subsection, the performance of the EDTA is assessed and the results compared to those produced by the MILP model, herein identified as the OPT. The EDTA was tested for various combinations of values of κ and λ, and the combination $\kappa = 0.5$ and $\lambda = 0.5$

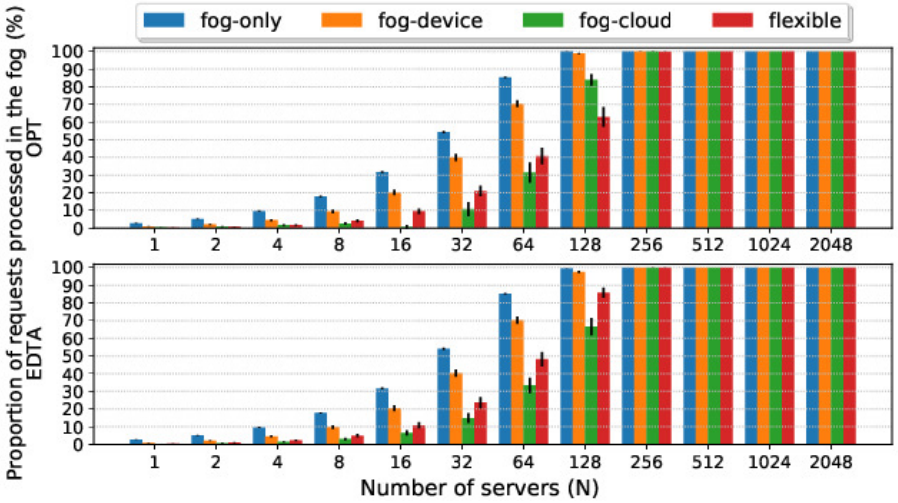


Figure 5.3: Acceptance ratio of application classes considered.

produced results close to those obtained by OPT; this combination was employed in the evaluation. This subsection is divided into three parts. First, results concerning different request input sizes are discussed, showing trends and highlighting possible energy savings. Second, results produced under different traffic scenarios are evaluated. Finally, the feasibility of the EDTA for large-scale inputs is shown. Two metrics are evaluated: energy consumption and the acceptance ratio for processing in the fog. Energy consumption is the energy consumed by all mobile devices. The acceptance ratio is the proportion of each application class served at fog nodes. 95 % confidence intervals of the mean values obtained by replicated simulations are displayed in the graphics.

The first analysis discusses general trends produced by OPT and EDTA in the scenario in which the requests are equally distributed among all application classes (eq). The energy consumption is analyzed for all request input sizes. The acceptance ratio is presented only for an input of 100 Mb since the number of bytes to be transmitted does not affect significantly the acceptance ratio. This happens because the fog capacity puts a hard limit on the number of concurrent tasks, and the available bandwidth is large enough for this limited number of tasks, regardless of the largest possible input size per task.

The acceptance ratio is displayed in Figure 5.3. The number of applications processed in the fog increases as a function of the number of servers (N) up to $N = 128$. The proportion of the fog-only class processed in the fog is greater than that of the other classes due to the main objective chosen (Equation (5.1)). Similarly, the proportion of the fog-device applications processed in the fog is greater than that of fog-cloud and flexible classes because the execution of the fog-device workload in the fog leads to energy savings when compared to the execution of this workload on the mobile device (the second objective chosen, Equation (5.2)). For all values of N, the EDTA produces results very close to those given by the OPT, which shows that the EDTA manages to deploy servers in the locations from which the demand comes, thus reducing the energy consumption of end-user devices.

Figures 5.4–5.7 display the energy consumption under the eq scenario for various

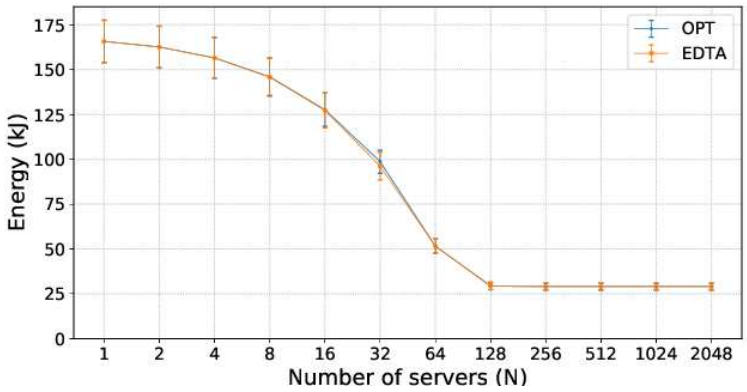


Figure 5.4: Energy consumption for the *eq* scenario and 100 Kb.

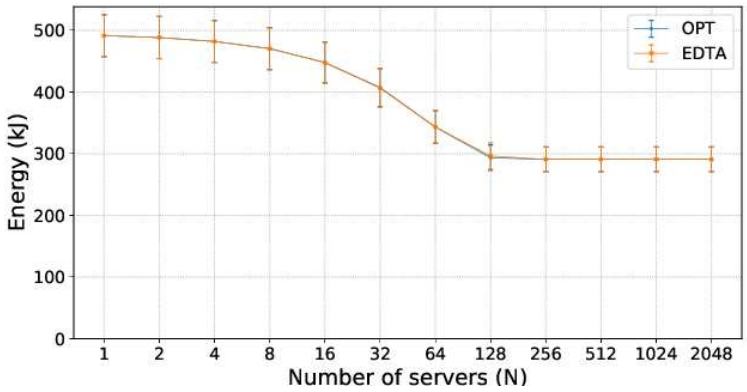


Figure 5.5: Energy consumption for the *eq* scenario and 1 Mb.

different values of request input size. The availability of wifi at all locations (for $N \geq 128$) reduces the energy consumption since users do not need to use the cellular interface. For input of 1 Mb, 10 Mb, and 100 Mb, the deployment of fog nodes at all locations reduced the energy consumption by about 40 % in comparison with a scenario with a single fog node ($N = 1$). For input of 100 Kb, however, reduction was nearly 80 % due to the fact that when using the cellular interface, the energy consumed in the promotional phase is greater than that in the transmission phase for 100 Kb. For $N \geq 128$, no energy is consumed in the promotional phase, so that about 80 % of the consumption is eliminated by the use of wifi. The curves of the energy consumption produced by the EDTA overlap those of the OPT for all sizes of input and values of N, which shows that the use of the EDTA algorithm promotes excellent energy savings, preventing battery drain and allowing mobile users to take advantage of more services from the application service provider.

To assess the efficiency of the EDTA, the following part of this subsection discusses results obtained for different scenarios. The acceptance ratio and energy consumption values produced in the fd, fc, and fl scenarios reveal a trend similar to that obtained for *eq*. However, in the fo scenario, the energy consumption tends to differ from that in the fd, fc, and fl scenarios because, in the former scenario, 70 % of the workload is fog-only, whereas, in the other scenarios, this accounts for at most 25 %. Since fog-only requests are the only ones that can be blocked, the service provided for these requests impacts more significantly on the metrics evaluated. From now on, the analysis of the effectiveness of

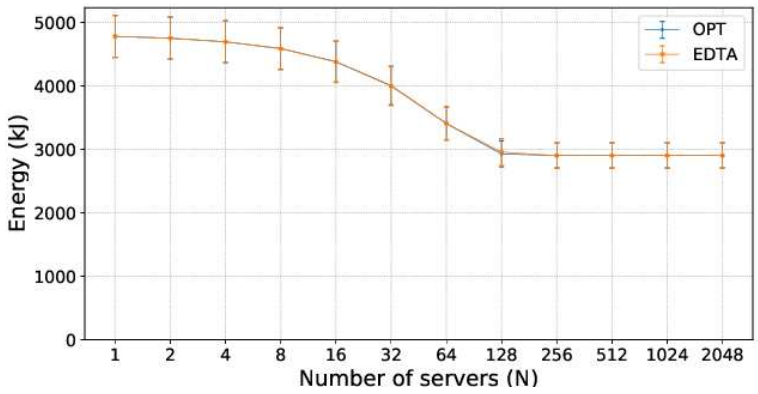


Figure 5.6: Energy consumption for the *eq* scenario and 10 Mb.

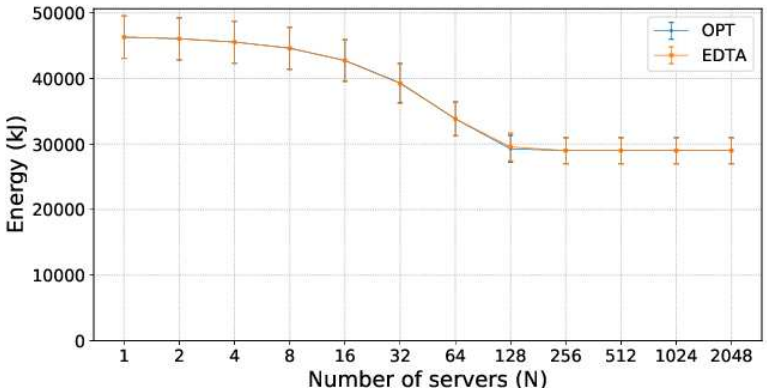
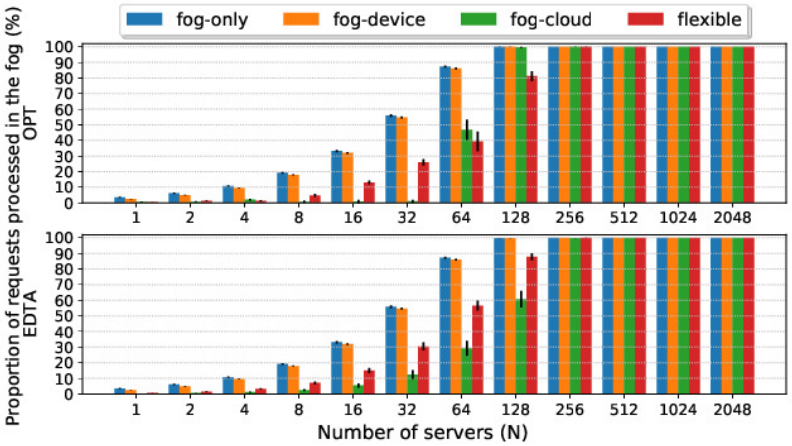
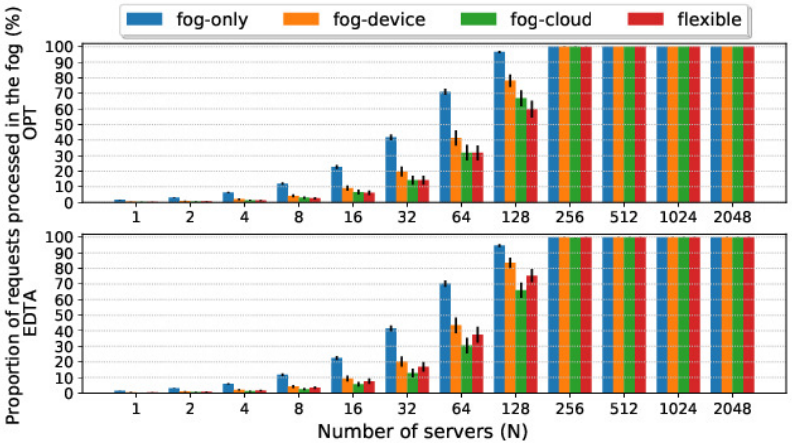


Figure 5.7: Energy consumption for the *eq* scenario and 100 Mb.



(a) fl.



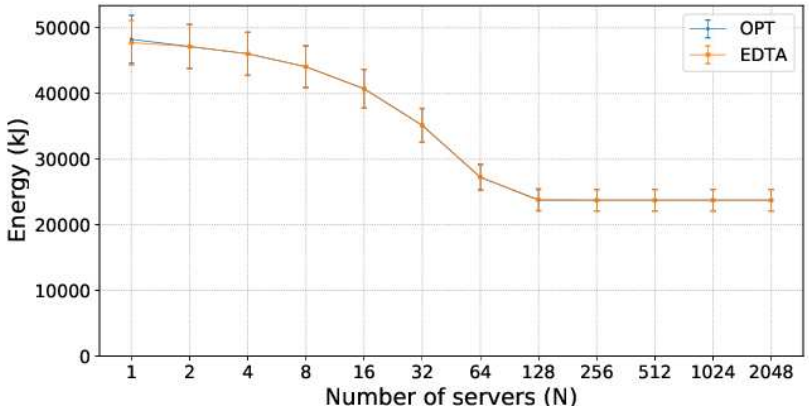
(b) fo.

Figure 5.8: Acceptance ratio of application classes considered.

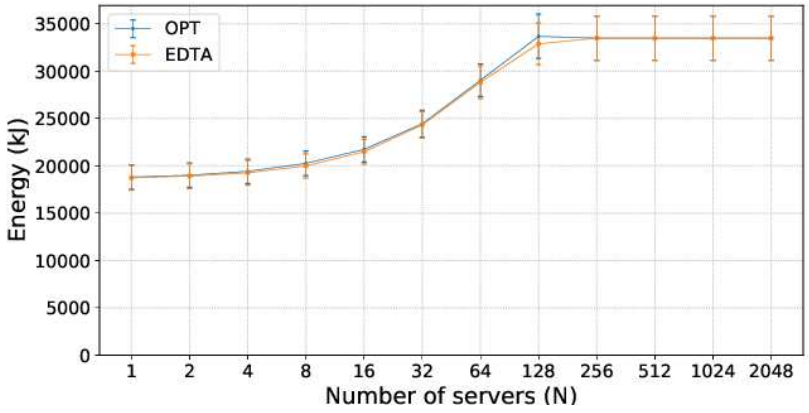
the EDTA will be presented for the fl and fo scenarios since they represent these two trends.

The acceptance ratio of workloads in the fl and fo scenarios is shown in Figure 5.8, and the energy consumption is displayed in Figure 5.9. Similar to what happens in the eq scenario, the energy consumption decreases as a function of the number of servers (N) in the fl scenario; however, it increases in the fo scenario. This apparent lack of correlation is justified by the greater proportion of fog-only requests in the fo scenario. In the fl scenario, larger values of N allow the processing of most requests (fog-device, fog-cloud, and flexible) in the fog instead of in the cloud or on the device, thus reducing the energy consumption. This reduction also takes place in the fo scenario, but the energy consumption increases with the number of fog-only requests processed. Since most requests are fog-only in the fo scenario, this increase in the energy consumption is greater than the reduction promoted by the fog for the other requests, thus the consumption increases as a function of N.

The results discussed so far are related to a limited number of locations ($L = 100$). However, the whole metropolitan area from the data set [8] requires the analysis of a much larger number of locations, $L = 895$. Solutions for this entire area obtained by the OPT

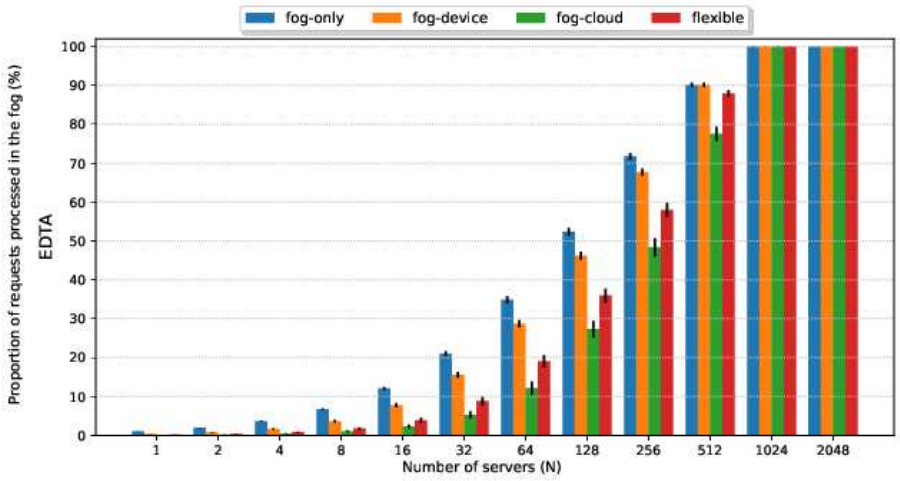


(a) fl.

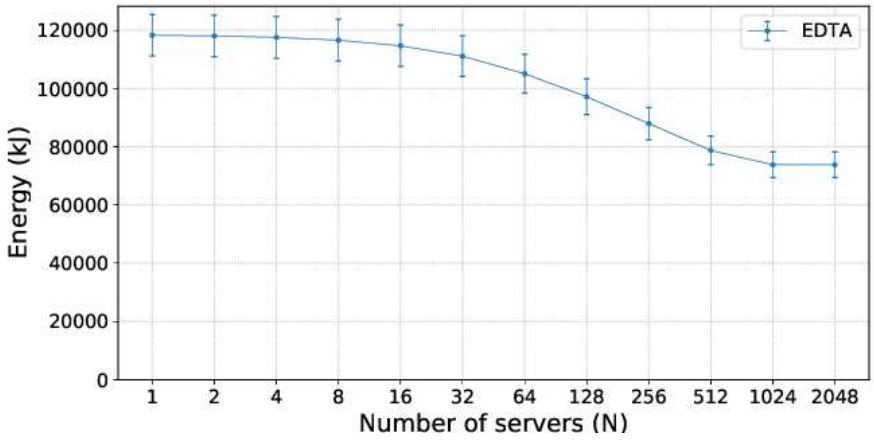


(b) fo.

Figure 5.9: Energy consumption for the fo and fl scenarios and request input size 100 Mb.



(a) Acceptance of workloads.



(b) Energy consumption.

Figure 5.10: Results for EDTA for $L = 895$ and *eq* scenario.

require several days of computation for each point in the graphics and, in most cases, are not achieved due to the large amount of memory required by the solver. On the other hand, the EDTA obtains results much faster, as displayed in Figure 5.10. More servers were required ($N \geq 1024$) to satisfy the demands of a larger number of mobile users when compared to $L = 100$. The ratio between the energy consumption for $N = 1024$ and $N = 1$ is approximately the same as that obtained by the OPT for $L = 100$. For the same scenario and different number of locations, this ratio will not vary significantly, i.e., the employment of fog nodes cannot decrease the energy consumption beyond a certain limit. Consequently, there is evidence to suggest that the EDTA leads to energy savings close to the maximum achievable ones when executed for the whole metropolitan area ($L = 895$).

In summary, this subsection presented the results obtained by both the OPT and the EDTA for several scenarios with different proportions of application classes and different provider budgets (N). For all evaluated scenarios, the EDTA produces results close to the optimum. EDTA also produces solutions quickly and is able to evaluate input with a large

number of locations and mobile users. EDTA also manages to increase the fog utilization for all application classes as does OPT, which improves the latency delivered to end-users. By employing the EDTA, fog nodes are properly deployed in a fog-cloud infrastructure, thus allowing the processing of fog-only applications as well as a reduction in the energy consumption for the other classes of applications, which reduces the battery drain of mobile devices, encouraging end-users to stay connected longer to the infrastructure and leading to more profit for service providers.

5.6 Conclusions

This chapter has addressed the problem of locating fog nodes in fog-cloud infrastructures in order to minimize the energy consumed by mobile devices. By planning the infrastructure in this manner, battery drain is reduced, thus allowing end-users to stay connected to the network longer. By allowing that, providers obtain higher profits. Therefore, the employment of energy-aware fog nodes can be an advantage for both users and providers.

The problem in this chapter was modeled using a multicriterial MILP model, with the solution obtained by a solver using the multi-level programming approach to deal with the multiple criteria. For a large number of input variables, however, the solver requires an unreasonable time to reach a solution. Thus, a heuristic called Energy and Demand Trade-off Algorithm has been proposed to produce solutions more rapidly.

The solutions were evaluated by using real traces of mobile users in a metropolitan area as input, and different classes of applications in terms of processing and latency. The results obtained for various traffic scenarios show that the heuristic solution produced results very similar to those obtained by the MILP model, selecting locations for the deployment of fog nodes so that requests which depend on the fog are processed and fog nodes benefit the other requests by reducing the energy consumed by the mobile devices as well as the delay to end-users. The EDTA was shown to be feasible for large inputs with millions of variables. The results obtained in this chapter can serve as the basis for future deployments of fog-cloud infrastructures for smart cities, so that an adequate infrastructure can be created for mobile devices with limited energy access.

Part III

Aerial Infrastructure

Chapter 6

Location of Fixed and UAV-based Fog Nodes

6.1 Overview

Fog nodes are usually deployed as fixed nodes in different locations to support the computational needs of local users and support the strict latency requirements of applications. However, nodes in a fixed infrastructure may need to be over-dimensioned to cope with variable processing demands. One possible solution to alleviate such a problem is the employment of mobile fog nodes that can process end-user demands in different locations. Unmanned aerial vehicles (UAVs) have been considered for integration into cellular networks to serve as base stations [75, 9], allowing providers to expand their coverage area in case of occasional demands or failure of the terrestrial infrastructure. However, the employment of UAVs as fog nodes is still its infancy.

This chapter aims at assessing the advantages of employing UAVs as fog nodes for dealing with variable workload demands generated by mobile users. In this perspective, this chapter studies the problem of where to locate UAVs as fog nodes (the fog node location problem) in a metropolitan area with the aim of offering cloud services at the edge. In contrast to work in Part II, this chapter considers both fixed nodes and mobile UAV nodes. Fixed nodes are always available due to continuous energy supply, but, once deployed, they cannot be easily migrated to another location. On the other hand, UAV nodes can fly between different locations to augment the processing capacity of fixed fog nodes, especially to process workload in excess of the capacity of fixed nodes. However, UAV nodes are mobile and operate on batteries which limit the length of time they can process user workload. The consideration of both fixed and mobile nodes enables the deployment of an infrastructure capable of handling the variabilities in workload demand; this helps reduce the underutilization of over-dimensioned fixed nodes for processing eventual peak demands.

An algorithm, called UAV Fog Node Location (UFL), is introduced to determine the best combination of fixed nodes and UAVs in an infrastructure. The UFL algorithm initially finds an exact solution to the fog node location problem considering only fixed nodes, and then attempts to replace underutilized servers in fixed nodes by UAVs which

can change their location to cope with processing demand at different locations. Simulations considering diverse demand patterns across a metropolitan area as well as real UAV characteristics such as cost, battery capacity and processing capabilities [3] are employed to address the question: *are UAVs worth adopting to replace fixed nodes in a fog infrastructure?* The wireless communication channel is considered ideal, so that results are actually a bound on the value of UAVs used as processing nodes. Results show that the current cost of UAVs is a limiting factor to their usage as fog nodes. In the future, however, and assuming that costs continue to decrease, UAVs could provide interesting solutions for optimally infrastructure dimensioning. Results show that when UAV costs are the same as those of fixed nodes, UAV deployment is indeed advantageous. In this case, the coverage of several locations using UAVs reduces CAPEX and provides more flexibility in coping with unexpected and timely increases in processing demands.

The remainder of this chapter is organized as follows. Section 6.2 introduces the system model adopted. Section 6.3 presents the linear programming formulation of the problem. Section 6.4 introduces the UAV Fog Node Location Algorithm. Section 6.5 presents the experimental setting and numerical evaluation of the UFL algorithm. Lastly, Section 6.6 concludes this chapter.

6.2 System model

We consider a system in which mobile users in a metropolitan area request services at different locations, and the processing demand at these locations is a function of user mobility. Applications such as augmented reality and traffic navigation have strict latency requirements and cannot be processed in the cloud; and users need to connect to a nearby fog node to offload the processing of these applications. If there is no fog node available, requests are rejected (blocked) since they cannot be migrated to more distant nodes due to latency requirements.

A fog node is a small facility that has processing, storage, and networking capabilities. It can process workload offloaded by end users without the need to send it to the cloud through the Internet, thus considerably reducing the response time of applications. Fog nodes serve users in their coverage area, and nodes can have more than one compute server. However, not all servers of a node are continuously needed since the processing demand varies over time.

UAVs can travel from one node to the other to increase the processing capacity of a destination node. UAVs can land to process the workload instead of just hovering. When on ground, UAVs have greater autonomy, since the energy consumption of a UAV is much lower than when hovering [3], with energy consumed only by communication and processing.

UAVs can be in one of four different states: turned-off, stand-by, processing, and flying. UAVs are initially turned off, with their battery fully charged. When a UAV is turned off, it does not consume energy. When its service is needed, a UAV starts its operation and remains on until service is no longer required. In the stand-by state, a UAV is on ground but not processing and its energy consumption is fixed; moreover, it

can be quickly switched to the processing or flying state. In the processing state, the UAV is also on ground, but it consumes energy for processing and data transmission. Finally, in the flying state, the UAV is moving between different locations; flights are allowed only to complement the capacity of a fog node at the destination. The flying state is the one that consumes the largest amount of energy, with a consumption depending on the distance traveled and the speed of traveling, both horizontally and vertically. The sum of the energy consumed by all operations must be lower than the UAV battery energy capacity, which demands a precise location plan to extend the drone operational time. Other sources of consumption such as environmental factors are not considered in this thesis.

The fog node location problem consists of deciding on the locations where fog nodes should be deployed. The main input to this problem is the set of potential locations for hosting fog nodes, the workload demand, and the available budget for acquiring fixed and UAV fog nodes. The output is the set of locations selected for the deployment of nodes as well as the number of servers at each node. Additionally, the number of UAVs and their flight plan should be determined. The primary goal is to process the maximum possible amount of workload, and the secondary goal is to reduce the infrastructure cost.

6.3 Formulation

The fog node location problem is formulated as an optimization problem and is modeled as a mixed-integer linear programming formulation. The model is bi-criteria and optimizes the served workload as well as the deployment cost. The notation in Table 6.1 is used in the formulation.

The model considers discrete time and $\mathcal{T}$ is the set of discretized intervals. The set of all candidate locations for fog nodes is given by $\mathcal{L}$. The workload demand is variable in each location over time, indicated by $\mathcal{W}_{lt}$. All fixed servers have the same capacity, their cost and capacity are $\mathcal{C}^S$ and $\mathcal{K}^S$, respectively, and UAVs cost and capacity are $\mathcal{C}^U$ and $\mathcal{K}^U$, respectively.

The output of the formulation is the workload served, the fixed servers used, number of UAVs employed, their trajectories as well as their states during the evaluation. The served workload is given by w_{lt} for each location l and time t. The number of fixed servers in each location l is indicated by n_l, and the number of used UAVs is n^u. $\mathcal{U}$ is set of all possible UAVs. The total battery capacity of any UAV is denoted by E, and energy cost for a trip between locations k and l is indicated by E_{kl}. The number of intervals required by a UAV to travel between a location k to a location l is given by D_{kl}.

The states of UAVs are indicated by the variables a_{ult}, s_{ult}, f_{uklt}, p^f, and p^l; a_{ult} indicates that the UAV u is in processing state at location l during time t. Similarly, s_{ult} indicates that the UAV u is inactive at location l and time t. Being inactive means that the UAV is either in stand-by or in power-off state. A UAV plan starts in the turned-off state; then, after being turned on, it can be in the other three states. If turned off or the battery capacity has exhausted, it goes to the power-off state. The power-off state is denoted by p^f_{ut} and p^l_{ut}. If a UAV u remains in power-off state in the first T^f time

Table 6.1: Notation used in the fog node location problem formulation.

Input parameters	
Notation	**Description**
General parameters	
$\mathcal{T}$	Set of discrete time intervals: $\mathcal{T} = \{1, 2, \ldots, T\}$
$\mathcal{L}$	Set of candidate locations for fog nodes: $\mathcal{L} = \{1, 2, \ldots, L\}$
$\mathcal{W}_{lt}, l \in \mathcal{L}, t \in \mathcal{T}$	Workload at location l during time slot t
Cost	
$\mathcal{C}^S$	Cost of a single fixed server
$\mathcal{C}^U$	Cost of a UAV carring a mobile server
$\mathcal{C}$	Available budget
Capacity	
$\mathcal{K}^S$	Capacity of a single fixed server
$\mathcal{K}^U$	Capacity of a UAV mobile server
UAV parameters	
$\mathcal{U}$	Set of available UAVs, $\mathcal{U} = \{1, \ldots, U\}$
E	Total battery capacity of a UAV
E^P	Energy required by a UAV in processing state during one time interval
E^S	Energy required by a UAV in stand-by state during one time interval
$E_{kl}, k, l \in \mathcal{L}$	Energy consumption per time window required by a UAV to travel between locations l and k
$D_{kl}, k, l \in \mathcal{L}$	Number of discrete time intervals required by a UAV to travel between locations l and k
Decision variables	
Notation	**Description**
$w_{lt}, l \in \mathcal{L}, t \in \mathcal{T}$	Workload originating at location $l \in \mathcal{L}$ at time $t \in \mathcal{T}$ and processed by the fog node
$n_l, l \in \mathcal{L}$	The number of fixed servers deployed at the fixed fog node located at l.
n^u	Number of required UAVs
$i_u, u \in \mathcal{U}, l \in \mathcal{L}, t \in \mathcal{T}$	1 if UAV u was in inactive, i.e. it was in stand-by state over all time intervals; 0 otherwise
$a_{ult}, u \in \mathcal{U}, l \in \mathcal{L}, t \in \mathcal{T}$	1 if UAV u is at location l at time t in processing state; 0 otherwise
$s_{ult}, u \in \mathcal{U}, l \in \mathcal{L}, t \in \mathcal{T}$	1 if UAV u is at location l at time t in stand-by or power-off state; 0 otherwise
$f_{uklt}, u \in \mathcal{U}, k \in \mathcal{L}, l \in \mathcal{L}, t \in \{2, \ldots, T\}$	1 if UAV u is traveling between locations k and l at time t; 0 otherwise
$p^f_{ut}, u \in \mathcal{U}, t \in \mathcal{T}$	1 if UAV u is in power-off state at all time intervals $t*$ such that $t* \leq t$; 0 otherwise
$p^l_{ut}, u \in \mathcal{U}, t \in \mathcal{T}$	1 if UAV u is in power-off state at all time intervals $t*$ such that $t* \geq t$; 0 otherwise

intervals, $p^f_{ut} = 1$ for all $1 \leq t \leq T^f$. Similarly, if a UAV u goes to power-off state at time T^l, then $p^l_{ut} = 1$ for all $T^l \leq t \leq T$. The flying state is represented by the variable f_{uklt} that indicates whether the UAV u is traveling between locations k and l during time t. Finally, if a UAV u is not required, i.e., if it remains inactive during all time intervals in $\mathcal{T}$, then $i_u = 1$.

The formulation of the fog node location problem is given by Equations (6.1)–(6.18):

$$maximize \sum_{l \in \mathcal{L}} \sum_{t \in \mathcal{T}} w_{lt} \tag{6.1}$$

$$minimize \quad C^S \sum_{l \in \mathcal{L}} n_l + C^U n^u \tag{6.2}$$

$$\mathcal{C}^S \sum_{l \in \mathcal{L}} n_l + \mathcal{C}^U n^u \leq \mathcal{C} \tag{6.3}$$

$$w_{lt} \leq \mathcal{K}^S n_l + \mathcal{K}^U \sum_{u \in \mathcal{U}} a_{ult}, l \in \mathcal{L}, t \in \mathcal{T} \tag{6.4}$$

$$w_{lt} \leq \mathcal{W}_{lt}, l \in \mathcal{L}, t \in \mathcal{T} \tag{6.5}$$

$$\sum_{k \in \mathcal{L}} \Big(a_{ukt} + s_{ukt} + \sum_{l \in \mathcal{L} \setminus \{k\}} f_{uklt} \Big) = 1, u \in \mathcal{U}, t \in \mathcal{T} \tag{6.6}$$

$$\begin{aligned} a_{ult} + s_{ult} \leq a_{ul(t-1)} + s_{ul(t-1)} + \sum_{k \in \mathcal{L} \setminus \{l\}} f_{ukl(t-1)}, \\ u \in \mathcal{U}, l \in \mathcal{L}, t \in \{2, \ldots, T\} \end{aligned} \tag{6.7}$$

$$\begin{aligned} f_{uklt} \leq a_{uk(t-1)} + s_{uk(t-1)} + f_{ukl(t-1)}, \\ u \in \mathcal{U}, k \in \mathcal{L}, l \in \mathcal{L} \setminus \{k\}, t \in \{2, \ldots, T\} \end{aligned} \tag{6.8}$$

$$\begin{aligned} a_{ukv} + s_{ukv} \leq 1 - a_{ult} - s_{ult}, \\ u \in \mathcal{U}, k \in \mathcal{L}, l \in \mathcal{L} \setminus \{k\}, t \in \mathcal{T}, \\ v \in \{t+1, \ldots, t + D_{lk}\} \end{aligned} \tag{6.9}$$

$$i_u \leq \frac{1}{T} \sum_{l \in \mathcal{L}} \sum_{t \in \mathcal{T}} s_{ult}, u \in \mathcal{U} \tag{6.10}$$

$$i_u > \Big(\sum_{l \in \mathcal{L}} \sum_{t \in \mathcal{T}} s_{ult} \Big) - T, u \in \mathcal{U} \tag{6.11}$$

$$p^f_{ut} \leq \sum_{l \in \mathcal{L}} s_{ult}, u \in \mathcal{U}, t \in \mathcal{T} \tag{6.12}$$

$$p^l_{ut} \leq \sum_{l \in \mathcal{L}} s_{ult}, u \in \mathcal{U}, t \in \mathcal{T} \tag{6.13}$$

$$p^f_{ut} \leq p^f_{u(t-1)}, u \in \mathcal{U}, t \in \{2, \ldots, T\} \tag{6.14}$$

$$p^l_{ut} \leq p^l_{u(t+1)}, u \in \mathcal{U}, t \in \{1, \ldots, T-1\} \tag{6.15}$$

$$p^f_{ut} + p^l_{ut} \leq 1, u \in \mathcal{U}, t \in \mathcal{T} \tag{6.16}$$

$$n^u = U - \sum_{u \in \mathcal{U}} i_u \tag{6.17}$$

$$\begin{aligned} \sum_{t \in \mathcal{T}} \sum_{l \in \mathcal{L}} E^A \cdot a_{ult} + \\ \sum_{t \in \mathcal{T}} (\sum_{l \in \mathcal{L}} E^S \cdot s_{ult} - p^f_{ut} - p^l_{ut}) + \\ \sum_{t \in \{2,\ldots,T\}} \sum_{k \in \mathcal{L}} \sum_{l \in \mathcal{L} \setminus \{k\}} E_{kl} \cdot m_{uklt} \leq E, \\ u \in \mathcal{U} \end{aligned} \tag{6.18}$$

The objective function in Equation (6.1) is the maximization of the served workload. Equation (6.2) minimizes of the cost to build the infrastructure, considering the cost of both fixed servers and UAVs. These goals are subject to the constraints given by Equations (6.3)–(6.18). Equation (6.3) limits the cost of the infrastructure to the available budget. Equation (6.4) guarantees that the workload served is smaller than the capacity of the fog node, considering both fixed and UAV servers. Equation (6.5) limits the workload served to the demand at each location in each time interval. Equation (6.6) guarantees that every UAV is at exactly one single state at any moment. Equations (6.7)–(6.8) check whether the transitions between states are valid considering time and location limitations. In sequential time intervals, UAVs cannot change their location without going to travel state (Equation (6.7)), and the travel state can only be reached if the UAV was in travel state or in the source location in the previous time interval (Equation (6.8)). Equation (6.9) guarantees that the minimum trip time is respected when a UAV flies between different locations. Equations (6.10)–(6.11) check whether each UAV was inactive, i.e., in stand-by over all intervals. Equations (6.12)–(6.16) guarantee the right value of p^f_{ut} and p^l_{ut}, identifying the intervals of power-off state. Equation (6.17) accounts n^u, the number of required UAVs. Finally, (6.18) guarantees that the autonomy of each UAV is respected by summing the energy spent in each state and limiting it to the battery capacity.

The deployment planning given by this formulation is a network design problem, and, as such, is typically solved off-line. However, solving this problem optimally using existing solvers does not scale to large problem instances. The modeling of potential flight routes, the location of UAVs at every time interval, and the UAVs activity/inactivity periods lead

to an exponential growth in the number of constraints. To circumvent these limitations, a heuristic algorithm is proposed next.

6.4 UAV Fog Node Location Algorithm

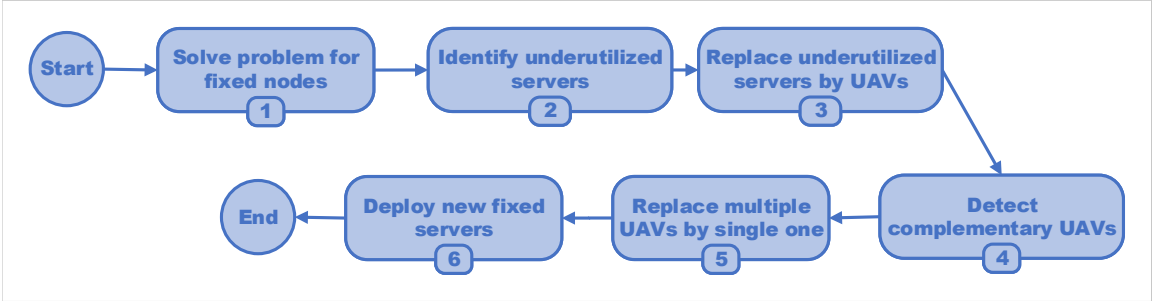
This chapter proposes a heuristic algorithm called the UAV Fog Node Location (UFL) algorithm. Figure 6.1a shows its flowchart. The algorithm starts by solving the formulation with only fixed nodes, and the pre-defined budget limiting the number of servers and nodes that can be deployed. Based on the solution obtained, the algorithm identifies servers that can potentially be replaced by UAVs; these servers are typically underutilized and deployed only to deal with peak demands. The UFL algorithm then attempts to use UAVs to cover several locations at different times to reduce the deployment cost. The algorithm considers the ratio between the cost of UAVs and the cost of fixed servers.

The first step of the algorithm considers only fixed nodes with the result obtained using an optimization solver (Step 1). The next step is the identification of servers to be replaced (Step 2). For each fog node, the algorithm identifies if a server can be replaced by a UAV, a situation which arises if a server is not processing requests for all time intervals, the UAV processing capacity is greater than the offered workload, and the energy that will be consumed by the UAV is less than its available battery capacity. The energy needed is the sum of the energy spent in processing and that in stand-by in periods associated with the potential replacement. The identified servers are then replaced by UAVs (Step 3).

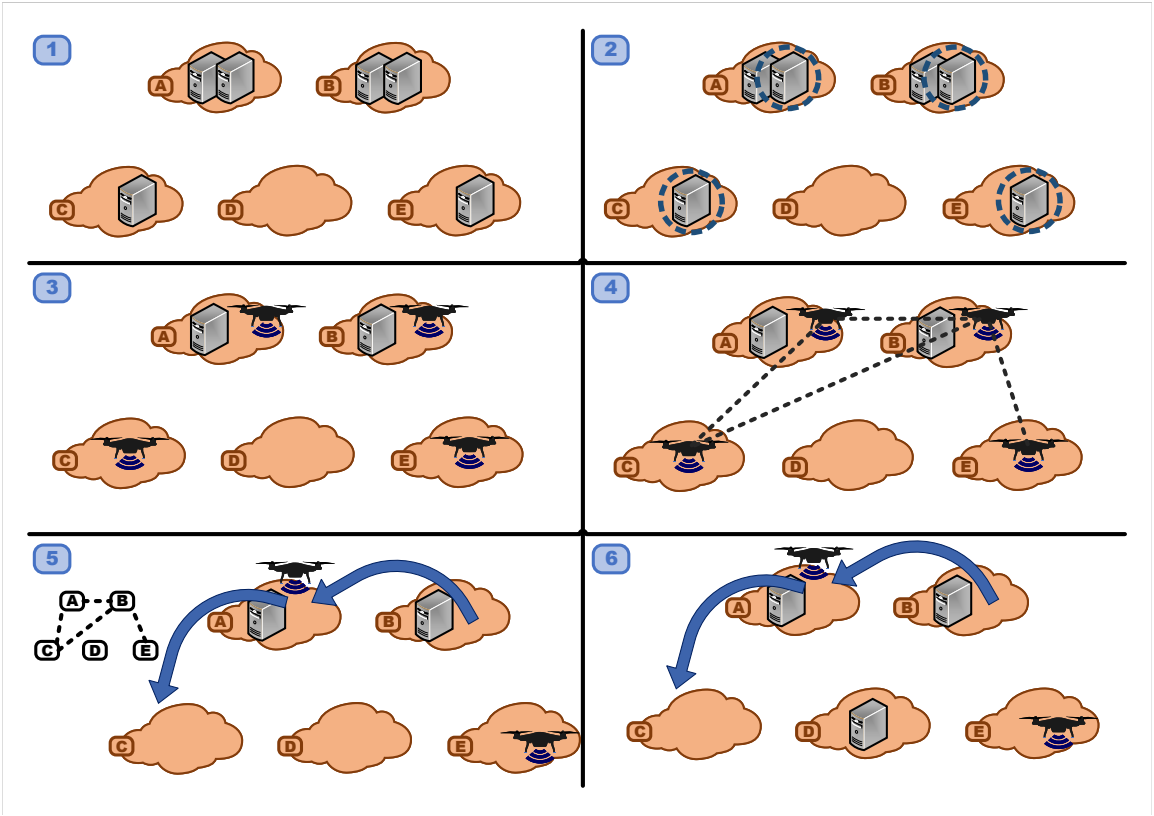
To reduce the infrastructure cost (secondary objective), all pairs of UAVs are considered to be replaced by a single UAV (Step 4). Two UAVs can be replaced by a single one if three conditions are fulfilled. First, the two UAVs must be in processing state in different time periods. Second, the time for traveling between the two locations is less than the time elapsed between the end of the processing at the node for which the UAV departs and the beginning of processing at the destination node. Third, the UAV battery should be sufficient to support full operation, including the flight between the fixed fog nodes. If all conditions are met and after serving the workload at a location, a UAV can fly to another location to serve the workload at the new location.

The algorithm evaluates a potential reduction in the number of UAVs (Step 5). Such an evaluation is carried out by considering a graph in which each UAV is a vertex and each potential pair of locations for replacement is an edge of the graph. Then, the algorithm finds maximal cliques, which determines the minimum number of UAVs to be deployed. If the solution still leaves a backlog of unprocessed workload and the number of UAVs has been reduced in the last step, the unused budget can be employed to further reduce the unserved workload (Step 6).

Figure 6.1b exemplifies the steps involved in planning fog nodes in five locations. The budget comprises six servers, and the solution obtained in Step 1 indicates fog nodes in locations A and B with two servers each due to their larger processing demand; other locations have a low processing demand, with only one server in locations C and E. Four fixed servers are identified as being underused in Step 2, and are then replaced by UAVs



(a) Algorithm flowchart.



(b) Example of execution.

Figure 6.1: UFL algorithm.

(Step 3). In Step 4, the algorithm detects the pairs of servers that have complementary processing demands in time, and that the battery of a single UAV being adequate to support the operation in both locations; the dashed lines indicate these pairs. Step 5 shows the graph of UAVs, which contains two maximal cliques. Locations A, B, and C can be served by a single UAV, i.e., during a discretized time interval, a UAV can either process the workload in one location or fly to another location to provide service at the destination. Since one UAV has replaced three fixed servers, an extra server can be deployed in location D, thus increasing the overall workload served (Step 6). The final solution has three fixed servers and two UAVs, thus employing fewer devices than the initial solution.

The UFL algorithm, as presented here, considers that any fixed server can be replaced by a UAV server. If the price of a UAV is greater than that of a fixed server, the solution obtained may not always be worth adopting. As a consequence, alternative solutions should be obtained which consider different ratios between the cost of a UAV node and that of a fixed node. Such an analysis facilitates long term planning for the evaluation of the infrastructure, thus helping avoid unnecessary expenses in the deployment of the original infrastructure.

The complexity of the UFL algorithm depends on Step 1, which has an exponential time complexity due to the exact solution of fixed node location problem. Nonetheless, similar to the work in Chapter 4, this operation can be quickly performed by a solver. Therefore, although the time complexity of the UFL algorithm is exponential, it can obtain scalable solutions to the fog node location problem.

6.5 Performance Evaluation

To answer the question of whether UAVs are worth adopting for replacing fixed fog nodes, extensive simulations of the UFL algorithm involving realistic scenarios were carried out. Subsection 6.5.1 describes the experimental settings adopted, and Subsection 6.5.2 discusses numerical results.

6.5.1 Experimental settings

The parameter values defining the scenarios in the simulations are summarized in Table 6.2. The UAVs considered are rotary-wing drones which have the capacity to land in limited spaces. The characteristics of the simulated UAVs are based on real drones described in previous work [3, 75]. The consumption model is the one derived in [3], described in Subsection 2.5.3. Since the UAV used in [3] does not have a powerful battery, the present evaluation also considers different battery models [75]. Moreover, two other parameters were varied as a function of the fixed servers: the UAV processing capacity and UAV price. The locations and the workload demands were based on the data set [8] reviewed in Section 3.5. Similar to the evaluation in Chapter 4, the BSs are used as the candidate locations for fog nodes, with 1150 locations. The workload is the one taken from the data set [8], without considering individual requests. 144 time intervals with 10-minute length are considered, evaluating 24 hours of demands.

Table 6.2: Parameters adopted in the UAV simulations. H refers to the vertical distance traveled, either upwards or downwards.

Operation	**Energy consumption**
Fly horizontally	245.2815 W
Fly vertically up	$(-16.9396H^2 + 216.6944H - 157.9473)$ J
Fly vertically down	$(4.6817H^2 - 11.9708H + 135.3118)$ J
Stand-by state	8.2637 W
Processing state	15.7637 W
Operation	**Speed**
Horizontal	10 m/s
Vertical up/down	1 m/s
Variable parameters	**Values**
Battery capacity	4500 mAh/14.8 V, 27000 mAh/22.2 V, and 34200 mAh/22.8 V
Processing capacity	50 % and 100 % of a fixed server
UAV price	1, 2, 3, and 4 times the price of a fixed server

6.5.2 Numerical results

The UFL algorithm was coded in Python, and its first step (bi-criteria formulation for the deployment of fixed nodes only) was solved using the Gurobi Optimizer solver. The results produced by the UFL algorithm were compared to those obtained by the solver. Two metrics were evaluated, related to the two objectives of the problem: the acceptance ratio of workload and the number of devices deployed (servers and UAVs). The first metric is the ratio between the workload served and the total workload requested by the end users. The second metric is the number of servers used for the solution using only fixed nodes, and the number of servers and UAVs employed computed by the UFL algorithm. Sixty executions were carried out to derive each value with a 95 % confidence interval. The number of available servers for deployment (N) was varied from 1 to 2048. UAV^C denotes the ratio between the cost of a UAV and the cost of a fixed server. Similarly, UAV^P is the ratio between the processing capacity of a UAV server and that of a fixed server.

The acceptance ratio using UAVs with the most powerful battery and the same capacity as a fixed server is shown in Figure 6.2. The acceptance ratio increases until $N = 1280$, when servers are sufficient to deal with all the demand. The demand served depends predominantly on the fixed infrastructure capacity due to the limited autonomy of UAVs to stay powered for long periods. Nevertheless, improvements were noticed when UAVs and fixed servers have the same cost and $N \geq 128$ since UAVs could be widely deployed. In these cases, UAVs improved the acceptance of workload and, in some cases ($N \geq 1536$), provided 100 % acceptance of workload. Higher costs of UAVs limited their number considerably and, as a consequence, the workload acceptance did not change in relation to deployment with only fixed nodes.

The results described below show the impact of the cost of UAVs, UAV processing capacity, and their autonomy on the deployment. Variations in these parameters do not make significant changes in the workload acceptance when compared to those already

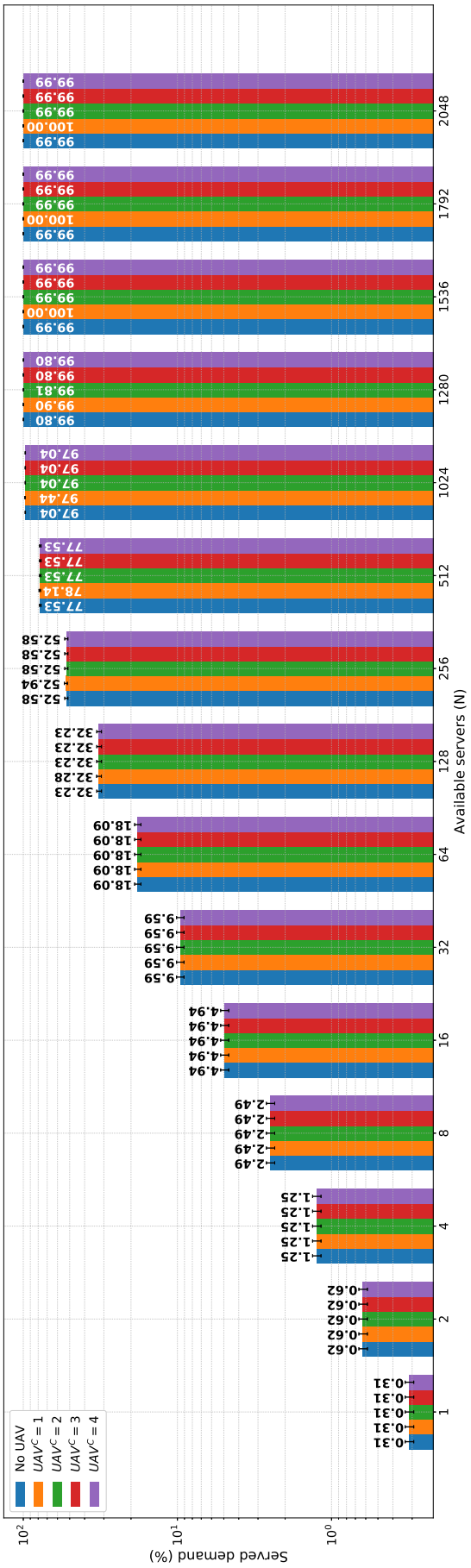


Figure 6.2: Acceptance ratio for the 34200 mAh/22.8 V battery and $UAV^P = 1$.

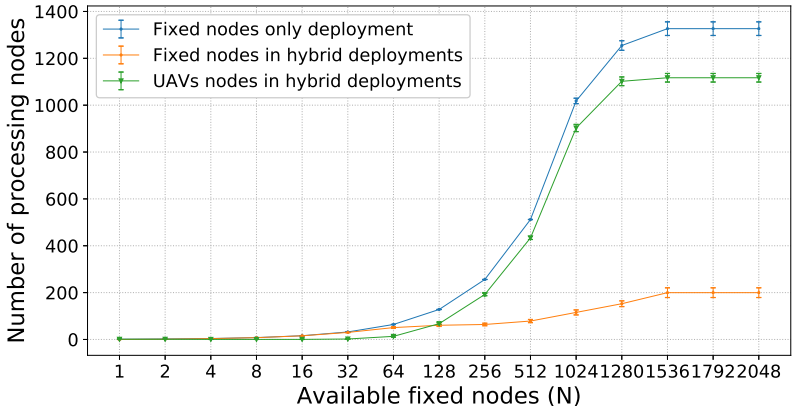


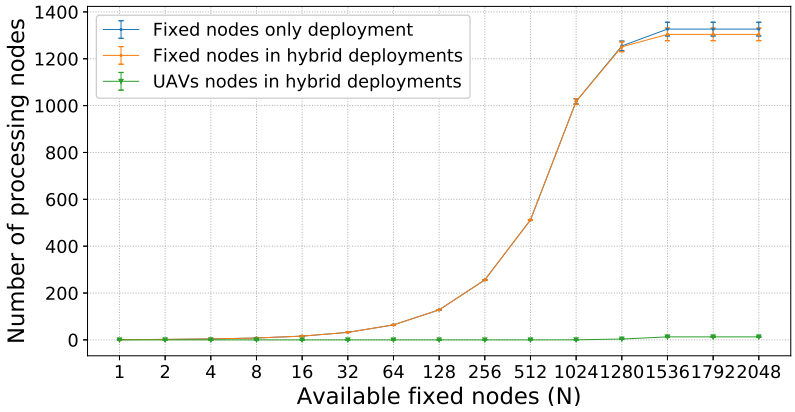
Figure 6.3: Number of servers and UAVs used for the 34200 mAh/22.8 V battery, $UAV^C = 1$, and $UAV^P = 1$.

shown (Figure 6.2). This is due to the fact that the workload acceptance is predominantly optimized by the fixed servers, with only slight improvements by the effect of using UAVs. In line with these results, only number of UAVs and servers are considered in the following experiments.

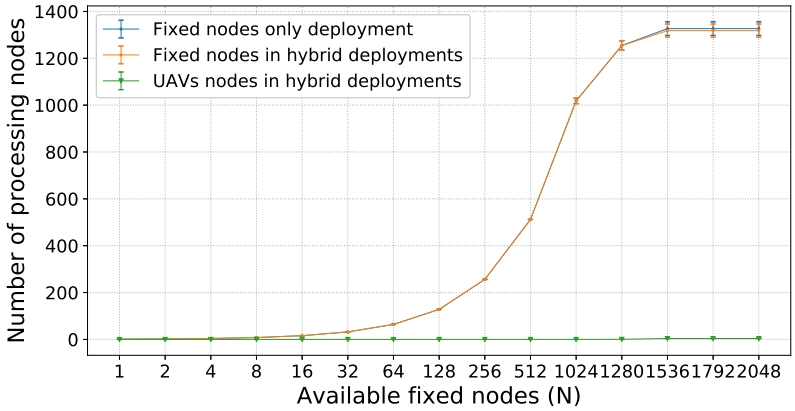
Figures 6.3–6.6 show the number of employed devices for an optimal deployment with only fixed nodes as well as for hybrid deployments (fixed servers and UAVs) obtained by the use of the UFL algorithm. Figure 6.3 shows the results for the greatest battery capacity, with UAV cost and processing capacity equal to those of fixed servers. For $N < 128$, almost all fixed servers in fog nodes were heavily used for long periods of time so that replacing servers with UAVs is not possible. When a larger number of devices is available for deployment, a large number of fixed servers is replaced by UAVs, which shows that, despite the large number of locations (1150), only about 200 fixed servers could not be replaced by aerial servers, i.e. an infrastructure with only 20 % of the locations being fixed nodes and UAVs serving the remaining 80 % of the locations.

Nowadays, UAVs cost is three to four times the cost of a traditional fixed server, and, under these circumstances, the employment of several UAVs is not advantageous. Figure 6.4 shows the results for UAVs two to four times more expensive than a fixed server. Even for a low cost (Figure 6.4a) and $N \leq 1280$, the average number of employed UAVs is very close to zero. This low number of flying servers is due to the fact that using the same UAV to serve two different locations is not always possible because of the required flight time between the locations, which led to quickly drain of the UAV battery. UAVs with costs between three and four times the price of a fixed were seldom used, showing that the UAV price is decisive to be considered in large deployments.

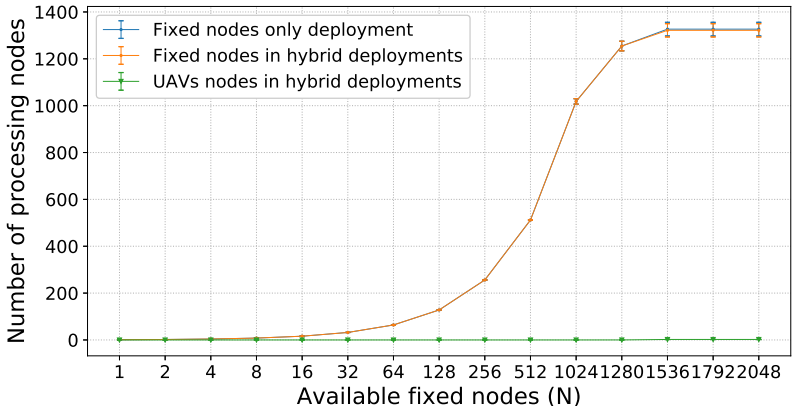
The results considering UAV servers having limited processing capacity compared to a fixed server are shown in Figure 6.5. The greater the capacity, the larger the number of servers replaced by UAVs, since UAVs with a limited processing capacity cannot always deal with the peak demands supported by a fixed server. The increase in the processing capacity and the increase in the number of UAVs is not linear: a four-fold increase in the UAV processing capacity (25 % to 100 %, Figure 6.5a) leads to an increase in less



(a) 34200 mAh/22.8 V battery, $UAV^C = 2$ and $UAV^P = 1$.

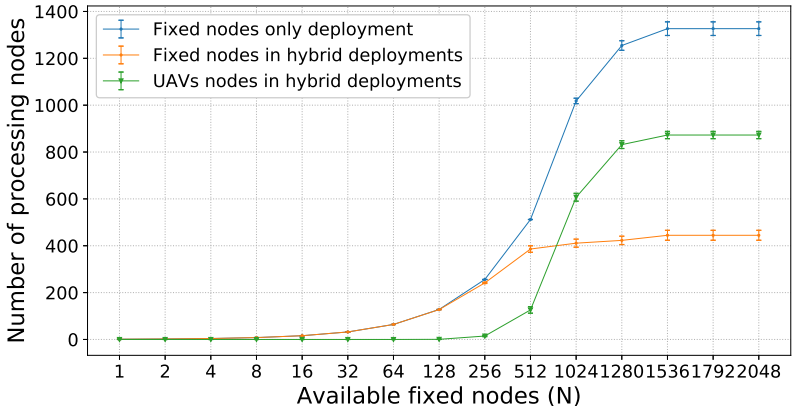


(b) 34200 mAh/22.8 V battery, $UAV^C = 3$ and $UAV^P = 1$.

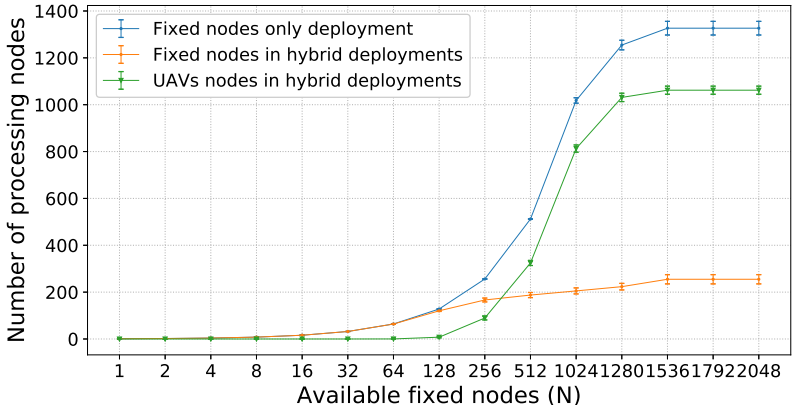


(c) 34200 mAh/22.8 V battery, $UAV^C = 4$ and $UAV^P = 1$.

Figure 6.4: Number of servers and UAVs used for the 34200 mAh/22.8 V battery, $UAV^P = 1$, and different values of UAV^C.



(a) 34200 mAh/22.8 V battery, $UAV^C = 1$ and $UAV^P = 0.25$.

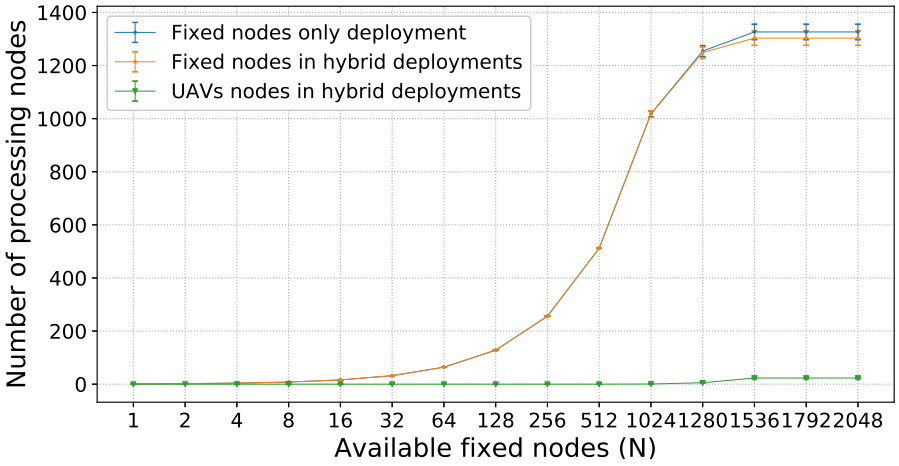


(b) 34200 mAh/22.8 V battery, $UAV^C = 1$ and $UAV^P = 0.50$.

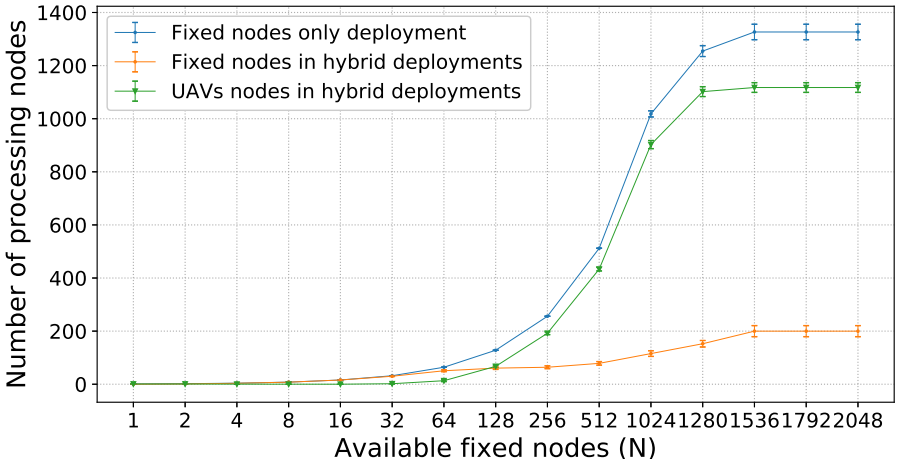
Figure 6.5: Number of servers and UAVs used for the 34200 mAh/22.8 V battery, $UAV^C = 1$, and different values of UAV^P.

than 5 % in the number of UAVs for $N > 1280$, while the two-fold increase from 50 % to 100 % (Figure 6.5b) leads to an increase in less than 10 %. This is explained by the pattern of the frequency of peak demands. Servers replaced by UAVs are seldom used, and, therefore, they deal with rather sporadic demands. Given these low demands, UAVs with powerful computers are not required, making UAVs with 50 % of the fixed server capacity significantly useful.

The final analysis concerns the battery capacity, with results for different battery capacities presented in Figure 6.6. Results for an intermediate battery (Figure 6.6b) did not lead to great differences in the results, but the smallest battery capacity (Figure 6.6a) has a very limited autonomy, thus it has little use in such an infrastructure. Only when all the demand was met ($N \geq 1536$) can approximately 25 UAVs replace fixed servers. The problem is the autonomy of the batteries, which prevents the replacement of a single underloaded server with a UAV. To further increase the number of UAVs, the battery life would have to be sufficient to maintain the UAVs turned on for several hours, which is not a realistic assumption for battery-constrained UAVs. Technologies for charging batteries without interrupting the operation can help to extend UAVs operation in fog



(a) 4500 mAh/14.8 V battery, $UAV^C = 1$ and $UAV^P = 1$.



(b) 27000 mAh/22.2 V battery, $UAV^C = 1$ and $UAV^P = 1$.

Figure 6.6: Number of servers and UAVs used for $UAV^C = 1$, $UAV^P = 1$, and different battery capacities.

infrastructures.

A comparison between the UFL algorithm and the dispatching scheme in [99] was carried out. Figure 6.7 depicts the mean percentage difference in the number of UAVs demanded by the two schemes, $\frac{\text{number of UAVs required by [99]} - \text{number of UAVs required by UFL}}{\text{number of UAVs required by UFL}} \times 100$. The scheme in [99] differs from UFL in three ways: first, it assumes unlimited energy; second, it can dispatch UAVs to process the workload at every time interval without evaluating future demands; and third, UAVs do not fly between different locations. We imposed battery limitation in the scheme in [99] for the sake of fair comparison. We denoted the original solution with battery limitation "single location". We also implemented a version that allows a UAV to serve multiple locations, denoted "multiple locations". UAVs are used in multiple locations if the battery can support the flight and the processing of the workload at the destination.

The results indicate that most fixed servers process heavy loads in small infrastruc-

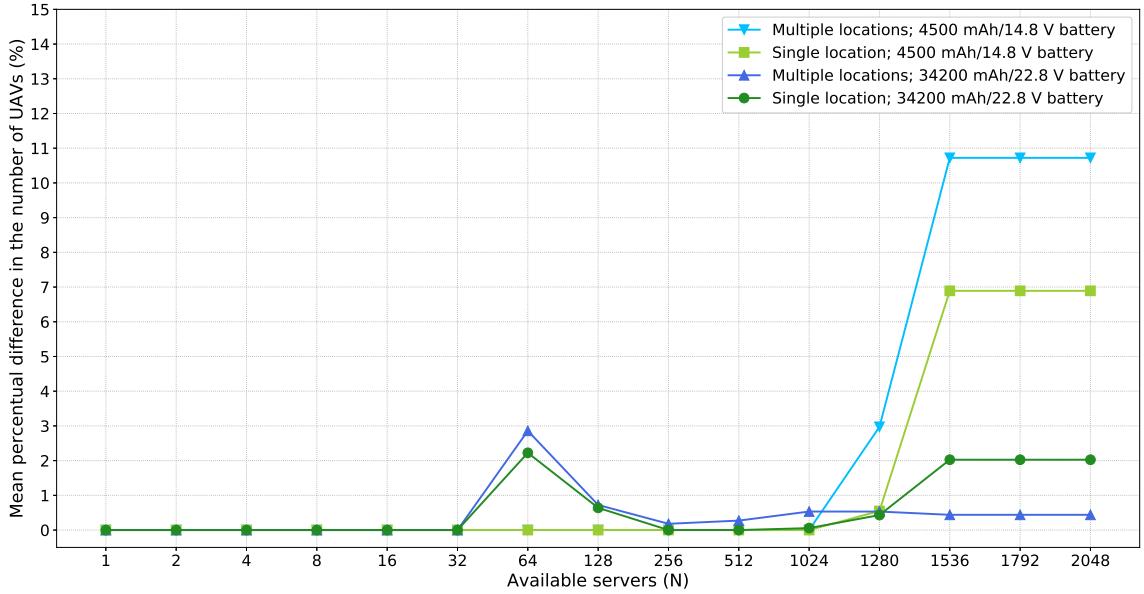


Figure 6.7: Difference in the number of UAVs required by the dispatching scheme in [99]. Results obtained for $UAV^C = 1$, and $UAV^P = 1$.

tures ($N \leq 32$), which does not create opportunities for replacing fixed nodes with UAVs. For large infrastructures ($N \geq 1536$), the scheme in [99] requires a greater number of UAVs when compared to UFL. When UAVs have batteries with large capacity, the dispatching scheme employs up to 2 % more UAVs to process the same workload. Moreover, for batteries with small capacity, the scheme in [99] requires, on average, 7 % more UAVs when they can serve only a single location and almost 11 % when they can serve multiple locations. The battery capacity has an impact on the number of required UAVs, since flights consume large amounts of energy. Such consumption reduces the UAV operational time, especially with a small battery capacity, preventing the processing of future workload, and, consequently, calls for more UAVs. The UFL algorithm produces more efficient deployments due to its consideration of energy consumption and planning of UAVs trajectories.

The present findings have revealed advantages and disadvantages in relation to the adoption of such hybrid infrastructures with both fixed and UAV nodes. One advantage is that hybrid infrastructures can simplify the deployment of fixed nodes where processing demands are low, thus reducing costs for deploying and maintaining nodes with underused servers continuously turned on. However, as long as UAV prices are higher than those of traditional servers, their use will remain limited. In case of price reduction, UAVs in fog infrastructures may become much more wide-spread.

6.6 Conclusions

This chapter has investigated the employment of unmanned aerial vehicles as fog nodes by solving a fog node location problem. By considering UAVs in this early stage, it is possible to plan the best deployment and avoid placing fixed servers in locations with low demand.

This chapter has described the UFL algorithm which first solves the problem optimally by considering only fixed servers, and then tries to replace underutilized servers by UAVs, which can potentially serve more than one location. Results were obtained by varying different UAV characteristics and using a publicly available dataset. The UFL algorithm can be used for long-term planning of large fog infrastructures. Results show that a significant portion of the infrastructure could be replaced by UAVs depending on their price evolution. An additional benefit of using UAVs is the energy saved compared to an infrastructure with only fixed servers constantly powered all the time. This investigation has revealed that such a deployment depends on the prices of UAVs being close to that of traditional servers. Currently, UAVs cost three to four times more than traditional servers, but prices are expected to decrease in the future as a function of mass production and wide use of unmanned aircraft.

Chapter 7

Location of Fog Nodes mounted on Rotary-wing UAVs

7.1 Overview

UAVs vary in size, aerodynamics, and autonomy. One type of UAV is the rotary-wing UAV [105, 99, 23, 75, 88, 82, 25, 110], a versatile aircraft that operates either in the air or on the ground since it can both hover and fly vertically. Despite the advantages, rotary-wing UAVs typically have a very limited autonomy due to the high energy consumption for propulsion, and their operation must be carefully planned to maximize operational time.

In order to plan an efficient fog infrastructure with UAVs, this chapter attempts to answer the following question: *what should be the location and operation period of fog nodes mounted on rotary-wing UAVs to maximize the number of end-users served while reducing the delay between ground nodes and UAVs?* To answer this question, this chapter solves the fog node location problem by considering rotary-wing UAVs. Several papers [74, 52, 64, 99, 114] dealt with fog nodes without taking the autonomy of UAVs in large deployments into consideration nor fully exploring the potential of rotary-wing UAVs. To fill these gaps, this chapter proposes a solution to the location problem and evaluates different modes of operation for rotary-wing UAVs as well as different latency constraints. In a more realistic evaluation, characteristics of real drones are considered and energy consumption is accounted for. Ways of extending operational time to enhance user service are also discussed. A fog computing environment with latency requirements of a few milliseconds was considered, supporting a broad range of applications. Such an environment calls for high data transmission rates between aerial fog nodes and ground users.

The fog node location is an optimization problem and this chapter formulates it as an integer linear programming (ILP) model. The model has a large number of constraints and decision variables, limiting the scalability to obtain optimal solutions, which has motivated the proposal of a heuristic called Sequential UAV Fog Node Location (SUL). Peaks of user requests tend to be concentrated in a few regions and last for limited times. This suggests that deploying UAVs into these locations can cope with the varying demands for resources. The SUL algorithm builds a solution to the location problem by greedily deploying one

UAV at a time into locations with high demands. It is scalable and can be used to plan large fog infrastructures. We performed simulations using real data, representing time and space variable demands of users collected in a cellular network, to assess the effectiveness the SUL algorithm. Numerical results show that UAVs can support strict delays, but their operational time is highly impacted by hovering, which indicates that the employment of landing spaces can significantly reduce deployment costs. Moreover, the UAV hovering altitude is assessed, and results showed that higher altitudes required more UAVs to serve the same number of users due to the different energy. However, UAVs at lower altitudes can be quite useful to provide nearby users with fast communications. Results also validated the performance of the SUL algorithm by comparing it with the optimal solution for small instances of the problem.

The remainder of this chapter is organized as follows. Section 7.2 introduces the system model adopted. Section 7.3 presents the linear programming formulation of the problem. Section 7.4 introduces the UAV Fog Node Location Algorithm. Section 7.5 presents the experimental setting and numerical evaluation of the SUL algorithm. Lastly, Section 7.6 concludes this chapter.

7.2 System Model

Fog computing complements the traditional cloud infrastructure by supporting low-latency requirements of end-user applications; fog nodes are usually deployed in different geographical locations to cope with variable demands. Users run various applications on mobile devices and typically visit various locations during the day. A device can send a request using the wireless interface to a fog node, transferring the workload to the fog node for processing, and the results are sent back to the user. Most fog nodes are small fixed facilities that host processing devices geographically close to users. However, when there is no infrastructure or when it is overloaded, a UAV equipped with an onboard computer can be used to process the requests solicited by end-users.

The aim of this chapter is to provide tools for the planning of an aerial infrastructure that will complement the resources of terrestrial fog nodes. The aerial nodes to be considered are rotary-wing drones that can land on limited spaces or hover maintaining the same position. The UAVs are equipped with onboard computers, in charge of processing end-user requests, as well as battery. Connections to ground nodes are wireless and the channel conditions depend on the distance and line-of-sight (LoS) between the UAVs and the ground nodes. The infrastructure provider needs to decide what areas the UAVs will cover. A UAV will be dispatched to the given location at the specified time and will operate until there are no more requests in the area, or the battery can no longer support the processing of requests. UAVs do not fly between locations because, as seen in the previous chapter, flying considerably reduces their potential time of operation, which would lead to the requirement of a larger number of UAVs.

UAVs can be in three different states: turned-off, stand-by, and processing. In the turned-off state, a UAV does not consume energy and cannot be used, while in the stand-by state it is turned on but does not process any workload. In the processing state,

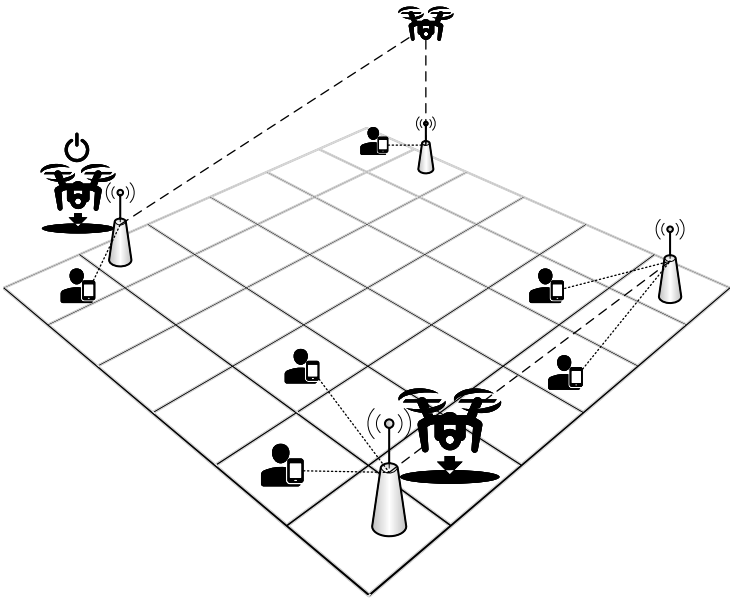


Figure 7.1: Visual example of the system model.

the onboard computer processes users' requests. Rotary-wing UAVs can operate either on the ground or hovering. When on the ground, UAVs save the energy required for propulsion, thus extending operational time. When hovering, energy consumption is high, but connectivity with more distant users on the ground is improved by increasing LoS links. In case a UAV switches to stand-by state, it can benefit from landing to save energy until it will again be required to process data, which will only be possible if on a rooftop or a drone landing pad. This chapter considers different deployment scenarios that define if the UAV remains on the ground constantly, only during stand-by, or the UAV is constantly hovering.

While in processing state, a UAV can receive requests submitted by end-user devices. Mobile devices submit requests first to a nearby BS. The UAV connects to this BS, receives the workload to be processed, process it, and sends the results back to the BS. All requests have delay constraints that limit the time available for transferring the workload; if a UAV is too far from the BS, the wireless channel may also prevent a UAV from processing the request.

Figure 7.1 illustrates the system model, with four BSs, three UAVs, and various mobile users. The UAV at the top of the figure is hovering and providing its computational resources to two ground BSs, processing the requests of two users. Since it is flying, the line-of-sight with the ground nodes is improved and it can communicate with more distant BSs. However, the delay in communication with distant BSs can be long due to the path loss, thus preventing the UAV from processing distant requests. The UAV at the bottom of the figure is also in the processing state, but it has landed to reduce the energy required for propulsion. The UAV in the left part of the figure is in the stand-by state, although it will eventually be switched to the processing state.

This chapter introduces a solution for a specific fog node location problem that determines where fog nodes mounted on UAVs should be deployed from a set of candidate locations. The location problem is multi-criterial; the main goal is to process the highest

number of requests; a second objective is to reduce the costs, i.e., to minimize the number of UAVs required; a third goal is to reduce overall delay, favoring locations where UAVs will remain closer to the demands, favoring good wireless channel conditions for aerial fog nodes and BSs. The input for the problem is the set of potential locations where UAVs can hover or land, the requests to be met, their origin and time, and the number of available UAVs for deployments. The output is the number of UAVs used, their locations, and their states throughout all time intervals. If the fog node location problem is solved, UAVs can be deployed to the best positions, thus improving the service for applications that require low latency.

7.3 Formulation

This section formulates the fog node problem as an ILP model. The solution for the problem includes the locations where UAVs will be deployed for the fulfillment the three objectives of the problem. The mathematical notation in Table 7.1 will be used in the formulation of the problem. Pre-processing of data is required: the altitude and coordinates of UAV candidate positions and BSs are used to calculate physical distance and, consequently, the wireless channel state. The constant D_{rl} is the transmission delay for processing the request r by a UAV located at location l; the value is calculated using the channel model and the number of bytes to be transmitted. Another pre-processing requirement is the energy consumed by the UAV either in processing or in the stand-by state. The formulation assumes discretized time, and the energy model is used to calculate the power of a UAV, with the energy consumption calculated for the interval when in processing (E^{PROC}) and in stand-by (E^{STAND}) states.

The ILP model is given by Equations (7.1)–(7.14):

$$maximize \sum_{r \in \mathcal{R}} \sum_{l \in \mathcal{L}} \mu_{rl} \tag{7.1}$$

$$minimize \sum_{u \in \mathcal{U}} \sum_{l \in \mathcal{L}} \gamma_{ul} \tag{7.2}$$

$$minimize \sum_{r \in \mathcal{R}} \sum_{l \in \mathcal{L}} D_{rl}\ \mu_{rl} \tag{7.3}$$

$$\sum_{l \in \mathcal{L}} \mu_{rl} \leq 1, \forall r \in \mathcal{R} \tag{7.4}$$

$$\mu_{rl} D_{rl} \leq D_r^{MAX}, \forall r \in \mathcal{R}, \forall l \in \mathcal{L} \tag{7.5}$$

$$\sum_{r \in \mathcal{R} | X_{rt}=1} \mu_{rl} P_r \leq \mathcal{K} \sum_{u \in \mathcal{U}} p'_{ult}, \forall l \in \mathcal{L}, \forall t \in \mathcal{T}, \tag{7.6}$$

$$p_{ut} + s_{ut} + i_{ut}^{start} + i_{ut}^{end} = 1, u \in \mathcal{U}, t \in \mathcal{T} \tag{7.7}$$

Table 7.1: Notation used in the fog node location problem formulation.

Notation	**Description**
	INPUT
$\mathcal{T}$	Set of discrete time intervals: $\mathcal{T} = \{1, 2, \ldots, T\}$
$\mathcal{L}$	Set of candidate locations for UAVs: $\mathcal{L} = \{1, 2, \ldots, L\}$
$\mathcal{R}$	Set of requests: $\mathcal{R} = \{1, 2, \ldots, R\}$
$\mathcal{U}$	Set of UAVs for deployment: $\mathcal{U} = \{1, 2, \ldots, U\}$
$X_{rt}, r \in \mathcal{R}, t \in \mathcal{T}$	1 if request r is made at time t, 0 otherwise
$D_{rl}, r \in \mathcal{R}, l \in \mathcal{L}$	delay if request r is processed at the location l
D_r^{MAX}	Maximum delay tolerated by request r
P_r	Processing requirement of request r in processing units
$\mathcal{K}$	Capacity of a UAV server in processing units per time interval
E^{BAT}	Total battery capacity of a UAV
E^{PROC}	Energy required by a UAV in processing state during one time interval
E^{STAND}	Energy required by a UAV in stand-by state during one time interval
	DECISION VARIABLES
$\gamma_{ul}, u \in \mathcal{U}, l \in \mathcal{L}$	1 if UAV u is assigned to location l, 0 otherwise
$\mu_{rl}, l \in \mathcal{L}, l \in \mathcal{L}$	1 if request r is processed by a UAV at l, 0 otherwise
$p_{ut}, u \in \mathcal{U}, t \in \mathcal{T}$	1 if UAV u is in processing state at time t, 0 otherwise
$s_{ut}, u \in \mathcal{U}, t \in \mathcal{T}$	1 if UAV u is in stand-by or at time t, 0 otherwise
$i_{ut}^{start}, u \in \mathcal{U}, t \in \mathcal{T}$	1 if UAV u is in a turned-off state at all time intervals $t*$ such that $t* \leq t$, 0 otherwise
$i_{ut}^{end}, u \in \mathcal{U}, t \in \mathcal{T}$	1 if UAV u is in a turned-off state at all time intervals $t*$ such that $t* \geq t$, 0 otherwise
$p'_{ult}, u \in \mathcal{U}, l \in \mathcal{L}, t \in \mathcal{T}$	1 if UAV u is at location l at time t in processing state, 0 otherwise

$$p_{ut} + s_{ut} \leq p_{u(t-1)} + s_{u(t-1)} + i^{start}_{u(t-1)}, u \in \mathcal{U}, t \in \mathcal{T}, t \neq 1 \tag{7.8}$$

$$i^{start}_{ut} \leq i^{start}_{u(t-1)}, u \in \mathcal{U}, t \in \{2, \ldots, T\} \tag{7.9}$$

$$i^{end}_{ut} \leq p_{u(t-1)} + s_{u(t-1)} + i^{end}_{u(t-1)}, u \in \mathcal{U}, t \in \{2, \ldots, T\} \tag{7.10}$$

$$\gamma_{ul} + p_{ut} - 1 \leq p'_{ult}, u \in \mathcal{U}, l \in \mathcal{L}, t \in \mathcal{T} \tag{7.11}$$

$$2p'_{ult} \leq \gamma_{ul} + p_{ut}, u \in \mathcal{U}, l \in \mathcal{L}, t \in \mathcal{T} \tag{7.12}$$

$$\sum_{l \in \mathcal{L}} \gamma_{ul} \leq 1, \forall u \in \mathcal{U} \tag{7.13}$$

$$E^{PROC} \sum_{t \in \mathcal{T}} p_{ut} + E^{STAND} \sum_{t \in \mathcal{T}} s_{ut} \leq E^{BAT}, u \in \mathcal{U} \tag{7.14}$$

The formulation has three objective functions (Equations (7.1)–(7.3)). Equation (7.1) maximizes the number of processed requests. Equation (7.2) minimizes the number of employed UAVs (budget). Equation (7.3) minimizes the delay. Prioritization of these objectives is possible, so a multi-level programming approach has been adopted (Section 2.4), with the highest priority objective being the first addressed, and the others are optimized only considering the solutions in the Pareto front.

Equations (7.4)–(7.14) model the constraints of the problem. Equation (7.4) ensures that requests are processed only once. Equation (7.5) guarantees the maximum delay of a request is not surpassed, while Equation (7.6) guarantees that the requests processed at a location will be limited to the capacity of the UAVs in the processing state at the location and the time interval. The states of the UAVs and their transitions are given by Equations (7.7)–(7.12). Equation (7.7) guarantees that any UAV is in a single state at any given time, while Equations (7.8)–(7.10) model the state transitions. Equations (7.11)–(7.12) define the correct value of variable p'_{ult}. Finally, Equation (7.13) guarantees that a UAV will be assigned to only one location, while Equation (7.14) guarantees that the energy consumed by any UAV is not greater than the battery capacity.

This formulation was coded in a solver, which was able to handle simulations for a few UAVs. This scalability issue is the result of the exponential growth of the number of constraints in the model. If there is a single UAV to be deployed, there are L possible locations for this deployment. However, deploying two UAVs at the L possible locations leads to L^2 possibilities. In general, for U UAVs and L locations, L^U pairs of UAVs and locations must be analyzed. Inputs with several requests and UAVs impose heavy processing load to the solver, and solutions for large networks cannot be obtained, even with days of computation.

7.4 Sequential UAV Fog Node Location

This section presents the Sequential UAV Fog Node Location (SUL) algorithm which can obtain an efficient solution for the fog node location problem. The algorithm optimizes the same objectives of the ILP model while providing a scalable heuristic. The SUL algorithm has a greedy approach to achieve these goals. To maximize the number of processed requests, the algorithm takes advantage of the patterns of demands. The number of requests vary with time and location in a non uniform way, with peaks in demands common and tending to take place in small areas and last for short periods. Therefore, locations and time intervals in which peaks of demands take place should receive fog nodes first. The SUL algorithm addresses these peaks using a greedy approach.

Instead of considering all possible combinations of UAVs and locations, the SUL algorithm builds the solution by sequentially placing UAVs. The first deployed UAVs will process the largest possible number of requests. Such an operation is more scalable than the ILP formulation in various ways. First, the second objective (reduction of costs) is not optimized for each iteration, but rather addressed by stopping the SUL algorithm when all requests have been processed to avoid the employment of unnecessary UAVs. Second, the number of Constraints (7.7)–(7.14) is reduced by a factor of U. Third, the number of pairs of UAV and locations analyzed is greatly reduced, from L^U in the ILP model to $L \cdot U$. The SUL algorithm has at most U iterations, and a single UAV can be matched to L locations, thus, at the end of the SUL algorithm, a maximum of $L \cdot U$ pairs will have been analyzed, whereas the ILP solver will have analyzed L^U pairs. Finally, the SUL algorithm starts with R requests, but this number decreases for each iteration, thus reducing the number of Constraints (7.4)–(7.5). A direct consequence of this reduction in complexity makes the SUL algorithm scalable, yet producing good results when compared to the ILP model.

The SUL algorithm has the same objectives of the ILP model. The first objective (number of requests processed, Equation (7.1)) as well as the third one (minimization of delay, Equation (7.3)) are optimized during each iteration. When solving the location problem with a single UAV, the fog node is allocated to a location that maximizes the number of requests processed (first objective). If two or more locations could maximize this objective, the algorithm chooses the one where the UAV minimizes the delay of the processed requests (third objective). The second objective (minimization of costs, Equation (7.2)) is implemented by avoiding unnecessary UAVs in the solution: once all requests have been processed, the SUL algorithm terminates.

Algorithm 7.1 details the SUL algorithm, which consists of an initialization phase (Line 1), followed by a main loop (Line 2). For each iteration, a single UAV is positioned using the proposed formulation. Requests served by this UAV are not considered during further iterations, this reduces the number of locations to be served by UAVs in future iterations. The variable $availableUAVs$ stores the number of available UAVs, initialized with the value U in Line 1, and updated for each iteration (Line 5). The main loop (Line 2) performs two operations. The first obtains an exact solution to the problem using a single UAV (Line 3) using the same formulation from the Section 7.3. The second objective (Equation (7.2)) is not considered at this stage, since only a single UAV is

considered ($U = 1$). The second operation is to update the set of requests left unserved, removing the ones processed by the UAV that has been just deployed. The algorithm continues until the entire budget has been consumed ($availableUAVs = 0$) or all requests have been processed.

```
availableUAVs ← U;
while availableUAVs > 0 and ℛ ≠ ∅ do
    Solves single-UAV formulation;
    Removes requests r from ℛ if μ_rl = 1 for any l ∈ ℒ;
    availableUAVs ← availableUAVs − 1;
```

Algorithm 7.1: SUL algorithm.

In the SUL algorithm, an exact solution of the single-UAV formulation is obtained up to U times. These exact solutions require an exponential time complexity. Therefore, the SUL algorithm has an exponential complexity, but its design allows it to be scalable, different from solving the complete formulation presented in Section 7.3.

7.5 Performance Evaluation

Evaluating a fog node location solution for real deployments is a hard and costly task due to the large number of UAVs and requests. To evaluate the potential gains from using rotary-wing UAVs as fog nodes, the formulation in Section 7.3 and the SUL algorithm were coded using Python and the Gurobi Optimizer solver. This section discusses the results obtained from several simulations and is organized as follows. First, this section presents the simulation settings: UAV deployment scenarios, energy and channel models (Subsection 7.5.1), and workload (Subsection 7.5.2). Last, Subsection 7.5.3 validates the SUL algorithm by comparing its results with those obtained by the ILP model, and Subsection 7.5.4 discusses the results of the employment of rotary-wing UAVs as fog nodes.

7.5.1 UAV characterization

Rotary-wing UAVs can operate in a variety of setups by changing their hovering height or operating on the ground. On the ground, UAVs save a considerable amount of energy. However, landing is not always possible and, when it is possible, it tends to reduce connectivity with distant nodes. Therefore, we evaluated five different UAV deployment scenarios.

Table 7.2 summarizes the deployment scenarios, indicating if a UAV is on the ground or hovering during the stand-by and processing states. In the `ground` deployment, UAVs are on the ground during both stand-by and processing states, using a landing drone pad close to the BS at 25 meters in height. In the scenarios identified by `hoverH`, UAVs hover during stand-by and processing states, with the height `H` in meters during hovering. Finally, in the scenarios identified by `mixH`, the UAVs remain on the ground during stand-by state, but hover at height `H` during the processing state .

The energy consumption of the UAV was calculated using the power models derived in [3], detailed in Subsection 2.5.3. The estimated power $P(h)$ in watts for hovering

Table 7.2: UAV deployments.

	stand-by state	**processing state**
`ground`	on the ground	on the ground
`hover30`	hovering at 30 m	hovering at 30 m
`hover50`	hovering at 50 m	hovering at 50 m
`mix30`	on the ground	hovering at 30 m
`mix50`	on the ground	hovering at 50 m

depends on the height h (in meters) and is calculated as $P(h) = 13.0397h + 196.8490$. The energy for the processing state is calculated as the energy for stand-by, added to the energy consumed by a Jetson TX2, a typical onboard computer for drones, with an average of 7.5 W power. A 34200 mAh/22.8 V battery was considered [75].

The propagation delay is calculated using the number of bits to be transmitted and the data rate of the channel model. The wireless channel state is modeled by the path loss model proposed by 3GPP [2], described in Subsection 2.5.4. The path loss is used to calculate the channel capacity (in bits per second) using the Shannon-Hartley theorem. In this chapter, the transmission power is 23 dBm, the noise -60 dBm, and the bandwidth 10 mHz [2, 99, 20]. The path loss and the channel capacity are calculated for each pair request-location to obtain the delay D_{rl}. UAVs cover a maximum 4 km radius [2]: if $d_{2D} > 4000\ m$, D_{rl} is considered infinite.

7.5.2 Workload

The locations and the workload demands were taken from the data sets [8, 78] reviewed in Section 3.5. The BSs are used as the candidate locations for fog nodes, considering 100 possible locations for deployment of UAVs. In this chapter, individual requests are considered, and the constant Z (Section 3.5) was used to obtain such a value. The altitude of BSs was obtained from the Shuttle Radar Topography Mission [37] data set, and it is used to calculate the vertical distance between UAVs and BSs. The BS positions are the candidate locations for the deployment of UAV fog nodes. A 24-hour interval is considered, producing $\mathcal{T} = \{1, 2, \ldots, 144\}$ due to the aggregation into 10-minute intervals.

In 5G and 6G networks, end-to-end latencies range from a few microseconds to a few milliseconds [108] and the transmission of a few kilobytes should not last more than a few milliseconds. To model low-latency applications, this chapter characterizes them by the processing requirements, the number of bytes to be transmitted to the fog node, and the maximum tolerated delay. We have considered applications with low latency requirements, identified as `lowlatency6`, `lowlatency10`, `lowlatency20`, and `lowlatency50`, with delay requirements of 6 ms, 10 ms, 20 ms, and 50 ms, respectively. The requirement is the propagation delay to transmit 100 Kb, a common amount of data for various fog applications [4], over the wireless link. The processing requirement is given in processing units (PUs), and the UAVs have an onboard computer with a capacity of 100 PUs per time slot.

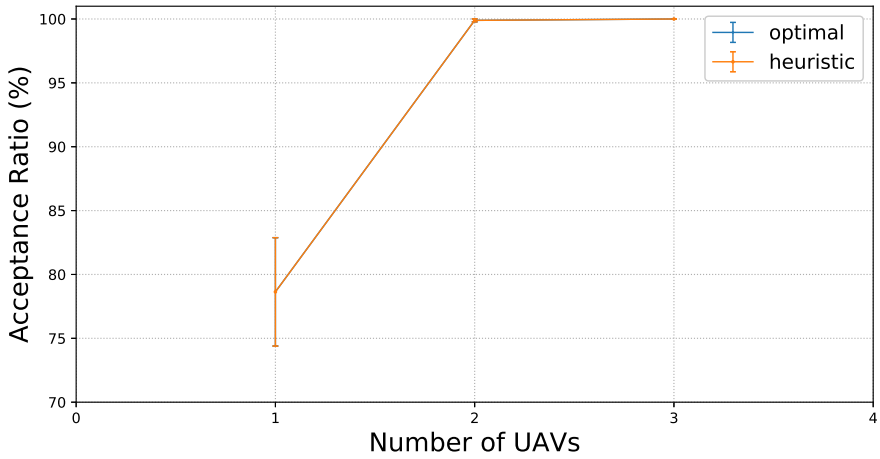


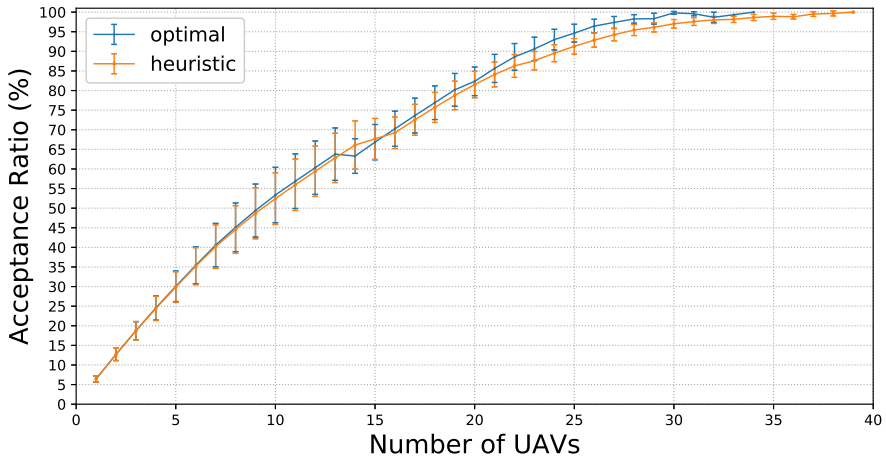
Figure 7.2: Acceptance ratio as a function of the number of UAVs for a `lowlatency6` application and the `ground` deployment.

7.5.3 Validation of the SUL Algorithm

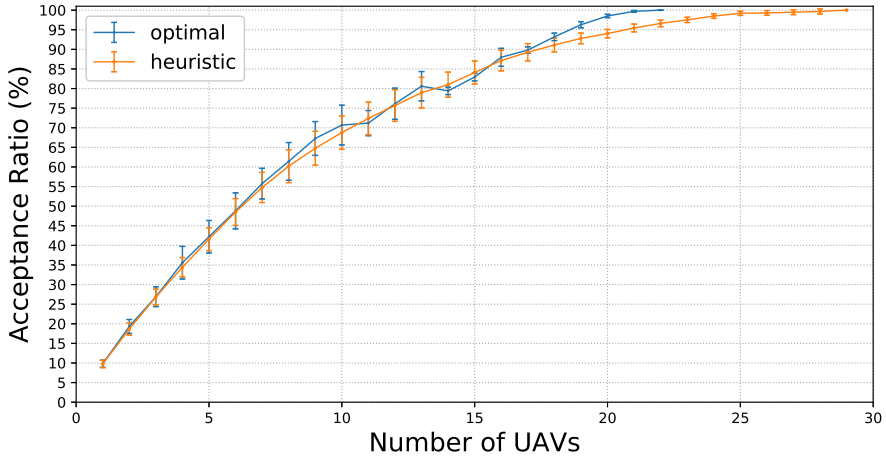
The SUL algorithm was validated by comparing its performance to that of the ILP model. The optimal solution had very limited scalability and, for large instances, after a long execution time, crashes occurred due to the high demand of main memory. Therefore, we carried out this validation with a limited number of locations and requests. Sequential executions were started with a single UAV, increasing this number by one until all requests had been processed. In this subsection, the average acceptance ratio is discussed as a function of the number of UAVs available for deployment, and differences between the optimal solution and the SUL algorithm are discussed. Only 5 locations were considered, and the parameter Z was set to 0.0002, reducing the number of requests to less than one hundred. 95 % confidence intervals are displayed in the graphics.

Figure 7.2 shows the acceptance ratio for the `ground` deployment. Only a `lowlatency6` application is presented, since other applications produced similar results. This deployment scenario requires the fewest number of UAVs because the operational time is longer. When constantly on the ground, a UAV does not consume any energy in propulsion. Therefore, the battery operational time is greatly extended, allowing the UAV to remain active for long periods and process requests of more users. Consequently, only 2 UAVs were sufficient to deal with all the requests. The SUL algorithm and the optimal solution produced the same results under the `ground` deployment.

Figure 7.3 presents the acceptance ratio for the `hover50` deployment and two different applications. In this deployment scenario, UAVs are constantly flying, which considerably increases their energy consumption. Therefore, to process all requests, more UAVs were required compared to the `ground` deployment. The delay requirement also impacts on the number of UAVs: the lower the latency required, the greater is the number of UAVs. This is due to the fact that strict delay requires higher data rates, which is realized when UAVs are deployed in close proximity to end-users. This close proximity increases the number of UAVs needed. The maximum number of UAVs to process 100 % of the



(a) `lowlatency6`.

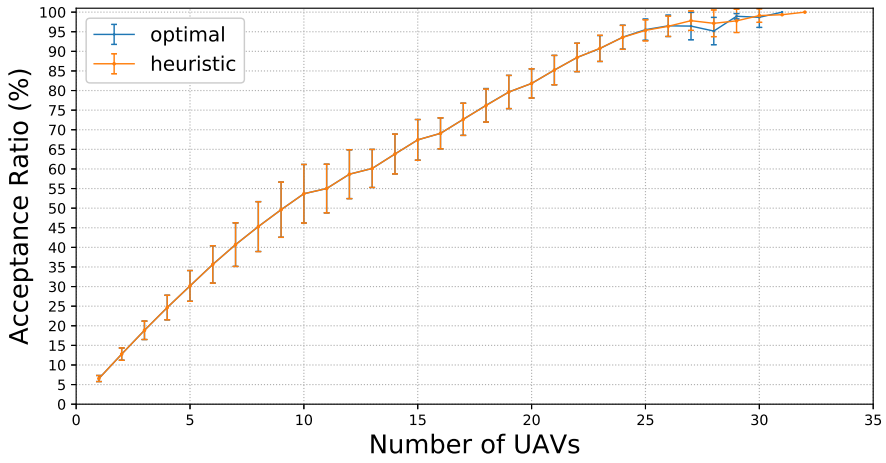


(b) `lowlatency50`.

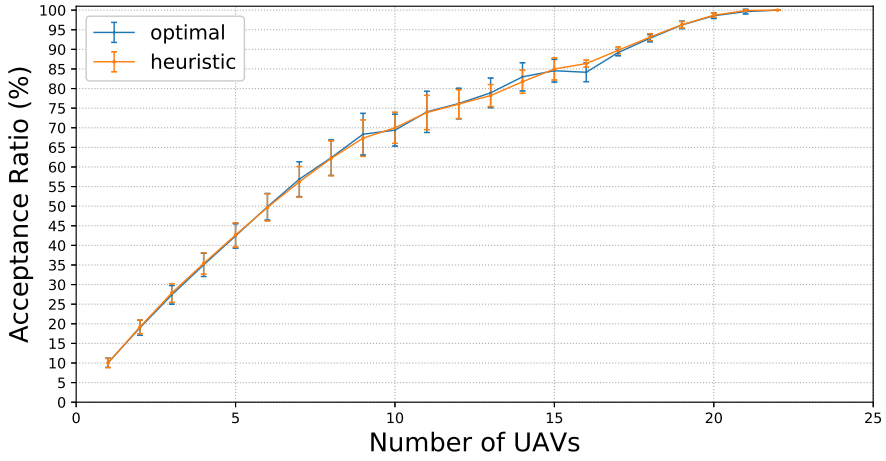
Figure 7.3: Acceptance ratio as a function of the number of UAVs for `hover50` deployment and different applications .

requests was greater with the SUL algorithm than it was for the optimal solution. This happens because the SUL algorithm evaluated a single UAV at time, while the optimal solution considered all of them simultaneously. This meant that the SUL algorithm could leave specific requests unserved that, in a later iteration, would require the deployment of UAVs to process only a few requests to guarantee 100 % acceptance of requests. As a consequence, up to seven extra UAVs were needed for `hover50` conditions. Since the budget of a fog provider typically does not cover 100 % of users, such discrepancies can potentially be neglected.

The last analysis concerns the acceptance for the `mix50` deployment scenario, shown in Figure 7.4. In this deployment, UAVs take advantage of the idle time in stand-by to remain on the ground until they are needed. The savings in stand-by allow a significant reduction in the number of UAVs needed, especially when a more strict delay is required (Figure 7.4a). If shared landing pads are installed to receive idle UAVs in a city, the



(a) `lowlatency6`.



(b) `lowlatency50`.

Figure 7.4: Acceptance ratio as a function of the number of UAVs for `mix50` deployment and different applications .

unnecessary energy consumption can be avoided. In the `mix50` deployment scenario, the differences between the SUL algorithm and the optimal solution are minor, which demonstrates the efficiency of the proposed solution.

The results show that the energy consumption play a crucial role in the development of a fog computing infrastructure since it limits the operational time and increases the number of required UAVs. The results also showed that the SUL algorithm produced results similar to those obtained by the optimal solution. Certain drawbacks do exist in a few cases, suggesting that the proposed solution may require the employment of additional UAVs to cope with all requests in specific deployment scenarios. In these cases, the SUL algorithm can be executed for a much larger number of requests. Moreover, if a 100 % acceptance ratio is not required, the SUL algorithm produces results very close to the optimal ones.

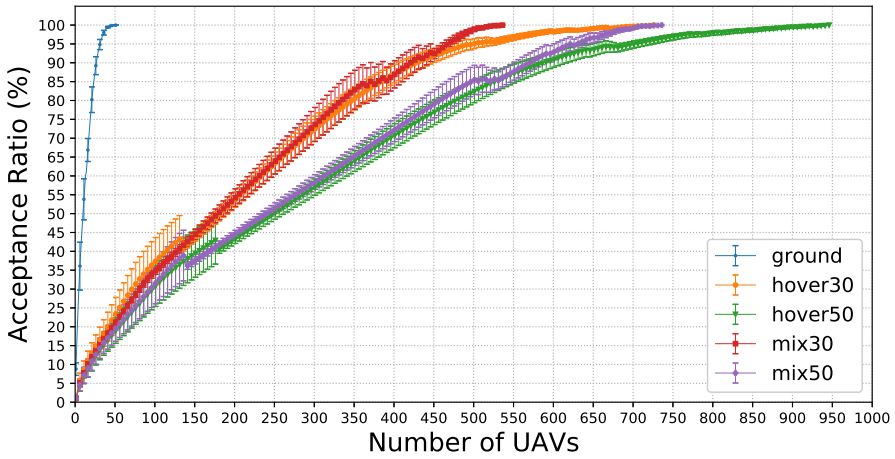


Figure 7.5: Acceptance ratio as a function of the number of UAVs for the `lowlatency6` application.

7.5.4 Qualitative Analysis

The analysis in this subsection considered 100 locations and Z equals to 0.0005, which produced thousands of requests. The metrics are the average acceptance ratio, the average number of UAVs to process all requests, the average delay, and the average data rate. The results are reported as a function of the number of available UAVs. As in the previous subsection, the SUL algorithm was started with a single UAV, in subsequent iterations increasing the number of available UAVs by one until all requests had been processed. 95 % confidence intervals are displayed in the graphics.

Figures 7.5–7.8 shows the acceptance ratio as a function of the number of UAVs. As expected, the `ground` deployment requires the smallest number of UAVs for the service of all users. The results of the remaining deployment scenarios show the effect of the UAV height in the decision about location. As seen in Subsection 7.5.1, the energy consumption increases with the height of the UAV. Thus, UAVs with deployment of `hover30` and `mix30` consume less energy and remain operational longer, therefore processing more requests, and leading to fewer employed UAVs than do the deployments of `hover50` and `mix50`. This trend is observed for all applications.

Figures 7.5–7.8 also shows the impact of different delay requirements in the solution. For `lowlatency6`, an application with very strict delay requirements, more than 700 UAVs were needed with `hover30` deployment, and more than 900 for the `hover50`. However, when delays are more flexible (Figure 7.8), these values are less than 300 and 400 UAVs, respectively. As seen in Figures 7.2–7.4, this is explained by the possibility of serving distant users with different delay requirements. When a UAV communicates with two different LoS ground nodes, the connection to the furthest one will function at a lower data rate because the path loss increases with distance. When limited delays are required, UAVs cannot process requests from distant users, which leads to solutions with UAVs at more locations. Another aspect to be noted is the similarity of the results produced by the `lowlatency20` and `lowlatency50` applications (Figures 7.7–7.8): the number of UAVs

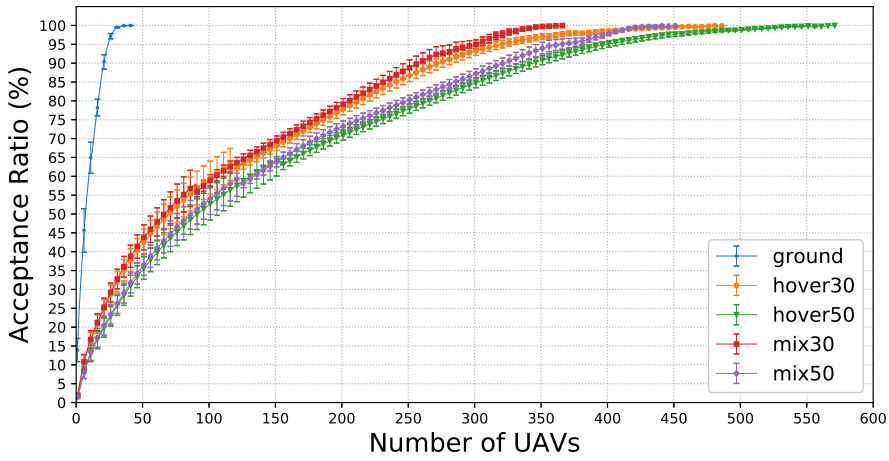


Figure 7.6: Acceptance ratio as a function of the number of UAVs for the `lowlatency10` application.

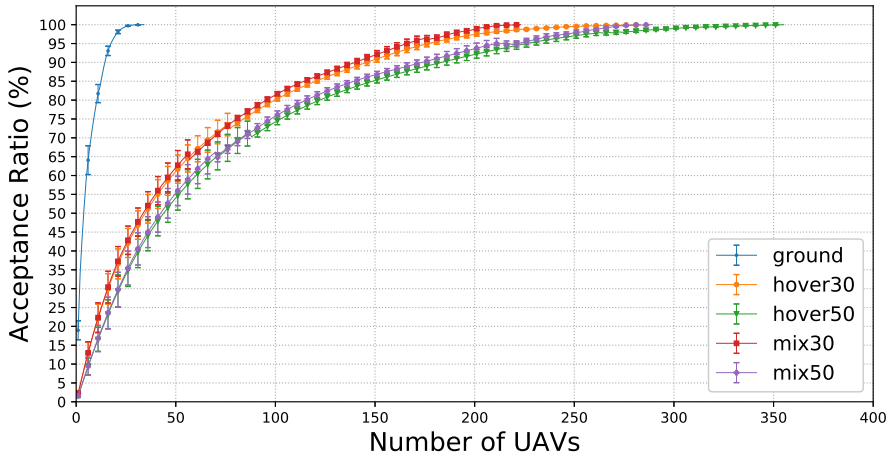


Figure 7.7: Acceptance ratio as a function of the number of UAVs for the `lowlatency20` application.

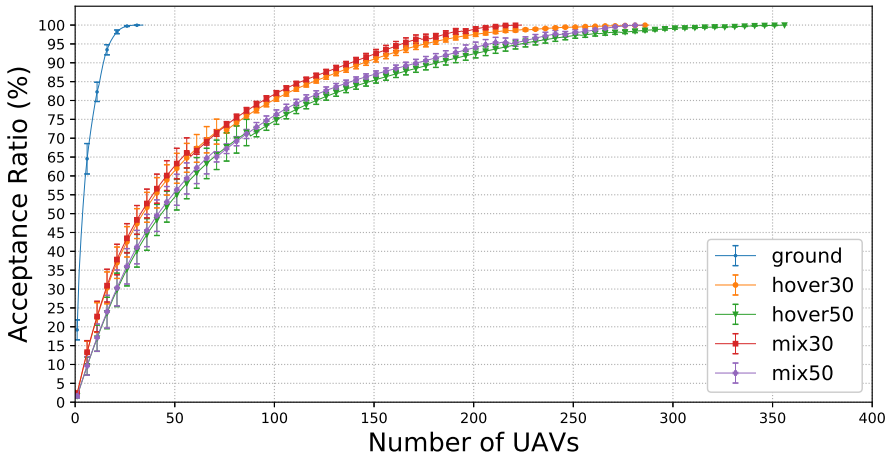


Figure 7.8: Acceptance ratio as a function of the number of UAVs for the `lowlatency50` application.

could not be reduced when the delay was relaxed from 20 ms to 50 ms. This was due to the fact that wireless communication is limited when serving very distant users and, even with a relaxation of the delay requirement, UAVs are not be able to process requests made by distant users.

Figure 7.9 shows the average number of UAVs needed for processing 100 % of the requests. A comparison between `hover30` and `mix30`, or between `hover50` and `mix50`, reveals that the budget can be considerably reduced if landing is possible during stand-by. The reduction reaches nearly 20 % for all deployments and altitudes. Although building landing pads or using rooftops in all locations can be expensive or even impossible, UAVs at nearby locations should be able to share landing pads in strategic places in a city, thus reducing overall deployment costs.

The average delay is shown in Figures 7.10–7.13. In the first iterations, the SUL algorithm prioritizes locations where UAVs process as many requests as possible, as long as the delay is respected, this produces an initial trend to increase. After several iterations, however, UAVs deal with fewer requests and can be deployed where the delay is reduced (objective function in Equation (7.3)), this reduces the delay as a function of the number of UAVs. The delay values produced by deployment scenarios with UAVs at the same height (`hover30` and `mix30`, `hover50` and `mix50`) are very similar. This is due to the same distance between UAV and BSs that produces the same wireless channel conditions during processing state. Furthermore, UAVs closer to the ground reduce the length of the wireless link, thus allowing higher data rates, and reducing the delays. Every iteration of the SUL algorithm places UAVs where the delay is reduced, optimizing the average delay to at most about 75 % of the tolerated delay. When more flexible delays are tolerated, such as 20 and 50 ms, the average delay remains at about 10 ms, which provides a reasonable quality wireless channel for end-users.

To further explore the impact of the deployment scenario, Figure 7.14 presents the data rates for the `lowlatency6` and `lowlatency50` applications. The `ground` scenario produces the highest data rates, which is explained by the proximity with ground nodes

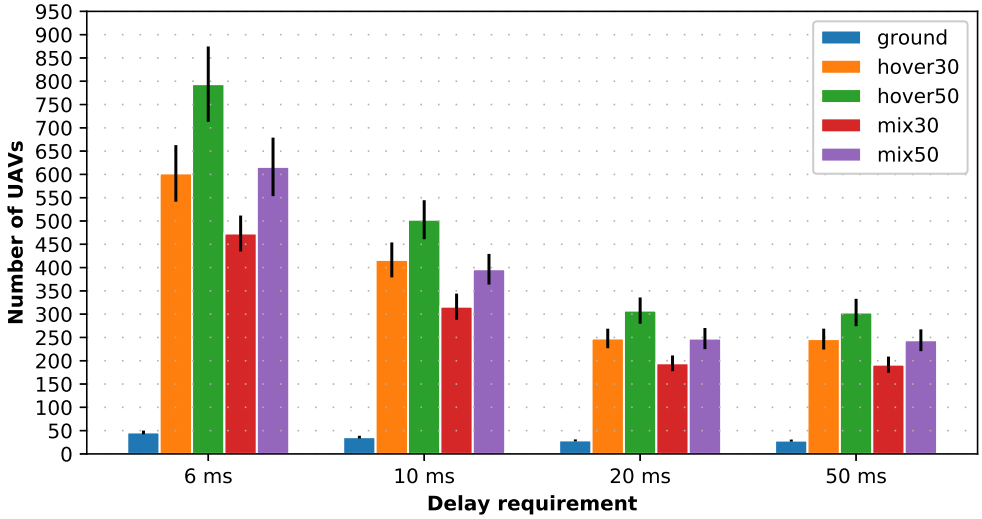


Figure 7.9: Average number of UAVs to process all requests.

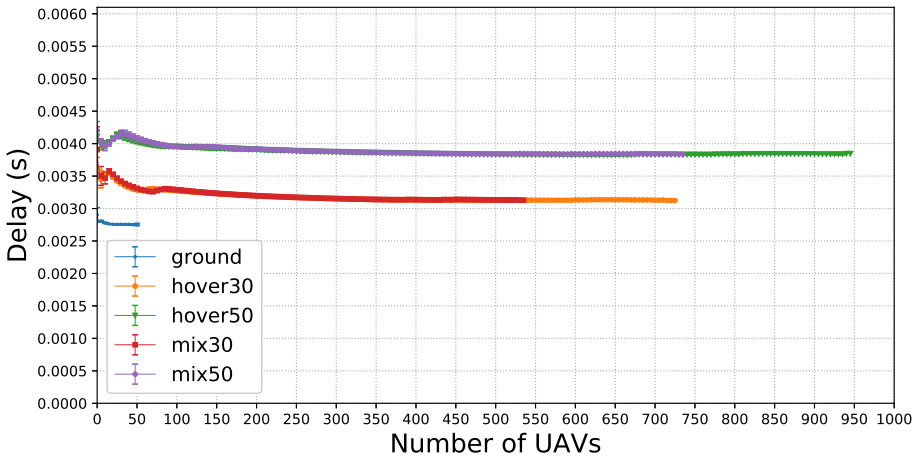


Figure 7.10: Average delay as a function of the number of UAVs for the `lowlatency6` application.

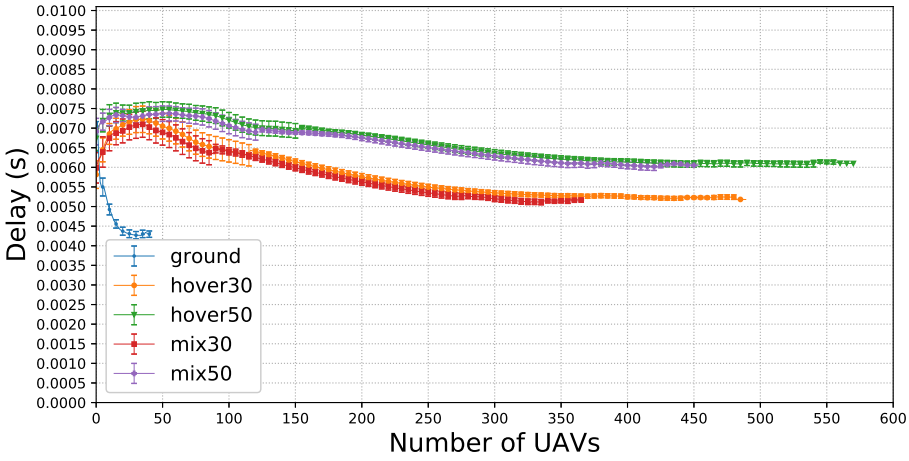


Figure 7.11: Average delay as a function of the number of UAVs for the `lowlatency10` application.

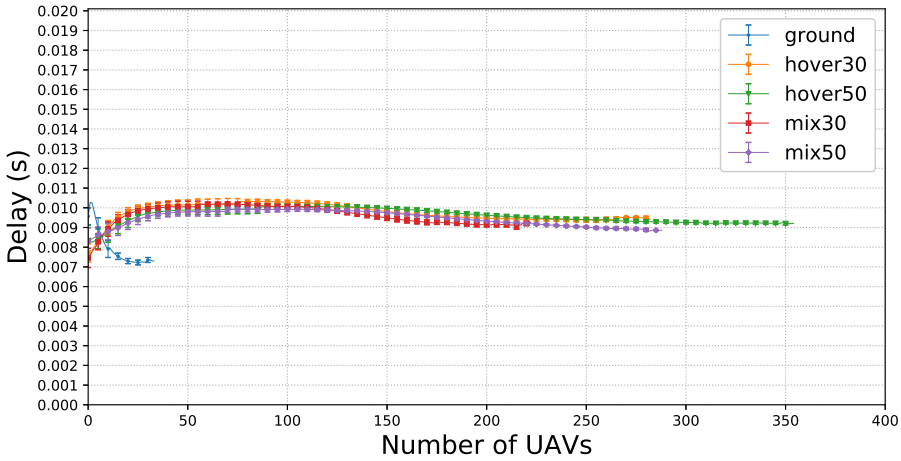


Figure 7.12: Average delay as a function of the number of UAVs for the `lowlatency20` application.

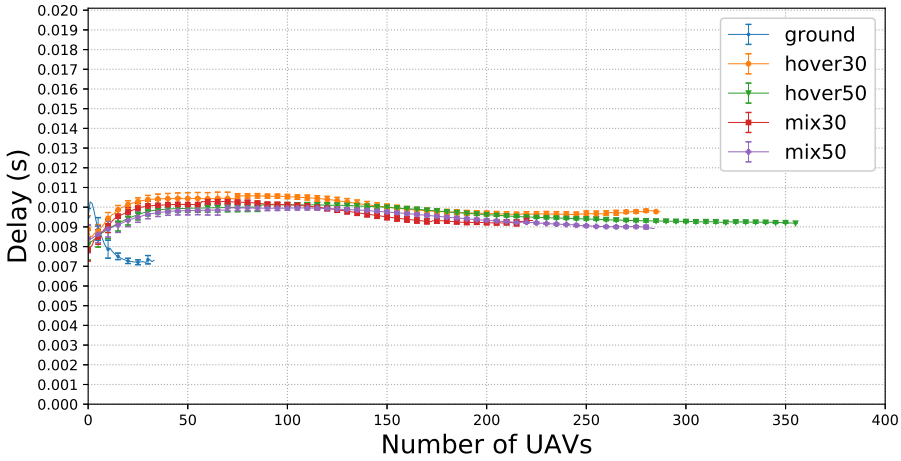


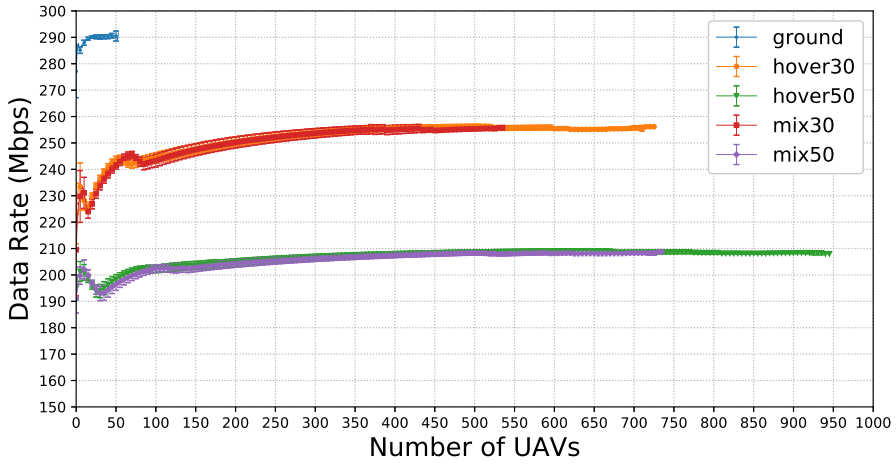
Figure 7.13: Average delay as a function of the number of UAVs for the `lowlatency50` application.

that reduces the path loss. Nonetheless, even the smaller data rates are sufficient to provide fast communications between aerial and terrestrial nodes. The data rate is also affected by stricter delay requirements. For `lowlatency6`, in which 100 Kb should be transmitted in under 6 ms, data rates above 200 Mbps are needed, which requires that the UAVs should be deployed very close to the BSs serving the users. On the other hand, data rate less than 100 Mbps are needed under `lowlatency50`. These results show that the more restricted delays, the closer to the users the aerial fog nodes should be.

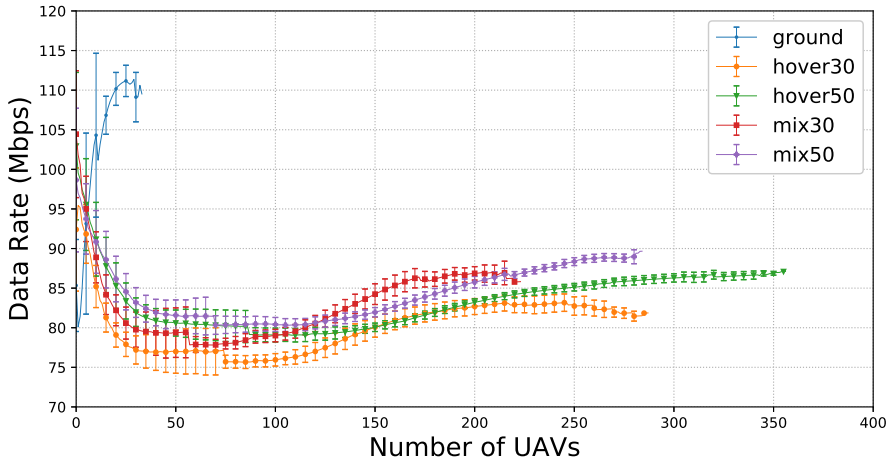
The main findings in this subsection are summarized as follows. First, as seen in the previous chapters, the energy consumption of UAVs is a crucial aspect to be considered when deploying UAVs as fog nodes. Second, a trade-off between physical distance and data rate must be addressed by fog providers. UAVs at higher altitudes have more LoS links with terrestrial users, but the higher altitude also implies a greater path loss, thus reducing the data rate. Therefore, when deploying a fog node on a UAV, several factors must be jointly considered: UAV energy efficiency, delay or data rate requirements, and distribution of users in time and space. There is no one-fits-all solution for any fog infrastructure. Third, UAVs can potentially share landing places for stand-by under `mix` deployments, reducing about 20 % in relation to `hover` deployments. Finally, the stricter the delay requirements are, the greater the number of UAVs will be needed to bring the computing resources close to users.

7.6 Conclusions

UAVs are potential candidates for the building of future fog computing infrastructures. This chapter has studied the fog node location problem, a network design problem that aims at choosing the best locations for fog nodes to improve the service provided for end-users. We formulated the problem as a multi-objective ILP model and proposed the Sequential UAV Fog Node Location algorithm. To discuss the importance of the decision about locations and to validate the proposed algorithm, we performed simulations using



(a) `lowlatency6`.



(b) `lowlatency50`.

Figure 7.14: Average data rate as a function of the number of UAVs for different application classes.

realistic data, and the results showed the feasibility of UAVs for supporting applications with very restricted latencies. Results showed that UAVs can process requests and communicate with end-users with very low latencies. Results suggest that a key limitation in the deployment of UAVs is energy consumption which limits the operational time so that fewer requests can be processed by a single UAV. The provisioning of landing pads for rotary-wing UAVs by fog providers, even if used only when UAVs are idle awaiting requests, could significantly extend operational time. Moreover, we showed that high data rates can be achieved by rotary-wing UAVs, which can make them effective in fog computing.

Chapter 8

Location of Fog Nodes mounted on Fixed-wing UAVs

8.1 Overview

Different types of unmanned aerial vehicles (UAVs) have different characteristics; the choice of a specific UAV type directly affects the operation as fog nodes. Previous authors considered mostly the employment of rotary-wing UAVs [105, 113, 52, 53, 64, 111]. One issue in the use of such rotary-wing UAVs is their high energy consumption, limiting the time they remain operational, but fixed-wing UAVs provide an alternative with greater autonomy and longer operational periods since the broad fixed wings that generate lift reduce the power required from the engine. This extended operational time is essential for the fog nodes to deal with variable loads. However, fixed-wing UAVs cannot hover; so their trajectory must be carefully planned.

One essential decision for the deployment of fog nodes is where to locate them, as this will determine which users can be served by this fog infrastructure. For fog nodes mounted on UAVs, this decision is especially important, since the UAVs will remain active for only a short period at the destination location. The fog node location problem for rotary-wing UAVs was explored in the previous chapter, and this chapter investigates the employment of fixed-wing UAVs. The longer endurance of fixed-wing UAVs can greatly improve the delivery of fog services, extending the time the UAVs remain in the air and, consequently, serving more users on the ground. In order to evaluate the use of fixed-wing UAVs as fog nodes, this chapter aims at answering the question *how should fixed-wing UAVs should be positioned to provide a fog computing infrastructure to deal efficiently with variable demands in relation to time and space, as well as maximize the number of processed requests?* In an attempt to answer this question, this chapter introduces a solution for the fog node location problem which can determine the location of fog nodes mounted on fixed-wing UAVs and the period they should be active. With this in mind, this chapter introduces the Spatio-Temporal UAV Fog Node Location (STUFog) algorithm, which analyzes the variability of requests and attempts to locate UAVs at those locations facing high demands, thus maximizing the number of requests processed, minimizing costs, and reducing latency.

This chapter models the problem of locating fog nodes mounted on fixed-wing UAVs with an integer linear programming (ILP) formulation. The mathematical formulation models all characteristics of these aircraft as well as the variability of end-user requests in relation to time and space. Circular trajectories for fixed-wing UAVs and variable wireless channels are considered. This chapter also introduces the STUFog algorithm to solve the fog node location problem for instances with several locations, requests, and UAVs. This chapter employed data from real users in a cellular network to evaluate the proposed algorithm, considering circular trajectories with different radii for the fixed-wing UAVs. The results are compared to those for rotary-wing UAVs. Results showed that fixed-wing UAVs can balance the quality of the wireless links with energy consumption, thus reducing the number of UAVs required.

The remainder of this chapter is organized as follows. Section 8.2 introduces the system model adopted. Section 8.3 presents the linear programming formulation of the problem. Section 8.4 introduces the STUFog algorithm. Section 8.5 presents the experimental setting and numerical evaluation of the proposed algorithm. Lastly, Section 8.6 concludes this chapter.

8.2 System Model

A fog computing infrastructure relies on fog nodes, which are small facilities that host processing, storage, and networking equipment. Such nodes can reduce latency of end-user requests by the avoidance of the propagation delays to access data centers in the cloud. UAVs can help by supporting services at locations with fog nodes overloaded, or where the deployment of fixed nodes is not feasible. However, UAVs typically have limited processing and battery capacity. Therefore, the deployment of fog nodes on UAVs requires proper planning; otherwise the UAVs will quickly consume large amounts of energy and become inoperative.

To overcome autonomy restrictions of battery-powered UAVs, the use of tethered UAVs has been considered [60]. These UAVs are connected to ground stations (GSs) by a tether which furnishes energy and connectivity. However, these tethers reduce mobility and limit the distance between UAVs and GSs. This chapter evaluates the employment of both tethered and untethered UAVs.

End-users are mobile and visit several locations; they can submit requests to the fog infrastructure at any time. Their demands vary in time and space: this requires the positioning of fog nodes at different locations and their activation for different periods. This chapter investigates the use of fixed-wing UAVs for the deployment of resources to various different locations. They cannot hover nor land in limited spaces; the solution proposed here assumes that UAVs perform circular trajectories with a predefined radius centered at a specific location. Once the UAV starts flying, it stays powered until there is no pending request, or the battery has been depleted. The UAV flies continuously, without pauses. The available trajectories cannot overlap, and different UAVs should not share a given trajectory to prevent collisions.

Requests made by end-users are characterized by the minimum data rate of transmis-

sion of a workload to the UAV (data rate requirement), and the number of instructions to be processed (processing requirement). These requests are initially sent to a base station (BS), which then forwards this workload to a nearby UAV. The guarantee of a rapid transfer of content requires the maintenance of a minimum data rate for communication between the BS and the UAV. A failure to maintain this data rate results in the failure of the UAV to process the requests. If no nearby UAV can meet the data rate requirement, a request will be rejected. The quality of transmission (QoT) of a wireless channel is also variable as a result of UAV mobility. UAV is only suitable for processing a request if it can support the required data rate even when located at the point most distant from the BS.

The location of fog nodes is crucial since the number of UAVs for the creation of an infrastructure is limited. The location of the UAVs should be determined so that the processing of end-user requests is independent of the moment these were submitted or their geographical location. The location of fog nodes is thus a special case of the classical facility location problem. The input is a set of locations, and the output the locations where UAVs will be placed. In the fog node location problem, the facilities are fog nodes, and the main objective is to maximize the number of processed requests. In this chapter, facilities only remain operational for short periods since they are UAVs. The formulation of the problem is multi-criterial. First, the number of requests processed should be optimized, guaranteeing that requests will not be rejected. Second, the number of UAVs should be reduced as much as possible, decreasing the deployment costs with the infrastructure. Finally, the average data rate of all processed requests should be maximized to improve the fog service, that can be achieved by locating UAVs close to the end-users.

8.3 Formulation

The location of fog nodes mounted on fixed-wing UAVs is an optimization problem formulated in this chapter as an integer linear programming formulation. The objectives of the formulation are to maximize the number of requests processed, reduce the costs for the infrastructure, and maximize the data rate between UAVs and ground nodes. The decision variables indicate the locations where UAVs fly, the period they will be active, and the UAVs to be used to process each request to optimize the goals of the problem.

The formulation employs the notation presented in Table 8.1. Time is discretized, and each time interval $t \in \mathcal{T}$ has the same duration. The budget is indicated by U, the maximum number of UAVs to be deployed over the L possible locations. Individual requests are established (set $\mathcal{R}$), each one associated with a single location and time interval. Solving the model with accurate results depends on the choice of adequate values of the constants in the formulation. For example, E^{BAT} is the battery capacity and must represent an actual UAV battery, while E^{T} is the energy consumed by a UAV when flying during the length of a single time interval, which must represent the actual consumption of a real aircraft during a typical flight. Moreover, the data rate of the link between UAVs and BSs is pre-calculated using the channel model, and the distance

Table 8.1: Notation used in the formulation.

Input	
Notation	**Description**
$\mathcal{R}$	Set of requests: $\mathcal{R} = \{1, 2, \ldots, R\}$
$\mathcal{L}$	Set of candidate locations for UAVs: $\mathcal{L} = \{1, 2, \ldots, L\}$
$\mathcal{U}$	Set of candidate UAVs: $\mathcal{U} = \{1, 2, \ldots, U\}$
$\mathcal{T}$	Set of discrete time intervals: $\mathcal{T} = \{1, 2, \ldots, T\}$
$D_{rl}, r \in \mathcal{R}, l \in \mathcal{L}$	Data rate to be delivered to request r if it is processed by a UAV at location l
$D_r^{MIN}, r \in \mathcal{R}$	Minimum data rate required by request r
$X_{rt}, r \in \mathcal{R}, t \in \mathcal{T}$	Binary. 1 if request r is made at time t; 0 otherwise
$P_r, r \in \mathcal{R}$	Processing requirement of request r in processing units
$\mathcal{C}$	Capacity of a UAV in processing units per time interval
E^{BAT}	Total battery capacity of a UAV
E^T	Energy required by an active UAV during one time interval
Decision variables	
Notation	**Description**
$\mu_{rl}, r \in \mathcal{R}, l \in \mathcal{L}$	Binary. 1 if request r is processed by a UAV at l; 0 otherwise
$\gamma_{ul}, u \in \mathcal{U}, l \in \mathcal{L}$	Binary. 1 if UAV u is assigned to location l; 0 otherwise
$\alpha_{ut}, u \in \mathcal{U}, t \in \mathcal{T}$	Binary. 1 if UAV u is active during time t; 0 otherwise
$\beta_{ult}, u \in \mathcal{U}, l \in \mathcal{L}\ t \in \mathcal{T}$	Binary. 1 if UAV u is assigned to location l and active during time t; 0 otherwise

between the BS and the candidate UAV location.

The ILP model is given by Equations (8.1)–(8.12):

$$maximize \sum_{r \in \mathcal{R}} \sum_{l \in \mathcal{L}} \mu_{rl} \tag{8.1}$$

$$minimize \sum_{u \in \mathcal{U}} \sum_{l \in \mathcal{L}} \gamma_{ul} \tag{8.2}$$

$$maximize \sum_{r \in \mathcal{R}} \sum_{l \in \mathcal{L}} D_{rl}\ \mu_{rl} \tag{8.3}$$

$$\sum_{l \in \mathcal{L}} \mu_{rl} \leq 1, \forall r \in \mathcal{R} \tag{8.4}$$

$$\mu_{rl} D_{rl} \geq D_r^{MIN}, \forall r \in \mathcal{R}, \forall l \in \mathcal{L} \tag{8.5}$$

$$\sum_{r \in \mathcal{R} | X_{rt}=1} \mu_{rl} P_r \leq \mathcal{C} \sum_{u \in \mathcal{U}} \beta_{ult}, \forall l \in \mathcal{L}, \forall t \in \mathcal{T}, \tag{8.6}$$

$$E^T \sum_{t \in \mathcal{T}} \alpha_{ut} \leq E^{BAT}, u \in \mathcal{U} \tag{8.7}$$

$$\sum_{l \in \mathcal{L}} \gamma_{ul} \leq 1, \forall u \in \mathcal{U} \tag{8.8}$$

$$\sum_{l \in \mathcal{L}} \beta_{ult} \leq 1, \forall u \in \mathcal{U}, \forall u \in \mathcal{T} \tag{8.9}$$

$$\gamma_{ul} + \alpha_{ut} - 1 \leq \beta_{ult}, u \in \mathcal{U}, l \in \mathcal{L}, t \in \mathcal{T} \tag{8.10}$$

$$2\beta_{ult} \leq \gamma_{ul} + \alpha_{ut}, u \in \mathcal{U}, l \in \mathcal{L}, t \in \mathcal{T} \tag{8.11}$$

$$\frac{1}{t} \sum_{t' \in \{1,\ldots,t-1\}} \alpha_{ut'} - \alpha_{u(t+1)} + \alpha_{u(t+2)} \leq 1, \quad u \in \mathcal{U}, t \in \{1, \ldots, T-2\} \tag{8.12}$$

The fog node location problem aims at optimizing three objectives. The first objective function (Equation (8.1)) maximizes the number of requests processed by all UAVs. The second objective (Equation (8.2)) minimizes the infrastructure cost by reducing the number of employed UAVs. Lastly, the objective function given by Equation (8.3) maximizes the data rate delivered to end-users. These objectives are hierarchical. Optimizing an objective only makes sense if all previous ones have already been optimized. Therefore, a multi-level programming approach (Section 2.4) is adopted, optimizing higher priority objectives first and then considering only solutions in the Pareto front to optimize the remaining goals.

The constraints of the ILP model are given by Equations (8.4)–(8.12). Equation (8.4) guarantees that requests are processed by a single UAV. Equation (8.5) assures the minimum data rate for all processed requests. Equation (8.6) guarantees that the processing capacity of the UAVs is greater than the processing demand of all requests processed during a single time interval. Equation (8.7) guarantees that the sum of the energy spent by each UAV over all time intervals of activity is not greater than the UAV battery capacity. Equations (8.8) and (8.9) are related to the location of UAVs; the former ensures UAVs are deployed at only one location, and the latter ensures that only a single UAV is assigned to the same location, i.e. performing the same trajectory, simultaneously to avoid collisions. The decision variable β_{ult} can be calculated as $\beta_{ult} = \alpha_{ut} \cdot \gamma_{ul}$; linear Equations (8.10) and (8.11) avoid such a non-linear constraint. Finally, fixed-wing UAVs cannot land in the middle of their operation; thus, sequential time intervals are allocated. Equation (8.12) guarantees that the active time intervals of for each UAV be sequential.

8.4 STUFog Algorithm

This section presents the Spatio-Temporal UAV Fog Node Location (STUFog) algorithm designed to the main issues of the fog node location problem: the requirement of UAVs to be active to process requests when they are close to fixed overloaded nodes, and the

consideration of limitations of fixed-wing UAVs.

The STUFog algorithm attempts to maximize the requests served. STUFog considers the number of requests that a UAV can handle at every location during each time interval. Such information identifies when and where requests are generated so that UAVs can be positioned to process most of the requests. Furthermore, STUFog accounts for the physical limitations of fixed-wing UAVs, such as battery autonomy, to schedule them to potential locations in sequential time intervals. The STUFog algorithm assumes circular trajectories for fixed-wing UAVs and the QoT of the wireless links. The STUFog algorithm does not overdimension the number of UAVs needed since it does not deploy a new UAV if not necessary. If a UAV cannot process all requests made in a given period, the STUFog algorithm prioritizes UAVs close to the users to maximize the data rate, and leaves any remaining request to other UAVs.

A matrix M with spatio-temporal data related to requests is used by STUFog. Time is discretized, and each column of M represents a single time interval t; each row represents a candidate location l. Entries in M store the requests that a UAV at l during time t can handle. The STUFog algorithm associates sequential columns to each UAV. The maximum number of columns gives the maximum time a UAV can remain operational without recharging. Moreover, requests are included in an entry of M if and only if a fixed-wing UAV flying at that location can guarantee the minimum required data rate for the request. Matrix M is used in the identification of locations with large demands that helps obtaining solutions to large infrastructures in an acceptable time frame.

Algorithm 8.1 presents the STUFog algorithm. It employs the matrix M that represents the spatio-temporal data of the requests. The columns represent the available time intervals, rows indicate the locations. Each entry stores the requests that a UAV can process when positioned in the corresponding location during a given time. The STUFog algorithm assigns UAVs to locations in consecutive time intervals, represented by a group of sequential columns which is limited to the maximum time a UAV can fly without recharging. The elements of matrix M are used to identify peaks of demands and its use is crucial to make the algorithm scalable.

The STUFog algorithm first initializes M (Line 1), by filling all entries with an empty list. Then, requests are added to each entry if it is possible to process them by a UAV at that location, i.e., if the data rate and processing requirements can be assured for processing the request. The variable $availableUAVs$ is initialized with the maximum number of UAVs available for deployment (Line 2), and this value is decreased by one when the algorithm decides on the positioning of a UAV (Line 12). Then, the algorithm moves to the main loop, which is divided into two parts. In the first one (Lines 5–8), it evaluates all entries. If no UAV has been deployed to a location l and its starting operation time is t, the algorithm obtains the list $requests[l, t]$ of the requests that a UAV deployed at t and l could process. This list is calculated using the data in M, but limiting the total number of requests in each time interval to the processing capacity of the UAV. If a UAV cannot process all the requests made during that time t, the STUFog algorithm favors those requests with higher data rates, thus maximizing the third objective (Equation (8.3)). After this step, the algorithm sets the variables mL and mT with that location and time, respectively, so that the number of requests processed by the new UAV

```
1  Initialize M;
2  availableUAVs ← U;
3  continueDeployment ← True;
4  while continueDeployment do
5      foreach [l, t] ∈ ℒ × 𝒯 do
6          if There is no UAV at location l and time t then
7              Calculate requests[l, t];
8      [mL, mT] ← [l, t] such that requests[l, t] is maximum;
9      if requests[mL, mT] ≠ ∅ then
10         Deploy UAV at location mL and time mT;
11         Update M;
12         availableUAVs ← availableUAVs − 1;
13     else
14         continueDeployment ← False;
15     if availableUAVs = 0 then
16         continueDeployment ← False;
```

Algorithm 8.1: STUFog algorithm.

will be maximized.

Lines 9 to 16 constitute the second part of the main loop. If a viable solution was found, a UAV is deployed at location mL starting its operation at time mT (Line 10). In this case, the algorithm updates two variables: the requests assigned to the deployed UAV are removed from M, and $availableUAVs$ is decreased by one. The STUFog algorithm continues until new UAVs can no longer process the remaining requests (Line 14), or all available UAVs have been deployed (Line 16).

The STUFog is a viable alternative to the exact formulation in Section 8.3 that has a large number of constraints. For example, in the ILP model, all L^U pairs of UAVs and locations are considered, which generates an exponential number of Constraints (8.10) and (8.11). Differently, the STUFog algorithm has an execution with polynomial time complexity. This complexity can be derived by analyzing the most time-consuming operation of the STUFog algorithm, the one in Line 7. This operation is inside two loops: the immediate loop (Line 5), and the main loop (Line 4). The main loop can be executed up to U times, one for each available UAV, and, consequently, the immediate loop is executed up to $U \cdot L \cdot T$ times. Each time the operation in Line 7 runs, there is an evaluation of all requests that were not assigned to a UAV, at most R, and the sequential time intervals of the UAV operation, at most T. Therefore, the time complexity of the STUFog algorithm is polynomial and given by $\mathcal{O}(ULT(T + R))$. In practical scenarios, the number of requests has a higher order of magnitude than that of the number of time intervals; in such cases, the complexity of STUFog can be simplified to $\mathcal{O}(ULTR)$.

8.5 Performance Evaluation

This section presents the results of simulations of the proposed algorithm and the ILP model. Real workload and data to simulate fixed-wing UAVs were used. This section

is organized as follows. Subsection 8.5.1 describes the characteristics of UAVs, such as speed, altitude, energy consumption, and wireless channel. Subsection 8.5.2 describes how requests were modeled and the data sets employed. Finally, Subsections 8.5.3 and 8.5.4 discuss numerical results; with the former comparing the results obtained by the STUFog algorithm and the ILP model, and the latter discussing the results obtained by the proposed algorithm in large scenarios.

8.5.1 UAV characterization

The energy consumption of the UAV was calculated using the power models derived in [106], detailed in Subsection 2.5.3. UAVs perform circular trajectories with a radius r at a constant speed V; the energy consumed in such trajectory is given by Equation (2.3). The values for the parameters employed were based on [106]: $c_1 = 9.26 \cdot 10^{-4}$, $c_2 = 2250$, $g = 9.8m/s^2$, and $V = 30m/s$. The value of r was varied to evaluate the impact of different radius on the energy consumption and on the wireless channel; the adopted values were 100, 200, and 300 m. UAVs are considered to fly 100 m above the ground. The configuration of the battery considered is 34200 mAh/22.8 V [75].

In order to assess the advantages of fixed-wing UAVs, a comparison between fixed-wing and rotary-wing UAVs is made. Rotary-wing UAVs hover continuously at the position to which they are assigned. The energy consumed during hovering is calculated using Equation (2.1), which can be simplified to $P_0 + P_{\text{ind}}$ for $V = 0$, where the constants P_0 and P_{ind} represent the blade profile power and induced power, with values of $79.85628W$ and $88.62793W$, respectively. The same altitude and battery configuration were adopted for both types of UAV.

The data rate is calculated using the path loss model described in Subsection 2.5.4. The Doppler effect due to the motion of the UAV was assumed to have been perfectly compensated for [106]. The path loss is used to calculate the channel capacity (in bits per second) using the Shannon-Hartley theorem. In this chapter, the transmission power is 23 dBm, the noise -60 dBm, and the bandwidth 10 mHz [2, 99, 20]. The path loss and the channel capacity are calculated for each pair request-location to obtain the network requirements. The maximum UAV coverage is assumed 4 km [2].

8.5.2 Workload

The locations and the workload demands were taken from the data sets [8, 78] reviewed in Section 3.5. The BSs are used as the candidate locations for fog nodes, considering 100 possible locations for deployment of UAVs. In this chapter, individual requests are considered, and the constant Z (Section 3.5) was used to obtain such a value. The altitude of BSs was obtained from the Shuttle Radar Topography Mission [37] data set, and it is used to calculate the vertical distance between UAVs and BSs. The BS positions are the candidate locations for the deployment of UAV fog nodes. A 24-hour interval is considered, producing $\mathcal{T} = \{1, 2, \ldots, 144\}$ due to the aggregation into 10-minute intervals.

The coordinates around which the UAVs perform circular trajectories were defined as equally distant points on the map, and the distance between them is greater than

the radius, avoiding intersections. In the case of rotary-wing UAVs, these coordinates were the position where the UAVs hovered. The UAV server capacity was 100 processing units (PUs) per time interval. Each request was characterized in terms of processing and networking requirements. The processing requirement was fixed to 10 PUs and the requirement was the required data rate, which can be 50, 75, or 100 Mbps. Such a variation of data rate values helped to analyze the performance of fixed-wing UAVs under different conditions.

8.5.3 Validation of the STUFog algorithm

In this subsection, the STUFog algorithm is validated by comparing its performance to that of the ILP model. Obtaining exact solutions for several locations and requests was impossible due to the execution time and main memory constraints. Therefore, the comparison in this subsection was carried out with a limited number of locations and requests. The metric evaluated is the acceptance ratio, given by the ratio between the number of requests processed by UAVs and the total number of requests. Results are presented as a function of the number of available UAVs. The experiments started with a single UAV available for deployment, and sequential executions increased this number by one until the inclusion of another UAV no longer improved the acceptance of requests. For this validation, requests were obtained from 10 BSs, 25 locations were available for UAV deployment, and Z was set to 0.0002, thus reducing the number of requests to less than one hundred. 95 % confidence intervals are displayed in the graphics.

Figures 8.1–8.4 present the results obtained by the STUFog algorithm and the exact model for different UAV deployments. For the fixed-wing UAV, deployment with a 100 m radius trajectory required more UAVs to process all the requests than did the others, which is explained by the energy efficiency of fixed-wing UAVs. Trajectories with a shorter radius increased the energy consumption of the UAV, thus reducing its operation time and requiring the deployment of more UAVs to cope with the same demands. The energy consumption also explains why the use of rotary-wing UAV requires more UAVs than do the other deployments: a hovering rotary-wing UAV consumes about the same amount of energy as a fixed-wing flying with a trajectory having 100 m radius. Figures 8.1–8.4 also show that the increase in the number of UAVs did not necessarily lead to a proportional increase in the acceptance ratio. If only few UAVs are available, they will be located where most of the demands are. Then, the remaining UAVs will not be dispatched to fulfill peak demands but rather to process the remaining requests sparse in time and space.

The performance of the STUFog algorithm was very close to that of the exact model. For most points in the graphics, the difference in the acceptance ratios for the two schemes was less than 3 %. The significant difference is that, for 100 % acceptance, the proposed algorithm employed up to two extra UAVs. This increase in the number of UAVs was due the fact that, for the exact model, all UAVs were positioned simultaneously, whereas with the STUFog algorithm, this takes place sequentially, leaving punctual unprocessed requests and requiring extra UAVs to cope with all the demands. Although a such performance was not ideal, the results obtained by the STUFog algorithm were very close to 100 %, where an exact solution achieved 100 %, demonstrating that the impact of the

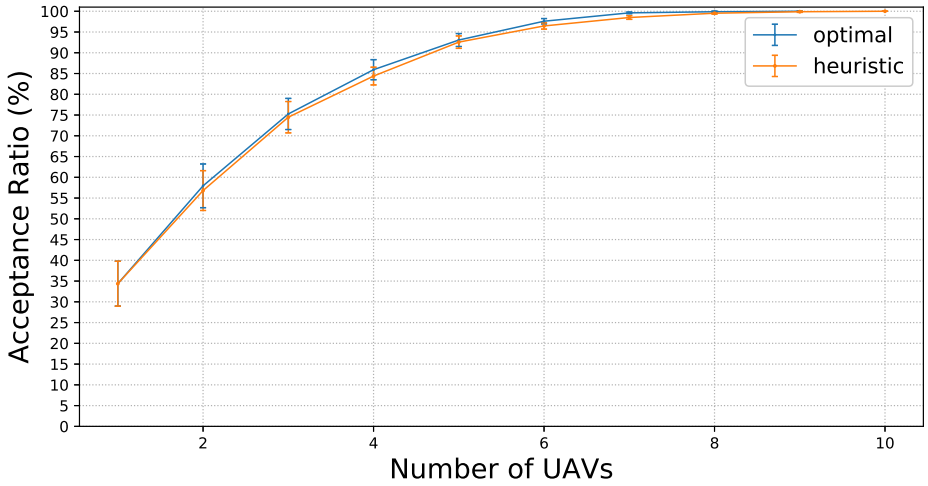


Figure 8.1: Acceptance ratio as a function of number of UAVs resulting from the use of the STUFog algorithm and the ILP model for fixed-wing UAVs flying with a 100 m radius.

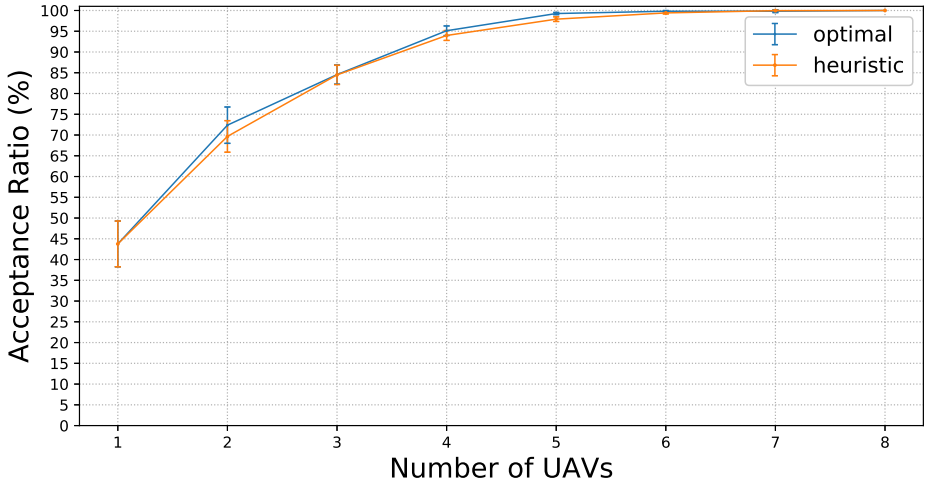


Figure 8.2: Acceptance ratio as a function of number of UAVs resulting from the use of the STUFog algorithm and the ILP model for fixed-wing UAVs flying with a 200 m radius.

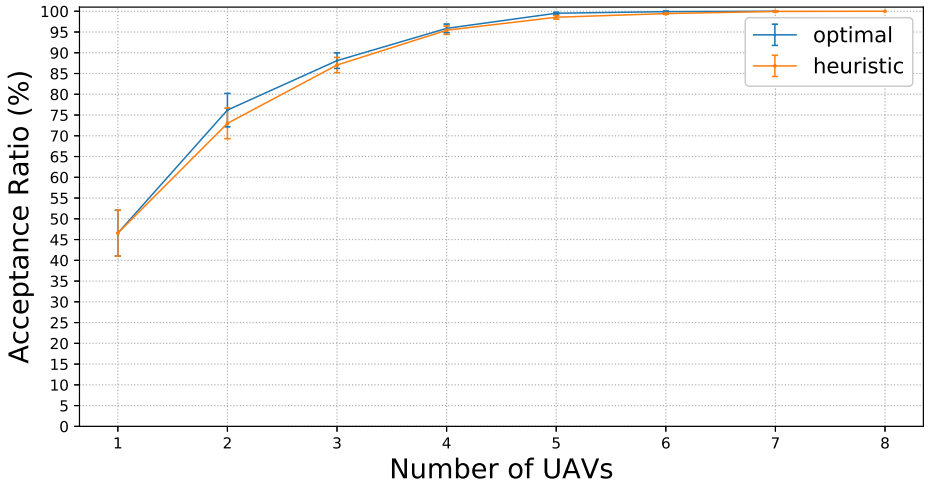


Figure 8.3: Acceptance ratio as a function of number of UAVs resulting from the use of the STUFog algorithm and the ILP model for fixed-wing UAVs flying with a 300 m radius.

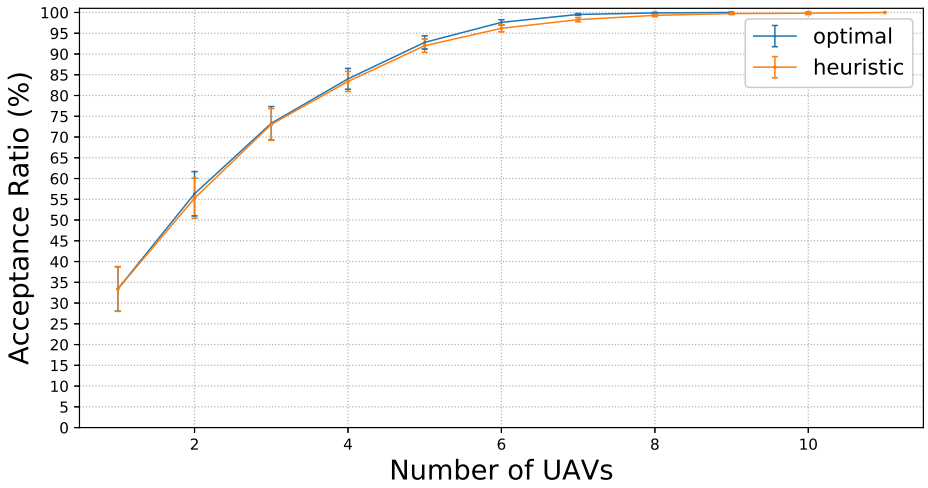


Figure 8.4: Acceptance ratio as a function of number of UAVs resulting from the use of the STUFog algorithm and the ILP model for rotary-wing UAVs.

additional UAVs is minimal.

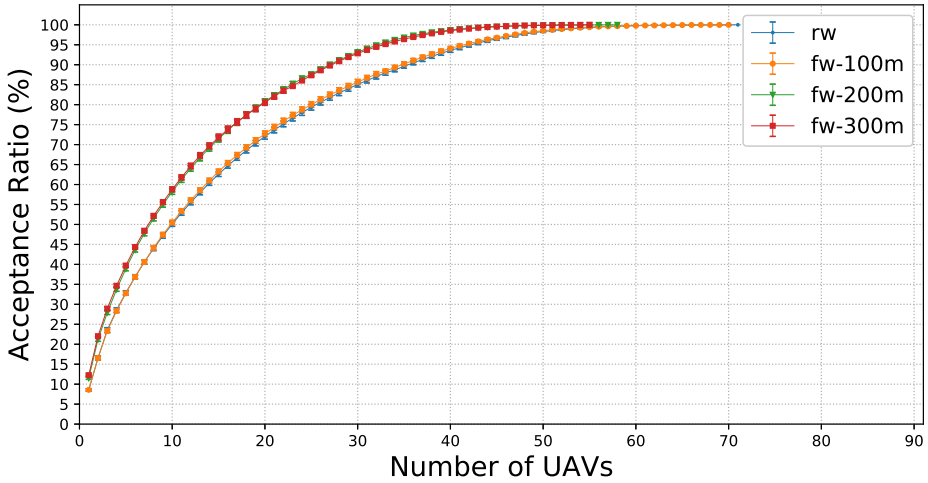
8.5.4 Numerical discussion

This subsection evaluates the employment of fixed-wing UAVs in large scenarios. The evaluation considered requests from 100 BSs, a variable number of candidate locations, and Z equals 0.0005, which led to thousands of requests. The metrics evaluated are the acceptance ratio and the average data rate delivered to end-users, both presented as a function of the number of available UAVs. As in the previous subsection, the experiments started with a single UAV available for deployment, with sequential executions increasing this value by one until the inclusion of a new UAV no longer increased the acceptance of requests. For the sake of readability, deployments of rotary-wing UAVs are identified as `rw`, and deployments of fixed-wing UAVs are identified as `fw-RRRm`, where `RRR` is the radius of the circular trajectory in meters. 95 % confidence intervals are displayed in the graphics. This subsection is organized into four discussions. First, general trends of different deployments are discussed; then, the impact of different data rate requirements on the infrastructure is investigated, and the effect of the number of available locations is assessed. Finally, the use of tethered deployments is discussed.

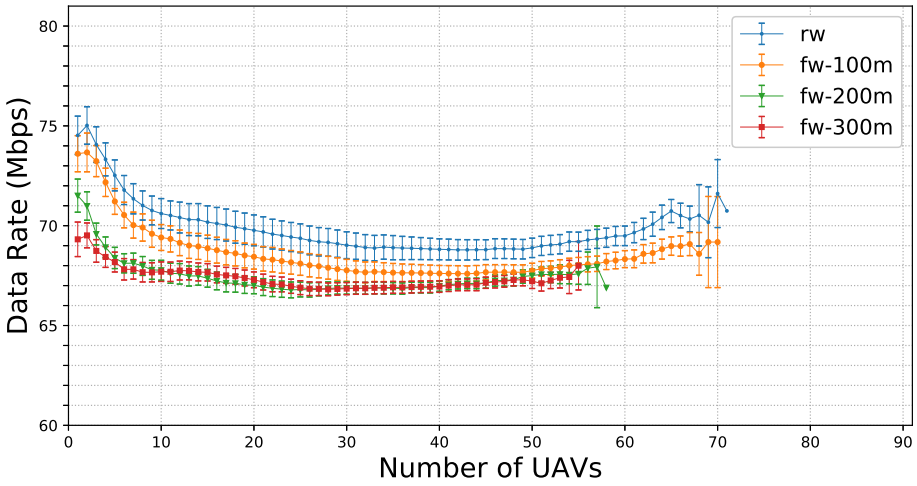
Figure 8.5 displays the results for deployments with a 50 Mpbs data rate requirement and 100 candidate locations available for UAV deployment. As discussed in the previous subsection, the shorter the radius, the greater is the UAV energy consumption. Consequently, UAVs with trajectories with a large radius usually process more requests, thus reducing the total number of UAVs to achieve the same acceptance ratio. This trend can be seen in Figure 8.5a, with deployments of `fw-200m` and `fw-300m` resulting in greater acceptance than the one of `fw-100m`. These deployments with a large radius obtained the best results, serving many users in different time intervals. The `fw-100m` and `rw` deployments could also have achieved 100 % acceptance of requests, but with a higher number of UAVs than the other deployments due to the energy efficiency.

The average data rate (Figure 8.5b) was in the range 65–75 Mbps, higher than the minimum requirement (50 Mbps). The highest data rate was that produced by the rotary-wing UAV deployment since such UAVs maintain the same position in the air, guaranteeing better wireless links. Differently, the data rate for the fixed-wing UAVs was calculated considering the coordinates of the circular trajectory that maximized their distance from the BS. When the UAV communicates with the ground, for example, it can be closer to the BS, which improve the data rate delivered; therefore, the results for fixed-wing deployments represent the minimum data rate that can be guaranteed. Trajectories with greater radii lead to only small reductions in data rate.

In these experiments, fixed-wing UAVs with 300 m radii represented the most advantageous deployment, with small energy consumption and good wireless links. However, if rotary-wing UAVs are cheaper than fixed-wing ones, rotary-wing employment can be quite advantageous. Currently, a variety of models of both types of UAVs is available, and prices vary significantly. Deployments with rotary-wing UAVs can be more advantageous if their price is lower than that of rotary-wing UAVs. To obtain an acceptance ratio of 90 %, rotary-wing UAVs provide the best results if their price is at least 25 % lower than



(a) Acceptance ratio

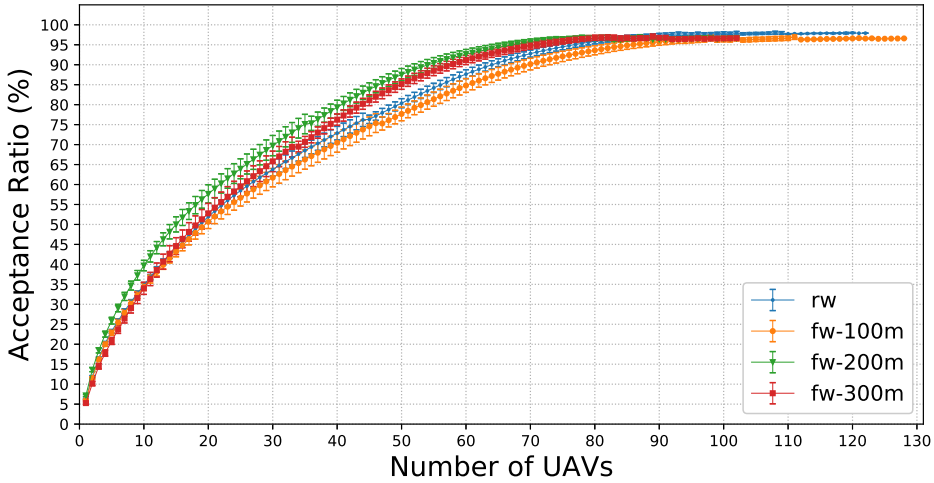


(b) Data rate

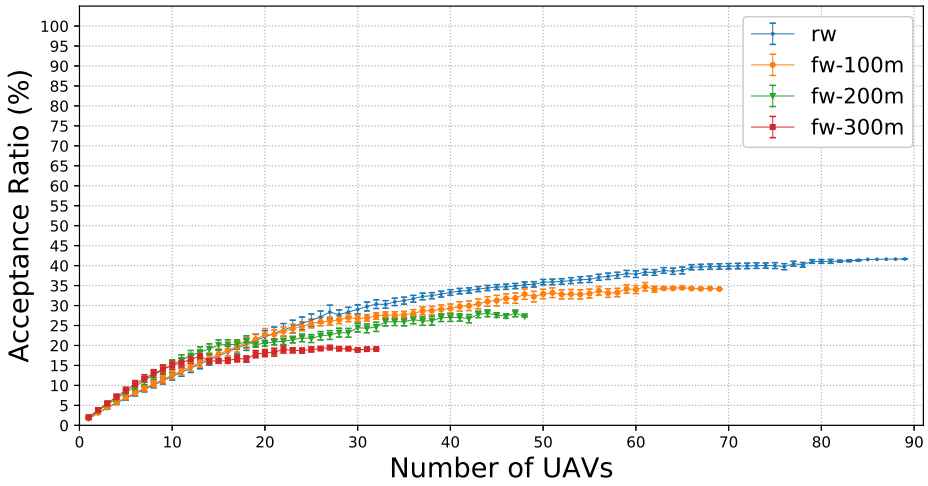
Figure 8.5: Results for the 50 Mpbs data rate requirement and 100 candidate locations.

that of fixed-wing ones since, in this case, they can obtain a higher acceptance ratio with the same budget and data rate will be slightly higher.

Figure 8.6 shows the acceptance ratio for different data rate requirements with 100 candidate locations for UAV deployment. Because of the path loss, wireless links need to be short to support high data rates; so UAVs must be physically close to the BSs, which may not always be possible. Therefore, when the requirement is 75 Mbps (Figure 8.6a), achieving 100 % acceptance was not possible; a nearby UAV could not cover a small portion of the users. The 75 Mbps requirement also presented a different trend in relation to the 50 Mbps requirement (Figure 8.5): the performance of the `fw-100` deployment was inferior, making that of the `fw-200` solution the most interesting one. The wireless link lengths explain those results for a 50 Mbps rate, in which both UAV deployments could provide sufficient data rates to support the ground users. However, a with 75 Mbps rate, UAVs needed to be closer to the ground to provide higher data rates. Therefore, although



(a) 75 Mpbs



(b) 100 Mpbs

Figure 8.6: Acceptance ratio as a function of the number of UAVs for different data rate requirements and 100 candidate locations.

the `fw-200` consumes more energy than does the `fw-300`, it manages to provide the best trade-off between energy consumption and network quality.

For the most strict requirement (100 Mbps, Figure 8.6b), there is a massive impact on the acceptance, allowing at most 45 % acceptance. The results are similar to those for the 50 Mbps requirement (Figure 8.5) for a low number of available UAVs. However, as the number of available UAVs increases, deployments of fixed-wing UAVs with large radii trajectories (`fw-200m` and `fw-300m`) achieve an upper limit of the acceptance earlier, while those with `fw-100m` and `rw` deployments continue to increase the acceptance. Specifically, if there are no budget constraints, rotary-wing UAVs can achieve the best results due to their stability in the air that improves the wireless channel, which shows that the decision onto type of UAV depends on both required QoS requirements and budget constraints.

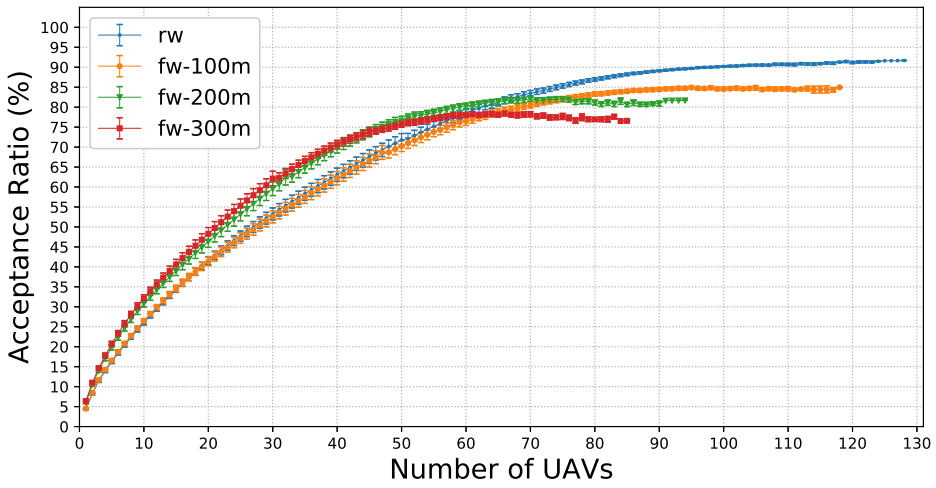
Fog providers can exert limitations on the available locations for deployment of UAVs

as this may be prevented by local regulations from usage of airspace near airports, tall buildings, private industries, or areas where certain animals live. Therefore, the effects of a few specific candidate locations for the processing of the same requests were investigated. In this case, the distance between these locations increases, since these few locations are positioned equally distant on the same map. Although results for different data rate requirements lead to different values in the acceptance ratio of requests, the same tendencies were observed. Therefore, only the 75 Mbps data rate requirement is presented in Figure 8.7. The small number of locations increases the average distance between UAVs and the BSs, reducing the data rate for the wireless links. Consequently, at most 95 % of requests are processed when 49 locations are considered (Figure 8.7a), and only up to 50 % for 25 locations (Figure 8.7b). For 49 locations, the best UAV type and radius depend on the available budget. When few UAVs are available, deployments with fixed-wing UAVs performing a 200–300m radius (`fw-200` and `fw-300`) are better due to their energy efficiency. However, for more than 70 available UAVs, the `rw` deployment is much better because it maintains a smaller distance to ground BSs, thus resulting in better wireless links, and obtaining a higher acceptance ratio than do fixed-wing UAVs. When only 25 locations are involved, a similar tendency is found, but the small number of candidate locations has a more significant impact on the acceptance, even for the rotary-wing UAVs (`rw`). Reducing the candidate locations or increasing the data rate requirement have a similar effect on the results: the acceptance of requests is reduced because the requirements cannot be met. Nonetheless, intermediate values for these parameters still allow satisfactory deployments, especially in scenarios with a limited budget.

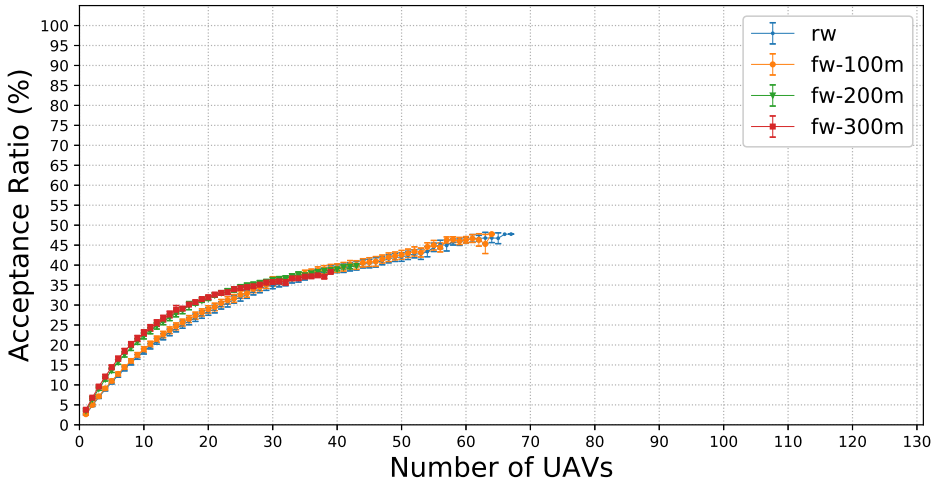
Figures 8.8–8.10 presents the acceptance ratio obtained for tethered deployments and 100 locations available for these deployments. The energy capacity is assumed to have no limit, and, consequently, a UAV can be deployed to any of the locations and remain powered at all the time. Without the energy limitation, the number of required UAVs to achieve 100 % acceptance of request is reduced remarkably to less than 30 UAVs. With no energy limitation, the `rw` deployment led to the best results, followed by the fixed-wing UAV deployments in order of radius, i.e., the best deployments were the ones closer to the ground nodes. For tethered deployment, rotary-wing UAVs are the best option unless fixed-wing UAVs are cheaper. Moreover, the deployments of fixed-wing UAVs with radii of 200 m or more are not a good choice since the tether tend to be in the range of 80–150m [60].

Deployments with tethered UAVs can be quite valuable for reducing costs, but they impose significant limitations. The ground infrastructure for tethered UAVs involves deployment and maintenance costs, such as connecting the tether to other networks, installing the energy supply infrastructure, and renting the rooftops of tall buildings. Moreover, tethering UAVs requires a larger support staff over the served region, while flying UAVs can be operated remotely most of the time. Therefore, the decision to tether UAVs depends on factors other than the UAVs.

The main findings of this evaluation can be summarized as follows. Fog nodes mounted on fixed-wing UAVs can be quite efficient for dealing with the end-user demand variable in time and space as long as their location is properly selected. The decision between fixed-wing and rotary-wing UAVs depends on the traffic demands, UAV costs, and budget



(a) 49 locations



(b) 25 locations

Figure 8.7: Acceptance ratio as a function of the number of UAVs for the 75 Mbps data rate requirement and different number of candidate locations.

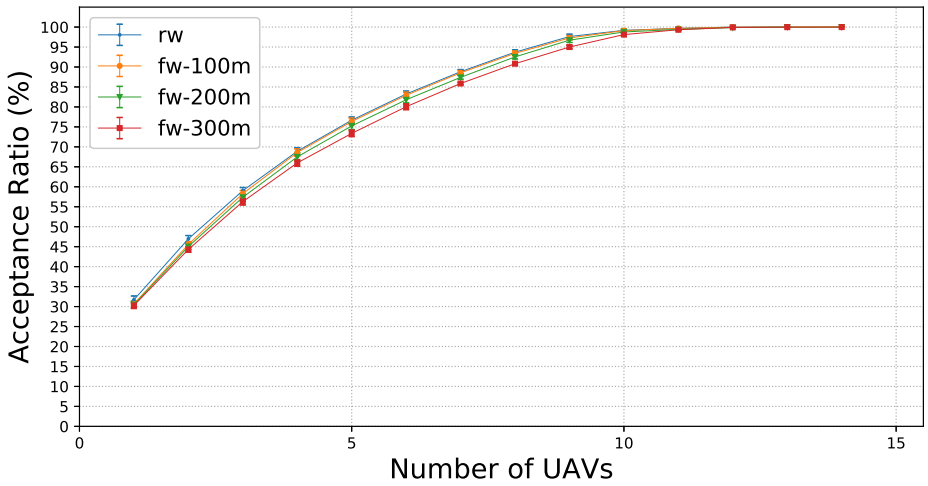


Figure 8.8: Acceptance ratio as a function of the number of UAVs for tethered deployments, 50 Mpbs data rate requirement, and 100 candidate locations.

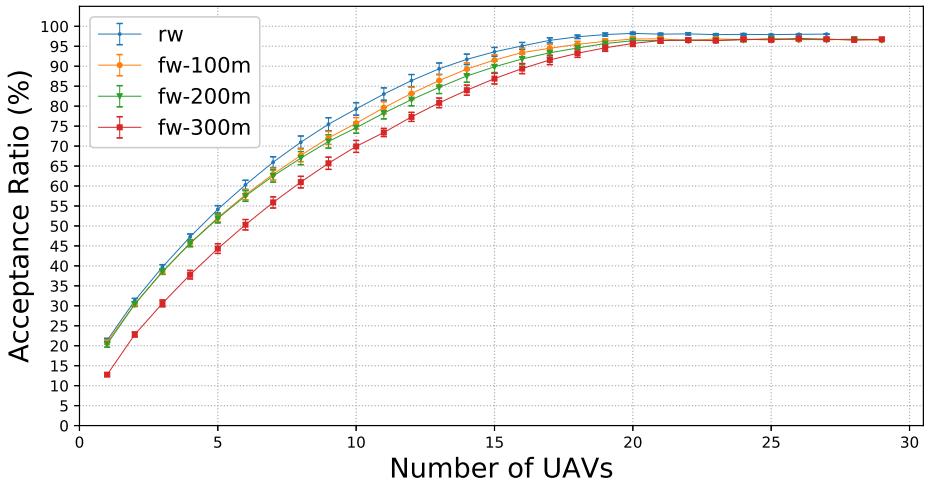


Figure 8.9: Acceptance ratio as a function of the number of UAVs for tethered deployments, 75 Mpbs data rate requirement, and 100 candidate locations.

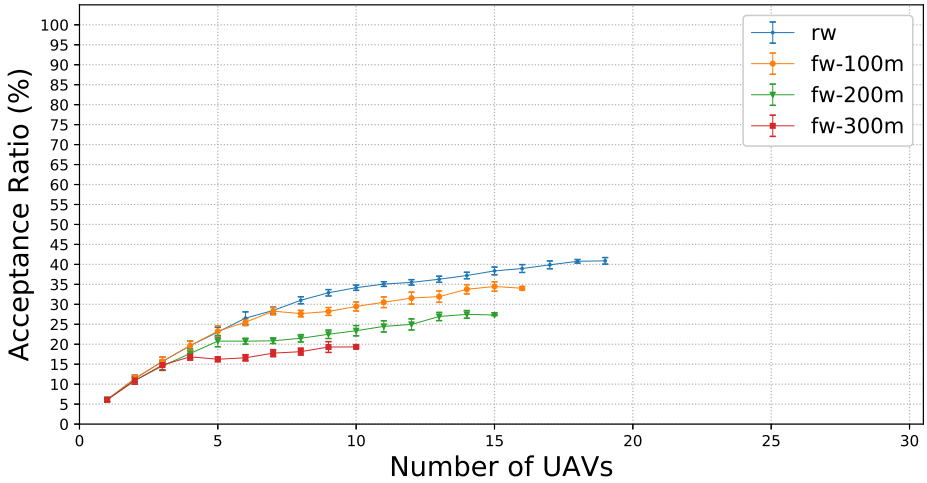


Figure 8.10: Acceptance ratio as a function of the number of UAVs for tethered deployments, 100 Mpbs data rate requirement, and 100 candidate locations.

constraints. In all scenarios evaluated, rotary-wing UAVs could offer the best networking conditions, especially in strict demand scenarios or tethered deployments. However, their energy consumption is relatively high, requiring more UAVs than deployments with fixed-wing ones. Therefore, fixed-wing UAVs are generally the better option, with the radius of their circular trajectory the maximum possible to extend their battery operation, but not large enough to prevent efficient wireless links. In this evaluation, radii between 200 m and 300 m were quite efficient for dealing with the workload, unless the number of location is severely limited. The cost of different UAV models must also be taken into consideration, since for many conditions, rotary-wing UAVs have to be much cheaper than fixed-wing ones to provide the same acceptance of requests at the same cost.

8.6 Conclusions

UAVs can complement the traditional fixed fog computing infrastructure by covering peaks of user requests on demand. Different types of UAVs can be employed, and this chapter has shown that fixed-wing UAVs can be quite valuable as fog nodes. The reduced energy consumption leads to extended operational time, serving more end-users. This chapter has highlighted the benefits of fixed-wing UAVs by an investigation of the fog node location problem with the aim of selecting the locations and time intervals for which UAVs will be active to deal with the demands of end-users in relation to time and space. We formulated the problem as an ILP model and presented the STUFog algorithm for obtaining solutions for large deployments. We have also evaluated the STUFog algorithm with actual data of mobile users, and results showed that fixed-wing UAVs can be quite efficient as fog nodes as long as their trajectory is adjusted to reduce the energy consumption and meet the networking requirements. Using intermediate values for the radius of circular trajectories provides an excellent trade-off between energy consumption and wireless link quality: compared to the most energy-efficient deployment (300 m radius), a 200 m radius provides basically the same acceptance, yet with better data rates. The evaluation has also shown that if fixed-wing and rotary-wing UAVs have a similar price, the former tend to be more advantageous.

Part IV

Resource Allocation

Chapter 9

Resource Allocation Mechanism for Fog-Cloud Infrastructures

9.1 Overview

A typical management issue in a fog-cloud infrastructure is in which (fog or cloud) node a workload should be processed. Such a decision must consider resource availability on fog nodes and the latency for processing the workload in the fog as well as the latency for processing in the cloud. This chapter proposes a mechanism to solve this management issue that avoids overutilization of resources in limited fog nodes while improving overall latency for end-users. To achieve this goal, this chapter proposes the Gaussian Process Regression for Fog-Cloud Allocation (GPRFCA) mechanism which decides where to run the tasks of an application. The infrastructure considered is composed of a fog layer and the cloud. Users submit requests to the fog nodes and the workload can be executed in the fog, in the cloud, or partially executed in the fog and the cloud. The GPRFCA mechanism decides where to schedule a workload to be processed taking into account the resource availability and the latency overhead.

The GPRFCA mechanism favors the utilization of the fog node rather than the cloud. The motivation for that is two-fold. First, fog nodes present smaller delays, which improves the end-user experience. Second, fog nodes presents advantages in relation to energy efficiency compared to clouds [51, 86, 7]. Specifically, small fog nodes may not require cooling infrastructures similar to those of clouds, whose consumption can be almost 50 % of the total energy delivered to the data center [17]. The ratio between the total energy used in a data center and the energy delivered to its computational resources is known as Power Usage Effectiveness (PUE); data centers can present high values for PUE. By forwarding workload to the fog node, potential gains in energy consumption may be obtained.

To optimize the utilization of limited fog resources, a Gaussian Process Regression (GPR) is employed to predict the arrival of future requests based on the history of arrivals. Such prediction helps the provisioning of resources to future requests, especially the real-time ones which can only be processed in the fog, reducing blocking and improving overall utilization. GPR was chosen due to the versatility of the covariance function, whose

selection favors characteristics from the input data. Previous analysis of cloud traces [49] showed that virtual machine arrival and departure processes present self-similarity. Therefore, in this chapter, GPR was employed with a rational quadratic (RQ) covariance function suitable for prediction of self-similar series [10].

This chapter presents the GPRFCA mechanism and its evaluation. None of the existing work considered the history of previous requests for the resource allocation in fog-cloud architectures. Results derived by simulation show that the employment of the proposed solution balances the workload between the available nodes, providing a good trade-off between energy consumption, latency, and blocking. The GPRFCA mechanism manages to maintain the utilization of the fog node at a high level but it prevents blocking, which leads to efficient use of the infrastructure.

The remainder of this chapter is organized as follows. Section 9.2 introduces the system model adopted. Section 9.3 introduces the GPRFCA mechanism. Section 6.5 presents the experimental setting and numerical evaluation of the GPRFCA mechanism. Lastly, Section 6.6 concludes this chapter.

9.2 System Model

A fog-cloud layered architecture is considered with a cloud layer is at the top, followed by the fog layer, and end-user devices are at the bottommost layer. Access to the fog or cloud nodes implies different delays. The cloud is represented as a data center with plenty of physical servers, while the fog node is modeled as a mini data center capable of hosting virtual machines (VMs) to host end-user workload. The third layer is composed of user devices that connect to the fog node using wireless interfaces. Figure 9.1 illustrates the considered architecture.

Requests are modeled using the distributed dataflow approach [41]. In this model, an application is represented by a directed graph in which nodes represent tasks, actuators, or sensors, and edges represent the data flow. Sensors gather data either from the environment or from the users while actuators interact with users or perform actions. An application has at least one task. Tasks are in charge of processing data of an application, such processing is triggered by a sensor signal or a message from a different task. After processing, tasks send messages to other tasks or actuators. To make the distributed dataflow approach feasible, all tasks need to be assigned to VMs, and VMs can be created either in a fog node or in the cloud data center.

The authors of [45] identified that different classes of applications in fog computing have different requirements in terms of security, availability, location, mobility, and scalability. To deal with such requirements, a task must be properly instantiated on a layer with a certain maximum delay and minimum resources. In the considered system, users submit requests to the nearest fog node. Once it is received, the fog node analyzes the requirements of the request, deciding the servers in the fog and cloud layers that will host the VMs associated with each task, and this information is further used in the instantiation of VMs.

This work considers two classes of tasks: tasks that require VMs instantiated in the

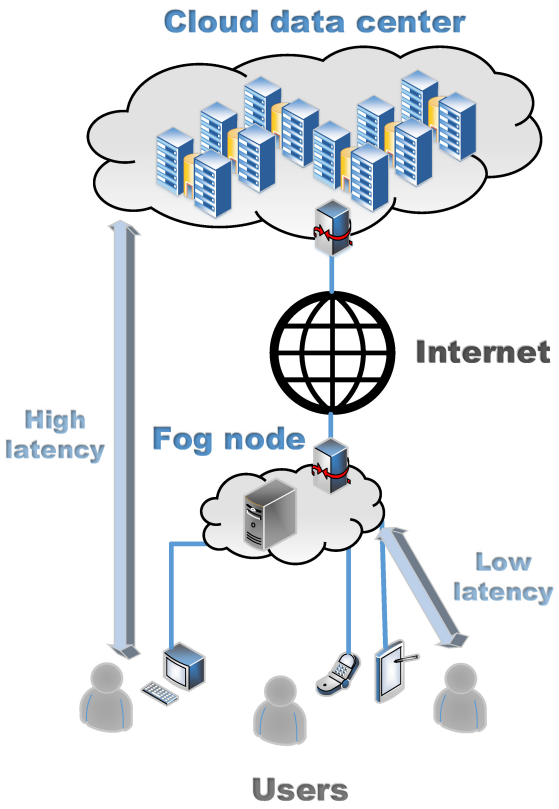


Figure 9.1: Cloud and fog architecture.

fog and those which can be instantiated either in the cloud or in the fog. The former are delay-sensitive tasks that demand a quick response to the users, while the latter are tasks with flexible response time. Applications can have multiple tasks with different classes; in this case, a request can have part of its processing in the cloud and another part in the fog. There are agents in the cloud and in the fog which decide on which physical server a VM should be instantiated. If the resource cannot be allocated for a task, the entire request is blocked.

This chapter proposes a mechanism that must decide on where VMs of a request should be instantiated (fog or cloud). Each request is submitted at a different time, and the mechanism must avoid overloading the fog node, which would prevent future service for other users. The time each request remains active is limited, which constantly changes the occupation of the fog and the cloud.

9.3 GPRFCA mechanism

The Gaussian Process Regression for Fog-Cloud Allocation (GPRFCA) mechanism analyzes the history of previously submitted requests for predicting future arrivals of tasks with strict latency requirements. Using this prediction, the GPRFCA mechanism can reserve resources in the fog node for future delay-sensitive tasks, reducing blocking of requests and increasing the utilization of the fog node. This section details the GPRFCA mechanism.

The notation used to describe the GPRFCA mechanism is presented in Table 9.1. This

Table 9.1: Notation used in this chapter.

Notation	Description
	Input
$\mathcal{C}$	Cloud data center
$\mathcal{F}$	Fog node
$\mathcal{H} = h_1, h_2, \ldots, h_n$	Historical sequence with the number of strict-latency tasks arrived in the previous n time intervals.
$\mathcal{R}$	Set of requests
	Functions
$X.allocate(), X \in \{\mathcal{F}, \mathcal{C}\}$	Instantiates a virtual machine to host the task t either in the fog $\mathcal{F}$ or in the cloud $\mathcal{C}$.
$\mathcal{F}.getAvailableVMs()$	Obtains the number of available slots for new virtual machines.
$r.numberOfFlexibleTasks(), r \in \mathcal{R}$	Obtains the number of flexible tasks of the request r
$r.numberOfStrictTasks(), r \in \mathcal{R}$	Obtains the number of strict tasks of the request r
$r.getTasks(), r \in \mathcal{R}$	Obtains the tasks associated with the request r
$t.getType(), t \in r.getTasks(), r \in \mathcal{R}$	Obtains the type of the task t, which can be $STRICT$ for a task with high latency requirements or $FLEXIBLE$ for tasks which can be instantiated in the cloud.
$emptyList()$	Creates a new empty list.
$l.add(r)$	Appends a new item to a list l.
$gaussianRegression(\mathcal{H})$	Returns the next predicted value based on the historical set $\mathcal{H}$.
$min(a, b)$	Returns the minimum value between a and b.
$max(a, b)$	Returns the maximum value between a and b.

chapter considers that requests arrive at the fog node every minute, thus two sequential executions of the GPRFCA mechanism are equally spaced over time. The number of strict-delay tasks arriving in each minute is recorded in the data set $\mathcal{H} = h_1, h_2, \ldots, h_n$, i.e., h_1 is the number of strict VMs that were submitted in the first minute, h_2 in the second minute, and so forth. All tasks require the same processing and memory resources. The mechanism, however, is not limited by this assumption and it can also handle heterogeneous demands.

The GPRFCA mechanism is described in Algorithm 9.1. It is divided into five parts: initialization (Lines 1 to 3), instantiation of strict tasks (Lines 4 to 11), prediction (Lines 12 to 14), instantiation of flexible tasks (Lines 15 to 24), and update of arrivals history (Line 25).

The initialization phase (Lines 1 to 3) consists of initializing the variables used. $totalStrictTasks$ is the variable records the total number of strict tasks that arrived in this time interval, used to update the historical data. $listInstantiatedRequests$ is the list of requests which could be instantiated in VMs in the fog without overloading the fog node. $availableVMs$ represents the number of VMs available to be created in the fog. The command in Line 3 initializes this variable with the maximum number of available VMs before any allocation, and this value is updated later.

Input: $\mathcal{R}$,$\mathcal{H}$, $\mathcal{F}$,$\mathcal{C}$

```
1  totalStrictTasks ← 0
2  listInstantiatedRequests ← emptyList()
3  availableVMs ← F.getAvailableVMs()
4  ∀r ∈ R
5      totalStrictTasks ← totalStrictTasks + r.numberOfStrictTasks()
6      if r.numberOfStrictTasks() ≤ availableVMs then
7          ∀task ∈ r.getTasks()
8              if task.getType() = STRICT then
9                  F.allocate(task)
10                 availableVMs ← availableVMs − 1
11         listInstantiatedRequests.add(r)
12 if availableVMs > 0 then
13     G ← gaussianRegression(H)
14     availableVMs ← max(0, availableVMs − G)
15 ∀r ∈ listInstantiatedRequests
16     flexibleTasksOnFog ← min(r.numberOfFlexibleTasks(), availableVMs)
17     availableVMs ← availableVMs − flexibleTasksOnFog
18     ∀task ∈ r.getTasks()
19         if task.getType() = FLEXIBLE then
20             if flexibleTasksOnFog > 0 then
21                 F.allocate(task)
22                 flexibleTasksOnFog ← flexibleTasksOnFog − 1
23             else
24                 C.allocate(task)
25 H.add(totalStrictTasks)
```

Algorithm 9.1: Gaussian Process Regression for Fog-Cloud Allocation mechanism.

Subsequently, the mechanism performs the instantiation of strict tasks (Lines 4 to 11). This phase has two goals: deciding which requests will be accepted and instantiating the strict tasks of these requests. This phase is important to guarantee that current requests have priority over future requests, reducing the chances of blocking in future iteractions due to flexible tasks allocated in the fog. This phase iterates over all requests, updates the variable *totalStrictTasks* (Line 5), and then, if the fog can accommodate all tasks of the current request (Line 6), the mechanism allocates the strict tasks of the request (Line 9), updating the number of available slots (Line 10). If the request can be successfully created, it is appended to list *listInstantiatedRequests* (Line 11).

Before instantiating the flexible tasks, the prediction phase (Lines 12 to 14) updates the number of available VM slots. If there are still available slots after the instantiation of strict requests (Line 12), the prediction is made based on a Gaussian Process Regression (Line 13). The number of predicted requests is subtracted from $availableVMs$, and, in case this operations leads to a negative value, $availableVMs$ is set to zero, indicating that no current flexible VMs will be created in the fog (Line 14).

The details of Gaussian Process Regression (Line 13) are explained as follows. The *gaussianProcess* function receives as input a time series to make prediction based on the Gaussian Process Regression (GPR), explained next. Considering the observed values $X(t_i),\ i = 0, 1, \ldots, n-1$ at n time instants t_i, then $X(t_i) = f(t_i) + \varepsilon_i$, where $f(t_i)$ is a mapping function and ε_i is an independent Gaussian noise with zero mean and variance σ^2. To make prediction, it is assumed that $f(t_i) \sim GP(m(t), k(t_i, t_j; \theta))$, where GP is a stochastic Gaussian process with mean $m(t)$ and covariance function $k(t_i, t_j; \theta)$ with hyperparameters θ. The decision of the covariance function depends on the actual covariance of the data. Previous analysis of cloud data centers [49] identified that the virtual machine arrival process can be modeled as a self-similar process. Furthermore, the work in [10], which employed GPR to predict traffic in real networks, identified that the rational quadratic covariance function k_{RQ} is suitable for modeling self-similar series. This chapter employs the covariance function k_{RQ}, given by Equation (9.1).

$$k_{RQ}(r; l, \alpha) = s^2 \cdot \left(1 + \frac{r^2}{2\alpha l^2}\right)^{-\alpha} \tag{9.1}$$

Three hyperparameters are used, namely the variance s, length-scale parameter l and magnitude parameter α. For self-similar series, $\alpha = 1 - H$ [10], where H is the Hurst parameter of the series. These hyperparameters are optimized by a local search algorithm that searches their values in an interval between 0 and 100 for s, 0 and four times the maximum value in $X(t_i)$ for l, and 0 and 0.5 for α.

The instantiation of the flexible tasks is conducted in Lines 15–24. All requests which had their strict tasks instantiated are visited (Line 15), and the number of flexible tasks to be instantiated is decided, limited by the value of $availableVMs$ (Line 16). Then, the tasks of each request are iterated (Line 18), and only flexible ones are considered, since strict tasks were already created. If there are available slots, the current task is created in the fog (Line 21), otherwise the cloud is used (Line 24). By using the fog node for some flexible requests, the GPRFCA mechanism improves fog occupation, reducing latency for more clients, yet reducing blocking due to the prediction.

Table 9.2: Infrastructure configuration and virtual machine instance description.

	PUE	Specification	Delay to users
Fog	1.0	2 servers Hp Proliant Dl380 G7 3.06 GHz Xeon X5675 processor 1 core, 4 Gb RAM	10ms
Cloud	1.4	100 servers Hp Proliant Dl380 G7 3.06 GHz Xeon X5675 processor 2 cores, 8 Gb RAM	100ms
Virtual Machine		1000 MIPS, 256 Mb RAM	

The final step is the update of arrivals history $\mathcal{H}$ (Line 25), so that future predictions of the GPRFCA mechanism consider updated time series.

9.4 Performance Evaluation

The GPRFCA was coded in Java and compared to other mechanisms from the literature; results of this evaluation are reported in this section. Subsection 9.4.1 describes the simulation settings. Subsection 9.4.2 presents the other mechanisms evaluated in this chapter. Subsection 9.4.3 describes the workload model, and Subsection 9.4.4 the energy consumption model. Finally, Subsection 9.4.5 discusses numerical results.

9.4.1 Simulation settings

The iFogSim simulator [46], presented in Section 3.6 was employed, and Gaussian Process Regression prediction was implemented with gptools Python package [26]. Results were obtained by 30 different executions for each point in the graphs and using a 95 % confidence interval derived by the independent replication method.

The cloud data center is modeled as a set of physical servers. As suggested in [93], mini data centers employed as fog nodes have similar hardware to that of cloud data centers. Table 9.2 displays the hardware configuration of servers and the VM requirements.

Each task runs in a VM hosted either in the fog or in the cloud. The mapping of a VM onto a physical server in the fog node or in the cloud data center is performed by a VM placement algorithm which chooses the first available server in sequential order to host a task, therefore consolidating the workload on a few machines to save energy. If all servers are fully utilized and cannot host a VM, the task is blocked as well as its entire request.

9.4.2 Evaluated mechanisms

The performance of the proposed mechanism was compared to that of four other mechanisms, namely cloudwards and fogwards. These are simple strategies that favor the utilization of either the fog or the cloud [46]. Whenever a task can be hosted either in the data center or in a fog node, cloudwards always places it on the cloud, but it always sends

to the fog tasks with strict latency requirements. Fogwards tries to send all the tasks to the fog, independent of their latency requirements, however, if a task can be sent to the cloud and no resources are available in the fog to host it, fogwards places the task in the cloud. Since plenty of resources is available in the cloud, allocation in the data center is virtually always possible, and cloudwards does not send tasks to the fog unless they have strict latency requirements.

9.4.3 Workload

Two applications are considered: remote VM and augmented reality. Remote VM (Figure 9.2a) is a simple application in which a user accesses a terminal (thin client) and interacts with a remote operational system. User's actions are processed in a VM and results are sent back to be displayed on the terminal screen. Simulated scenarios have two classes of requests: requests with one task which can only be placed in the fog, and requests with one task that can be deployed either in the fog or in the cloud.

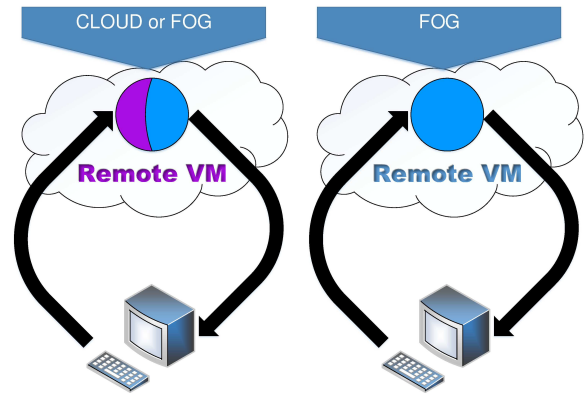
For the augmented reality application (Figure 9.2b), virtual objects are overlaid on an image acquired from a camera and then rendered on a display. The simulated application model was presented in [97] and is composed of six tasks: video source (a sensor fetching images from a camera), renderer (screen acting as an actuator), tracker (analysis of video frames), mapper (map generation and refinement), relocalizer (relocation of camera position) and object recognizer (localization of known objects). Video source and renderer are built on devices and do not require processing on a VM. Tracker and relocalizer virtual machines must be instantiated in a fog node, and the mapper and object recognizer can be hosted either in the fog or in the cloud. The flow of exchanged data is shown in Figure 9.2b.

Virtual machine arrival and departure process in cloud data centers exhibit self-similarity [49]. Consequently, the system request arrival was modeled as a self-similar series. The algorithm in [81] is used to generate such series, i.e., the number of requests received per minute. The input for this generator is the values of the mean and standard deviation of the series, and the Hurst parameter H. The output is a sequence of numbers representing the number of requests arrived each minute. Different series were created using different parameters for simulating various loads; these load scenarios and their parameters are shown in Table 9.3, all employing $H = 0.7$.

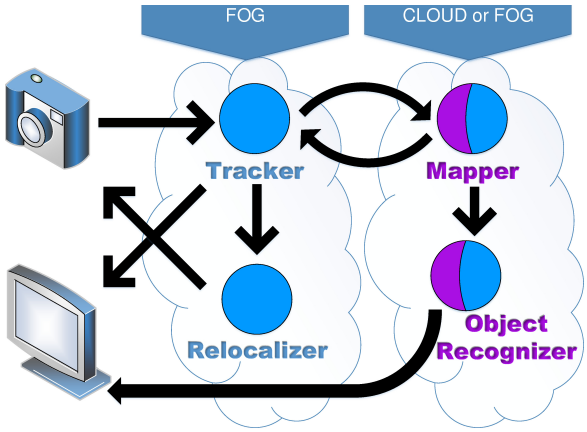
All VMs of the same request are started and released at the same time; a 5-minute lifetime is adopted. Scenarios with several requests of a single type of application were simulated. In the simulation of remote VM application, which requires a single VM per request, half of the requests needed be hosted in the fog and the other half can be hosted either in the fog or in the cloud. For the augmented reality application such a distinction is not necessary since it is already formed by tasks with different latency requirements.

9.4.4 Energy consumption model

The energy consumed in the fog node and the cloud data center is calculated as the sum of the energy consumed by the physical servers; such consumption depends on the



(a) Remote VM



(b) Augmented reality

Figure 9.2: Simulated applications. Each circle represents a task: blue represents exclusive fog tasks, while blue and purple tasks can be hosted either in fog or cloud. Arrows represent communication between tasks.

Table 9.3: Simulated scenarios and parameters employed in the generator [81]. The standard deviation (SD) is half the value of the mean arrival rate.

Scenario	**Mean**	**SD**
m2	2	1
m4	4	2
m6	6	3
m8	8	4
m10	10	5
m12	12	6
m14	14	7
m16	16	8
m18	18	9
m20	20	10

Table 9.4: Power of cloud and fog servers.

CPU Load (%)	**0**	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
Consumption (W)	52.3	93.6	106	116	126	136	147	163	180	199	222

CPU usage. CPUs fully utilized lead to the highest server energy consumption, but idle servers consume about 70 % of this value. A linear model is employed to model intermediary values of consumption, as in [12]. The energy consumption model is based on a real benchmark of SPEC power, which measured power for different CPU usage levels [1]. These values are interpolated to calculate the energy consumption, considering the current load imposed by the hosted VMs. Table 9.4 presents reference values for the employed server.

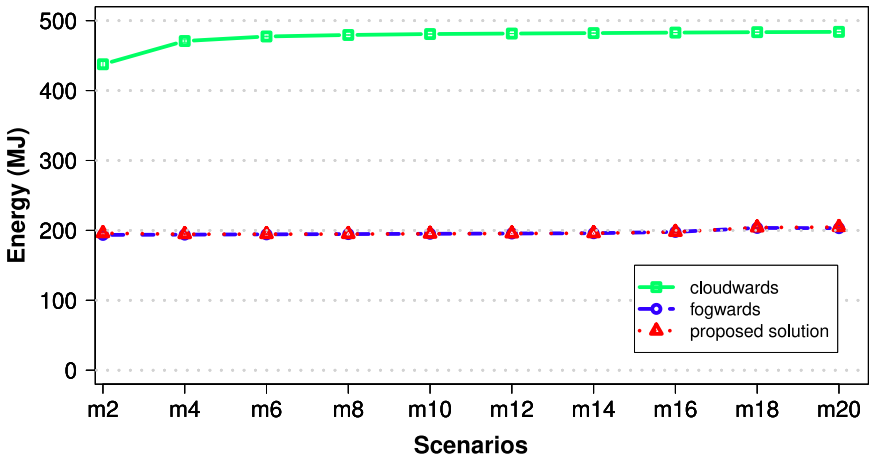
As suggested in [93], fog nodes present similar hardware to those of cloud data centers; thus, only a single energy consumption model was adopted. However, the fog does not employ the same cooling and network equipment. Data in the report [17] identifies that state-of-the-art data centers in 2008 had a PUE metric equals 1.4, which means that for each watt spent in computation, 0.4 watt is pent in cooling for these data centers. Therefore, different PUE values are employed in the simulations: 1.4 for the cloud and 1.0 for the fog.

9.4.5 Numerical results

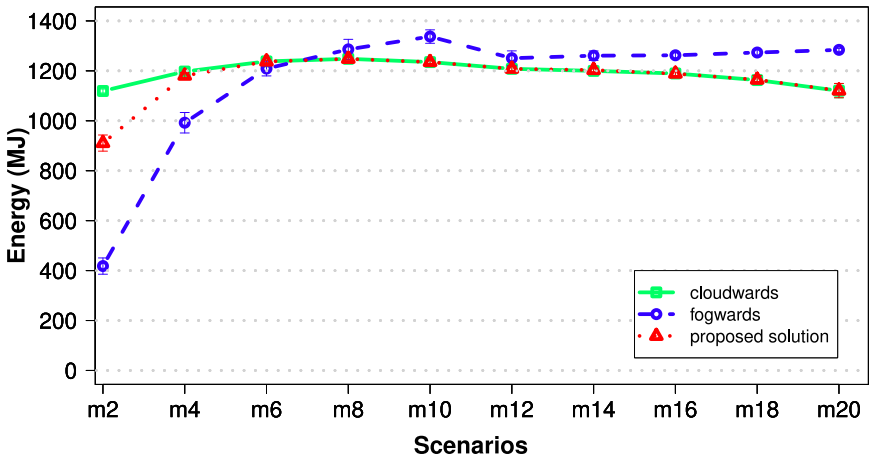
The metrics evaluated are energy consumption, blocking ratio, and latency. The energy consumption was accounted for the fog and cloud infrastructures, considering the PUE. Figure 9.3 shows the results for the energy consumption. For the remote VM application, fogwards produced the lowest energy consumption. Remote VM application requires only one virtual machine per request; thus, it hardly causes an overload of resources in the fog, allowing the GPRFCA mechanism to obtain energy efficiency similar to that of fogwards. Augmented reality application had a different tendency since each request contains four virtual machines. Under low arrival rates, the fog manages to host several VMs; thus, fogwards policy produced the lowest energy consumption. However, starting from the m8 scenario, the system becomes overloaded, forcing VMs to be hosted on the cloud. In these cases, the fogwards policy overloads the fog, increasing blocking. The GPRFCA mechanism, however, produced similar results to those of cloudwards.

Figure 9.4 shows the blocking ratio, which is the ratio between the number of applications that were not instantiated and the number of submitted requests. Blocking happens whenever a VM does not have its requirements satisfied due to the lack of physical resources in the fog. Remote VM produced no blocking due to the low demand of VMs. In contrast, augmented reality application, which submits four VMs per request, produced great blocking. Cloudwards produced lower blocking than does fogwards since it forwards all possible VMs to the cloud, where resources are abundant. Blocking under cloudwards occurs due to VMs which must be placed in the fog and there are no physical resources available for that. In fogwards, on the other hand, blocking happens because the fog node

[1] `https://www.spec.org/power_ssj2008/`

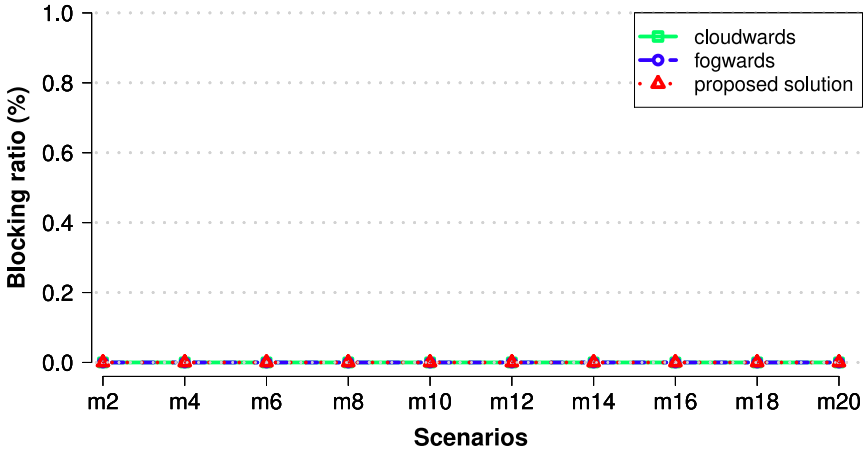


(a) Remote VM

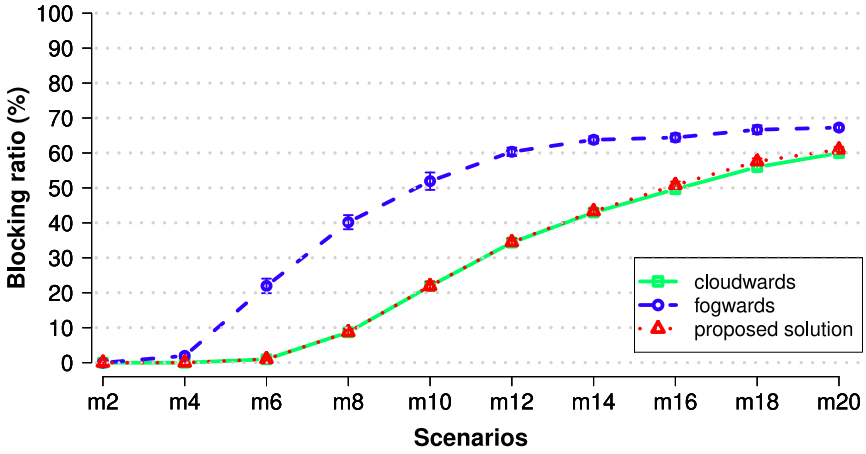


(b) Augmented reality

Figure 9.3: Total energy for different applications.



(a) Remote VM

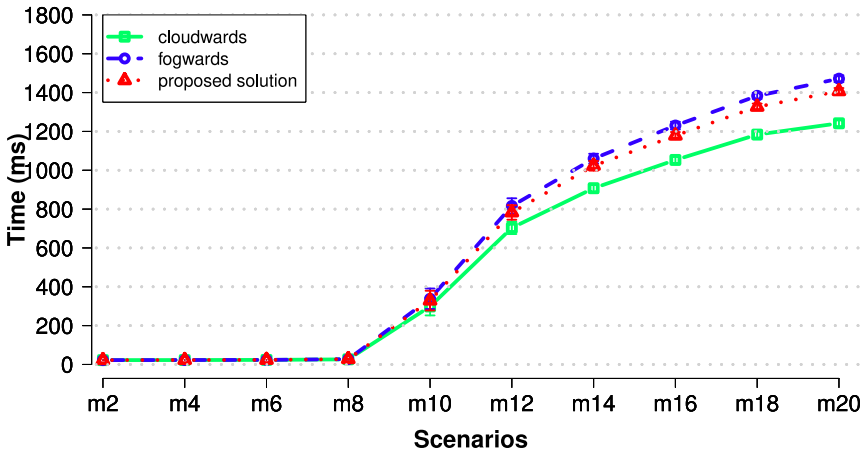


(b) Augmented reality

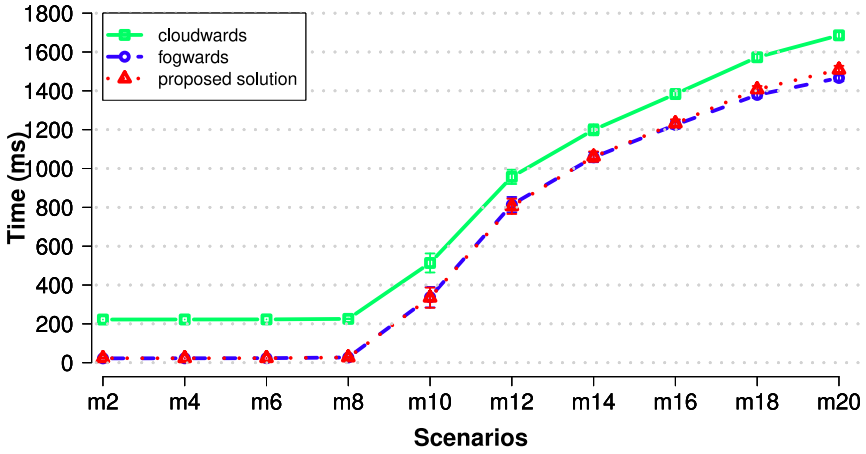
Figure 9.4: Blocking ratio.

is overloaded with tasks that are not sensitive to delays. The GPRFCA mechanism tends to forward virtual machines to the edge, saving resources for future requests, which decreased the load on the fog node. As a consequence, it produced a blocking ratio similar to that produced by cloudwards, while maintaining the energy consumption at a reasonable level.

Figure 9.5 presents the values for latency in the Remote VM application for tasks with strict latency requirements and also flexible tasks. The latency is the average time elapsed between an action of a user at the remote terminal (a pressed key or mouse movement) and rendering its results on the user's screen. Fogwards produces the smallest latency values since VMs are closer to end-users. The GPRFCA mechanism produces latency values between those produced by the two other policies, but values are closer to those given by fogwards. Under load conditions equal or heavier than that of scenario m14, the latency is higher than 1 second, which can jeopardize the application performance. High



(a) Strict latency virtual machines which can be placed only on the fog



(b) Flexible virtual machines which can be placed either on the fog or on the cloud

Figure 9.5: Latency for Remote VM application.

latency values indicate that more resources in the fog should be deployed.

The advantage of employing the GPRFCA mechanism can be summarized as follows. The energy consumption is always between those produced by cloudwards and fogwards and it is always close to the lowest produced value by these two mechanisms. GPRFCA tries to produce high utilization in fog nodes and consequently reduces the energy consumption and blocking ratio. The GPRFCA mechanism does not increase the latency of requests which have VMs that can be placed in the cloud. Moreover, it produces low latency for tasks that must be hosted in the fog.

9.5 Conclusions

This chapter proposed a novel mechanism for the provisioning of resources in a fog and cloud environment named the Gaussian Process Regression for Fog-Cloud Allocation mechanism, which aims at reducing the overall energy consumption while supporting latency requirements. Gaussian process regression was employed to estimate the demand of future arrivals so that fog nodes can host future requests, preventing the blocking of new requests. Results obtained show that the GPRFCA mechanism maintains the energy consumption at acceptable levels and avoids overloading the fog reducing the blocking of requests. Latency was also kept at reasonable levels and decreased for applications with flexible latency requirements. The GPRFCA mechanism manages to take advantage of the good features of both cloudwards and fogwards mechanisms.

Part V

Final Remarks

Chapter 10

Conclusions

This chapter concludes this thesis and is divided into four sections: Section 10.1 presents the main findings, Section 10.2 lists the main limitations and challenges, and Section 10.3 suggests future research directions.

10.1 Main findings

The fog node location is a crucial decision in the deployment of a fog computing infrastructure since fog nodes are distributed and need to be spatially close to end-users to meet their latency constraints. If inadequate decisions are made, users' requests will not be processed and the computational infrastructure will remain underloaded. Solutions presented in this thesis alleviate such problems by a proper design of the infrastructure, assuming different goals and types of fog nodes. Results obtained showed that our solutions can be quite efficient to design a fog computing infrastructure. The remainder of this section draws the overall conclusions of all chapters.

Dealing with variable demands in time and space is a challenge to solve the fog node location problem. The work about the terrestrial infrastructure (Part II) showed that reducing the number of fog servers does not have a proportional impact on the acceptance ratio due to the low demands. In line with that, Chapter 6 showed that a large part of the terrestrial infrastructure could be replaced by UAVs. Future solutions to the fog node location problem should address the demand variability so that deployment costs are optimized.

Obtaining exact solutions to the fog node location problem is not always possible, especially for a large number of locations, for formulations considering individual requests, or when mobile fog nodes are adopted. The number of locations increases the possible arrangements of fog nodes, increasing the number of constraints in an exact model. A small number of locations is useful to make comparisons between heuristics and exact models, but it seldom represents realistic deployments, such as large cities or metropolitan areas. Considering individual requests (e.g. as in Chapters 5, 7, and 8) requires a long computation time compared to approaches in which the workload and fog node capacity are given by real numbers (Chapters 4 and 6). However, individual requests in the formulation are needed to quantify the energy consumption or the delay. Future formulations

may combine both models to speed up the execution time. Finally, considering mobile fog nodes (UAVs) leads to formulations with a very large number of constraints. This is due to all possible locations and states UAVs can be at different time intervals. Nonetheless, heuristics can benefit from locating one UAV at a time to avoid an exponential increase in time complexity.

UAVs can be pretty efficient as fog nodes to complement the terrestrial network. UAVs present better flexibility than do other vehicles, but their energy consumption must be accounted for to guarantee an efficient operation. Both rotary-wing and fixed-wing UAVs can be used as fog nodes; the former is useful to provide stable and low latency processing, while the latter can endure a longer operation due to reduced energy consumption. If UAVs can operate tethered or on rooftops, operation time can be largely improved.

This thesis presented different solutions to the fog node location problem, with exact models and heuristics, and their evaluation under several conditions. There is no ready-to-use formulation: the characteristics of the problem must be taken into account to solve the fog node location. The work in this thesis pushes the boundaries of this problem and can serve as a basis for future work.

10.2 Limitations and challenges

This section lists some limitations and challenges in the development of this thesis. The limitations do not diminish the quality of this work but rather suggest new directions for research. This section also guides future researchers on the possible challenges they will face while conducting research in the areas of fog node location and UAV communications.

In all approaches to the location of fog nodes, the deployment cost was taken into consideration by limiting the number of fog nodes. However, there are other sources of costs that were not accounted for in this thesis, such as the cost of the facility to host fixed servers (physical infrastructure, power and network connections, cooling, security), the infrastructure to support UAV operations, and human resources. One limitation to evaluating these aspects is the estimation of such costs since this information is not easily found in the literature and these costs usually vary for different countries, providers, and technologies. Nonetheless, the cost components considered in this thesis are a significant part of the deployment costs, and solutions can be easily adapted to consider other components.

An important decision in this thesis was the data set used in the evaluation, mainly the one from [8], described in Section 3.5. Evaluating the solutions in this thesis with other data sets could have led to a broader discussion; however, there are only a few open data sets representing variable demands in time and space. For example, some cellular data sets used in previous papers are not publicly available. Moreover, pre-processing data sets is a time-consuming task. The data sets used in this thesis led to notable results, comparable to those obtained by other authors that employed different data, which suggests this work is not biased towards a single data set. Nonetheless, future work can employ other data sets without the need to develop new formulations.

The solutions in this thesis can be improved to consider more characteristics of applica-

tions and fog nodes. Applications can be characterized by specific processing, networking, and storage requirements. Formulations can also be adapted to specific hardware, such as different processor architectures or UAV models. Different UAVs trajectories can be evaluated. Considering these aspects require some modifications in the solutions proposed in this thesis. However, such modifications lead to formulations tailored to specific situations; this thesis introduces broad solutions that can be adapted to different needs.

There was an effort in this thesis to make realistic assumptions in the simulations. However, a testbed with real hardware would be useful to evaluate situations not easily simulated, such as weather conditions' effect on UAVs, software response time, and actual wireless channels. Such testbeds are quite expensive and require several hours of work to make them operational. These testbeds would have required more researchers involved in the project and a less limited budget, which was not possible during the Ph.D. Nonetheless, results in this thesis can serve as a basis for physical deployments. Moreover, the employment of simulations allowed a broad evaluation and replication of tests that might not have been possible with only a physical testbed.

Finally, the work about UAVs in this thesis did not consider solutions for recharging UAVs during their operation. However, in real deployments, UAVs are periodically recharged to avoid the need for many aircraft. Aspects to be further explored are the employment of recharge stations for UAVs, especially for UAVs employed as fog nodes. Despite the importance of recharging, studying such an aspect leads to completely new research problems out of the scope of the location of fog nodes.

10.3 Future work

In addition to studies to fill the gaps listed in Section 10.2, this section suggests opportunities for future investigation. The work is in this thesis can be extended in different manners and we intend to explore some of these gaps in the future.

This thesis considered only the employment of LAPs. However, HAPs can also be used as fog nodes. Future solutions for the fog node location problem can consider only HAPs as well as the integration between the two types of UAVs. HAPs fly at much higher altitudes and can move at different speeds compared to LAPs. These aspects should be taken into consideration in the design of fog computing infrastructures based on HAPs.

Another research direction is the employment of other energy and wireless channel conditions. For example, UAV and environment parameters (such as weight, wing size, and air density) could be used to estimate the energy consumption of specific UAV models, allowing the use of solutions of this thesis for a variety of aircraft. Moreover, channel models based on measurements of real cities could be used to evaluate the impact of different city densities. Other possissibilities of future work include studying handover when a user connects to multiple UAVs, and predicting future demands to dispatch UAVs in advance.

Bibliography

[1] *Cisco Data Center Infrastructure 2.5 Design Guide*. December 2007.

[2] 3GPP. Technical Specification Group Radio Access Network; Study on Enhanced LTE Support for Aerial Vehicles. Technical Report (TR) 36.777, 3rd Generation Partnership Project (3GPP), 12 2017. Version 15.0.0.

[3] Hasini Viranga Abeywickrama, Beeshanga Abewardana Jayawickrama, Ying He, and Eryk Dutkiewicz. Empirical power consumption model for uavs. In *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, pages 1–5, 2018.

[4] Arif Ahmed, HamidReza Arkian, Davaadorj Battulga, Ali J. Fahs, Mozhdeh Farhadi, Dimitrios Giouroukis, Adrien Gougeon, Felipe Oliveira Gutierrez, Guillaume Pierre, Paulo R. Souza Jr au2, Mulugeta Ayalew Tamiru, and Li Wu. Fog computing applications: Taxonomy and requirements, 2019.

[5] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. *SIGCOMM Comput. Commun. Rev.*, 38(4):63–74, August 2008.

[6] S. Alanazi, M. Dabbagh, B. Hamdaoui, M. Guizani, and N. Zorba. Reducing data center energy consumption through peak shaving and locked-in energy avoidance. *IEEE Transactions on Green Communications and Networking*, 1(4):551–562, Dec 2017.

[7] E. Baccarelli, P. G. V. Naranjo, M. Scarpiniti, M. Shojafar, and J. H. Abawajy. Fog of everything: Energy-efficient networked computing architectures, research challenges, and a case study. *IEEE Access*, 5:9882–9910, 2017.

[8] Gianni Barlacchi, Marco De Nadai, Roberto Larcher, Antonio Casella, Cristiana Chitic, Giovanni Torrisi, Fabrizio Antonelli, Alessandro Vespignani, Alex Pentland, and Bruno Lepri. A multi-source dataset of urban life in the city of milan and the province of trentino. *Scientific Data*, 2, Oct 2015.

[9] Riccardo Bassoli, Fabrizio Granelli, Claudio Sacchi, Stefano Bonafini, and Frank H.P. Fitzek. Cubesat-based 5g cloud radio access networks: A novel paradigm for on-demand anytime/anywhere connectivity. *IEEE Veh. Technol. Mag.*, 15(2):39–47, 2020.

[10] A. Bayati, V. Asghari, K. Nguyen, and M. Cheriet. Gaussian process regression based traffic modeling and prediction in high-speed networks. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7, Dec 2016.

[11] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, 28(5):755–768, 2012. Special Section: Energy efficiency in large-scale distributed systems.

[12] Anton Beloglazov and Rajkumar Buyya. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurrency and Computation: Practice and Experience*, 24(13):1397–1420, 2012.

[13] L. F. Bittencourt, E. R. M. Madeira, and N. L. S. Da Fonseca. Scheduling in hybrid clouds. *IEEE Communications Magazine*, 50(9):42–47, September 2012.

[14] Luiz F. Bittencourt, Javier Diaz-Montes, Rajkumar Buyya, Omer F. Rana, and Manish Parashar. Mobility-aware application scheduling in fog computing. *IEEE Cloud Computing*, 4(2):26–35, 2017.

[15] Luiz Fernando Bittencourt and Edmundo Roberto Mauro Madeira. Hcoc: a cost optimization algorithm for workflow scheduling in hybrid clouds. *Journal of Internet Services and Applications*, 2(3):207–227, Dec 2011.

[16] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh 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, page 13–16, New York, NY, USA, 2012. Association for Computing Machinery.

[17] Richard Brown, Eric Masanet, Bruce Nordman, Bill Tschudi, Arman Shehabi, John Stanley, Jonathan Koomey, Dale Sartor, and Peter Chan. Report to congress on server and data center energy efficiency: Public law 109-431. *Lawrence Berkeley National Laboratory*, 2008.

[18] Giuseppe Bruno, Andrea Genovese, and Gennaro Improta. A historical perspective on location problems. *BSHM Bulletin: Journal of the British Society for the History of Mathematics*, 29(2):83–97, 2014.

[19] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose, and Rajkumar 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.

[20] X. Cao, J. Xu, and R. Zhang. Mobile edge computing for cellular-connected uav: Computation offloading and trajectory optimization. In *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pages 1–5, 2018.

[21] Massimiliano Caramia and Paolo Dell´Olmo. *Multi-objective Management in Freight Logistics*. Springer London, 2008.

[22] Aaron Carroll and Gernot Heiser. The systems hacker's guide to the galaxy energy usage in a modern smartphone. In *Proceedings of the 4th Asia-Pacific Workshop on Systems*, APSys '13, pages 5:1–5:7, New York, NY, USA, 2013. ACM.

[23] Ayon Chakraborty, Eugene Chai, Karthikeyan Sundaresan, Amir Khojastepour, and Sampath Rangarajan. Skyran: A self-organizing lte ran in the sky. In *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT)*, page 280–292. Association for Computing Machinery, 2018.

[24] Longbiao Chen, Linjin Liu, Xiaoliang Fan, Johnthan Li, Cheng Wang, Gang Pan, Jérémie Jakubowicz, and Thi-Mai-Trang Nguyen. Complementary base station clustering for cost-effective and energy-efficient cloud-ran. In *2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pages 1–7, 2017.

[25] Yunfei Chen, Xiaonan Liu, Nan Zhao, and Zhiguo Ding. Using multiple uavs as relays for reliable communications. In *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, pages 1–5, 2018.

[26] M.A. Chilenski, M. Greenwald, Y. Marzouk, N.T. Howard, A.E. White, J.E. Rice, and J.R. Walk. Improved profile fitting and quantification of uncertainty in experimental measurements of impurity transport coefficients using gaussian process regression. *Nuclear Fusion*, 55(2):023012, 2015.

[27] Miguel T. Covas, Carlos A. Silva, and Luis C. Dias. Multicriteria decision analysis for sustainable data centers location. *International Transactions in Operational Research*, 20(3):269–299, 2012.

[28] Yong Cui, Jian Song, Kui Ren, Minming Li, Zongpeng Li, Qingmei Ren, and Yangjun Zhang. Software defined cooperative offloading for mobile cloudlets. *IEEE/ACM Trans. Netw.*, 25(3):1746–1760, June 2017.

[29] Nelson L. S. da Fonseca and Raouf Boutaba. *Cloud Services, Networking, and Management*. John Wiley & Sons, Ltd, 2015.

[30] R. A. C. da Silva and N. L. S. da Fonseca. Energy-aware migration of groups of virtual machines in distributed data centers. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2016.

[31] Rodrigo A. C. da Silva and Nelson L. S. da Fonseca. Topology-aware virtual machine placement in data centers. *Journal of Grid Computing*, pages 1–16, 2015.

[32] David Daly. A Not-So-Short History of Unmanned Aerial Vehicles (UAV). Online. Available at `https://consortiq.com/short-history-unmanned-aerial-vehicles-uavs/` [Accessed: 24/11/2021].

[33] R. Deng, R. Lu, C. Lai, and T. H. Luan. Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing. In *2015 IEEE International Conference on Communications (ICC)*, pages 3909–3914, June 2015.

[34] Q. Fan and N. Ansari. Cost aware cloudlet placement for big data processing at the edge. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–6, May 2017.

[35] Giuseppe Faraci, Christian Grasso, and Giovanni Schembra. Fog in the clouds: Uavs to provide edge computing to iot devices. *ACM Trans. Internet Technol.*, 20(3):1–26, August 2020.

[36] Reza Zanjirani Farahani and Masoud Hekmatfar, editors. *Facility Location*. Physica-Verlag HD, 2009.

[37] Tom G. Farr, Paul A. Rosen, Edward Caro, Robert Crippen, Riley Duren, Scott Hensley, Michael Kobrick, Mimi Paller, Ernesto Rodriguez, Ladislav Roth, David Seal, Scott Shaffer, Joanne Shimada, Jeffrey Umland, Marian Werner, Michael Oskin, Douglas Burbank, and Douglas Alsdorf. The shuttle radar topography mission. *Reviews of Geophysics*, 45(2), 2007.

[38] C. Fiandrino, N. Allio, D. Kliazovich, P. Giaccone, and P. Bouvry. Profiling performance of application partitioning for wearable devices in mobile cloud and fog computing. *IEEE Access*, 7:12156–12166, 2019.

[39] Boris Galkin, Jacek Kibilda, and Luiz A. DaSilva. Uavs as mobile infrastructure: Addressing battery lifetime. *IEEE Commun. Mag.*, 57(6):132–137, 2019.

[40] Andres Garcia-Saavedra, Pablo Serrano, Albert Banchs, and Giuseppe Bianchi. Energy consumption anatomy of 802.11 devices and its implication on modeling and design. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '12, pages 169–180, New York, NY, USA, 2012. ACM.

[41] N. K. Giang, M. Blackstock, R. Lea, and V. C. M. Leung. Developing iot applications in the fog: A distributed dataflow approach. In *2015 5th International Conference on the Internet of Things (IOT)*, pages 155–162, Oct 2015.

[42] Fabio Giust, Xavier Costa-Perez, and Alex Reznik. Multi-access edge computing: An overview of etsi mec isg. *IEEE 5G Tech Focus*, 1(4):4, 2017.

[43] Marta C. Gonzalez, Cesar A. Hidalgo, and Albert-Laszlo Barabasi. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, June 2008.

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

[45] J. C. Guevara, L. F. Bittencourt, and N. L. S. da Fonseca. Class of service in fog computing. In *2017 IEEE 9th Latin-American Conference on Communications (LATINCOM)*, pages 1–6, Nov 2017.

[46] Harshit Gupta, Amir Vahid Dastjerdi, Soumya K. Ghosh, and Rajkumar Buyya. 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(9):1275–1296, 2017.

[47] LLC. Gurobi Optimization. Gurobi Optimizer. Online. Available at `https://www.gurobi.com/` [Accessed: 03/12/2021].

[48] D. Han, W. Chen, and J. Liu. Energy-efficient uav communications under stochastic trajectory: A markov decision process approach. *IEEE Trans. Green Commun. Netw.*, 5(1):106–118, 2021.

[49] Yi Han, J. Chan, and C. Leckie. Analysing virtual machine usage in cloud computing. In *Services (SERVICES), 2013 IEEE Ninth World Congress on*, pages 370–377, June 2013.

[50] Junxian Huang, Feng Qian, Alexandre Gerber, Z. Morley Mao, Subhabrata Sen, and Oliver Spatscheck. A close examination of performance and power characteristics of 4g lte networks. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, MobiSys '12, pages 225–238, New York, NY, USA, 2012. ACM.

[51] Fatemeh Jalali, Kerry Hinton, Robert Ayre, Tansu Alpcan, and Rodney S. Tucker. Fog computing may help to save energy in cloud computing. *IEEE Journal on Selected Areas in Communications*, 34(5):1728–1739, 2016.

[52] S. Jeong, O. Simeone, and J. Kang. Mobile edge computing via a uav-mounted cloudlet: Optimization of bit allocation and path planning. *IEEE Trans. Veh. Technol.*, 67(3):2049–2063, 2018.

[53] Jiequ Ji, Kun Zhu, Changyan Yi, and Dusit Niyato. Energy consumption minimization in uav-assisted mobile-edge computing systems: Joint resource allocation and trajectory design. *IEEE Internet Things J.*, 8(10):8570–8584, 2021.

[54] M. Jia, J. Cao, and W. Liang. Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks. *IEEE Transactions on Cloud Computing*, 5(4):725–737, Oct 2017.

[55] Karen E. Joyce, Karen Anderson, and Renee E. Bartolo. Of course we fly unmanned—we're women! *Drones*, 5(1), 2021.

[56] Krishna P. Kadiyala and Jorge A. Cobb. Inter-as traffic engineering with sdn. In *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 1–7, 2017.

[57] A. Kapsalis, P. Kasnesis, I. S. Venieris, D. I. Kaklamani, and C. Z. Patrikakis. A cooperative fog approach for effective workload balancing. *IEEE Cloud Computing*, 4(2):36–45, March 2017.

[58] Wahab Khawaja, Ismail Guvenc, David W. Matolak, Uwe-Carsten Fiebig, and Nicolas Schneckenberger. *A Survey of Air-to-Ground Propagation Channel Modeling for Unmanned Aerial Vehicles*, chapter 2, pages 17–70. John Wiley & Sons, Ltd, 2020.

[59] W. Kim and S. Chung. User-participatory fog computing architecture and its management schemes for improving feasibility. *IEEE Access*, 6:20262–20278, 2018.

[60] Mustafa Kishk, Ahmed Bader, and Mohamed-Slim Alouini. Aerial base station deployment in 6g cellular networks using tethered drones: The mobility and endurance tradeoff. *IEEE Veh. Technol. Mag.*, 15(4):103–111, 2020.

[61] Daniel Guimaraes do Lago, Edmundo R. M. Madeira, and Luiz Fernando Bittencourt. Power-aware virtual machine scheduling on clouds using active cooling control and dvfs. MGC '11, New York, NY, USA, 2011. Association for Computing Machinery.

[62] F. Larumbe and B. Sansò. Cloptimus: A multi-objective cloud data center and software component location framework. In *2012 IEEE 1st International Conference on Cloud Networking (CLOUDNET)*, pages 23–28, Nov 2012.

[63] F. Larumbe and B. Sansò. A tabu search algorithm for the location of data centers and software components in green cloud computing networks. *IEEE Transactions on Cloud Computing*, 1(1):22–35, Jan 2013.

[64] Mushu Li, Nan Cheng, Jie Gao, Yinlu Wang, Lian Zhao, and Xuemin Shen. Energy-efficient uav-assisted mobile edge computing: Resource allocation and trajectory optimization. *IEEE Trans. Veh. Technol.*, 69(3):3424–3438, 2020.

[65] Xingqin Lin, Vijaya Yajnanarayana, Siva D. Muruganathan, Shiwei Gao, Henrik Asplund, Helka-Liina Maattanen, Mattias Bergstrom, Sebastian Euler, and Y.-P. Eric Wang. The sky is not the limit: Lte for unmanned aerial vehicles. *IEEE Commun. Mag.*, 56(4):204–210, 2018.

[66] Feng Luo, Chunxiao Jiang, Shui Yu, Jingjing Wang, Yipeng Li, and Yong Ren. Stability of cloud-based uav systems supporting big data acquisition and processing. *IEEE Trans. Cloud Comput.*, 7(3):866–877, 2019.

[67] Tero Lähderanta, Teemu Leppänen, Leena Ruha, Lauri Lovén, Erkki Harjula, Mika Ylianttila, Jukka Riekki, and Mikko J. Sillanpää. Edge computing server placement with capacitated location allocation. *Journal of Parallel and Distributed Computing*, 153:130–149, 2021.

[68] Redowan Mahmud, Kotagiri Ramamohanarao, and Rajkumar Buyya. Latency-aware application module management for fog computing environments. *ACM Trans. Internet Technol.*, 19(1), nov 2018.

[69] Redowan Mahmud, Kotagiri Ramamohanarao, and Rajkumar Buyya. Latency-aware application module management for fog computing environments. *ACM Trans. Internet Technol.*, 19(1):1–21, nov 2018.

[70] Eva Marín-Tordera, Xavi Masip-Bruin, Jordi García-Almiñana, Admela Jukan, Guang-Jie Ren, and Jiafeng Zhu. Do we all really know what a fog node is? current trends towards an open definition. *Computer Communications*, 109:117 – 130, 2017.

[71] X. Masip-Bruin, E. Marín-Tordera, G. Tashakor, A. Jukan, and G. Ren. Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud computing systems. *IEEE Wireless Communications*, 23(5):120–128, 2016.

[72] Peter Mell and Tim Grance. The NIST Definition of Cloud Computing. Technical report, July 2009.

[73] Antti P. Miettinen and Jukka K. Nurminen. Energy efficiency of mobile clients in cloud computing. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, pages 4–4, Berkeley, CA, USA, 2010. USENIX Association.

[74] Nader Mohamed, Jameela Al-Jaroodi, Imad Jawhar, Hassan Noura, and Sara Mahmoud. Uavfog: A uav-based fog computing for internet of things. In *2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pages 1–8, 2017.

[75] Mehrdad Moradi, Karthikeyan Sundaresan, Eugene Chai, Sampath Rangarajan, and Z. Morley Mao. Skycore: Moving core to the edge for untethered and reliable UAV-based LTE networks. In *MobiCom '18*, page 35–49, New York, NY, USA, 2018.

[76] Mobile Cloud Computing. Online, 8 2016. Available at `https://www.nist.gov/programs-projects/mobile-cloud-computing` [Accessed: 22/11/2021].

[77] E. M. R. Oliveira and A. C. Viana. From routine to network deployment for data offloading in metropolitan areas. In *2014 Eleventh Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 126–134, June 2014.

[78] OpenCellID. Online. Available at `http://www.opencellid.org` (accessed on 1 January 2019).

[79] OpenFog Reference Architecture. Online, 2017. Available at `https://www.openfogconsortium.org/ra/` [Accessed: 24/05/2017].

[80] Cisco White Paper. Cisco visual networking index: Global mobile data traffic forecast update, 2017–2022. February 2019.

[81] Vern Paxson. Fast, approximate synthesis of fractional gaussian noise for generating self-similar network traffic. *SIGCOMM Comput. Commun. Rev.*, 27(5):5–18, October 1997.

[82] Quoc-Viet Pham, Ming Zeng, Rukhsana Ruby, Thien Huynh-The, and Won-Joo Hwang. Uav communications for sustainable federated learning. *IEEE Trans. Veh. Technol.*, 70(4):3944–3948, 2021.

[83] Xuan-Qui Pham, Nguyen Doan Man, Nguyen Dao Tan Tri, Ngo Quang Thai, and Eui-Nam Huh. A cost- and performance-effective approach for task scheduling based on collaboration between cloud and fog computing. *International Journal of Distributed Sensor Networks*, 13(11):1550147717742073, 2017.

[84] Carlo Puliafito, Diogo M. Gonçalves, Márcio M. Lopes, Leonardo L. Martins, Edmundo Madeira, Enzo Mingozzi, Omer Rana, and Luiz F. Bittencourt. Mobfogsim: Simulation of mobility and migration for fog computing. *Simulation Modelling Practice and Theory*, 101:102062, 2020. Modeling and Simulation of Fog Computing.

[85] Walid Saad, Mehdi Bennis, Mohammad Mozaffari, and Xingqin Lin. *Wireless Communications and Networking for Unmanned Aerial Vehicles*. Cambridge University Press, 2020.

[86] Subhadeep Sarkar, Subarna Chatterjee, and Sudip Misra. Assessment of the suitability of fog computing in the context of internet of things. *IEEE Transactions on Cloud Computing*, 6(1):46–59, 2015.

[87] Mahadev Satyanarayanan, Paramvir Bahl, Ramon Caceres, and Nigel Davies. The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, 8(4):14–23, 2009.

[88] Shanza Shakoor, Zeeshan Kaleem, Dinh-Thuan Do, Octavia A. Dobre, and Abbas Jamalipour. Joint optimization of uav 3-d placement and path-loss factor for energy-efficient maximal coverage. *IEEE Internet Things J.*, 8(12):9776–9786, 2021.

[89] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.

[90] V. B. Souza, X. Masip-Bruin, E. Marín-Tordera, W. Ramírez, and S. Sánchez-López. Proactive vs reactive failure recovery assessment in combined fog-to-cloud (f2c) systems. In *2017 IEEE 22nd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pages 1–5, June 2017.

[91] V. B. C. Souza, W. Ramírez, X. Masip-Bruin, E. Marín-Tordera, G. Ren, and G. Tashakor. Handling service allocation in combined fog-cloud scenarios. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–5, 2016.

[92] Vitor Barbosa Souza, Xavi Masip-Bruin, Eva Marin-Tordera, Wilson Ramirez, and Sergio Sanchez. Towards distributed service allocation in fog-to-cloud (f2c) scenarios. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2016.

[93] X. Sun, N. Ansari, and Q. Fan. Green energy aware avatar migration strategy in green cloudlet networks. In *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 139–146, Nov 2015.

[94] M. Taneja and A. Davy. Resource aware placement of iot application modules in fog-cloud computing paradigm. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 1222–1228, May 2017.

[95] M. Usman, A. Akhtar, M. Qaraqe, and F. Granelli. Remote cloud vs local mobile cloud: A quantitative analysis. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, Dec 2018.

[96] Muhammad Usman. *Energy Efficiency and Privacy in Device-to-Device Communication*. PhD thesis, University of Trento, 12 2017. available at https://www.semanticscholar.org/paper/Energy-Efficiency-and-Privacy-in-Device-to-Device-Usman/c47aeb7ff44142b3166d363cbeec8d366671a828.

[97] Tim Verbelen, Pieter Simoens, Filip De Turck, and Bart Dhoedt. Cloudlets: Bringing the cloud to the mobile user. In *Proceedings of the Third ACM Workshop on Mobile Cloud Computing and Services*, MCS '12, pages 29–36, New York, NY, USA, 2012. ACM.

[98] R. Vilalta, V. Lopez, A. Giorgetti, S. Peng, V. Orsini, L. Velasco, R. Serral-Gracia, D. Morris, S. De Fina, F. Cugini, P. Castoldi, A. Mayoral, R. Casellas, R. Martinez, C. Verikoukis, and R. Munoz. Telcofog: A unified flexible fog and cloud computing architecture for 5g networks. *IEEE Communications Magazine*, 55(8):36–43, 2017.

[99] J. Wang, K. Liu, and J. Pan. Online uav-mounted edge server dispatching for mobile-to-mobile edge computing. *IEEE Internet Things J.*, 7(2):1375–1386, Feb 2020.

[100] X. Wang, S. Leng, and K. Yang. Social-aware edge caching in fog radio access networks. *IEEE Access*, 5:8492–8501, 2017.

[101] Y. Yang, K. Wang, G. Zhang, X. Chen, X. Luo, and M. Zhou. Meets: Maximal energy efficient task scheduling in homogeneous fog networks. *IEEE Internet of Things Journal*, 5(5):4076–4087, Oct 2018.

[102] Hong Yao, Changmin Bai, Deze Zeng, Qingzhong Liang, and Yuanyuan Fan. Migrate or not? exploring virtual machine migration in roadside cloudlet-based vehicular cloud. *Concurrency and Computation: Practice and Experience*, 27(18):5780–5792, 2015.

[103] D. Ye, M. Wu, S. Tang, and R. Yu. Scalable fog computing with service offloading in bus networks. In *2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud)*, pages 247–251, 2016.

[104] 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. *Journal of Systems Architecture*, 98:289 – 330, 2019.

[105] Yong Zeng, Jie Xu, and Rui Zhang. Energy minimization for wireless communication with rotary-wing uav. *IEEE Trans. Wireless Commun*, 18(4):2329–2345, 2019.

[106] Yong Zeng and Rui Zhang. Energy-efficient uav communication with trajectory optimization. *IEEE Trans. Wireless Commun*, 16(6):3747–3760, 2017.

[107] Yongs Zeng, Qingqing Wu, and Rui Zhang. Accessing from the sky: A tutorial on uav communications for 5g and beyond. *Proceedings of the IEEE*, 107(12):2327–2375, 2019.

[108] G. Zhang, F. Shen, Z. Liu, Y. Yang, K. Wang, and M. Zhou. Femto: Fair and energy-minimized task offloading for fog-enabled iot networks. *IEEE Internet of Things Journal*, 6(3):4388–4400, June 2019.

[109] Kaiyuan Zhang, Xiaolin Gui, Dewang Ren, and Defu Li. Energy–latency tradeoff for computation offloading in uav-assisted multiaccess edge computing system. *IEEE Internet Things J.*, 8(8):6709–6719, 2021.

[110] Shuhang Zhang, Hongliang Zhang, Qichen He, Kaigui Bian, and Lingyang Song. Joint trajectory and power optimization for uav relay networks. *IEEE Commun. Lett.*, 22(1):161–164, 2018.

[111] Mingxiong Zhao, Wentao Li, Lingyan Bao, Jia Luo, Zhenli He, and Di Liu. Fairness-aware task scheduling and resource allocation in uav-enabled mobile edge computing networks. *IEEE Trans. Green Commun. Netw.*, pages 1–14, 2021.

[112] Z. Zhao, G. Min, W. Gao, Y. Wu, H. Duan, and Q. Ni. Deploying edge computing nodes for large-scale iot: A diversity aware approach. *IEEE Internet of Things Journal*, 5(5):3606–3614, Oct 2018.

[113] Fuhui Zhou, Yongpeng Wu, Rose Qingyang Hu, and Yi Qian. Computation rate maximization in uav-enabled wireless-powered mobile-edge computing systems. *IEEE J. Sel. Areas Commun.*, 36(9):1927–1941, 2018.

[114] Yi Zhou, Cunhua Pan, Phee Lep Yeoh, Kezhi Wang, Maged Elkashlan, Branka Vucetic, and Yonghui Li. Communication-and-computing latency minimization for uav-enabled virtual reality delivery systems. *IEEE Transactions on Communications*, 69(3):1723–1735, 2021.