UNIVERSITÀ
DEGLI STUDI
DI PALERMO

DEPARTMENT OF ENGINEERING
PhD IN INFORMATION AND COMMUNICATION TECHNOLOGIES
XXXV CYCLE

Fabio Antonino Busacca

AI for Resource Allocation and Resource Allocation for AI: a two-fold paradigm at the network edge

PhD THESIS

PhD Coordinator
Prof.ssa Ilenia Tinnirello

Advisor
Prof. Sergio Palazzo

ACADEMIC YEAR 2021/2022

Abstract

5G-and-beyond and Internet of Things (IoT) technologies are pushing a shift from the classic cloud-centric view of the network to a new edge-centric vision. In such a perspective, the computation, communication and storage resources are moved closer to the user, to the benefit of network responsiveness/latency, and of an improved context-awareness, that is, the ability to tailor the network services to the live user's experience. However, these improvements do not come for free: edge networks are highly constrained, and do not match the resource abundance of their cloud counterparts. In such a perspective, the proper management of the few available resources is of crucial importance to improve the network performance in terms of responsiveness, throughput, and power consumption.

However, networks in the so-called *Age of Big Data* result from the dynamic interactions of massive amounts of heterogeneous devices. As a consequence, traditional model-based Resource Allocation algorithms fail to cope with this dynamic and complex networks, and are being replaced by more flexible AI-based techniques as a result. In such a way, it is possible to design intelligent resource allocation frameworks, able to quickly adapt to the ever-changing dynamics of the network edge, and to best exploit the few available resources.

Hence, Artificial Intelligence (AI), and, more specifically Machine Learning (ML) techniques, can clearly play a fundamental role in boosting and supporting resource allocation techniques at the edge. But can AI/ML benefit from optimal Resource Allocation?

Recently, the evolution towards Distributed and Federated Learning approaches, i.e. where the learning process takes place in parallel at several devices, has brought important advantages in terms of reduction of the computational load of the ML algorithms, in the amount of information transmitted by the network nodes, and in terms of privacy. However, the scarceness of energy, processing, and, possibly, communication resources at the edge, especially in the IoT case, calls for proper resource management frameworks. In such a view, the available resources should be assigned to reduce the learning time, while also keeping an eye on the energy consumption of the network nodes.

According to this perspective, a two-fold paradigm can emerge at the network edge, where AI can boost the performance of Resource Allocation, and, vice versa, optimal Resource Allocation techniques can speed up the learning process of AI algorithms.

Part I of this work of thesis explores the first topic, i.e. the usage of AI to support Resource Allocation at the edge, with a specific focus on two use-cases, namely UAV-assisted cellular networks, and vehicular networks.

Part II deals instead with the topic of Resource Allocation for AI, and, specifically, with the case of the integration between Federated Learning techniques and the LoRa LPWAN protocol. The designed integration framework has been validated on both simulation environments, and, most importantly, on the Colosseum platform, the biggest channel emulator in the world.

List of publications

- F. Busacca, C. Cirino, G. Faraci, C. Grasso, S. Palazzo, and G. Schembra, 2020, June. Multi-layer offloading at the edge for vehicular networks. In *2020 Mediterranean Communication and Computer Networking Conference (MedComNet)*.
- F. Busacca, L. Galluccio, and S. Palazzo, 2020, July. Drone-assisted edge computing: a game-theoretical approach. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*.
- F. Busacca, L. Galluccio, J.S. Mertens, D. Orto, S. Palazzo, and S. Quattropani, 2020, September. An experimental testbed of an Internet of Underwater Things. In *Proceedings of the 14th International Workshop on Wireless Network Testbeds, Experimental evaluation & Characterization*.
- F. Busacca, G. Faraci, C. Grasso, S. Palazzo, and G. Schembra, 2021. Designing a multi-layer edge-computing platform for energy-efficient and delay-aware offloading in vehicular networks. *Elsevier Computer Networks*.
- A.A. Brincat, F. Busacca, L. Galluccio, J.S. Mertens, A. Musumeci, S. Palazzo, and A. Panebianco, 2022. An integrated acoustic/LoRa system for transmission of multimedia sensor data over an Internet of Underwater Things. *Elsevier Computer Communications*.
- F. Busacca, C. Grasso, S. Palazzo, and G. Schembra, 2022. A smart road side unit in a microeolic box to provide edge computing for vehicular applications. *IEEE Transactions on Green Communications and Networking*.
- F. Busacca, S. Mangione, I. Tinnirello, S. Palazzo, and F. Restuccia, 2022. SDR-LoRa: dissecting and implementing LoRa on software-defined radios to advance experimental IoT research. In *Proceedings of*

the 16th ACM Workshop on Wireless Network Testbeds, Experimental evaluation & CHaracterization.

- F. Busacca, L. Galluccio, and S. Palazzo, 2022, November. A Marketplace Model for Drone-assisted Edge Computing in 5G scenarios. Accepted for publication In *Elsevier Computer Networks*.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The last decades have witnessed an unprecedented evolution of wireless and mobile networks. The number of interconnected nodes has grown considerably, and their nature has changed as well. Indeed, more and more people can access mobile networks and terminals all over the world. Due to the recent emergence of new technologies, such as the Internet of Things (IoT) and 5G and beyond paradigms, there has been a shift from a cloud-centric to an edge-centric vision of the network. In such a perspective, the storage, computational and communication services are moved at the edge of the network, and closer to the users. Among the main advantages of this approach are [1]: i) the increased responsiveness and reduced latency of the system. Network resources are moved closer to the final users, resulting in smaller communication delays, as well as in a reduced congestion of the network (traffic is kept at the edge servers, rather than processed at a single central cloud node); ii) Context Awareness, i.e. the ability to supply customized services based on the live user's experience. The edge servers can exploit the physical proximity to capture useful real-time information, enabling a plethora of applications at different levels of the stack, starting from the application level (e.g. virtual reality [2, 3], content caching [4, 5], computation offloading [6, 7]), to the network level (for instance, load balancing [8, 9], and virtual network function provisioning [10, 11]), down to the physical level (e.g., spectrum hole detection [12, 13], radio fingerprinting [14, 15]).

Nevertheless, this advantages do not come for free; indeed, unlike its cloud counterpart, the edge infrastructure presents several constraints at various degrees of level. For instance, an edge server is not nearly as powerful as a cloud server, and may represent a computational bottleneck if the network resources are not properly managed. Another example is represented by mobile networks extended by Unmanned Aerial Vehicles (UAVs): for instance, overloading one specific drone may lead to a quick battery-depletion,

causing a service outage. Finally, this issue is even more critical in IoT networks, where the energy, computational, and communication resources are extremely constrained, and call for a proper management.

In other words, the problem of resource management and allocation at the edge is of utmost importance, as it can drastically improve the network throughput, efficiency, and lifetime.

The traditional approach to solve resource allocation problems is through mathematical modeling and optimization. In such a view, the first step is to model the dynamics and performance of the network, and to accordingly solve a mathematical problem (e.g. a constrained maximization of the network throughput, or a constrained minimization of the network energy consumption). While optimal, this approach proves impractical and/or unfeasible in modern edge networks.

Actually, 5G and beyond technologies, together with the IoT bring a tremendous growth in the number of users, and, consequently, in the volume of generated data and interactions. In other words, modern wireless networks have entered the so-called "Age of Big Data", and are the result of complex interactions among heterogeneous devices. Thus, traditional model-based approaches for network design, deployment and management prove obsolete and inefficient. If old-fashioned wireless networks could be easily described and evaluated through mathematical models, the same does not apply to modern networks, which can only be described with either complex but intractable models or simple but inaccurate models.

If the model-based approach is doomed to fail, what could instead cope with the increased complexity and variability of wireless networks? The answer is Artificial Intelligence (AI). AI and its implementation through Machine Learning (ML) have attracted great interest in both academia and industry, as they promise to deal with problems which are either too complex to be solved with the aforementioned model-based algorithms, or cannot be modeled at all. ML systems, such as deep neural networks (DNNs), are capable of learning how to perform a task from large sets of examples, i.e., data coming from either real-world scenarios or simulations.

Neural networks do not require the development of an explicit model, and can exploit the large volume of data generated in modern networks to their own advantage, i.e. to learn. Most notably, ML systems are also able to adapt to the circumstances, and can optimally react to the dynamics of modern edge networks. Given such remarkable features, intelligent systems prove to be i) easier to deploy, and ii) more flexible than model-based optimization algorithms. In short, intelligent systems are the answer to the renewed complexity and variability of the edge wireless networks.

With all of this in mind, it is clear how the flexibility and adaptability of

AI represent the most effective solution to the problem of resource allocation at the network edge. The first part of this work of thesis, i.e. Part I, focuses indeed on two specific edge scenarios where AI can be employed to solve the problem of resource allocation.

Chapter 3 deals with the already mentioned scenario of UAV-assisted cellular networks, where UAVs behave as a natural extension of the fixed mobile infrastructure. In such a case, the UAVs can either act as a bridge between rural/remote areas and the mobile network, or support the existing cellular infrastructure in case of flash-crowds and/or unexpected traffic growth. In this specific case, the UAVs act as Virtual Network Functions (VNFs) providers and compete to maximize their own utility. The UAVs can establish a competition by buying a certain amount of bandwidth, i.e. the resource to be assigned, from the network Telco Operator. Moreover, the drones can take autonomous decisions by means of a game-theoretical framework supported by exponential learning mechanism.

Chapter 4 is instead focused on the problem of AI-based job offloading in vehicular networks. The scenario is the following: two battery-powered Road Side Units (RSUs) receive the jobs generated by the vehicles moving along a roadway. The RSUs can activate a variable number of computing elements, either to speed up the job processing, or to save energy. In order to balance the network load, each RSU can offload its respective jobs to the other. Both the decisions over the number of jobs to offload, and the number of computing elements to turn on are taken according to a model-based Reinforcement Learning mechanism. The latter can operate according two different policies, either to privilege energy-saving and increase the system autonomy, at the expense of the job processing latency, and vice versa.

If AI can help improve and optimize the process of resource allocation at the edge, what about the opposite? Can resource allocation frameworks help improve and speed-up the process of learning in the networks? The answer is yes.

Recently, vanilla AI/ML techniques have evolved towards the so-called Distributed Learning approach. Intuitively, Distributed Learning takes place in parallel at several devices, as opposed to the traditional setting where learning occurs at a single powerful node. In such a view, the individual nodes produce either part of a global ML model or a local version. The result of the computation is then sent to a central server in place of the actual training data. The distribution of the learning process brings three main advantages: i) the computation load is split among several devices, ii) the amount of information transmitted by the devices is significantly reduced, and iii) sensitive data is kept local, for the sake of privacy and security. In such a view, Federated Learning (FL) is gaining an important momentum

in both research and academia. FL is an evolution of regular Distributed Learning, designed to deal with massively distributed, non-IID and unbalanced data sets [16]. In other words, in a typical FL setting, training data samples are spread among a huge number of nodes, and the average size of local data sets is significantly smaller than the total number of devices in the system; also, data from different nodes are characterized by different distributions, and the number of training samples per-node can vary from device to device.

On one hand, FL and the IoT represent a match made in heaven. Indeed, IoT networks are made up of a massive amount of sensing devices scattered over a large geographic area. Due to these features, the sensor nodes in the network collect location-specific data sets, where local data follow heterogeneous, i.e. non-i.i.d., probability distributions. For instance, Figure 1.1 depicts a possible use case, where Federated Learning supports an application of Spectrum Sensing in the IoT. However, on the other hand, the two paradigms present opposite and contrasting requirements. In fact, some critical aspects to take into account are [1]:

- **Energy Resources**: Energy saving is an important aspect in mobile networks, and Federated frameworks should optimize the battery life of devices included in the learning process. Energy saving is even more important to IoT nodes, whose low capacity batteries are not expected to last for several months/years.

- **Processing Power**: Federated Learning and, more in general, Machine Learning require certain minimum computational capabilities to carry the learning process, e.g. CPUs or, even better, GPUs. IoT devices are, instead, equipped with low-end on-chip processors. Thus, even more advanced nodes equipped with GPUs may struggle to train a local model, especially in the case of complex neural networks.

- **Data Rate**: Modern IoT networks resort to Low Power Wide Area Networks (LPWANs) to communicate, such as LoRa [17], SigFox[18] technologies, and many others. Such protocols allow the transmission of information at long distances and with low energy consumption, at the expense of achievable data rates. Clearly, LPWAN protocols cannot be easily integrated with Federated Learning, as the latter requires the transmission of millions of parameters in a short amount of time.

Hence, the usage of Federated Learning techniques in IoT networks calls for proper allocation frameworks. In such a way, the few available resources

[1] Note how this issues can be also present in regular 5G and beyond networks, but are greatly exacerbated in the IoT

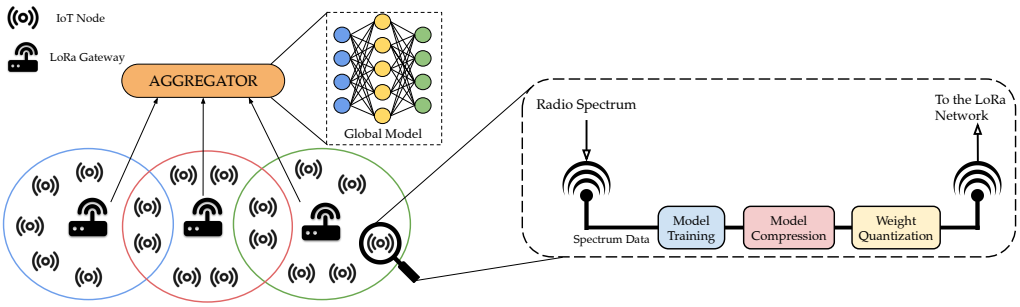


Figure 1.1: Federated Learning in the IoT: a possible use case

can be exploited in the best way possible, reducing both the learning time, and the system energy consumption (and, accordingly, the network lifetime).

Therefore, if Part I of this thesis explores how AI can support procedures of Resource Allocation at the edge, Part II overturns this idea, and proposes an example of an IoT Resource Allocation framework to support and speed-up Federated Learning schemes in the IoT. More in detail, Part II describes our Federated Learning framework for the popular Long Range (LoRa) communication protocol, including the design, implementation, and evaluation of the framework in both the Colosseum channel emulator platform [19], and in real scenarios.

Chapter 5 describes the implementation of LoRa Physical Layer over Software Defined Radio (SDR) from scratch. Although not directly related to the topic of Federated Learning, this represents a step of utmost importance towards the testing and evaluation of our FL framework over large-scale scenarios implemented on Colosseum. The latter actually is a general purpose channel emulator, and, as such, relies on the SDR technology to support a vast variety of communication protocols.

Chapter 6 focuses on the actual Federated Learning framework for the IoT, named `FedLoRa`. The chapter illustrates the theoretical intuitions behind the framework, a greedy solution for the resource allocation problem, as well as the comparison of our resource allocation technique with two baselines. The AI reference model for the numerical evaluation is a DNN for radio device fingerprinting, that is, a typical application of Spectrum Sensing.

Chapter 2

Related Work

The intertwining between AI and Resource Allocation promises to boost the performance of Edge Networks, yet it represents a very complex and scenario-specific topic. Indeed, the problem formulation and solution depend on a variety of factors, namely the specific use-case, the AI/ML technique taken into account, and the type of available resources. This thesis, in particular, focuses on three specific application scenarios, namely:

- Resource Allocation in UAV-based cellular networks
- Resource Allocation and Job Offloading in vehicular networks
- Resource Allocation and Federated Learning in the IoT

For the sake of clarity, this chapter summarizes the current state-of-the-art for each one of these scenarios.

2.1 UAV and Cellular Networks Integration

Integration between UAVs and wireless networks has been explored in several technical papers [20, 21, 22, 23, 24, 25, 26]. More in detail, the usage of drones as mobile cellular base stations, is thoroughly addressed in literature. Authors in [27, 28] study the employment of UAVs to improve the capacity of the existing mobile network, while authors in [29, 30] optimize the usage of drones as relays or aerial base stations in absence of a fixed infrastructure. A key aspect of the study proposed in this thesis is the usage of Game Theory as a tool to study the dynamics in UAV-based cellular networks and optimize their performance. Several other works in literature employ Game Theory in the study of these scenarios, albeit focused on different aspects. For instance, the authors in [31] focus on the problem of task offloading: Game

Theory supports the drones in taking an offloading decision, and allows the optimization of the trade-off between energy consumption, cost, and latency. In [32], the authors follow two different approaches based on Non-Cooperative Games and Evolutionary Algorithms to maximize the network coverage of flocks of drones. The work described in [33] illustrates a disaster scenario, where drones act as LTE base stations and try to fill the coverage gaps in the existing terrestrial WiFi infrastructure. Here, Game Theory is instrumental in selecting the optimal communication delay and minimize the interference, so to achieve a satisfactory throughput in both the WiFi and LTE networks.

For what concerns this work of thesis, we present a game-theoretic model of an edge computing scenario relaying on the exploitation of drones. Game Theory represents a fundamental tool to describe the dynamics of the scenario and the interactions between users and drone. Hence, not only Game Theory enables a fully distributed management of the available communication resources, but is also instrumental to prove the full convergence of the scenario. Artificial intelligence comes into play in real deployments, where it enables drones to autonomously allocate the available communication resources, and, in general, to eventually converge to the demonstrated game equilibrium.

Several novel aspects are considered here. First, we specifically target our model to 2 important 5G and beyond use cases, namely eMBB and URLLC, each characterized by its own relevant requirements and features. We also consider a hybrid trade off use case in between the requirements of eMBB and URLLC. We highlight the impact of delay requirements and bandwidth concerns and then we incorporate in our model also the effect of interference among users which has been shown to represent a critical feature in next generation 5G networks [34]. Finally, we also evaluate how interference impacts on the game equilibrium.

2.2 Resource allocation frameworks and job offloading in vehicular networks

In this section, we review the main studies in the current literature that regard the application field of job offloading and load balancing in vehicular networks. First, we present some papers regarding the integration of computation offloading techniques, computing paradigms and vehicular networks in the context of ITS. Then we move to discuss some papers regarding the application of offload to reduce processing delays in vehicular networks and, specifically, dealing with aspects related to energy saving. Finally, we review

some papers that optimize energy saving in MEC servers, and that introduce cooperation among MEC servers for load balancing purposes.

Several surveys deal with computation offloading applied to vehicular networks. The study in [35] provides a taxonomy based on the main aspects of this topic, also presenting the main challenges and directions to guide future research in this active area. Authors in [36] investigate application of computation offloading in vehicular environments, proposing a framework to support it, while some models for task scheduling in vehicular networks have been proposed in [37, 38, 39, 40]. Other papers [41, 42, 43] consider offload techniques focusing on maximizing resource utilization.

One of the main aspects of offload that has been taken into consideration in the past literature, overlapping a topic of this paper, is reducing computation tasks' processing delay [44]. To this purpose, large amount of research work has focused on the integration of vehicular networks and MEC [45, 46, 47, 48].

In [45], Islam et al. address the unbalanced load distribution issue and propose a secure intelligent vehicle federation model for balancing loads based on a blockchain-based decentralized architecture to enhance transparency in resource management. In [46], Tang et al. focus on integrating edge with autonomous vehicles, and design a scheduling pipeline for the transfer and fusion of different types of data between vehicles and edge nodes to achieve cooperative perception of the road environment.

In [47], Hui et al. develop a collaborative edge computing mechanism for sixth-generation (6G) space-air-ground integrated vehicular networks where each 6G infrastructure component (e.g., satellite, drone, and base station) cooperates with parked vehicles to provide services for mobile vehicles in order to improve the efficiency of the edge computing services. In [48], Buda et al. define an architecture with an efficient collaboration between vehicles, aimed at managing computation task processing with low latency. To this purpose, they propose a two-stage collaborative edge computing scheme for vehicular IoT towards a greener ITS, where vehicles are organized in clusters to increase efficiency of the networking and computing architecture. The same idea of clustering vehicles is used in [49], where a task-offloading scheme is introduced for cellular vehicle to everything (C-V2X) with the objective of improving offloading reliability and latency. Vehicles in need of assistance can transfer their task to other vehicles for processing through the vehicle-to-vehicle (V2V) link, or transfer their task to the mobile edge computing (MEC) server via the vehicle-to-network (V2N) link, with the goal of minimizing communication overhead and providing reliable routes for efficient task offloading. Matching theory is exploited for the task assignments. Nevertheless, although all the above papers have goals that are very similar to

this paper, use of iterative approaches require many interactions and causes slow convergence, which does not allow the framework to quickly adapt to dynamicity of the environment where it is applied. On the contrary, the approach presented in this thesis provides the two platform servers (i.e. VPS and MPS) with a mathematical tool to calculate the optimum analytically, and so adaptation to system variations is immediate. In addition, not having a global model of the environment, performance evaluation and design insights in the above papers are derived via simulation.

In addition, the above papers, although focusing on the same ITS scenario of this paper and having the same target of integrating vehicles with MEC servers, do not consider energy consumption aspect that, on the contrary, constitutes one of the main elements of this work of thesis. As regards this important aspect, the survey in [50] provides a classification of solutions of computation offloading for vehicular networks according to the techniques of partitioning, scheduling and data retrieval, while some other works concentrated their focus on achieving the maximum benefit of service providers via designing optimal resource management schemes [51].

In [52], using the game theory, the authors propose a task-offloading scheme for real-time and compute-intensive applications that optimizes energy consumption at mobile devices without violating such applications' strict latency requirements. In [53], a multi-agent RL algorithm is proposed with the task of optimizing the tradeoff among energy consumption, delay and task failure rate. The study in [54] has the target of jointly optimizing task offloading and resource allocation to minimize the energy consumption subject to the latency requirement. To address this issue, the general problem is decomposed into three sub-problems named offloading ratio selection, transmission power optimization, and subcarrier and computing resource allocation. An iterative algorithm is introduced to deal with them in a sequence, deriving the closed-form solution of offloading ratios, employing the equivalent parametric convex programming to obtain the optimal power allocation policy, and dealing with subcarrier and computing resource allocation by the primal-dual method. The work in [55] deals with the problem of user association and task offloading to MEC servers with spatial and temporal variations of computing power, channel quality and connection capacity between different MEC servers. Nevertheless, let us notice that most of the above works focus on optimizing task offloading decision problems to minimize energy consumption of mobile terminals like smartphones or IoT devices. This is also the case of [56], where minimization of processing time is included, and of [57, 58], where a battery with limited capacity is considered. Also in [55] mobile devices are smartphones, and the focus is to design an architecture to charge mobile devices from MEC servers with power transfer techniques.

Likewise, in [9], the authors investigate the average transmission-power minimization problem for Vehicular Edge Computing (VEC) under the tasks of QoS requirements.

Unfortunately, this does not apply to our case of vehicular networks because here mobile terminals are on vehicles that have a big battery that is also continuously recharged by the vehicle recharge system. Therefore, energy consumption of devices installed on vehicles does not constitute a main concern, whilst our focus for energy saving is on the MEC servers residing in the RSUs. Little work has been carried out focusing on this aspect. In [59] Mao et al. proposed a framework that jointly optimizes radio resource allocation and computation resource management, deriving the optimal CPU cycle frequency of mobile devices and MEC servers in closed form using an algorithm based on Lyapunov optimization and the Gauss–Seidel method. However, its focus is limited to calculate the optimal transmit power and bandwidth allocation only. More recently, in [60], Tang et al. proposed a cache enabled task offloading in the vehicular edge computing in hope to jointly optimize the response delay and the energy consumption at the RSU. More specifically, both the communication and computation models are refined and a greedy algorithm is then put forward to solve the optimization problem. However, in this work of thesis, we integrate the models defined at both the vehicular and the MEC domains with an accurate model of the vehicular traffic, which we define as a Markov modulated model to capture both first- and second-order statistics. Moreover, we define the queueing systems in both the domains through Markov chains allowing us to provide the platform servers (i.e. VPS and MPS) with an overall model supporting their decision process analytically.

Another aspect that cannot be neglected in applying edge computing for offloading, and that plays an important role in this thesis, is the computing limitation of MEC servers that may compromise the quality of offloading service. To this purpose, in the field of fog computing, there is a lot of research work on load balancing. However, only few works have considered cooperation among MEC servers in vehicular networks. To address this problem, the authors of [61] propose a hierarchical cloud-based offloading framework for vehicular networks, where a near backup computing server is introduced to make up for the deficit computing resources of MEC servers. The optimal multilevel offloading scheme, which maximizes the utilities of both the vehicles and the computing servers, was solved as a Stackelberg game. Another paper, where some task offloading schemes in vehicular edge computing networks are optimized by considering cooperative MEC servers is [62]. In that paper, computing resource allocation and the maximum latency of tasks are considered as constraints, but no Key Performance Indicators (KPIs) are

calculated to evaluate system reliability, and utility is considered only from the service provider perspective, and not from the vehicle viewpoint. Moreover, that paper sets the problem as a mixed integer non-linear programming problem (MINLP), which results NP-hard, and therefore it was necessary to resort to some heuristics to achieve a sub-optimal solution.

Utilities of both vehicles and providers of computing servers simultaneously were considered by few papers. The authors in [63] propose a contract-based offloading and computing resource allocation scheme, which maximize the benefit of the MEC service providers while enhancing the utilities of the vehicular terminals. In [44, 51], load balancing of the computation resources at the edge servers is approached by means of the game theory, with the same objective of minimizing the processing delay of all the computation tasks. The paper in [64] formulates a problem for load balance in vehicular networks, with the objective of minimizing the maximum load under some constraints in terms of transmit power, storage capacity, per-task completion time and energy consumption. A deep RL algorithm is used to manage network dynamics, while a coalition game based on deep reinforcement learning is used to maximize the total payoff for the selfishness of vehicles. In [65], Xu et al. approach optimization of task offloading in software-defined access networks formulating a mixed integer non-linear program problem to minimize the overall cost, and solve it by means of an algorithm based on convex optimization and matching theory to minimize the overhead. Moreover, they adopt alternating direction method of multipliers algorithm to optimize a problem of load balancing of edge servers. In [66], the task computation load dynamics are captured in cloud-based autonomous vehicular networks by means of a Markov Decision Process (MDP), with the objective of minimizing the expectation of a long-term total cost for imbalanced base-station computation load and task offloading decision switching with offloading latency constraints. Diverse options for computing offloading in scenarios of multiple edge servers are studied in [67], where an Ant Colony Multi-edge Load-balancing Offloading (ACMLO) strategy is proposed inspired by an ant colony algorithm. The problem is modeled as a general multi-constraint optimization problem with the goal of minimizing the weighted sum of energy consumption and delay, and is solved iteratively. In each iteration, the ant colony algorithm is used to make decisions on the division of subtasks and allocation of channel bandwidth to achieve load balancing.

Nevertheless, to the best of our knowledge, this work of the thesis is the first to proposes a RSU system as a whole, equipped with a portable microeolic power generator, named VMEC-in-a-Box, that is able to work at both vehicular and MEC domains aiming at supporting decisions maximizing utilities at both levels. Unlike most of the past literature, a model-based RL

approach is used in order to achieve high reactivity. Moreover, we introduce cooperation between VMEC Stations in order to maximize performance and decrease outage probability due to lack of battery charge, aspect that was never considered so far.

2.3 LPWANs and Federated Learning

This section reviews existing work about the integration of low-power wide-area networks (LPWANs) protocols, and of LoRa in particular, with Federated Learning techniques. A huge part of the study of this integration required the usage of Colosseum, the biggest channel emulator in the world [19]. However, since Colosseum relies on the SDR technology, a full LoRa stack implementation was required to set up the experimentation over the platform. Therefore, as part of this work of thesis, we developed the first full-fledged LoRa implementation for SDR. For this reason, this section is split in two parts: the first one explores existing LoRa implementations, while the second focuses on the topic of Federated Learning within the IoT.

2.3.1 LoRa and SDR

LPWAN technologies, and, specifically, LoRa, represent sources of great interest from both academia and industry, as they allow long range, yet low-energy communications over the wireless spectrum [68]. LoRa enables a wide variety of sensing and monitoring applications, including Smart Health, Smart Metering, Smart Agriculture and Smart Home [69], but also proves very effective for other use cases, such as image transmission [70], UAV communications [71], and localization [72]. Given the huge potentialities of LoRa, several works focus on the design of a SDR implementation of the protocol. From a theoretical point of view, Bernier et al. [73] offered a complete study of the preamble and start-of-frame synchronization procedure of LoRa, and also focused on the implementation of low-complexity frame synchronization algorithms. The authors provided some performance insights of such algorithms in terms of phase estimation error and synchronization failure probability. However, the implementation of the other components of the LoRa transceiver chain was neglected. Knight and Seeber attempt to implement a full LoRa physical layer (PHY) stack for SDRs in [74]. However, this implementation lacks some functionalities, e.g. the possibility to tune the Spreading Factor (SF) – the only allowed value is 8 – and the Coding Rate (CR), and does not properly implement the whitening functionalities.

Marquet et al. [75] provided a thorough description of the LoRa modulation and demodulation architecture. The implementation is exploited to offer some performance insight on the LoRa technology, such as an evaluation of the Bit Error Rate (BER) as a function of SF and CR. However, the implementation in [75] does not include time and frequency shift tracking for chirp spread spectrum (CSS) modulation, and is therefore unable to decode LoRa signals. Robyns et al. [76] provided an implementation called `gr-lora`, where the authors have reverse-engineered the functionalities of a LoRa receiver. This implementation is therefore successful at decoding LoRa signals generated by commercial devices. However, the transmitter has not been included in the implementation.

Tapparel et al. [77] provide an implementation that includes a Carrier Frequency Offset (CFO) estimation functionality, and is therefore able to communicate with LoRa commercial devices. Moreover, the authors validate their implementation through experiments on USRP SDR hardware. However, such experiments are run on dedicated cables connecting the transmitter to the receiver. Hence, the provided BER values do not include any possible performance degradation resulting from external interference. Finally, the performance analysis results from a fixed configuration, with SF = 7, a bandwidth of 250 kHz, and a payload of 64 bytes.

2.3.2 Federated Learning in the IoT

Another important research topic in the context of LoRa networks is the design of optimal resource allocation and optimization schemes. Authors in [78] model the energy consumption of a single LoRa node, and accordingly study the optimization of transmission parameters (bandwidth, SF, and transmission power) to minimize the node power consumption. However, the authors do not consider the potential interference from other LoRa devices, and focus on a single node configuration. Su et Al. [79] study the problem of energy minimization through optimal resource allocation in LoRa networks. The presented approach is similar to the one adopted in this work of thesis. Nevertheless, the authors do not allow nodes on the same LoRa channel to share the same SF, as instead is considered in our problem formulation. Similarly, authors in [80] deal with the problem of resource allocation in IoT networks, and focus on LoRa as an illustrative example. The proposed allocation scheme first groups users on several LoRa frequencies; then, users on the same frequency are assigned optimal SF and transmission power values, thus maximizing the network throughput. However, the authors simply ignore the effect of interference SFs in the process of resource allocation.

Federated Learning is a very popular machine learning techniques, as

it represents the first step towards ubiquitous and pervasive AI systems. Convergence in Federated Learning systems relies on accurate and fast data delivery from the devices to the central aggregator and vice versa. Given the scarceness of communication and energy resources (especially in IoT scenarios), optimal resource allocation in Federated Learning has been investigated in several works. Authors of [81] employ SignSGD, an optimized Federated Learning approach where the networks nodes only transmit 1-bit signs of the local gradients to the aggregator. Moreover, the authors derive both a local computation and a transmission model, to either improve the energy consumption or minimize the devices' outage probability. However, the envisioned system is based on OFDM-based channel access techniques, which are often unsuitable for IoT devices. Although Narrowband IoT (NB-IoT) [68] represents a notable exception, LoRa technology proves less power-hungry and more cost-efficient, and, last but not least, is license-free. [82]. Similarly, authors in [83] propose a resource allocation algorithm for IoT FL to improve energy consumption and transmission time of the devices. In spite of a solid allocation algorithm and convergence analysis, such a work is based on unrealistic assumptions for typical IoT communication protocols (e.g. the usage of maximum ratio combining techniques, or the availability of massive amounts of bandwidth). Moreover, conversely to this work of thesis, validation of the FL framework is based on computer vision datasets and models. Likewise, authors in [84] model the problem of resource allocation in FL IoT networks. Here, resource allocation is modeled as a non-linear programming model, and heuristically solved through an Alternative Direction algorithm. However, the system modeling is only valid for cellular-based IoT networks, which, as already specified, seems a quite unrealistic assumption.

To the best of our knowledge, the work described in Chapter 6 is the first to adopt LoRa as a supporting communication protocol for FL in IoT networks. Moreover, we are the first to propose an optimal resource framework for such a scenario, and to validate such a framework on a realistic testbed based on Colosseum wireless emulator.

Part I

AI for Resource Allocation

Introduction

The first part of this work of thesis discusses how Artificial Intelligence (AI) can be employed to improve the process of Resource Allocation in edge networks. More in detail, Chapter 3 deals with the use case of UAV-based cellular networks. The behavior of UAVs and mobile users is modeled by means of a Game Theoretical framework. The game provably converges to a unique equilibrium, where UAVs can take autonomous decisions on the Resource Allocation policy (i.e. the allocation of bandwidth to the users) with the support of Exponential Reinforcement Learning mechanisms.

The second use case is job offloading in vehicular networks. In such a perspective, the so-called Road Side Units (RSUs) can autonomosuly decide either to further offload the jobs generated by the vehicles or not, and also allocate a variable number of computing elements to the job processing procedure. The decisions can be taken thanks to a model-based Reinforcement Learning scheme, and also according to several perfomance policies, such as energy saving, delay minimization, and so on.

Chapter 3

AI-powered Resource Allocation in UAV-based cellular networks through Game Theory

3.1 Introduction

Nowadays, 5G networks represent a reality. As of June 2021, 188 operators in 86 countries have already activated at least one 5G site in their commercial networks[85]. Also, in the 2022 Mobile Economy Report the GSMA announced that it is expected the number of 5G global connections to reach one billion in 2022 and to double to two billion by 2025. By that time 5G connections will thus account for a quarter of all mobile connections [86]. Nevertheless, the 5G networks are still far from reaching their full potential. Indeed, according to its technical specifications, 5G has to support different typologies of users with heterogeneous and, often, contrasting requirements. Indeed 5G-specific use cases include Ultra Reliable Low Latency Communications (URLLC) and Enhanced Mobile Broadband (eMBB). The former gathers all the use cases where low-latency and reliability are of crucial importance (e.g., autonomous driving or remote surgery), while the latter comprises all about providing extremely high data rates provided to massive amounts of users.

Edge Computing proves to be a crucial enabler for 5G, especially for URLLC and eMBB use cases. Bringing the computing facilities closer to the network edge, can indeed improve the management of massive amounts of users and reduce the end-to-end communication latency. Still, the deployment of computation facilities at the edge may be particularly expensive, making resource over-provisioning only seldom possible. At times, capillary

deployment of computation facilities is not feasible at all, especially in the case of rural and/or remote areas. For these reasons, the Edge Computing paradigm could as it is, fail in providing ubiquitous connectivity and services, either because the limited edge facilities are not powerful enough to withstand sudden or unpredictable increases in the local traffic (e.g. in case of massively crowded events, natural disasters, and so on), or because the edge facilities simply do not exist at all.

Unmanned Aerial Vehicles (UAVs) like drones represent a valid answer to this problem, as cellular drone-cells are able to either provide additional support to the already existing edge infrastructure, or to form a new standalone edge cell from scratch.

In the recent past, the use of drones as a critical component of the communication network emerged [87, 88, 89, 90]. In particular, UAVs can help to support reliable connectivity in hostile areas with limited or unavailable communication infrastructure, where UAVs provide aided communication networks by integrating them into the peripheral network, e.g. as flying base stations (BSs) [91], relay nodes [92], or terminal devices [93]. UAVs thus act primarily as a means for enabling flexible communication services in remote areas when interconnections with existing networks are lost.

This chapter investigates a scenario where drones are identified as potential sellers of services to network users in the form of Virtual Network Functions (VNFs) and as clients buying bandwidth from the Telco Operator (TO). To this end, we propose a marketplace scenario to model interactions among different players, that is, drones performing as VNF servers trying to maximize their monetary revenue while also coping with energy features, and users which requiring servicse from a specific server (i.e., a drone), based on the requested fee or performance needs. More specifically, we target three realistic 5G use cases, namely URLLC, eMBB, and an intermediate use case, where multiple contrasting requirements arise.

The main aim of this work, therefore, is enhancing previous models and customizing them to specific system settings to describe network dynamics and players' behavior by means of a game-theoretical model. This theoretical framework, together with exponential reinforcement learning procedures, brings a certain degree of intelligence to the network, allowing both users and servers to take autonomous and decentralized decisions, as foreseen in next generation networks. More specifically, in the rest of this chapter, we propose a system model for the eMBB, the URLLC, and a hybrid trade off use case in 5G systems, including a customized definition of proper utility functions to describe behavior and objectives of the system players. The introduced two-stage game model, based on Evolutionary Game Theory and Stackelberg Games, analyzes the interactions among users, and between users and

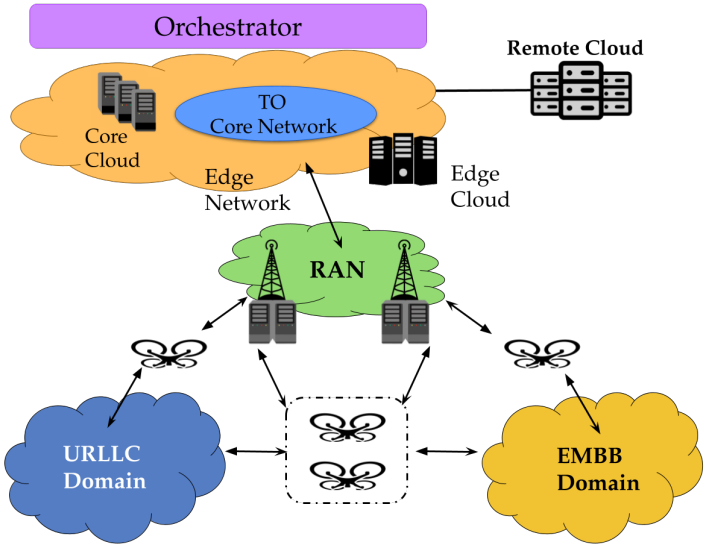


Figure 3.1: System architecture.

servers, while accounting in the different use cases for multiple contrasting requirements in terms of bandwidth, delay, price, and effect of interference caused by multiple users served by the same drone. The game is shown to converge to a unique point of equilibrium which can be explicitly estimated. A set of numerical estimations is also provided for the different use cases where, based on the diverse requirements, new network dynamics in terms of requested bandwidth and user distribution emerge. Finally, the numerical estimation also considers the impact of multiple available drones as well as the effect of interference experienced into the network as a consequence of multiple users served by the same drone.

3.2 System Model

In this section we present the system model considered for this study. More specifically, in Figure 3.1 the envisaged architecture is illustrated. It consists of a typical 5G and beyond network, made up of three main actors: a *Telco Operator* (TO), the *users*, and the *servers*. The TO provides connectivity and services to the network customers through the support of drones behaving as Mobile Edge Hosts (MEHs), which thus act as servers. The users can either be physical customers or devices, and request network and/or application functions and services with heterogeneous requirements.

In such a view, each entity in the network, on the one hand, has to incur

into various costs, e.g. energy costs and/or monetary costs, in order to take part into the system and accomplish its own tasks. On the other hand, all entities in the network also gain some benefits by participating to the system economy, e.g. the TO can sell services and bandwidth, the drones can earn money by providing VNFs, and the users can receive services and connectivity.

Therefore, the system equilibrium results from the interaction of the three aforementioned entities (i.e., the TO, the servers and the users), whose main goal is to maximize their own utility while obviously keeping the costs as low as possible. The study of the performance evolution in such a scenario calls for the creation of an accurate and flexible marketplace model, able to account for the large variety of possible use cases of 5G and beyond cellular networks.

For instance, the sketched system architecture in Figure 3.1 depicts two relevant use cases in new generation networks; actually, it includes users from an eMBB domain (e.g., smartphone users, 8K video streaming users,immersive VR and AR users), which request huge amounts of bandwidth, and users from a URLLC domain (e.g., vehicular nodes, or delay-sensitive medical systems, disaster recovery systems), whose main priority is reliability and very low latency.

Let us now describe, in detail, the business model for our system.

As previously stated, servers are drones, possibly owned by third parties, and act as a bridge between the users and the Edge Network. Hence, the servers can receive data and requests from the users, run network/application functions, and send the results back to the users. On the one hand, the servers can sell the services for a fee, and thus increase their monetary revenue. On the other hand, they incur the typical energy costs of drones, due to processing, communication, and hovering. Specifically, the energy consumption is dependent on both the drone maneuvering and flying system, and the carried hardware. The latter also influences the processing delay of a server (e.g., the more are the active CPU cores and the faster is the clock speed, the shorter is the processing delay). In addition, servers also incur into monetary costs, as they need to buy some bandwidth from the TO to provide the aforementioned services. Last but not least, depending on the number of connected users, if a server can increase its revenue, users on their side can experience a proportional reduction in bandwidth if the server is not prone to afford additional costs to improve the bandwidth availability, or can experience an increase of interference because of multiple users coexisting in the same ecosystem served by a specific drone.

Indeed the problem of estimating the impact of interference in 5G networks has been addressed in detail in [34] where the issue of noise and inter-

ference regimes in 5G Millimeter-Wave networks has been considered. The problem of appropriately weighting interference is of paramount importance since it can both provide benefits coming from inter-cellular interference coordination, coordinated beamforming and dynamic orthogonalization but, in case of high thermal noise, it can represent a major drawback. Also, differently from traditional cellular networks, in mmWave systems the relative strength of interference depends on many more factors, such as high directionality. size of the antenna arrays, antenna pattern, and level of local scattering and spatial multipath. Accordingly, in this work we consider the impact of interference in our model since it has shown to represent a key performance indicator in next generation 5G scenarios.

That said, the users can connect to specific servers according to the network and/or application functions they are interested in. Since the same function can be implemented by more than one drone, the users can choose to connect to a specific server according to the amount of offered bandwidth, the requested fee, the offered processing delay, and the expected job loss probability. For this reason, multiple users will connect to the same server. This is especially true for users moving together, or belonging to the same geographical area; for instance, vehicles moving together along the same road will probably connect to the same server unit. We therefore group these sets of users into *User Groups*.

The TO holds an important role as well, as it is responsible of the management of the whole system, thanks to the support of an *Orchestrator*. The latter is in charge of several crucial tasks related to the provisioning of virtual network and/or application functions. Indeed, the Orchestrator: i) shares a list of the available resources provided by the TO with the users ; ii) provides a template for each virtual network and/or application function. A template defines the operations needed to deploy and realize each application, network, storage or computation function and to manage its life cycle; iii) allocates the bandwidth resources to the servers in exchange of a monetary fee; iv) provides the users with a list of all the servers implementing a specific virtual function, as well as information about the provided bandwidth, the pricing applied, and the expected delay and loss probability of those servers.

Let us now discuss the mathematical model of the system.

Let $\mathcal{F}$ be the set of virtual network and/or application functions available to the users. Moreover, let $\mathcal{U}$ be the set of user Groups, where any User Group consists of users requesting provisioning of the same network function(s). Note how a User Group represents a convenience abstract definition, useful for modeling our system, rather than an actual group of users to be maintained and managed by the network.

Given a function $f_p \in \mathcal{F}$, let $p \in \mathcal{U}$ be a User Group made up of N_p user

devices that are interested in the virtual function f_p, and $\mathcal{S}$ be the set of servers providing f_p.

Let d_{ip} be the processing delay experienced by users from the User Group p when they are served by server $i \in \mathcal{S}$. Also let us denote as n_{ip} the number of users of User Group p served by server $i \in \mathcal{S}$.

The entities participating in these interactions are the servers running f_p, the users that request f_p for some of their flows, and the Orchestrator.

Each server incurs costs of heterogeneous nature. First, each server has to manage several user flows, and accordingly allocate storage and computing resources to those flows. Flight control and drone movements also imply a further energetic cost. Finally, VNF provisioning requires the servers to power on their hardware capabilities, and to withstand an additional cost for each new incoming user flow.

With all that in mind, the cost incurred by a server i to manage all the user flows in p can be modeled as:

$$C_{ip}^{(\mathcal{F})} = c_{ip}^{FC} + c_{ip}^{HW} + c_{ip} \cdot n_{ip} \tag{3.1}$$

In the above expression, c_{ip}^{HW} is the cost needed to activate gimbals, sensors, actuators, and communication modules of the drone, while c_{ip}^{FC} is the cost implied by flight control and hovering. Last, c_{ip} represents the incremental cost to allocate the required resources to each new incoming flow which requests function f_p.

Servers have also to lease some bandwidth from the TO to properly provide services and VNFs to the managed user flows. Clearly, the amount of leased bandwidth depends on the price the drones are willing to pay. Let b_{ip} be the bandwidth allocated by the server i to the User Group p, and $p_i^{(\mathcal{B})}$ the bandwidth-unit price charged by the TO network to the server i. Hence, the overall cost of the bandwidth lent to the server is:

$$C_{ip}^{(\mathcal{B})} = p_i^{(\mathcal{B})} \cdot b_{ip} \tag{3.2}$$

On the other hand, the servers earn some profit by selling VNFs to the users. However, the TO can claim part of the profit as a commission fee. Formally speaking, the revenue for the server i associated to the provision of f_p is proportional to i) the number n_{ip} of users that are using it, and ii) the fee price $\hat{p}_{ip}^{(\mathcal{F})}$ applied by the server. Given that the commission fee can be represented as the *commission parameter* $\psi \in [0, 1]$, the actual revenue of the server i related to the provision of function f_p to the User Group p is:

$$R_{ip} = p_{ip}^{(\mathcal{F})} \cdot n_{ip} \tag{3.3}$$

where $p_{ip}^{(\mathcal{F})} = \hat{p}_{ip}^{(\mathcal{F})}(1-\psi)$.

Accordingly, the utility function of the server i is:

$$U_{ip}^{(\mathcal{S})}(\mathbf{b}_p) = \beta_1 R_{ip} - \beta_2 C_{ip}^{(\mathcal{F})} - \beta_3 C_{ip}^{(\mathcal{B})} \tag{3.4}$$

where $\mathbf{b}_p = (b_{1p}, b_{2p}, \ldots, b_{Mp})$ is the *bandwidth vector* that specifies the bandwidth b_{ip} requested to the TO network by each server, and β_1, β_2 and β_3 are weighing constants for revenues and costs, respectively[1]

Users in Group p choose the server according to a variety of factors: the expected processing delay; the price applied by the server for the provision of the VNF f_p; the amount of available bandwidth at the server and the interference experienced upon connecting to a server which is also shared with other users. Note that all the above parameters are updated at each instantiation of the game and kept updated. Observe how, the higher the number of flows using the same server, the lower the bandwidth allocated to each of them. Specifically, the server equally splits the bandwidth among all the served users. In the end, each user receives an amount of fractional bandwidth equal to b_{ip}/n_{ip}.

That said, each user selects the server i that maximizes its utility function, i.e. [94]:

$$U_{ip}^{(\mathcal{U})}(\mathbf{n}_p) = \alpha_1 \ln\left(\frac{b_{ip}}{n_{ip}}\right) - \alpha_2 \hat{p}_{ip}^{(\mathcal{F})} - \alpha_3 d_{ip} - \alpha_4 \ln\left(k_{ip} n_{ip}\right) \tag{3.5}$$

where $\mathbf{n}_p = (n_{1p}, n_{2p}, \ldots, n_{Mp})$ is the *state vector* of the number n_{ip} of flows from the User Group p served by each server in $\mathcal{S}$; α_1, α_2, α_3 and α_4 are constants in the utility function of the user balancing the relevance given to the bandwidth received, the price applied by the server, the processing delay of the server, and the packet loss rate due to interference among users served by the same server, respectively. Also k_{ip} is a constant weighing the impact of the number of users served by the drone on the experienced interference. Note that this parameter can be different at the various servers because it can account for possible hardware features of the server device which make the effect of interference even more critical.

In the following of the chapter, we will refer to α_1, α_2, α_3, α_4, β_1, β_2 and β_3 as the *weighing parameters*.

[1] The coefficients are used to balance the contribution of the different revenue and cost terms in eq. (3.4) by making them comparable in size.

3.3 Game Model

This section introduces the two-staged game developed to model interactions between servers and users in the distributed management framework.

The adoption of a game theoretical model is easily motivated: in the depicted marketplace scenario, decisions taken by servers and users indeed depend not only on their individualistic interests, but also on decisions taken by counterparts.

Going into the details, we chose to model the scenario as a Stackelberg competition game between leaders and followers. Indeed, in real scenarios, servers act and make decisions by *anticipating* users. Therefore, Stackelberg Games represent the most natural choice, as interactions among servers and users can be modeled as a game where servers behave as the *leaders* and users as the *followers*. Another key point to our two-stage model is the ability of users to replicate other users decisions. Such a behavior best fits those scenarios whose entities can improve their own utility by replicating other users' behavior. [95, 96, 97, 98].

The rest of the section is organized as follows: in Section 3.3.1 we exploit Evolutionary Game Theory (**EGT**) and replicator dynamics to model the decision-making process of users of the same User Group $p \in \mathcal{U}$. The outcome is the game $\mathcal{G}_p^{(\mathcal{U})}$, and represents the first stage of our game theoretical model; In Section 3.3.2, we use non-cooperative game theory to define the game $\mathcal{G}_p^{(\mathcal{S})}$, which models competition among servers to serve users in the User Group p and represents the second stage of the model; finally, in Section 3.3.3 we describe the distributed reinforcement learning procedure employed to compute the equilibrium of the game $\mathcal{G}_p^{(\mathcal{S})}$.

As an output of the game, the distribution of users between servers can be determined as well as the consequent amount of bandwidth requested by each server to the TO. These two conditions represent a so-called *equilibrium* for our game.

However, the whole game sways from this equilibrium condition every time some conditions of the system change. Indeed, a variation of either the processing delay, a change in the pricing applied by a server for a given function or in the number of users interested in the function, or also the mobility of some drones and/or a deterioration in the wireless channel link conditions could determine a change in the utility function of some users. As a consequence, users start playing the game again, thus modifying the distribution of the users among the servers. This variation stimulates the servers to re-play the game in order to consequently check whether a modification

in the bandwidth requested to the TO network is needed.[2]

3.3.1 Evolutionary game $\mathcal{G}_p^{(\mathcal{U})}$ among users

Each user acts is a selfish way, as it takes decisions with the aim of maximizing its own utility $U_{ip}^{(\mathcal{U})}$, as defined in (3.5). Still, the higher the number n_{ip} of users in the User Group p supported by the server i, the lower the utility $U_{ip}^{(\mathcal{U})}$ of those users. Accordingly, the decision-making process of each user cannot ignore the decisions taken by the other users of the same group. Moreover, a user could adopt an imitation behaviour to improve the achieved utility. The idea is the following: the behaviour of users achieving the best utility could be imitated by other users in the group. For instance, if the user with the best utility in the group is served by server i, other users could migrate to that server, too. We refer to this occurrence as *imitation behavior*. The imitation behavior can be based on actual implicit or explicit exchange of information between the users. Indeed, in order to select a server (i.e. a drone), the users need some important information, such as the number of users served, the VNF cost, the processing delay and the level of interference at each drone. The drones will periodically broadcast this type of information, and a user can therefore leverage on it to autonomously select the best possible drone and eventually imitate the user in the group with the best utility.

The imitation dynamics among users can be accurately described with the help of the well-known and widely used *replicator dynamics* [99]. Therefore, for each User Group $p \in \mathcal{U}$, we define the evolutionary game $\mathcal{G}_p^{(\mathcal{U})}$ as follows:

- *Population*: the set of the N_p users in the User Group $p \in \mathcal{U}$.
- *Strategy*: the choice of the server $i \in \mathcal{S}$ to whom each user in the population p decides to associate.
- *Utility*: the utility $U_{ip}^{(\mathcal{U})}$ achieved by each user associated to the server $i \in \mathcal{S}$, as defined in (3.5).

The variation in the population p that associates to the available servers can be described by means of the *replicator equation*:

$$\dot{n}_{ip} = n_{ip} \left[U_{ip}^{(\mathcal{U})}(n_{ip}) - \frac{1}{N_p} \sum_{j \in \mathcal{S}} n_{jp} U_{jp}^{(\mathcal{U})}(n_{ip}) \right] \tag{3.6}$$

[2]In our work we focus on a system which is stable and does not fluctuate as a consequence of the appearance of dynamic temporal variations in system parameters. Accordingly, we characterize the behavior of users and drones upon neglecting these possible temporal fluctuations.

where $n_{ip} \in \mathbf{n}_p$ is the number of users in the User Group p which, as a strategy, have chosen to associate to the server i.

The first term in the right-hand side of (3.6), represents the utility of a user that connects to the server i, while the second term expresses the average utility of the population, and depends on the current distribution state vector $\mathbf{n}_p$. Therefore, the growth rate $\dot{n}_{ip}/n_{ip}$ of the number of users in the User Group p associated to the server i is equal to the difference between the benefit when choosing the strategy i, and the average benefit of the whole population[3].

According to a general result from Evolutionary Game Theory, when all users experience the same benefit, i.e., $U_{ip}^{(\mathcal{U})} = U_{jp}^{(\mathcal{U})}$ for all $i, j \in \mathcal{S}$, an equilibrium point for the replicator dynamics is reached. As shown in [100], the replicator equation (3.6) for each User Group p admits a unique solution for any bandwidth vector $\mathbf{b}_p$, and the corresponding equilibrium point can be characterized as:

$$n_{ip}^{*} = \frac{N_p}{\sum_{j \in \mathcal{S}} \left[\phi_{i,j}^{(p)} \left(\frac{b_{jp}}{b_{ip}} \right)^{\alpha_1} \left(\frac{k_{jp}}{k_{ip}} \right)^{\alpha_4} \right]^{\frac{1}{\alpha_1 + \alpha_4}}} \tag{3.7}$$

where $\phi_{i,j}^{(p)} = e^{\left[\alpha_2 \left(\hat{p}_{ip}^{(\mathcal{F})} - \hat{p}_{jp}^{(\mathcal{F})}\right) + \alpha_3 (d_{ip} - d_{jp})\right]}$, $\phi_{i,j}^{(p)} = 1/\phi_{j,i}^{(p)}$, $\phi_{i,i}^{(p)} = 1$, and $b_{ip} \in \mathbf{b}_p$.

An important outcome that stems from the uniqueness of the equilibrium point is the uniqueness of the convergence point, thus avoiding any possible oscillation among two or more equilibrium points.

3.3.2 Stackelberg game $\mathcal{G}_p^{(\mathcal{S})}$ between servers and users

For the sake of notation, let us first define two auxiliary variables, i.e.:

$$\tilde{p}_{ip} = N_p \left(\beta_1 p_{ip}^{(\mathcal{F})} - \beta_2 c_{ip} \right) \tag{3.8}$$

and

$$\pi_i = \beta_3 p_i^{(\mathcal{B})} \tag{3.9}$$

Accordingly, the utility function $U_{ip}^{(\mathcal{S})}$ of the server $i \in \mathcal{S}$ can be written

[3]Note that the values of the user utilities are communicated by the individual users to the servers, which then estimate the average utility of the population and disseminate this metric to allow all users to estimate the difference between the average benefit, i.e. utility, and their own individual benefit when choosing strategy i.

as follows:

$$U_{ip}^{(\mathcal{S})}(\mathbf{b}_p) = \tilde{p}_{ip} \frac{1}{\sum_{j\in\mathcal{S}} \left[\phi_{i,j}^{(p)} \left(\frac{b_{jp}}{b_{ip}}\right)^{\alpha_1} \left(\frac{k_{jp}}{k_{ip}}\right)^{\alpha_4}\right]^{\frac{1}{\alpha_1+\alpha_4}}} - \pi_i b_{ip} - \beta_2(c_{ip}^{FC} + c_{ip}^{HW}) \tag{3.10}$$

For each User Group $p \in \mathcal{U}$, we define the non-cooperative game $\mathcal{G}_p^{(\mathcal{S})}$ among servers as follows:

- *Player set*: the set $\mathcal{S}$ of servers.
- *Strategy*: the amount of bandwidth b_{ip} requested to the TO network by the server i to serve its associated users in the User Group p. For each User Group in $\mathcal{U}$, we assume that such amount of bandwidth is upper-bounded by B_i. Thus, the strategy set is $\mathcal{B} = \prod_{i\in\mathcal{S}} \mathcal{B}_i$, where $\mathcal{B}_i \in [0, B_i]$ and $\prod$ identifies the Cartesian product[4].
- *Utility*: the utility $U_{ip}^{(\mathcal{S})}$ achieved by each server $i \in \mathcal{S}$ as defined in (3.10).

In the following, we analyze the Stackelberg game $\mathcal{G}_p^{(\mathcal{S})}$ and provide some results about its *Stackelberg Equilibrium*, hereafter referred to as **SE**.

Let $\mathbf{b}_p^* \in \mathcal{B}$ be the bandwidth vector at the equilibrium. The strategy profile $(\mathbf{b}_p^*, \mathbf{n}_p^*)$ is a **SE** for the game $\mathcal{G}_p^{(\mathcal{S})}$ if, for all $\mathbf{b}_p \in \mathcal{B}$ and $i \in \mathcal{S}$, we have

$$U_{ip}^{(\mathcal{S})}(\mathbf{b}_p^*, \mathbf{n}_p^*) \geq U_{ip}^{(\mathcal{S})}(\mathbf{b}_p, \mathbf{n}_p^*)$$

where $\mathbf{n}_p^*$ is defined as in (3.7).

Note that $\mathbf{b}_p^* = (b_{ip}^*, \mathbf{b}_{p_{-i}}^*)$, where $\mathbf{b}_{p_{-i}}^*$ is the bandwidth vector of all players except i, i.e., $\mathbf{b}_{p_{-i}}^* = (b_{jp}^*)_{j\in\mathcal{S},\ j\neq i}$ with $b_{jp}^* \in \mathbf{b}_p^*$. The strategy $\mathbf{b}_p^* = (b_{1p}^*, b_{2p}^*, \ldots, b_{Mp}^*)$ is a *Stackelberg strategy* for the game $\mathcal{G}_p^{(\mathcal{S})}$ if, for all $i \in \mathcal{S}$:

$$b_{ip}^* = \arg\max_{b_{ip}\in\mathcal{B}_i} U_{ip}^{(\mathcal{S})}(b_{ip}, \mathbf{b}_{p_{-i}}^*, \mathbf{n}_p^*)$$

In this case, the value $U_{ip}^{(\mathcal{S})}(\mathbf{b}_p^*, \mathbf{n}_p^*)$ is referred to as the *Stackelberg utility* of server i in game $\mathcal{G}_p^{(\mathcal{S})}$.

[4]We do not consider the variable $\hat{p}^{(\mathcal{F})}$ as a strategy for the server. While the bandwidth varies fast in time, pricing policies are instead fixed or slowly-variant thus making the problem of selection of $\hat{p}^{(\mathcal{F})}$ less relevant in the considered problem.

As proven in [100], the game $\mathcal{G}_p^{(\mathcal{S})}$ admits a unique **SE**.

Therefore, the game $\mathcal{G}_p^{(\mathcal{S})}$ between users (i.e., the followers) and servers (i.e., the leaders) results in a unique **SE** $(\mathbf{b}_p^*, \mathbf{n}_p^*)$. Since the servers compete with each other in the Stackelberg game, the strategy profile $\mathbf{b}_p^*$ discussed above also represents a Nash Equilibrium (**NE**) [101] for the competitive game among servers.

3.3.3 Reinforcement Learning Procedure for game $\mathcal{G}_p^{(\mathcal{S})}$

In order to provide a robust mechanism to allow servers to independently reach the equilibrium of the game, we resort to an *exponential reinforcement learning* procedure [102][103].

For each server $i \in \mathcal{S}$ that serves users in Group $p \in \mathcal{U}$, the learning procedure is defined as follows:

$$\begin{cases} z_{ip}(m+1) = z_{ip}(m) + \gamma_m v_{ip}\left(\mathbf{b}_p(m)\right) \\ b_{ip}(m+1) = B_i \frac{e^{z_{ip}(m+1)}}{1+e^{z_{ip}(m+1)}} \end{cases} \tag{3.11}$$

where m represents the iteration index, $\mathbf{b}_p(m)$ is the bandwidth vector at iteration m, γ_m is the step-size of the learning procedure, and $v_{ip}\left(\mathbf{b}_p(m)\right)$ is the marginal utility of each player $i \in \mathcal{S}$ and is defined as $v_{ip}(\mathbf{b}_p) = \frac{\partial U_{ip}^{(\mathcal{S})}(\mathbf{b}_p)}{\partial b_{ip}}$.

Notably, it can be easily shown how the proposed exponential reinforcement learning procedure converges to the equilibrium of the game[5].

More in detail, given any variable step-size in the form $\gamma_m = 1/m^{\xi}$ with $\xi \in (0.5, 1]$, the procedure will always converge to the unique **SE** of the game $\mathcal{G}_p^{(\mathcal{S})}$.

3.4 Numerical analysis

This section discusses the numerical results related to the application of the proposed framework to relevant 5G use cases. The analysis is carried out for three scenarios, namely, the URLLC scenario, the eMBB scenario, and an intermediate use case which corresponds to a tradeoff between the first and the second one.

[5]Note that the use of exponential reinforcement learning procedures is easy to implement, and lightweight. Actually, by running a simulation of duration equal to 20000 time steps, the procedure required approximately 1 second to be executed and reach equilibrium in a medium-end machine equipped with an Intel Core i7-6700HQ 2.60GHz CPU [100].

Parameter \ Use Case	URLLC	eMBB	Trade-off
α_1	0.15	0.5	0.15
α_2	0.0001	0.0001	$1.5 \cdot 10^{-5}$
α_3	0.06	0.005	0.03
α_4	0.4	0.001	0.105

Table 3.1: Values of parameters α for the considered use cases

The number of deployed drones, unless otherwise specified, is equal to two, to better highlight the dynamics of the interactions among the users and the impact of the weighing parameters on the final outcome of the game.

For each use case, we consider a number of $N = 3000$ users, and we set $\beta_1 = 1$, $\beta_2 = 30$, and $\beta_3 = 1$. Note that typically $\beta_1 = \beta_3$ since the first and the third terms in eq. (3.4) are numerically comparable; on the other hand β_2 is usually much larger than the other two terms to allow numerically accounting also for the cost term associated to server management of flows. In our numerical analysis we will also investigate the impact of the choice of β_2.

Concerning parameters $\alpha_1, \alpha_2, \alpha_3, \alpha_4$, they are uniquely tailored to characterize each considered use case. Note how α parameters weigh the relative importance of the different terms in the user utility functions, and must be tuned taking the different scenario requirements into account. As an example, in a URLLC scenario where the processing delay d_{ip} is more relevant than the offered bandwidth b_{ip}/n_{ip}, parameters α_1 and α_3 must be tuned to satisfy the condition $\alpha_3 \cdot d_{ip} > \alpha_1 \cdot \ln\left(\frac{b_{ip}}{n_{ip}}\right)$. According to the different considered use cases, the effect of the α_i terms, where $i \in \{1, \dots 4\}$ is to let the utility function account for the specific requirements associated to the scenario (e.g., for the eMBB the terms associated to the bandwidth concern in eq. (3.5) should be the dominant one, as compared to other terms). In our case the three different use cases have been characterized in such a way that each of the four terms. namely the bandwidth term $BT = \frac{\alpha_1 \ln\left(\frac{b_{ip}}{n_{ip}}\right)}{U_{ip}^{(\mathcal{U})}(\mathbf{n}_p)}$, the price term $PT = \frac{\alpha_2 \hat{p}_{ip}^{(\mathcal{F})}}{U_{ip}^{(\mathcal{U})}(\mathbf{n}_p)}$, the delay term $DT = \frac{\alpha_3 d_{ip}}{U_{ip}^{(\mathcal{U})}(\mathbf{n}_p)}$, and the loss term $LT = \frac{\alpha_4 \ln(k_{ip} n_{ip})}{U_{ip}^{(\mathcal{U})}(\mathbf{n}_p)}$ in eq. (3.5), do exhibit a different percentage contribution to the overall user utility. Details on the settings of these terms in the three use cases are reported in Table 3.1 and discussed in the following subsections. Also, unless otherwise specified, the pricing fee $p_1^{(F)}$ applied by server S_1 is

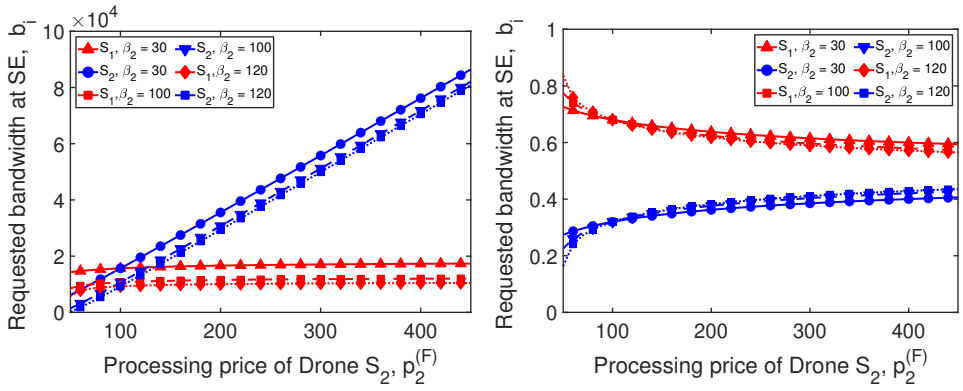


Figure 3.2: URLLC Scenario Variable β: Requested Bandwidth (a) and User Distribution (b) of drone S_2 at the Stackelberg equilibrium as a function of the price $p_2^{(F)}$ for three different values of β_2: 30, 100, 120).

equal to 100 Price Units (PUs), while the pricing applied by server S_2 is variable.

We assume the commission parameter ψ is equal to 0 (in other words, the drones do not owe any commission fee to the TO), and the bandwidth-unit price $p_i^{(B)}$ is the same for all drones in S and equal to $p_i^{(B)}$=1 PUs. The hardware and the flight control costs are equally set for all drones to $c_{ip}^{HW} = 32.5$ W and $c_{ip}^{FC} = 390$ W, respectively, according to the parameter values in [104] [105].

In the following sections, we thoroughly analyze the numerical results for each considered use case.

3.4.1 URLLC scenario

The following section is dedicated to the numerical results for the URLLC use case.

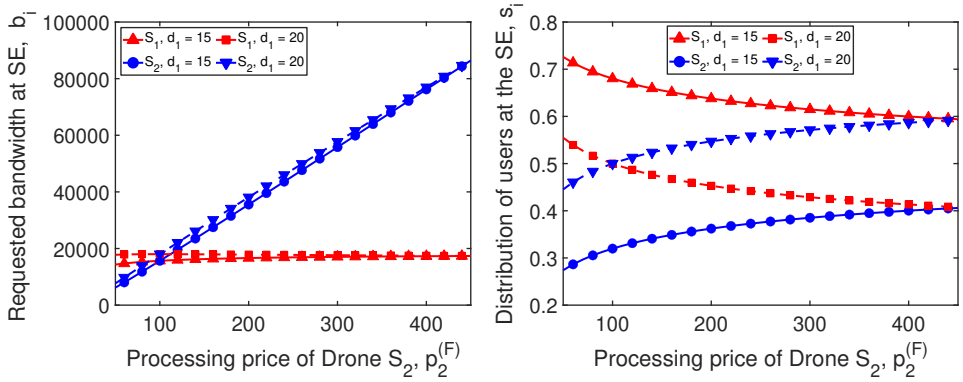
According to the URLLC features, users are interested in a low-latency and reliable service, and not in a large amount of bandwidth. As a consequence of this setting of the parameters, the corresponding weights of the bandwidth term (BT), price term (PT), delay term (DT) and loss term (LT) in eq. (3.5) respectively are BT=0.184, PT=0.006, DT=0.51, and LT=0.3, respectively. Correspondingly, the chosen weighing parameters are $\alpha_1 = 0.15, \alpha_2 = 0.0001, \alpha_3 = 0.06, \alpha_4 = 0.4$.

As a preliminary consideration we want to estimate the impact of the weigh terms β_j in eq. (4).

Figures 3.2a and 3.2b depict the requested bandwidth and the user distribution of server S_2 at the game SE as a function of $p_2^{(F)}$ for different values of the parameter β_2, namely 30, 100, and 120. It is possible to observe how different choices of the parameter (which accounts for the processing cost associated to flow management) do not play any relevant role. Thus in the rest of this chapter, for all use cases, we will assume $\beta_1 = \beta_3 = 1$ and $\beta_2 = 30$ to weigh the terms β_j in eq. (4).

Once decided the values of the β_j terms, Figures 3.3a and 3.3b, show the requested bandwidth and the distribution of users at the SE equilibrium as a function of price $p_2^{(F)}$. The results are depicted for $d_1 = 15$ DUs, $k_1 = 0.0015$ and $d_2 = 20$ DUs, $k_2 = 0.002$ (solid line), and $d_1 = d_2 = 20$ DUs, $k_1 = k_2 = 0.002$ (dashed line). Note that the amount of requested bandwidth results proportional to the applied pricing. Concerning the user distribution, let's first discuss the setting with heterogeneous latency and interference coefficients. In this case, the user distribution is unbalanced towards S_1. Indeed, the latter has a lower delay, and is more robust to interference than S_2. The increase in the bandwidth requested by S_2 produces only a slight variation in the user distribution. For instance, when $p_2^{(F)} < 100$, S_2 attracts about 30% of the users, which increases to about 40% when $p_2^{(F)} = 400$. In other words, S_2 has to request almost four times the bandwidth requested by S_1 to try attracting up to 40% of the users. Let's now discuss the second parameter setting, i.e. when the servers have equal latencies and interference coefficients. The outcome is, accordingly, very different. In this setting, no server holds a specific advantage over the others: the only way to attract more users is to offer more bandwidth. Accordingly, the user distribution is such that whoever offers more bandwidth, attracts more users. Indeed, on the one hand, the users tend to choose the server offering more bandwidth; on the other, they aim at minimizing the interference to improve reliability, and, therefore, at equally distributing themselves among the available servers. Generally speaking, regardless of the chosen set of parameters, the total amount of bandwidth is halved, as will be also evident from the following discussion of the eMBB case. This comes quite expected, as the URLLC scenario specifications do not focus on massive amount of bandwidth.

Figures 3.4a and 3.4b show how the system reacts to changes in the scenario and reaches a new equilibrium. For the URLLC case, we let the latency of server S_2 vary at specific time instants, while the delay of server S_1 is fixed at $d_1 = 20$ DUs. More in detail, d_2 gets the values 30, 25, 20, 15, 10 DUs at times interval 0, 20000, 40000, 60000 and 80000, respectively. Moreover, we set the price and interference coefficient for drones to $p_1^{(F)} = 100, p_2^{(F)} = 150, k_1 = k_2 = 0.002$. Similarly to the static case, as long as S_2



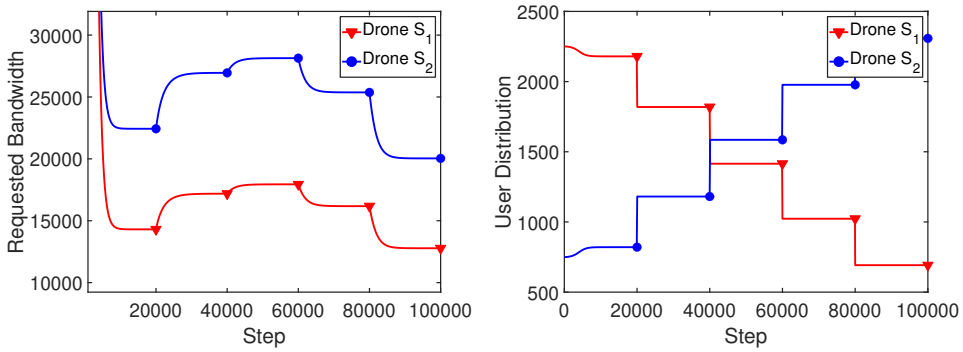
(a) (b)

Figure 3.3: URLLC Scenario: Requested Bandwidth (a) and User Distribution (b) at the Stackelberg equilibrium as a function of the price $p_2^{(F)}$

offers a lower delay than S_1, it is able to attract more users. When, instead, its delay increase, the majority of users connect to S_1 instead. Finally, when both the servers offer the same latency, the users tend to slightly prefer S_2: in fact due to the higher VNF fee, S_2 is indeed also able to offer more bandwidth to the users.

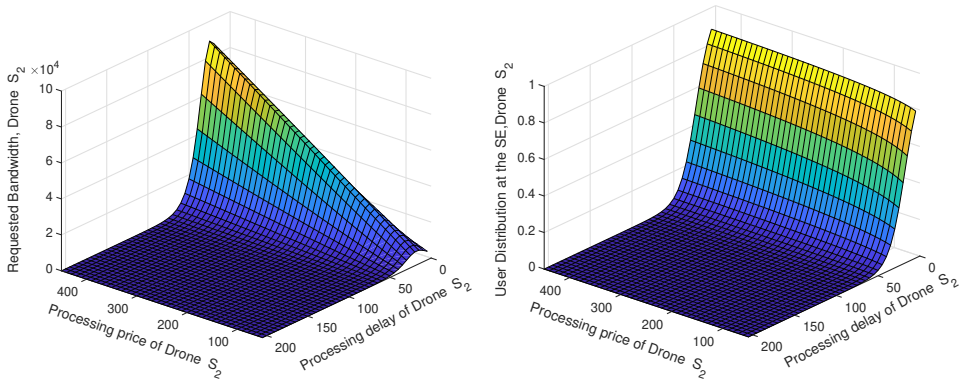
In Figures 3.5a and 3.5b, we report the requested bandwidth and the user distribution of server S_2 at the game SE as a function of both latency d_2 and pricing $p_2^{(F)}$. Once again, latency and pricing of drone S_1 are instead fixed and equal to $d_1 = 20$ DUs and $p_1^{(F)} = 100$ PUs, respectively. The plots show how, the more is the processing delay of S_2, the more is the transition towards a monopolistic scenario, where most of the users are attracted by S_1. The scenario is strongly competitive when $d_1 = d_2 = 20$. Indeed, at that point, the amount of requested bandwidth reaches its maximum, especially for high values of $p_2^{(F)}$. Moreover, the users are almost equally shared by the servers. When, instead, the delay offered by S_2 is particularly high, the amount of attracted users drops drastically, and so does the amount of requested bandwidth, too. When, instead, the latency offered by S_2 is extremely low, the plots show an opposite dynamic, where most, if not all the users, are attracted by S_2.

In order to estimate the effect of interference and loss (associated to term α_4 in eq. (5)) Figure 3.6 shows the evolution of the user distribution as the weight of the loss term (LT) increases. In order to better show the influence of the loss term over the user distribution, we set $k_1 = k_2 = 0.002$, and $p_1^{(F)} = p_2^{(F)} = 100$ PUs. Moreover, we set $d_1 = 15$ DUs, and $d_2 = 20$ DUs. When the weight of the loss term is low, S_1 holds the majority of



(a) (b)

Figure 3.4: URLLC Scenario: Evolution of Requested Bandwidth (a) and User Distribution (b) at the Stackelberg equilibrium as a function of time



(a) (b)

Figure 3.5: URLLC Scenario: Requested Bandwidth (a) and User Distribution (b) of drone S_2 at the Stackelberg equilibrium as a function of the price $p_2^{(F)}$ and of the delay d_2

users (about 70%), as it offers the lowest processing delay. However, as the importance of the loss term increases, more users choose S_2 instead, until an equal distribution is eventually achieved when the importance of the loss term is equal to 100%. The interference term is indeed minimized when the users are equally distributed among the available servers.

3.4.2 eMBB Scenario

In this section we detail the results obtained for the eMBB use case scenario. The corresponding weight of the bandwidth term (BT), price term (PT), delay term (DT) and loss term (LT) in eq. (3.5) are BT=0.96, PT=0.0047,

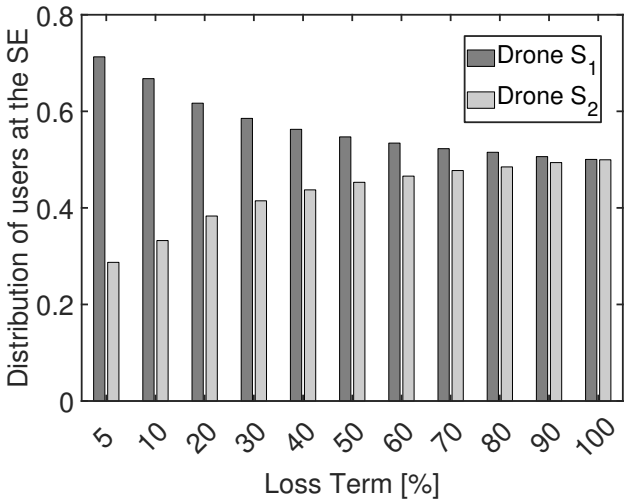
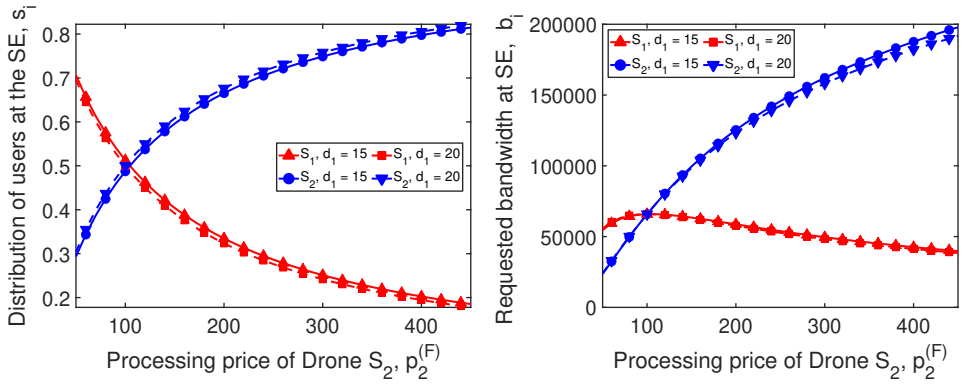


Figure 3.6: URLLC scenario: impact of the weight of the loss term on the User Distribution

DT=0.035, and LT=$3 \cdot 10^{-4}$, respectively. This setting is the result of the fact that users are mostly interested in bandwidth, while latency and interference are only minor concerns. Correspondingly, the weighing parameters set chosen for the eMBB scenario is $\alpha_1 = 0.5, \alpha_2 = 0.0001, \alpha_3 = 0.005, \alpha_4 = 0.001$.

Figures 3.7a and 3.7b show the variation of the requested bandwidth and the distribution of users at the SE equilibrium as the pricing applied by the server (i.e. drone) S_2 increases. Results are depicted for two different settings of latency and interference parameters, namely $d_1 = 15$ DUs, $k_1 = 0.0015$ and $d_2 = 20$ DUs, $k_2 = 0.002$ (solid line), and $d_1 = d_2 = 20$ DUs, $k_1 = k_2 = 0.002$ (dashed line). Note that the behaviour is interesting and counter intuitive: indeed the number of users connected to drone S_2 is proportional to its VNF fee. The reason is that, the more is the applied fee, the more bandwidth drone S_2 can buy. Since the users are interested in massive amounts of bandwidth, S_2 represents the most natural choice. Conversely, in the region where $p_2^{(F)}$ is smaller than $p_1^{(F)}$, S_2 requests less bandwidth than S_1, and thus attracts less users. Finally, an equal pricing leads to an identical distribution of the users among the two servers. Note how the exhibited behaviour is nearly identical for the two settings with identical or different delay features: in fact, users are mostly focused on bandwidth and the diversity in the choice of other parameters makes little or no difference.

Figures 3.8a and 3.8b show how the system reacts to changes in the scenario and reaches a new equilibrium. In the eMBB case, we let the pricing applied by S_2 vary at specific time instants, while the fee applied by S_1 is fixed at $p_1^{(F)} = 100$ PUs. More in detail, $p_2^{(F)}$ gets the values 400, 250, 50, 70, 100 PUs at times interval 0, 20000, 40000, 60000 and 80000, respectively.



(a) (b)

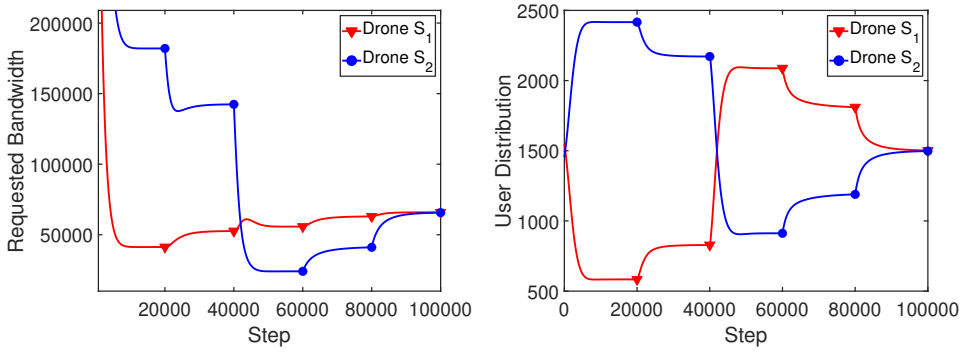
Figure 3.7: eMBB Scenario: Requested Bandwidth (a) and User Distribution (b) at the Stackelberg equilibrium as a function of the price $p_2^{(F)}$

Moreover, we set the delay and interference coefficient to be equal for both drones, with $d_1 = d_2 = 20, k_1 = k_2 = 0.002$

Once again, the behaviour is similar to the one depicted in the static case: as long as S_2 applies a higher fee than S_1, it can buy more bandwidth and attract more users. When, instead, its price drops, the majority of users connect to S_1 instead. When the pricing is the same, the users are equally shared by the drones.

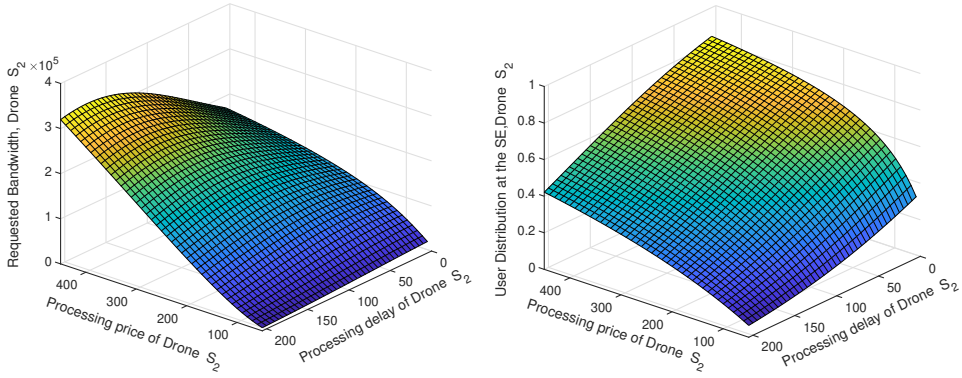
Figures 3.9a and 3.9b depict the requested bandwidth and the user distribution of server S_2 at the SE of the game as both the latency d_2 and pricing $p_2^{(F)}$ increase. The pricing and latency of drone S_1 are instead fixed and equal to $d_1 = 20$ DUs and $p_1^{(F)} = 100$ PUs, respectively. Similarly to their 2D counterparts, the plots highlight how, as the pricing applied by S_2 increases, the amount of requested bandwidth increases as well, driving more users towards S_2. Notably, in spite of the small relevance given to the server latency, a large difference between d_1 and d_2 is still able to impact on the user distribution and the amount of requested bandwidth. Indeed, note how, for high values of d_2, S_2 is forced to request more bandwidth to compensate the higher latency, and also attracts a lower amount of users. When d_2 is low, instead, S_2 attracts most of the users even with lower amounts of requested bandwidth.

In order to estimate the effect of interference and loss (associated to term α_4 in eq. (5)) Figure 3.10 depicts the user distribution as a function of the loss term weight. The system parameters are set as $k_1 = k_2 = 0.002$, $d_1 = d_2 = 20$ DUs, $p_1^{(F)} = 100$ PUs and $p_2^{(F)} = 400$ PUs. As long as the weight of the loss term is kept low, S_2 holds the majority of users. In fact, since



(a) (b)

Figure 3.8: eMBB Scenario: Evolution of Requested Bandwidth (a) and User Distribution (b) at the Stackelberg equilibrium as a function of time.



(a) (b)

Figure 3.9: eMBB Scenario: Requested Bandwidth (a) and User Distribution (b) of drone S_2 at the Stackelberg equilibrium as a function of the price $p_2^{(F)}$ and of the delay d_2.

$p_2^{(F)}$ is four times larger than $p_1^{(F)}$, S_2 can afford a sensibly higher amount of bandwidth. As the weight of the loss term increases, the amount of offered bandwidth becomes less important, until, as in the URLLC case, the users are equally distributed among the two servers when the importance of the loss term becomes equal to 100%.

3.4.3 Trade-off scenario

The following section illustrates the numerical results for a "trade-off" use case, i.e. an intermediate scenario. In this case the corresponding weight of the bandwidth term (BT), price term (PT), delay term (DT) and loss term

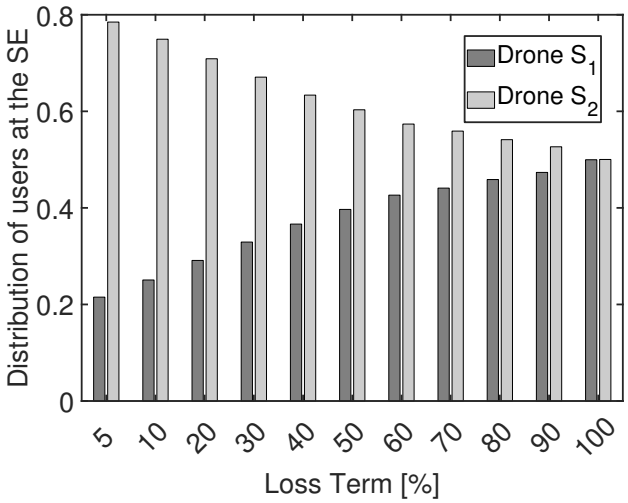


Figure 3.10: eMBB scenario: impact of the weight of the loss term on the User Distribution

(LT) in eq. (3.5) are BT=0.4477, PT=0.0014, DT=0.4312, and LT=0.1197, respectively. As a consequence, the chosen weighing parameters are $\alpha_1 = 0.15, \alpha_2 = 1.5 \cdot 10^{-5}, \alpha_3 = 0.03, \alpha_4 = 0.105$.

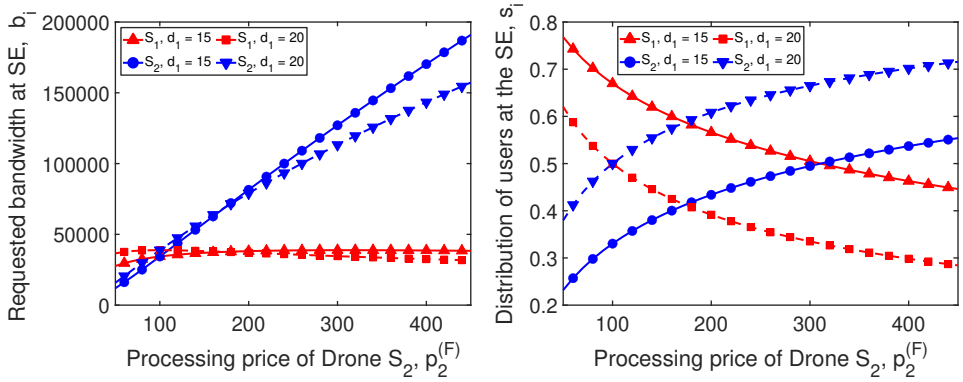
Figures 3.11a and 3.11b depict the requested bandwidth and the distribution of users at the SE equilibrium as a function of price $p_2^{(F)}$. Once again, latency and interference parameters are set as $d_1 = 15$ DUs, $k_1 = 0.0015$ and $d_2 = 20$ DUs, $k_2 = 0.002$ (solid line), and $d_1 = d_2 = 20$ DUs, $k_1 = k_2 = 0.002$ (dashed line). The larger is the VNF fee applied by $p_2^{(F)}$, the higher is the amount of bandwidth requested by S_1. The user distribution shows an intermediate behaviour between the URLLC and eMBB case. Let's focus on the case with $d_1 = 15$ DUs and $d_2 = 20$ DUs (solid lines). In the pricing interval below 300 PUs, the server with a lower processing delay, i.e. S_2, attracts the majority of the users. When $p_2^{(F)}$ passes the threshold value of 300 PUs, however, the amount of users managed by S_1 drops below 50 %. Indeed, while the users are still interested in the processing delay, they also keep an eye to amount of bandwidth offered by the drones. For this reason, when the pricing $p_2^{(F)}$ grows, the amount of bandwidth offered by S_2 increases. This behaviour can also be observed for the case with $d_1 = d_2 = 20$. However, the user behavior shifts a threshold value of 100 PUs, rather than the 300 PUs from the previous case.

Figures 3.12a and 3.12b show how the system reacts to changes in the scenario and reaches a new equilibrium as the latency of server S_2 varies over time. More in detail, the delay of server S_1 is fixed at $d_1 = 20$ DUs, while d_2 gets the values 30, 25, 20, 15, 10 DUs at time interval 0, 20000, 40000, 60000 and 80000, respectively. The price and interference coefficients

for both drones are set to $p_1^{(F)} = 100, p_2^{(F)} = 25, k_1 = k_2 = 0.002$. Generally speaking, S_2 always requests almost four times the bandwidth requeted by S_1, thanks to its higher VNF fee. Initially, the majority of users is attracted by S_1 due to its lower processing delay and high amount of offered bandwidth. As the delay d_2 decreases, however, S_2 attracts more users. Indeed, the lower amount of offered bandwidth is compensated by lower processing delays. More specifically, for $d_2 = 10$ DUs, the users are (almost) equally distributed among the two servers. In fact, on one hand, S_1 compensates a high processing delay with a high amount of offered bandwidth, while, on the other, S_2 is characterized by a low processing delay, but offers a low amount of bandwidth.

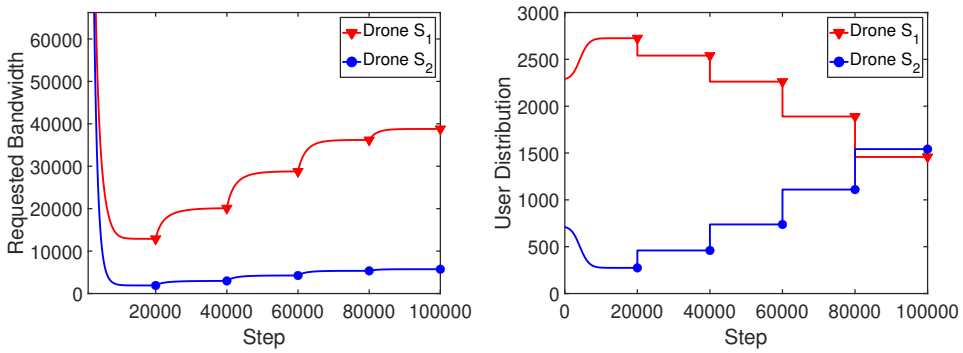
Figures 3.13a and 3.13b show how the system equilibrium varies with changes in the VNF fee $p_2^{(F)}$. More in detail, $p_2^{(F)}$ gets the values 400, 250, 50, 70, 100 PUs at time interval 0, 20000, 40000, 60000 and 80000, respectively. As in the previous case, $p_1^{(F)}$ is equal to 100 PUs. Moreover, we set the delay and interference coefficient to be equal for both drones, with $d_1 = 15$, $d_2 = 20$, $k_1 = k_2 = 0.002$. The bandwidth requested by S_2 varies in line with the VNF pricing. When $p_2^{(F)} > p_1^{(F)}$, S_2 can afford to request more bandwidth than S_1. Conversely, if $p_1^{(F)} > p_2^{(F)}$, the latter offers the most bandwidth. Let's now discuss the user behaviour. When, $p_2^{(F)} = 400$ PUs, S_2, can afford a considerable amount of bandwidth, and, in spite of its higher processing delay, attracts more users than S_1. Then, as the pricing of S_2 drops to 250 PUs, the users equally distribute among the servers. Finally, for $p_2^{(F)} = 50, 70$ and 100 PUs, S_1 can always afford more (or, at least, the same) bandwidth than S_2. Since S_2 is also characterized by a higher processing delay, S_1 attracts the majority of users as a consequence.

Figures 3.14a and 3.14b depict the requested bandwidth and the user distribution of server S_2 at the SE of the game as a function of both latency d_2 and pricing $p_2^{(F)}$. Similarly to the previous cases, pricing and latency of drone S_1 are instead fixed and equal to $d_1 = 20$ DUs and $p_1^{(F)} = 100$ PUs, respectively. The plots show how both the offered latency and requested bandwidth play a role in the choices of users. Indeed, the distribution of users at S_2 reaches its maximum when the drone both offers a fair amount of bandwidth, and a low processing delay. The minimum is instead reached when the drone offers a high latency, and a small amount of bandwidth. When, the offered latency is moderate (about 40-45 DUs) and the amount of offered bandwidth is sufficiently high, S_2 is still capable of attracting an acceptable amount of users, namely between 20 and 30 %. Indeed, latency and bandwidth have similar relevance, and a low delay can compensate for a small amount of requested bandwidth. Similarly, a fair amount of requested



(a) (b)

Figure 3.11: Tradeoff Scenario: Requested Bandwidth (a) and User Distribution (b) at the Stackelberg equilibrium as a function of the price $p_2^{(F)}$



(a) (b)

Figure 3.12: Tradeoff Scenario: Evolution of Requested Bandwidth (a) and User Distribution (b) at the Stackelberg equilibrium as a function of time

bandwidth compensates a higher delay.

Figure 3.15 depicts the impact of the weight of the loss term over the user distribution. We considered the following parameter configuration: $k_1 = k_2 = 0.002$, $p_1^{(F)} = 100$ PUs, $p_2^{(F)} = 50$ PUs. Moreover, we set $d_1 = 15$ DUs, and $d_2 = 20$ DUs. For low weights of the loss term, over 60% of the users choose server S_1, as it offers both a lower delay and a larger amount of bandwidth (thanks to its higher fee). When, instead, the loss term has a relevant importance, the users, as already shown for the URLLC and eMBB cases, tend to distribute more equally, to reduce the experienced interference. Once again, when the weight of the loss term is equal to 100 %, each server manages exactly half of the users.

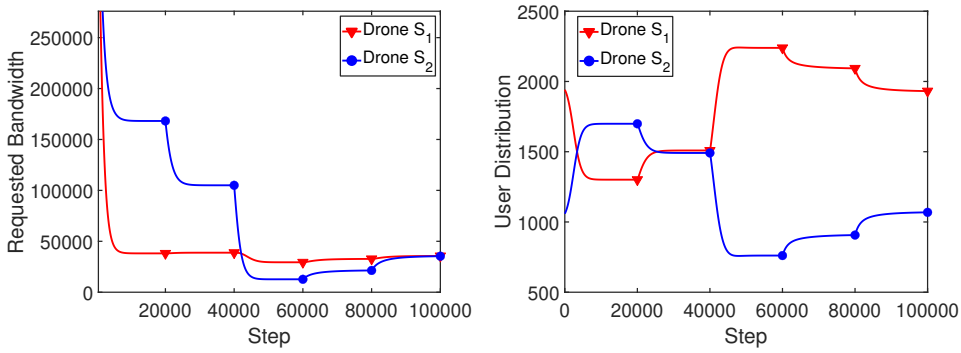


Figure 3.13: Tradeoff Scenario: Evolution of Requested Bandwidth (a) and User Distribution (b) at the Stackelberg equilibrium as a function of time

3.4.4 Multiple Drones Scenarios

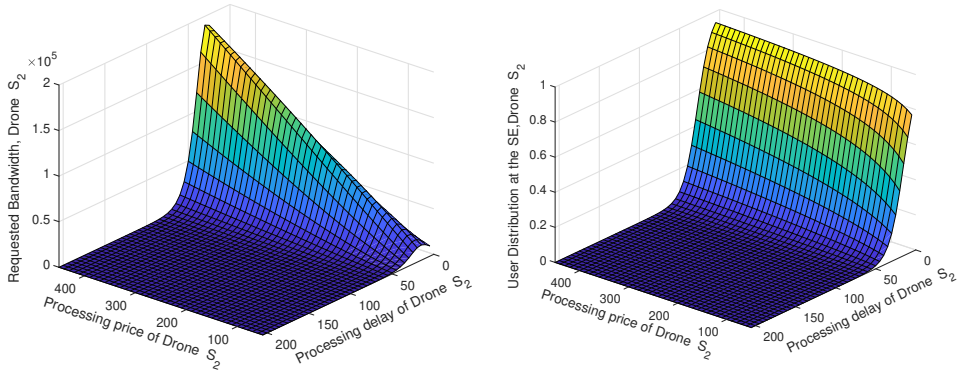
In this section, we analyze the impact of adding additional drones to each considered 5G use case. More in detail, we focus on the specific cases of $S = 3$ and $S = 4$ drones, respectively.

Let us focus on the case $S = 3$. The latency and interference configurations are the same for each scenario, with $d_1 = 14$ DUs , $d_2 = 15$ DUs, $d_3 = 16$ DUs, and $k_1 = k_2 = k_3 = 0.001$. Moreover, we fix $p_1^{(F)} = 250$ PUs and $p_3^{(F)} = 450$ PUs, while $p_2^{(F)}$ ranges from 200 to 450 PUs, with a step of 50 PUs. The remaining parameters are set as in the previous section for each specific use case.

URLLC Scenario Figure 3.16a reports the user distribution for the URLLC use case. As expected, given the slight difference between d_1, d_2, and d_3, the users are distributed in an almost equal way among all the drones, Note, however, how as the pricing increases, S_2 is able to buy more bandwidth and, hence, to attract slightly more users.

eMBB Scenario The user distribution for the eMBB use case is depicted in Figure 3.16b. As opposed to the URLLC case, the pricing applied by S_2 plays an important role on the final user distribution. Indeed, for $p_2^{(F)} = 200$ PUs, S_2 cannot afford to buy as much bandwidth as S_1 and S_2, and attracts only about 10% of the users. As the pricing applied increases, S_2 is able to buy more bandwidth, until it attracts about 45% of the users for $p_2^{(F)} = 450$ PUs.

Trade-off Scenario Last, the user distribution for the trade off use case is shown in Figure 3.16c. The exhibited behavior is intermediate between the ones in the eMBB and URLLC cases. Indeed, the user distribution is not as



(a) (b)

Figure 3.14: Tradeoff Scenario: Requested Bandwidth (a) and User Distribution (b) of drone S_2 at the Stackelberg equilibrium as a function of the price $p_2^{(F)}$ and of the delay d_2

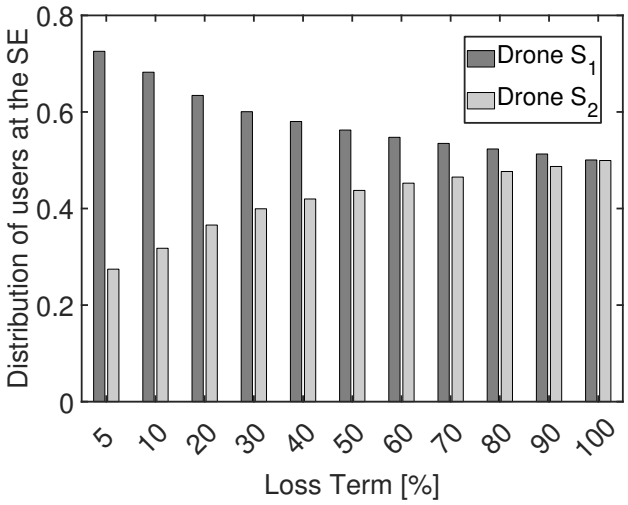


Figure 3.15: Tradeoff scenario: impact of the weight of the loss term on the User Distribution

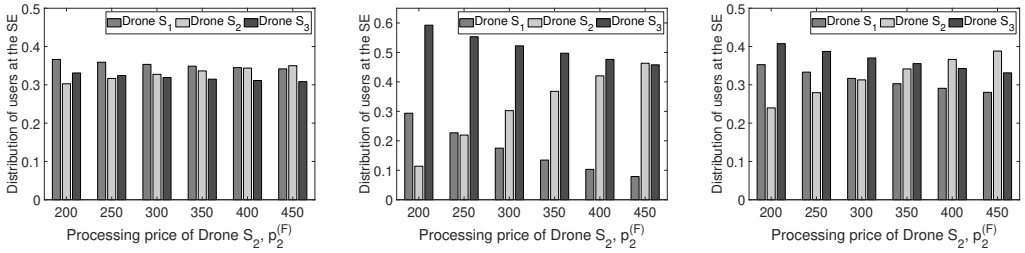


Figure 3.16: User distribution for $S = 3$ drones for the URLLC (a), eMBB(b), and Trade-off (c) scenarios

unbalanced as in the eMBB case, even for a pricing of $p_2^{(F)} = 200$ PUs. On the other hand, unlike in the URLLC scenario, the amount of offered bandwidth still has a moderate importance. Indeed, for $p_2^{(F)} = 400$, S_2 succeeds in attracting the majority of the users.

In the case with $S = 4$, the latency and interference configuration is instead the following: $d_1 = 14$ DUs , $d_2 = 15$ DUs, $d_3 = 16$ DUs, $d_3 = 17$ DUs and $k_1 = k_2 = k_3 = k_4 = 0.001$. Moreover, we fix $p_1^{(F)} = 250$ PUs, $p_3^{(F)} = 350$ PUs, $p_4^{(F)} = 450$ PUs, while, as in the previous case, $p_2^{(F)}$ ranges from 150 to 450 PUs, with a step of 50 PUs. The remaining parameters are set, once again, as in the previous sections.

Figure 3.16a reports the user distribution for the URLLC use case. As expected, given the slight difference between d_1, d_2, and d_3, the users are distributed in an almost equal way among all the drones, Note, however how, as the pricing increases, S_2 is able to buy more bandwidth and, hence, to attract slightly more users.

URLLC Scenario Figure 3.17a depicts the user distribution for the URLLC use case. As in the $S = 3$ case, the users are distributed in an almost equal way among all the drones, as implied by the slight difference between d_1, d_2, d_3, and d_4. The user distribution is subject to a marginal variation as a function of $p_2^{(F)}$; in fact, the number of users associated to S_2 increases from about 25% when $p_2^{(F)} = 200$ PUs to about 30% when $p_2^{(F)} = 450$ PUs.

eMBB Scenario The user distribution for the eMBB use case is shown in Figure 3.17b. As demonstrated in the previous sections, the amount of bandwidth requested by each drone strongly depends on the applied fee. Accordingly, when $p_2^{(F)} = 200$ PUs, S_2 attracts almost no user, while S_4, i.e. the drone with the highest fee, serves about 50% of the total number of users. As the pricing $p_2^{(F)}$ rises, the number of users connected to S_2 rises, and S_2 eventually surpasses S_4 for $p_2^{(F)} = 450$ PUs. Indeed, while both S_2 and S_4

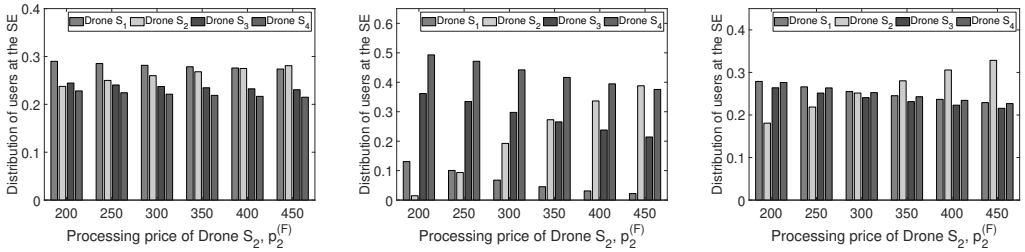


Figure 3.17: User distribution for $S = 4$ drones for the URLLC (a), eMBB(b), and Trade-off (c) scenarios

apply the same fee, and can thus request the same amount of bandwidth, S_2 offers a lower processing delay.

Trade-off Scenario The user behavior for the trade off use case is visible in Figure 3.17c. The user distribution is well balanced, especially in the case $p_2^{(F)} = 300$ PUs, due to the similar importance of the bandwidth and the delay in the users decision. Indeed, the lower amount of bandwidth requested by S_1 and S_2 is counterbalanced by their lower processing delay. Viceversa, the higher delays of S_3 and S_4 are compensated by their larger availability of bandwidth. Eventually, when $p_2^{(F)} = 400$, S_2 manages to attract the majority of users, as it can both offer a large amount of bandwidth, and a lower processing delay.

3.4.5 Benchmark comparison

In order to assess the effectiveness of the introduced approach, we also present a comparison between the proposed game theoretic framework and a benchmark 5G scenario where no marketplace is considered, and server nodes are simply characterized by certain delay and bandwidth features. More specifically, we compare our model with the eMBB and URLLC standard scenarios where users tend to simply choose the best bandwidth efficient or delay efficient server without any form of competition among the servers. In this case we consider the effect of the benchmark policy on the loss term LT which is proportional to the interference experienced when all user nodes are served by a single server (i.e. the best according to the specific considered use case). Figure 3.18 shows a comparison between the benchmark case, i.e. where all the users in the scenario connect to a single drone, and our game theoretical framework. In particular, the evaluation has been performed for $S = 2, 3$ and 4 drones. Note that the advantage of fostering the emergence of a marketplace is evident in terms of improvement of interference resilience.

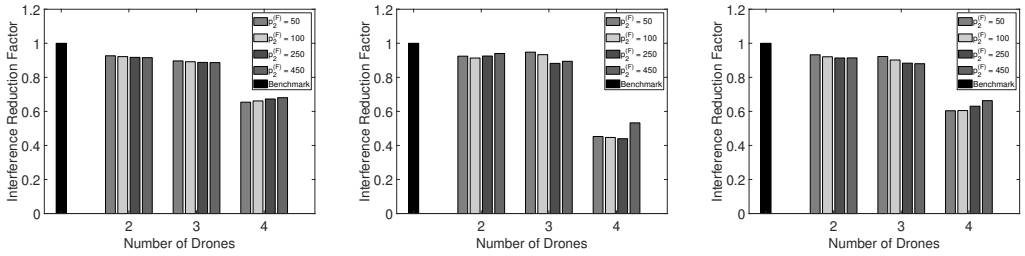


Figure 3.18: Benchmark vs Game-theoretical Framework: Interference reduction for the URLLC (a), eMBB(b), and Trade-off (c) scenarios

Actually, availability of additional providers implies a distribution of users among multiple drones playing as servers, with a consequent reduction in interference. This is particularly true for the $S = 4$ case. While, indeed, the experienced interference is reduced of about 10 to 15% in the case with $S = 2$ and 3 drones, the interference value drops to about 60% (or even less) with respect to the benchmark case with a single drone.

Chapter 4

AI-based Resource Allocation and Computation Offloading in Vehicular Networks

4.1 Introduction

In the last decades, traffic congestion, traffic accidents, and environmental pollution caused by road traffic have become major global issues, also affecting the economic growth of industrialized countries. The global interest in this problem is demonstrated by the fact that, for example, the United States, guided by the "VISION 2050: An Integrated National Transportation System", proposed to build transportation systems with integration, internationalization, inclusiveness, intelligence, and innovation, while the State Council of China identified the "Intelligent Traffic Management System" as a priority element [106]. An improvement in this direction has been registered thanks to the application of the Internet of Things (IoT) and wireless communication technologies. In fact, in the last years, vehicles are evolved from just engine-and-wheels systems to very complex platforms with a huge number of sensors, storage availability, intelligence and computational power [107]. Connected to each other (V2V) and with the fixed network infrastructure (V2I), and more in general to everything (V2X), vehicles constitute the so-called vehicular networks, which are becoming an important part of future Intelligent Transportation Systems (ITS) [108]. Besides applications like advertisements, path planning and navigation, there are a lot of more complex services that require the high perception of surrounding environment (e.g. adaptive traffic signal, autonomous intersection control and license plate recognition). There are also time-critical services, such as cooperative

adaptive cruise driving, autonomous driving, driver-safety enhancement and traffic monitoring that may require huge amounts of computation resources with very low delay.

Deployment of technologies for ITS support involves the installation of equipment both on the roadway, i.e. the Road Side Units (RSU), and inside the vehicles, the so-called On-Board Units (OBU). However, the huge amount of required RSUs to guarantee global coverage, and the limited processing and storage capacity of the OBUs installed on-board vehicles may slow down the spreading process of these technologies.

Recently, a lot of research has been focused on finding of quick deployment and alternative lower-cost methods and techniques to the ITS. With this purpose, 5G networks, the fifth generation of mobile communication technology, thanks to its goals of providing users and connected devices with evolved mobile broadband (eMBB), ultra-reliable and low-latency communications (URLLC) and massive machine-type communications (mMTC), will make a great impulse on the development of ITS [109]. Nevertheless, the success of application of 5G to ITS will depend not only on reliability and low latency of the radio links, but on the performance of the whole path made by the jobs generated by vehicles, including their processing that must be done in a very short time [110, 111]. The solution provided by 5G to support this plethora of services with heterogeneous requirements is the application of the Multi-Access Edge Computing (MEC) paradigm [112]. This paradigm pushes computing and storage resources at the edge of the network, specifically in Base Stations (BSs) and RSUs, in proximity of the end users, close to where the data are generated and likely consumed.

The presence of MEC servers allows the implementation of some offloading policies. However, compared to other MEC-assisted vertical markets, the automotive domain adds a layer of complexity to the process of computation offloading. Indeed, ITS scenarios may present strict requirements in terms of connection stability, quality and reliability caused by high mobility and vehicle speed [113]. Moreover, since RSUs can be deployed on highways in places very far from towns and villages, where Internet connections and even power grid may be not available, they have to be stand-alone computing systems with local computing facilities to perform computing locally, and be fitted with renewable power generators. Finally, due to the huge number of MEC servers that should be installed to cover the road network in capillary way, these should be cheap and easy to install.

With all this in mind, the main objective of the work presented in this chapter is to introduce VMEC-in-a-Box, a smart RSU combined with a MEC station, aimed at providing edge computing for vehicular applications. VMEC-in-a-Box is also equipped with a microeolic power generator that,

if compared with its larger counterpart, is portable, smaller and easier to install, with reduced noise, attractive aesthetics and localized power generation. It is also able to work in presence of lower levels of wind, by varying their computing capacity dynamically to pursue the best tradeoff between performance and power consumption at runtime, and to cooperate with each other by mutually offloading jobs (horizontal offload) with the aim of improving performance and reliability of the whole system. The main contributions of this work are as follows:

- We propose a two-layer architecture for decision making of job offloading, which is able to take into consideration the model of vehicular traffic and the time-variant load produced by vehicles in the area covered by each RSU, as well as offloading costs and performance requirements of the vertical vehicular applications. To this purpose, we define two reward functions, one at the Vehicular Domain and the other at the MEC Domain. These functions are used to weigh costs, performance and energy consumption to make the system reliable, that is, with mean delay, loss probability for queue overflow and outage probability for flat battery, lower than some given thresholds, and therefore acceptable.

- We define the decision-making process in the MEC Domain using a model-based Reinforcement Learning (RL) in order to make the system highly reactive even in cases when the external conditions change abruptly.

- We describe the environment by means of a Markov Decision Process (MDP) based on Markov Modulated discrete-time processes to capture both first- and second-order statistics of the real processes. This allows us to obtain the optimal solution offline, i.e. with no need to be trained online.

- We include in the design parameters the probability of service outage caused by lack of battery charge of the MEC servers, aspect that was never considered in the previous literature.

The above peculiarities make this work, to the best of our knowledge, the first work that proposes an integrated framework, which aims at supporting decisions maximizing reliability and performance in both Vehicular and MEC Domains.

The chapter is structured as follows. Section 4.2 describes the reference system. Section 4.3 formulates the problem, also describing the reward functions to be optimized at both the Vehicular and the MEC Domains,

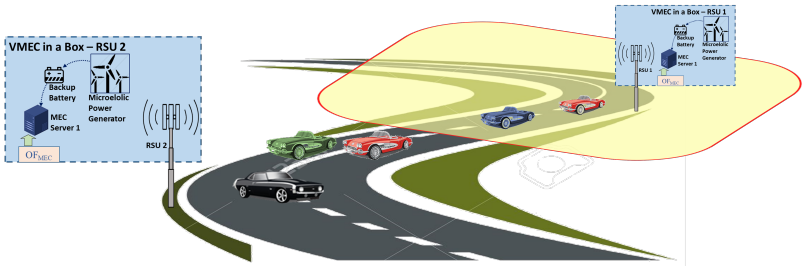


Figure 4.1: Reference System: Portion of road equipped with VMEC-in-a-Box Stations

while Section 4.4 describes the decision-making strategy. Models of the two Domains are presented in Sections 4.5 and 4.6. The main performance parameters are analytically derived in Section 4.7. Section 4.8 presents some numerical results, aimed at describing how the proposed platform works.

4.2 Reference System

The work presented in this chapter focuses on a portion of road, as depicted in Fig. 4.1, providing vehicles with advanced ITS services. Each RSU is equipped with a VMEC-in-a-Box Station, also referred to as VMEC Station for the sake of brevity. The architecture of the VMEC Box is depicted in Fig. 4.2. A VMEC Station is defined as a self-consistent box that is able to allow Telco Operators to create 5G network slices for third party ITS service providers to run application tools very close to the vehicles, at the edge of the network. We consider any kind of application service using objects installed on the vehicles (e.g. video surveillance with 360° cameras installed on the top of the vehicles, or traffic monitoring for smart driving and drive assistance with sensors and cameras).

In order to be easily installed anywhere with no constraints regarding the need of a socket of the power grid, the VMEC Station is equipped with a microeolic power generator. With the aim of improving reliability, a backup battery is also included to supply the station during periods when the power produced by the microeolic power generator is not sufficient to supply the station. The reference system architecture is sketched in Fig. 4.3. Two different domains are highlighted there. The Vehicular Domain is constituted by smart vehicles flowing on the road. In this domain, sensors and other kinds of devices installed on board vehicles generate data to be processed. A burst of data constitutes a job to be processed by a computing facility within a given time that depends on the specific ITS application. Some of these applications require low-latency processing to work in real-time.

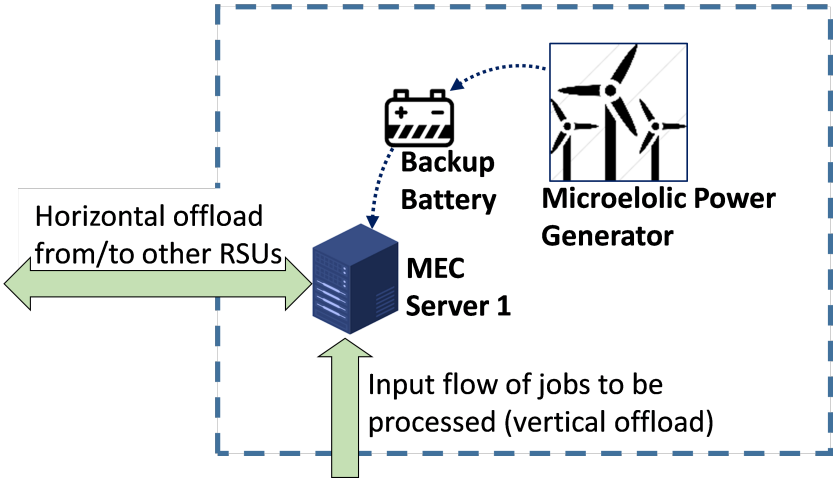


Figure 4.2: Reference System: VMEC-in-a-Box Station Architecture

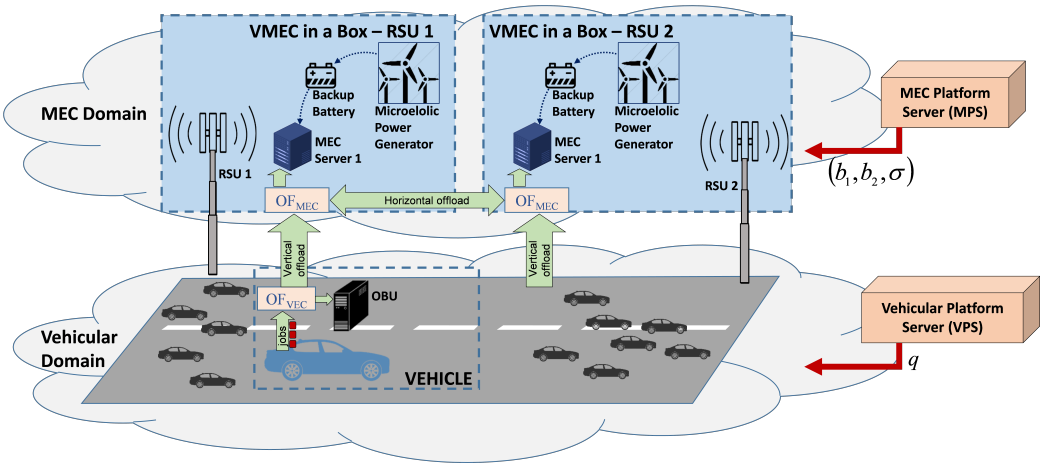


Figure 4.3: Reference System Architecture

This constitutes one of the most stringent QoS requirement for many ITS applications, and has to be taken seriously to enable these kinds of service. For this reason, as depicted in Fig. 4.3, each vehicle is equipped with a local computing station, the OBU, which is able to process jobs produced locally by the same vehicle. In addition, due to the limited processing capacity of an OBU, each vehicle can offload jobs to the closest VMEC Station, the one associated to the RSU used by it as network access point (this offloading operation is named vertical offload, and is indicated as OFVEC in Fig. 4.3).

As already said so far, also detailed in Fig. 4.3, each VMEC Station is constituted by a MEC server to process jobs received by the Vehicular Domain for offloading, a microeolic power generator to supply the whole

station, including the RSU. The backup battery is used to store surplus energy during periods of peak energy generation and to supply the station when the power generated by the microeolic power generator is not sufficient. In order to face overload periods, which are typical in vehicular environments that are characterized by traffic slowing down and job generation peaks, we introduce the possibility of horizontal offload between MEC servers belonging to near VMEC Stations. In this way, the set of VMEC Stations covering a portion of road work as a single entity providing vehicles with MEC services. For this reason, we refer to this set of VMEC Stations as the MEC Domain.

In the sequel, we will assume that the job generation process is statistically equal for all vehicles, and that all vehicles are equipped with OBUs having the same performance. For this reason, the percentage of jobs each vehicle needs to offload to the MEC Domain for a given ITS service application will be considered the same of the other vehicles. This percentage, q, is centrally decided by the Vehicular Platform Server (VPS). More specifically, the VPS decides the value of q maximizing a given reward function that weighs performance and costs, the latter due to the price applied by the MEC Domain to process the jobs received for vertical offloading.

Let us note that the decision of which jobs processing locally or offloaded to the MEC Domain (in case some jobs have greater importance and should be privileged with specific performance) is in charge of each vehicle, and can be done independently of the behavior of the VPS. This decision can be taken according to any strategy proposed in the previous literature regarding risk-aware user's behavior [43], but is out of the scope of this work. If some of such strategies is applied and the amount of jobs that have to be processed locally according to it is high, this would represent only an upper bound for q. However, for the sake of simplicity, in the sequel we do not consider this scenario, and so q ranges in the range [0, 1].

On the other side, in the MEC Domain, two elements play a fundamental role for its performance: the number of CEs, such as CPUs, which have to be maintained active in each VMEC Station, and the amount of horizontal offload from one VMEC Station to another one through a link of the backhaul network. As in [44], this last operation is indicated as OFMEC in Fig. 4.3. With respect to Fig. 4.3, MEC Server 1 can offload a fraction of the received jobs to the MEC Server 2 and vice versa. The direction of offload is decided according to the current load of the MEC servers: the least loaded MEC server can offload jobs to the other one.

The first decision, concerning the number of CEs to be maintained active, has to be taken considering that the higher the number of CEs that are activated in a VMEC Station, the better its performance, but the higher the risk of service outage due to lack of power supply in the backup battery.

Instead, the second decision should have to consider that horizontal offload, on one side, improves performance because it balances load among the MEC servers, but implies some offload costs due to the price applied by the backhaul infrastructure providers for the links connecting VMEC Stations. These links can be realized with fiber optical cables, wireless connections, or even with Unmanned Aerial Vehicles (UAVs) behaving as Base Stations (BS), as proposed in [45]. However, the specific type of these links is out of the scope of this work. Optimization of the above decisions is in charge of an entity named MEC Platform Server (MPS). As described in the sequel, it applies a model-based RL to find the best actions for each state of the MEC-Domain system. The list of these actions is provided to the RSUs that, thanks to a periodic mutual exchange of information regarding their states, implement these actions behaving according to the optimized behavior decided by the MPS.

4.3 Problem Formulation

In this section, we describe how the management servers working at the two different Domains, i.e. VPS and MPS, decide the parameters of their domains to optimize their reward functions. The main notation is synthesized in Table I.

At the Vehicular Domain, the VPS has the task of deciding the percentage q of flow that each vehicle has to forward to the MEC Domain. This is done by optimizing a reward function weighing the following terms: 1) the penalty due to the costs for offloading, $\overline{\omega}$; 2) the mean overall delay, $\overline{\phi}$; 3) the magnitude of the overall job loss probability, $\overline{\psi}^{(mag)}$.

The term $\overline{\omega}$ depends on the per-job offloading price, $\wp_{OL}$, applied by the MEC Domain to the Vehicular Domain for job offloading, and is proportional to the amount of offloaded jobs to the MEC Domain:

$$\overline{\omega} = \wp_{OL} \cdot {R_{OL}}^{(V \to M)} \tag{4.1}$$

where ${R_{OL}}^{(V \to M)}$ is the mean job offload rate from the Vehicular Domain to the MEC Domain.

The mean overall delay term $\overline{\phi}$ is calculated by averaging the mean job computation delay, $\overline{\phi}_V$, suffered in the OBU placed in the vehicle, and the mean job computation delay suffered in the MEC Domain, $\overline{\phi}_M$, by considering the percentage q of jobs to offload to the MEC Domain, and the remaining ones to be locally managed:

$$\overline{\phi} = q\overline{\phi}_M + (1-q)\overline{\phi}_V \tag{4.2}$$

The term $\overline{\psi}^{(mag)}$ represents the magnitude of the overall job loss probability, $\overline{\psi}$, calculated by averaging the loss probabilities suffered by non-offloaded and offloaded jobs at the MEC and the Vehicular Domains, $\overline{\psi}_M$ and $\overline{\psi}_V$, respectively, that is:

$$\overline{\psi}^{(mag)} = -\log_{10} \overline{\psi} \quad \text{where} \quad \overline{\psi} = q\overline{\psi}_M + (1-q)\overline{\psi}_V \tag{4.3}$$

Therefore, we define the reward function that the VPS applies to make decisions regarding the percentage q of flow to be forwarded to the MEC Domain as follows:

$$F_{RW}^{(VPS)} = -\gamma_1\overline{\omega} - \gamma_2\overline{\phi} - \gamma_3\overline{\psi}^{(mag)} \tag{4.4}$$

The constants γ_1 , γ_2 and γ_3 are weighing parameters that can be properly tuned according to some system management criteria defined by the VPS. The parameters $\overline{\omega}$, $\overline{\phi}$ and $\overline{\psi}^{(mag)}$ are normalized in the interval [0,1] in order to be comparable to each other.

On the other hand, according to the optimization model implemented by the MPS at the MEC Domain, the following decisions are applied at each slot:

1. the number b_i of CEs to be active during the time slot for each MEC Server i. The others are put in low-power state to reduce power consumption;

2. the MEC server that can perform offload;

3. the maximum number of jobs, σ, to be locally managed by the MEC server that is enabled to perform offload among the jobs arrived in the considered slot for vertical offload from the Vehicular Domain. The remaining jobs are offloaded to the other MEC server.

Of course, if the number of jobs to be offloaded by the MEC server decided in the above step 2 in a generic slot is higher than the number of rooms available in the queue of the other MEC server after departures and arrivals from its area, only jobs that can be accommodated there are offloaded. The other jobs are discarded by OF_{MEC} block of the first MEC server to avoid payment for their transmission when it is sure that they cannot be accommodated in the other MEC server. The offload direction is based on the current load, represented by the queue lengths in the two MEC servers: horizontal offload is performed by the MEC server with the longest queue towards the other one. Instead, the choice of b_i and σ is taken by each MEC server according to the indications received by the MPS, derived by solving the optimization

model described in Section V.B. To this purpose, the MPS uses a model-based RL with the following reward function considered as the immediate reward of the implemented action:

$$F_{RW}^{(MPS)} = -c_1\overline{\xi} - c_2\overline{\theta} - c_3\overline{\phi}_M - c_4\overline{\psi}_M \tag{4.5}$$

In the above equation, $\overline{\xi}$ represents the mean power consumed by the active CEs. It is proportional to the mean number of active CEs with a proportionality constant, $\wp_{CE}$, given by the mean power consumption of each CE. The term $\overline{\theta}$ is the mean net revenue at the MEC Domain, calculated considering the mean cost applied by the BIP for offloading towards another MEC server through the backhaul network, and the mean monetary gain received by accepting offload from the vehicles based on the price applied to the Vehicular Domain. It depends on the per-job cost applied by the backhaul network provider, $\Theta_{M\to M}^{(OL)}$, and the price applied by the MEC Domain to the Vehicular Domain, $\Theta_{V\to M}^{(OL)}$. Finally, $\overline{\phi}_M$ and $\overline{\psi}_M$ are the mean processing delay for a job and its loss probability suffered in the MEC Domain, both depending on the behavior of the process representing the number of jobs in the MEC queues. The constants c_1, c_2, c_3 and c_4 are parameters that can be properly tuned by the MPS according to some system management criteria (e.g. if minimization of the mean delay is the priority, the constant c_3 is chosen greater than the other ones). They are an input of the problem and their impact will be discussed in Section IX.

In addition, we consider a *performance constraint* regarding the *service outage probability*, $\overline{\Omega}_M$, of a MEC server due to the lack of power supply, despite the presence of a backup battery, which should be not higher than a given threshold $\overline{\Omega}_M^{(Max)}$. Of course, the higher the amount of traffic that is offloaded to the MEC Domain and the harder the requirements in terms of mean delay and loss probability, the higher the need for using a higher number of available CEs. This causes higher battery charge drain, and therefore more difficulties in satisfying the service-outage probability constraint $\overline{\Omega}_M < \overline{\Omega}_M^{(Max)}$. Moreover, let us observe that the service outage probability also depends on the power that the microeolic power generator is able to produce and the backup battery capacity, the latter defined as the amount of charge that can be stored in the battery. For this reason, in Section VIII, we will derive the service outage probability of a MEC server, $\overline{\Omega}_M$, in order that the backup battery, the microeolic power generator and the decision policy can be timely designed at the MEC Domain, and the percentage q of traffic to be vertically offloaded can be decided at the Vehicular Domain.

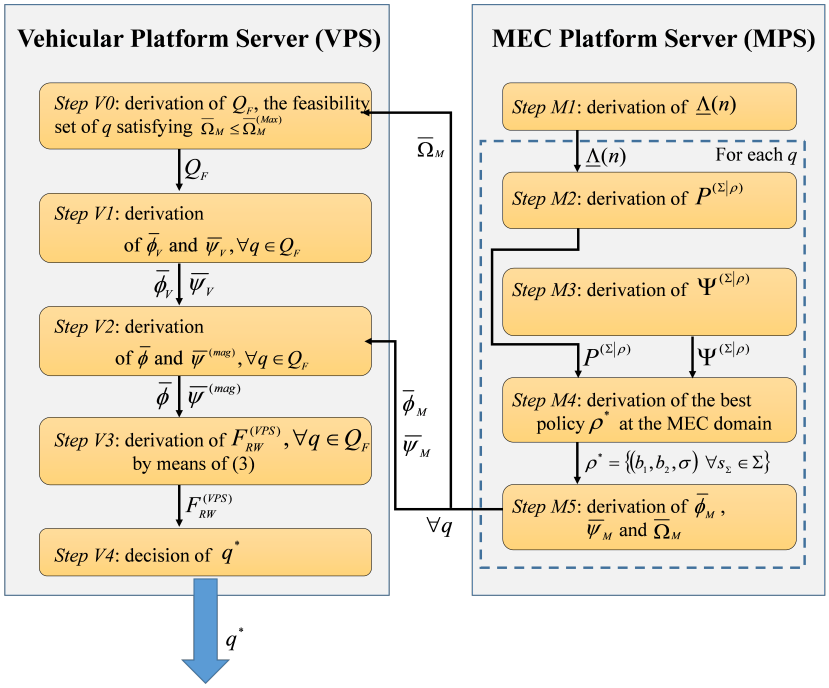


Figure 4.4: Decision Framework for the VPS and MPS domain

4.4 Decision-Making Strategy

In this section, we describe the framework proposed to take decisions by the VPS at the Vehicular Domain (Section V.A) and by the MPS at the MEC Domain (Section V.B). The whole strategy is synthesized in Fig. 4.4. We will apply a discrete-time approach, with a slot duration, Δ, equal to the time needed by a CE of the MEC server to process one job.

Offload Decision Making at the Vehicular Domain

At the Vehicular Domain, the VPS decides the optimal percentage q^* of job flow to be offloaded to the MEC Domain for each vehicle. For the sake of simplicity, we assume that the same percentage is applied for all the vehicles. As shown in Fig. 4.4, it consists of the following steps:

- Step V0: derivation of the feasibility set Q_F , defined as the set of q values such that the requirement $\overline{\Omega}_M < \overline{\Omega}_M^{(Max)}$ for the service outage probability is matched. It is based on the service outage probability $\overline{\Omega}_M$ calculated by the MPS at the Step M5;
- Step V1: derivation of the mean delay, $\overline{\phi}_V$, and the job loss probability, $\overline{\psi}_V$, suffered in the vehicle OBU for each offloading percentage $q \in Q_F$.

They will be derived from the model of the job-processing queue at the Vehicular domain, as described in Section VI;

- Step V2: derivation of the overall mean delay $\overline{\phi}$ and the magnitude of the overall job loss probability, $\overline{\psi}^{(mag)}$, suffered by all the jobs, calculated as in 4.2 and 4.3 by averaging the mean delay and the loss probability suffered by jobs that are offloaded to the MEC Domain, $\overline{\phi}_M$ and $\overline{\psi}_M$, and the ones suffered locally in the vehicles where jobs are generated, i.e. $\overline{\phi}_V$ and $\overline{\psi}_V$. The former are received from the MPS as result of the Step M5 of the offloading decision making strategy at the MEC Domain, while the latter are calculated in the Step V1.

- Step V3: derivation of the reward function $F_{RW}^{(VPS)}$ at the Vehicular Domain, according to 4.4.

- Step V4: derivation of the offloading percentage q^* as the best value of q that maximizes the function defined in 4.4 and calculated at the Step V3 for each value of $q \in Q_F$. This is the output of the VPS work.

4.4.1 Offload Decision Making at the MEC Domain

The best actions for all the states of the MEC Domain are decided by the MPS by using a model-based RL. They are performed periodically at the beginning of each time slot, whose duration will be indicated as Δ. As said so far, the objective of the MPS is to decide the number of CEs to be active during the time slot for each MEC server, b_1 and b_2, the MEC server that can perform offload, and the maximum number of jobs, σ, to be locally managed by the MEC server enabled for offload.

To this purpose, the MPS works according to the steps listed in Fig. 4.4 and described below.

- Step M1: derivation of the model, $\underline{\Lambda}(n)$, of the arrival process to each MEC server of the MEC Domain. This model will be described at the beginning of Section VII and used in the next step to describe the behavior of the MEC Domain.

- Step M2: derivation of the transition probability matrix $P^{(\Sigma|\rho}$ of the Markov Decision Process (MDP) Σ to be used in Step M4 by the MPS to find the optimal policy. It is done by RL, for different values of the Vehicular-Domain offloading probability q ranging between 0 and 1. Its derivation will be described in Section VII.A.

- Step M3: derivation of the immediate reward matrix, $\Psi^{(\Sigma|\rho}$, of the MDP Σ, according to the reward function, $F_{RW}^{(VPS)}$, defined in 4.5. Its derivation will be described in Section VII.A.

- Step M4: derivation of the best policy ρ^*. This is achieved by RL, solving the *Bellman optimality equation system* [114] applying the function $F_{RW}^{(MPS)}$ calculated in the previous step as the *Immediate Reward* $R(n)$. The *Cumulative Reward*, indicated as $G(n)$, is defined as follows:

$$G(n) = \sum_{k=0}^{\infty} \gamma^k \cdot R(n+k+1) \tag{4.6}$$

  where $\gamma \in [0, 1]$

- Step M5: derivation of the mean delay and the loss probability suffered by jobs that are offloaded to the MEC Domain, $\overline{\phi}_M$ and $\overline{\psi}_M$, and of the service outage probability at the MEC Domain, $\overline{\Omega}_M$. These derivations, done applying the optimal policy ρ^* calculated in the previous step, and for each considered value $q \in [0, 1]$ of offloading probability to the MEC Domain, will be described in Section VIII.

4.5 Model of the job processing queue in the vehicular domain

In this section, we model the local computing station (i.e. the OBU) of a vehicle, with the aim of deriving the computing delay, $\overline{\phi}_V$, and the loss probability, $\overline{\psi}_V$, suffered by non-offloaded jobs. They will constitute the results of the Step V1 to be provided to the Step V2 as input. To this purpose, we model the vehicle OBU as a $B/B/1/K_V$ queueing system. The first three parameters mean Bernoulli arrivals, Bernoulli departures, and only one service facility. The parameter K_V represents the maximum number of jobs that can be accommodated in the OBU, i.e. $K_v - 1$ jobs in the queue and one job in the queue service facility.

Let p_V be the per-slot job generation probability for one vehicle, and ξ_V the mean per-job service rate of the local service facility placed on the vehicle, coinciding with the per-slot job departure rate from the OBU service facility. Considering that a percentage q of the flow is offloaded to the MEC Domain, the per-slot arrival probability of a job to the OBU queueing system is $p_{\rightarrow QV} = (1-q)p_V$. The steady-state probability array $\underline{\pi}^{(QV)}$ can be derived as in [115]. Then, applying the Little law, we can derive the mean delay suffered in the OBU as follows:

$$\overline{\phi}_V = \overline{N}_{QV}/\overline{\Lambda}_{QV} \tag{4.7}$$

where $\overline{N}_{QV}$ is the mean number of jobs in the system while$\overline{\Lambda}_{QV}$ is the mean arrival rate, equal to the job emission rate reduced by the percentage of offloaded jobs. They can be calculated as follows:

$$\overline{N}_{QV} = \sum_{s_{QV}}^{K_V} s_{QV} \cdot \underline{\pi}_{[s_{QV}]}^{(QV} \quad \text{and} \quad \overline{\Lambda}_{QV} = p_{\rightarrow QV} = (1-q)p_V \tag{4.8}$$

The job loss probability, $\overline{\psi}_V$, suffered in the OBU by non-offloaded jobs as the ratio between the mean number of lost jobs in the generic slot n and the mean number of jobs arrived in the same slot:

$$\overline{\psi}_V = \frac{\overline{L}_{QV}}{\overline{\lambda}_{QV}} \tag{4.9}$$

The denominator has been calculated in 4.8. The numerator can be calculated as the probability that the queue is full, i.e. $S^{QV}(n) = K_V$, and there are one arrival (occurring with probability $p_{\rightarrow QV}$) and no departure (occurring with probability $1 - \xi_V$), that is, $p_{\rightarrow QV}(1-\xi_V) \cdot \underline{\pi}_{[K_V]}^{(QV}$

4.6 Modeling the MEC Domain

In this section, we will present the models needed to describe the MEC Domain system. More specifically, in Section VII.A we will derive the transition probability matrix and the short-term reward matrix of the MDP Σ, to be used as input of the Step M4. Then, in Section VII.B, we will describe how finding the optimal policy to be applied by the MPS. To simplify notation, we will consider a MEC Domain constituted by only two MEC servers, covering two adjacent areas on the road. Extension to more than two servers is easy, but would complicate notation and the model description. As regards the arrival process to the MEC Layer, as said so far, jobs are generated by devices installed on vehicles moving on the road. Due to the correlation between the number of vehicles in the two considered areas, we model the job arrival process to the MEC Domain coming from the Vehicular Domain as a bi-dimensional Markov modulated process $\underline{\Lambda}(n) = \left(\underline{\Lambda}_1(n), \underline{\Lambda}_2(n)\right)$, $\underline{\Lambda}_1(n)$ and $\underline{\Lambda}_2(n)$ representing the number of jobs that arrive from the Area 1 and the Area 2, respectively. The reader can refer to [115] to derive the transition probability matrix and the joint job emission probability matrix that characterize the arrival process $\underline{\Lambda}(n)$, whose generic elements are:

$$P^{(\Lambda)}_{[\underline{s}'_\Lambda, \underline{s}''_\Lambda]} = \Pr\Big\{ \underline{S}^{(\Lambda)}(n) = \underline{s}''_\Lambda \Big| \underline{S}^{(\Lambda)}(n-1) = \underline{s}'_\Lambda \Big\} \tag{4.10}$$

$$B^{(\Lambda)}_{[\underline{s}''_\Lambda, \lambda_1, \lambda_2]} = \Pr\Big\{ \Lambda_1(n) = \lambda_1, \Lambda_2(n) = \lambda_2 \Big| \underline{S}^{(\Lambda)}(n) = \underline{s}''_\Lambda \Big\} \tag{4.11}$$

Let us notice that, since the MPS uses a model-based RL, which works offline to derive the optimal set of actions related to all the possible states of the environment, the communication links between vehicles and their RSU are modeled with their average behavior. Moreover, due to the short distance between vehicles and RSUs, we assume that these links do not introduce significant delays. Therefore, they do not constitute a bottleneck, and so they are not considered in the Markov Decision Process used by the MPS.

4.6.1 Modeling Markov Decisions at the MEC Domain

With the aim of finding the best policy that optimizes the cumulative reward at the MEC Domain, we model the system at this level with a Markov Decision Process (MDP) Σ . More specifically, as described in Fig. 4.5, we define a three-dimensional discrete-time Markov chain:

$$\underline{S}^{\Sigma}(n) = \Big(\underline{S}^{\Lambda}(n), S^{Q1}(n), S^{Q2}(n) \Big) \tag{4.12}$$

where $\underline{S}^{\Sigma}(n)$ is the state of the underlying Markov chain of the bi-dimensional Switched Batch Bernoulli Process (SBBP) process $\underline{\Lambda}(n)$ defined so far, whereas $\underline{S}^{Q1}(n)$ and $\underline{S}^{Q2}(n)$ are the Markov chains modeling the behavior of the queues Q_1 and Q_2 located in the two MEC servers and used to access their active CEs. Let K be the maximum number of jobs that can be buffered in each of those queues. Therefore, $\underline{S}^{Q1}(n) \in [0 \ldots, K]$, with $i \in \{1, 2\}$. From 4.12, the resulting state space cardinality is $C_\Sigma = C_\Lambda K^2$, where C_Λ is the number of states used to model the job arrival process from the Vehicular Domain.

Jobs generated by vehicles moving in the Area 1, and that are not offloaded, together with jobs coming from vehicles in the Area 2 for offload, will suffer a delay due to the queue Q_1 . Likewise, jobs coming from the Area 2 and not offloaded, together with jobs offloaded by the Area 1, will suffer a delay due to the queue Q_2. Jobs that cannot find space in the queues are lost. We assume that each queue first accommodates jobs coming from the area directly served by it (the Area 1 for Q_1 and the Area 2 for Q_2), and then accommodates the jobs offloaded from the other area.

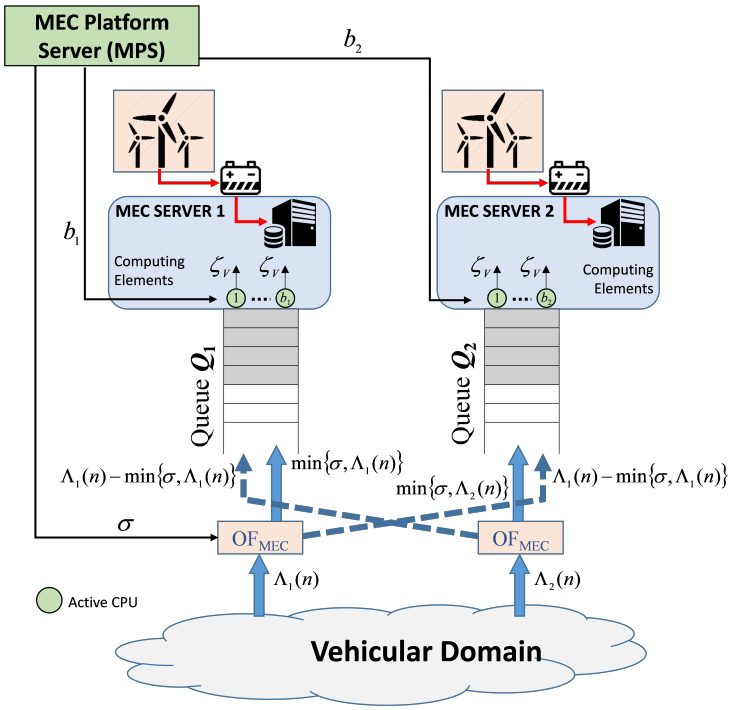


Figure 4.5: Model of the MEC Domain

The choice of the action for each state of the system is the optimal policy decided by RL. In order to calculate the optimal policy ρ^* that specifies an action a for each state of Σ, we derive the transition probability matrix $P^{(\Sigma|\rho)}$ needed in Step M2, and the immediate reward matrix $\Psi^{(\Sigma|\rho)}$. Section VII.B will describe how deriving the optimal policy and will calculate the steady-state probability array when the optimal policy is applied.

In order to calculate the transition probability matrix of the whole system at the MEC Domain, constituted by the job arrival processes coming from the Areas 1 and 2, and by the queueing processes of the MEC 1 and MEC 2 Servers, let us consider the following two generic states:

- $\underline{s}'_{\Sigma} = \left(\underline{s}'_{\Lambda}, \underline{s}'_{Q1}, \underline{s}'_{Q2}\right) = \underline{S}^{\Sigma}(n-1)$: the system state at the slot $n-1$;
- $\underline{s}''_{\Sigma} = \left(\underline{s}''_{\Lambda}, \underline{s}''_{Q1}, \underline{s}''_{Q2}\right) = \underline{S}^{\Sigma}(n)$: the system state at the slot n.

Let $a_{\underline{s}'_{\Sigma}}$ be the action chosen by the MPS at the beginning of the slot n, associated to the state according to the optimal policy ρ^*, decided by RL. To simplify notation, in the sequel we will omit the dependence of the action

from the starting state $\underline{s}_{\Sigma}^{'}$. The generic element of the transition probability matrix can be defined as follows:

$$P^{(Q1,Q2|a)}_{[(s^{'}_{Q1},s^{'}_{Q2}),(s^{''}_{Q1},s^{''}_{Q2})]}(s^{''}_{N1},s^{''}_{N2}) = \\ = \Pr\left\{ \begin{matrix} S^{(Q1)}(n) = s^{''}_{Q1} \\ S^{(Q2)}(n) = s^{''}_{Q2} \end{matrix} \middle| \begin{matrix} S^{(Q1)}(n-1) = s^{'}_{Q1}, \\ S^{(Q2)}(n-1) = s^{'}_{Q2}, A(n) = a \end{matrix} \right\} \tag{4.13}$$

All the details of the derivation are reported in Chapter A of the Appendix. As regards the short-term reward matrix $\Psi^{(\Sigma|\rho)}$, its generic element, representing the immediate reward received performing the action a at the slot n when the system transits from $\underline{s}_{\Sigma}^{'}$ to $\underline{s}_{\Sigma}^{''}$ is defined as follows:

$$\Psi^{(\Sigma|\rho)}_{[\underline{s}^{'}_{\Sigma},\underline{s}^{''}_{\Sigma}]} = E\left\{ R(n) \middle| \begin{matrix} \underline{S}^{(\Sigma)}(n-1) = \underline{s}^{'}_{\Sigma}, \underline{S}^{(\Sigma)}(n) = \underline{s}^{''}_{\Sigma} \\ A(n) = a \end{matrix} \right\} \tag{4.14}$$

It is the output of the Step M3 to be used in 4.6 as the immediate reward. It can be easily derived similarly to [115], but considering that in this case, according to 4.5, it is a weighed sum of the four key performance parameters characterizing the MEC Domain behavior, that is, the mean power consumption, the mean net revenue, the mean delay for a job processed by the MEC Domain and its loss probability. More in deep, from 4.5, we define the immediate reward associated to that transition as follows:

$$\Psi^{(\Sigma|a)}_{[\underline{s}^{'}_{\Sigma},\underline{s}^{''}_{\Sigma}]} = -c_1\overline{\xi}(a) - c_2\overline{\theta}(\underline{s}^{''}_{\Lambda},a) - c_3\overline{\phi}_M(\underline{s}^{''}_{\Sigma},a) - c_4\overline{\psi}_M(\underline{s}^{''}_{\Sigma},a) \tag{4.15}$$

All the details of its derivation are reported in in Chapter A of the Appendix.

4.6.2 Optimal policy and steady-state probability

The optimal policy ρ^* is the set of actions for each state of the MDP. Starting from the knowledge of the dependence of each element of the transition probability matrix and the short-term reward matrix calculated in the previous two subsections on each possible action, for each policy ρ, we define the state-value function of a state s_{Σ} as follows:

$$\nu_{\rho}(\underline{s}_{\Sigma}) = E_{\rho}\left\{ G(n) \middle| \underline{S}^{(\Sigma)}(n) = \underline{s}_{\Sigma} \right\} \tag{4.16}$$

where $E_{\rho}\{\cdot\}$ is the expected value given that the agent follows the policy ρ, and $G(n)$ is the cumulative reward, as defined in 4.6, representing the long-term expected return when the system starts from the state $\underline{s}_{\Sigma}$ and follows

the policy ρ. Substituting 4.6 in 4.16, and applying the theorem of total probability to all the possible arrival states $\underline{S}^{(\Sigma)}(n) = \underline{s}_{\Sigma}^{''}$ after the transition from the slot $n-1$ to the slot n , we derive the so-called Bellman equation [116] for the state $\underline{s}_{\Sigma}^{'}$:

$$\begin{aligned} \nu_{\rho}(\underline{s}_{\Sigma}^{'}) &= E_{\rho}\left\{\sum_{k=0}^{\infty}\gamma^{k}\cdot R(n+k+1)\middle|\underline{S}^{(\Sigma)}(n-1)=\underline{s}_{\Sigma}^{'}\right\} = \\ &= \sum_{\forall \underline{s}_{\Sigma}^{''}\in\Im^{(\Sigma)}} P_{[\underline{s}_{\Sigma}^{'},\underline{s}_{\Sigma}^{''},]}\cdot\left[\Psi_{[\underline{s}_{\Sigma}^{'},\underline{s}_{\Sigma}^{''}]}^{(\Sigma|\rho)}+\gamma\cdot\nu_{\rho}(\underline{s}_{\Sigma}^{''})\right] \end{aligned} \tag{4.17}$$

It gives a relationship between the value of a state $\underline{s}_{\Sigma}^{'}$ and the values of its successive states. It affirms that the value of the start state must equal the value of the expected next state, plus the reward expected along the way. Finding the value of each state for each policy ρ as the solution of a linear system constituted by the above Bellman equation applied to all the states of the Markov chain $\underline{S}^{(\Sigma)}(n)$, by RL we obtain the optimal policy ρ^{*} as the set of the best actions associated to each state of the MDP represented by $\underline{S}^{(\Sigma)}$ [114].

Then, starting from the transition probability matrix defined in 4.12 and calculated for the actions in , we can calculate the steady-state probability array for the MEC Domain, whose generic element is defined as $\underline{\pi}_{[\underline{s}_{\Sigma}]}^{(\Sigma)} = \lim_{n\to\infty}\Pr\left\{\underline{S}^{(\Sigma)}(n)=\underline{s}_{\Sigma}\right\}$. It can be derived by solving the steady-state equation system $\underline{\pi}^{(\Sigma)}\cdot P^{(\Sigma)} = \underline{\pi}^{(\Sigma)}$ with the condition $\underline{\pi}^{(\Sigma)}\cdot\underline{1}^{T}=1$ where, for the sake of conciseness, we have omitted the dependence on the optimal policy ρ^{*}. This will be done also in the next sections.

4.7 Performance Analysis

In this section, we derive the five main performance parameters that characterize the MEC Domain, that is, the mean power consumption, $\overline{\xi}$, the mean net revenue, $\overline{\theta}_{M}$, the mean delay, $\overline{\phi}_{M}$, the loss probability, $\overline{\psi}_{M}$, and the MEC server outage probability, $\overline{\Omega}_{M}$. We will also derive the two main performance parameters of the system as a whole, that is, the overall mean delay, $\overline{\phi}$, and the overall loss probability, $\overline{\psi}$.

To calculate the, we have to consider the number of active CEs decided for each state of the MEC Domain at the generic slot, multiplied by the power consumption of each CE, and weighed by the steady-state probability of that state:

$$\overline{\xi} = \sum_{\underline{s}_\Sigma \in \Im^{(\Sigma)}} \pi^{(\Sigma)}_{[\underline{s}_\Sigma]} \cdot (b_1 + b_2) \cdot \wp_{CE} \tag{4.18}$$

where b_1 and b_2 are the number of active CEs for both the MEC servers decided for the state $\underline{s}_\Sigma$. Energy saving can also be easily derived as follows:

$$E_s = (\overline{\xi}_L - \overline{\xi})/\overline{\xi}_L \cdot 100 \tag{4.19}$$

where $\overline{\xi}_L$ is the power consumption when L CEs are always active, i.e. $\overline{\xi}_L = L \cdot \wp_{CE}$.

Likewise, the mean net revenue, the mean delay and the loss probability can be easily derived by calculating these parameters for each state $\underline{s}_\Sigma$ of the system, and averaging them over all the system states using the associated steady-state probability. Details of their derivation can be found in in Chapter A of the Appendix.

The last key performance parameter at the MEC Domain is the MEC server outage probability. It is used as the metric to quantify the system reliability regarding cases in which the backup battery of a MEC server is empty. In these cases, the MEC server is not supplied, and therefore its service is not available. In order to calculate this probability, we assume that the batteries used to supply the MEC servers are lithium-ion polymer batteries, so with linear charging and discharging processes. For this reason, we model each battery as a buffer: at each slot, it is loaded by the energy produced by the microeolic power generator [117, 116], and dequeued of the energy consumed by the computing facility (CF) of the MEC servers in the same slot. The energy consumption process of the CF is time-variant since its electrical load depends on the number of active CEs according to the decisions of the MPS.

More in deep, for the generic MEC Server $i \in \{1, 2\}$, we model the backup battery (BT) behavior with the 3-dimensional Markov process $\underline{S}^{(BT)}(n) = \left(S^{(RG)}(n), S^{(QB)}(n), S^{(CF)}(n)\right)$, where $S^{(RG)}(n) \in \Im^{(RG)}$ is the state of the microeolic power generator, $S^{(QB)}(n) \in \Im^{(QB)}$ the state of the backup-battery charge level, and $S^{(CF)}(n) \in [1, L]$ the state of the computing facility, that is, the number of active CEs in the slot n decided by the MPS for the considered MEC Server i.

In order to model the state of the battery charge level, we quantize it in quantum of charge (QoC), and indicate the number of QoCs corresponding to the fully-charged battery as K_{QoB}. Let $\Im^{(QB)} = \left\{0, 1, \cdot, K_{QoB}\right\}$ be the space set of $S^{(QB)}(n)$ and $\Im^{(RG)}$ be the space set of $S^{(RG)}(n)$. The dimension of the battery model state space is $C_{BT} = C_{RG} \cdot K_{QoB} \cdot L$, where C_{RG} is

the number of states used to model the behavior of the microeolic power generator. Let us define the transition probability matrix of the battery system. To this purpose, let us consider two generic states of $\underline{S}^{(BT)}(n)$, i.e. $s_{BT}^{'} = \left(s_{(RG)}^{'}, s_{(QB)}^{'}, b^{'}\right)$ in the slot $n-1$, and $s_{BT}^{''} = \left(s_{(RG)}^{''}, s_{(QB)}^{''}, b^{''}\right)$ in the slot n. The generic element of the transition probability matrix of the battery system can be calculated as follows:

$$P^{BT}_{[\underline{s}^{'}_{BT}, \underline{s}^{''}_{BT}]} = \Pr\Big\{\underline{S}^{(BT)}(n) = \underline{s}^{''}_{BT} \Big| \underline{S}^{(BT)}(n-1) = \underline{s}^{'}_{BT}\Big\} \tag{4.20}$$

Its derivation is reported in the in Chapter A of the Appendix for the sake of readability.

Finally, after calculating the steady-state probability $\underline{\pi}^{(BT)}$ of the battery behavior process $\underline{s}_{BT}(n)$ by solving the steady-state equation system applied to the matrix $P^{(BT)}$, the service outage probability of a given MEC server i can be derived as follows:

$$\begin{aligned} \overline{\Omega}_{M,i} &= \Pr\Big\{\text{Outage for empty battery}\Big\} = \\ &= \Pr\Big\{S^{(QB)}(n) = 0\Big\} = \sum_{\forall \underline{s}_{BT} | s_{QB}=0} \pi^{(BT)}_{[\underline{s}_{BT}]} \end{aligned} \tag{4.21}$$

Let $\overline{\Omega}_{M,1}$ and $\overline{\Omega}_{M,2}$ be the service outage probability for the MEC Servers 1 and 2, respectively. We define the overall MEC Domain service outage probability as the probability that at least one of the two MEC servers are out because we assume that none of the servers is able to support the global traffic. Therefore, the overall MEC Domain service outage probability can be derived as follows:

$$\overline{\Omega}_M = 1 - 1\Big[(1 - \overline{\Omega}_{M1}) \cdot (1 - \overline{\Omega}_{M2})\Big] \tag{4.22}$$

Finally, we derive the two main performance parameters of the system as a whole, that is, the overall mean delay, $\overline{\phi}$, and the overall loss probability, $\overline{\psi}$. The overall mean delay and the overall loss probability can be calculated as in 4.2 and 4.3, respectively. The first terms of 4.2 and 4.3 are the mean delay $\overline{\phi}_V$ and the loss probability $\overline{\psi}_V$ suffered in the Vehicular Domain, and derived in 4.7 and 4.9. Instead, the terms $\overline{\phi}_M$ and $\overline{\psi}_M$ are the mean delay and the loss probability suffered in the MEC Domain, and can be calculated as follows:

$$\begin{aligned}\overline{\phi}_M &= \sum_{\forall \underline{s}_\Sigma \in \Im^{(\Sigma)}} \pi^{(\Sigma)}_{[\underline{s}_\Sigma]} \cdot \overline{\phi}_M(\underline{s}_\Sigma, a) \\ \overline{\psi}_M &= \sum_{\forall \underline{s}_\Sigma \in \Im^{(\Sigma)}} \pi^{(\Sigma)}_{[\underline{s}_\Sigma]} \cdot \overline{\psi}_M(\underline{s}_\Sigma, a)\end{aligned} \tag{4.23}$$

where $\overline{\phi}_M(\underline{s}_\Sigma)$ and $\overline{\psi}_M(\underline{s}_\Sigma)$ are the mean delay in the MEC Domain associated to the state $\underline{s}_\Sigma$ when the action a is applied. They can be easily calculated as reported in Chapter A of the Appendix.

4.8 Numerical Results

In this section, we present some numerical results to demonstrate the gain of the proposed system as compared to a standard management technique of the MEC Domain that does not apply intelligence to change the number of active CEs dynamically at runtime. Moreover, we will analyze the impact on the overall performance of the chosen set of weights of the reward functions at both the MEC and the Vehicular Domains. Finally, we will provide some design guidelines to match some given objectives pursued at the two domains according to the specific vertical vehicular application.

To this purpose, we will consider a real use case that will be described in Section IX.A. Section IX.B will be devoted to present some numerical results at the MEC Domain, while Section IX.C will present some results from the vehicular perspective. Finally Section IX.D will discuss on how dimensioning the battery capacity of MEC servers, and the impact of this choice on the feasibility region of the offload percentage q.

4.8.1 Use Case Setup

As a use case, we consider a video surveillance application realized by a set of video cameras mounted on cars and buses that produce photos and short video-clips to be processed by a video-processing unit based on machine learning to find garbage or other kinds of objects left on the roadsides. Measurements were realized on the stretch of 400 m of road depicted in Fig. 4.6. We have covered this stretch of road with two MEC Stations. The behavior of the number of vehicles in the two areas in one month of measurements is described in [118], while the Markov models of the vehicle arrival process $O(n)$ to the area 1, the vehicle departure process $\mu_1(n)$ from the area 1 towards the area 2, and the vehicle departure process $\mu_2(n)$f rom the area 2 are provided in [115]. Accordingly, assuming that each vehicle generates one

Figure 4.6: Stretch of road considered as use case

job in a slot with probability $p_V = 0.95$, the transition probability matrix and the joint job emission probability matrix that characterize the arrival process $\underline{\Lambda}(n)$ to both the areas can be computed as in Section VII.A.

At the Vehicular Domain, vehicles are equipped with a miniPC with an Intel Celeron Processor N4020 2-Cores/2-Threads 4GB DDR4 RAM, working as OBU, that is able to process jobs with a mean service rate of $\xi_V = 0.75$ job/slot and having a queue size of $K_V = 10$ jobs.

As far as the MEC Domain is concerned, we consider MEC servers equipped with $L = 4$ CEs, each realized with a ThinkCentre M75n with AMD Ryzen 5 PRO 3500U Processor with 4 cores and consuming, on average, $\wp_{CE} = 47.4$W. Let $K = 10$ jobs be the size of the MEC server queue. Decision making is performed by RL with a discount factor $\gamma = 0.8$. The MEC Domain applies a job-offloading price $\Theta^{(OL)}_{V\to M} = 0.1$ PrU to the Vehicular Domain, where PrU is the unit of price. The same price is applied for horizontal offload of one job from one MEC server to another one, i.e. $\Theta^{(OL)}_{V\to M} = 0.1$ PrU.

In the numerical analysis, we will consider three strategies applied by the MPS at the MEC Domain and two strategies applied by the VPS at the Vehicular Domain. These strategies are characterized by different weighing parameters. Among all the strategies, S_{M1} is the more balanced, giving all the components of the reward function at the MEC Domain the same importance, i.e. with $c_1 = c_2 = c_3 = c_4 = 1$. The other two strategies are, instead, more focused on delay and loss, setting $c_3 = c_4 = 2$ for S_{M2}, and $c_3 = c_4 = 6$ for S_{M3}. The weight $c_1 = 1$ is used for both these last strategies, while the weight of the mean net revenue, c_2 , is set $c_2 = 0.5$ for S_{M2} and

	S_{M1}	S_{M2}	S_{M3}
c_1	1	1	1
c_2	1	0.5	1
c_3	1	2	6
c_4	1	2	6

Table 4.1: MEC-Domain Strategy parameters

	S_{V1}	S_{V2}
γ_1	1	1
γ_2	1	6
γ_3	1	6

Table 4.2: Vehicular-Domain Strategy parameters

$c_2 = 1$ for S_{M3}. In such a way, S_{M2} and, even more, S_{M3} strongly privilege delay and loss probability as compared with power consumption and mean net revenue.

As regards the Vehicular Domain, we consider the strategy S_{V1} weighing all the components of the reward function at the same way, that is, with $\gamma_1 = \gamma_2 = \gamma_3 = 1$, while the strategy S_{V2} strongly weighs mean delay and loss probability as compared with the penalty due to the costs for offloading. This is achieved by setting $\gamma_2 = \gamma_3 = 6$, while maintaining $\gamma_1 = 1$. The parameters of these strategies are synthesized in Tables 4.1 and 4.2.

4.8.2 Perfomance Evaluation of the MEC Domain

In this section, we evaluate the gain achieved by changing the number of active CEs at runtime by means of RL. To this purpose, we will compare the performance of the proposed VMEC-in-a-Box system with a standard baseline model.

The latter employs RL to decide how many jobs to send to the other MEC server for horizontal offload, but maintains the number of active CEs constant. More specifically, we will refer to the model proposed for the VMEC-in-a-Box system as the RL-All policy. Instead, the policies that apply RL only to manage the horizontal offload while keeping the number of active CEs constant and equal to $L = 2$, $L = 3$ and $L = 4$ are named RL-OL2, RL-OL3, RL-OL4, respectively.

Fig. 4.7 and 4.8 present a comparison of the proposed policy, RL-All, with the other policies introduced so far, in terms of mean delay and mean loss probability, calculated as in 4.23, while Fig. 4.9 shows the energy saving percentage calculated as in 4.19. These figures also show a comparison between the three considered RL-All policies at the MEC Domain, that is, S_{M1}, S_{M2} and S_{M3}.

As far as the mean delay is concerned, we can observe that S_{M3} performs better than the other RL-All strategies, while S_{M1} offers the worst

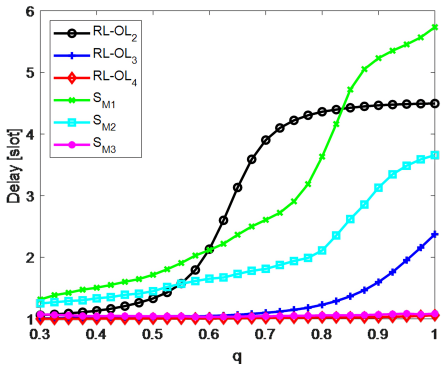


Figure 4.7: Mean Delay at the MEC Domain

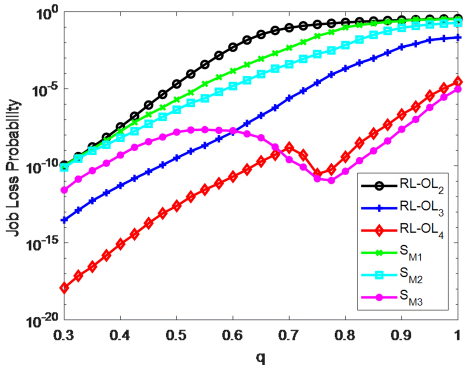


Figure 4.8: Loss Probability at the MEC Domain

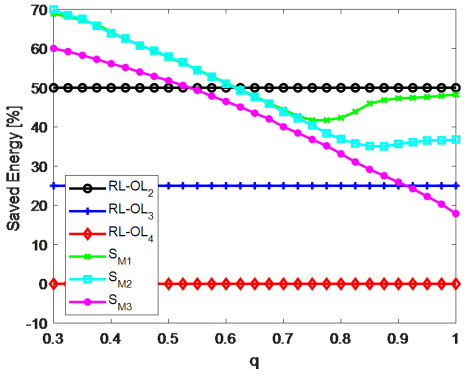


Figure 4.9: Energy saving percentage

performance. Indeed, S_{M1} results in a mean delay that is even higher than RL-OL2, which uses only two CEs, at least for q smaller than 0.6 and bigger than about 0.8. This is due to the importance that the three strategies give to the mean delay, which is maximum for S_{M3} and minimum for S_{M1}, as illustrated in Table 4.1. As regards the three policies with fixed active CEs, the higher the number of active CEs, the lower the mean delay. The same reasoning can be applied for loss probability, shown in Fig. 8.

The non-monotonic behavior of some curves is due to the need of balancing the privileges provided to the loss probability with the ones of the other parameters in the reward function.

However, it is necessary to analyze Fig. 4.7 and 4.8 together with energy saving percentage shown in Fig. 4.9 to better understand the overall performance of each technique. Indeed, if, on one hand, it is evident that

RL-OL4 obtains a delay of only one slot, i.e. the time needed to process one job, and a negligible loss probability, on the other hand, it consumes a lot of energy to maintain all the CEs active. On the other side, the policy RL-OL2 presents the advantage of saving 50% of energy, but causes high delays and loss probability even with very low values of the offload percentage q. Instead, the curves regarding the three strategies considered for RL-All are able to achieve a good tradeoff between delay and loss performance on one side, and energy saving on the other side, and this tradeoff can be tuned by a timely chosen strategy: the highest energy saving is achieved by S_{M1}, as it gives more importance to this aspect. Indeed, we can observe how S_{M1}, although characterized by delay and loss probability performances that are comparable with RL-OL2, achieves higher energy saving, even when compared to the other RL-All strategies.

Similarly, S_{M3} (especially in the case of a high load of offloaded jobs, i.e. for q higher than 0.6) achieves a behavior comparable to RL-OL4 in terms of loss and delay, but with a sensible amount of saved energy.

As expected, power consumption has a strong impact on the service outage probability, as shown in Fig. 4.22, where this parameter has been calculated according to 4.22 for a battery capacity of 20 Ah in the MEC Server 1, needed to manage a bigger amount of traffic, and of 12 Ah in the MEC Server 2. Let us notice that in the study of this subsection, these values of battery capacity are an input of the problem, but they can be designed thanks to the proposed model, as described in Section IX.D. Plots regarding RL-OL2, RL-OL3, RL-OL4 are flat because they maintain the number of active CEs constant independently of the input traffic coming from the Vehicular Domain. Moreover, we can observe that RL-OL3 and RL-OL4, in spite of their good performances in terms of mean delay and loss probability, are characterized by unacceptable service outage probabilities, while the RL-OL2 strategy provides better service outage probability, although very high, but unacceptable delays and losses. On the contrary, in Fig. 4.10, we can appreciate how the proposed framework is able to guarantee a very low service outage probability for a wide range of q. More in deep, let us note that service outage probability for the three RL-All policies is negligible for values of q approximately less than 0.6. For higher values of q, the outage probability starts to increase more rapidly; however, the fact that we have chosen a bigger battery in the MEC Server 1 and that we use horizontal offload for load balancing purposes, determines a similar behavior of the outage probability curves for both the MEC servers. Indeed, as highlighted by the mean number of per-slot offloaded jobs plot shown in Fig. 4.11, horizontal offload is not much used for low values of q, while it increases when the MEC Domain starts to become more stressed. As expected, S_{M2} offloads more

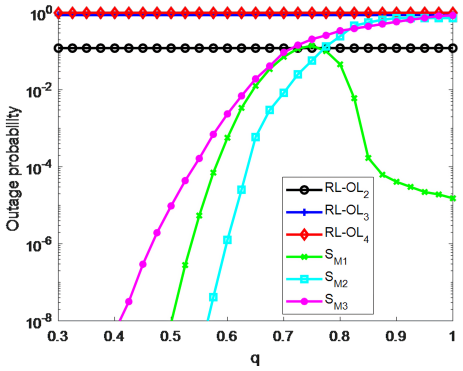


Figure 4.10: Service Outage Probability at the MEC Domain

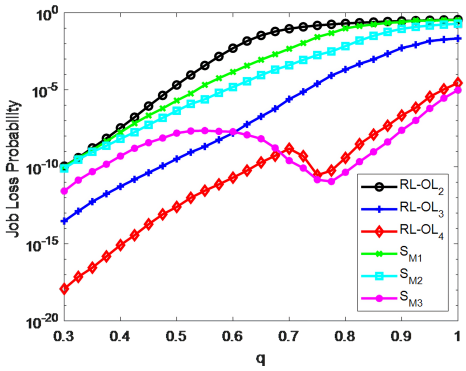


Figure 4.11: Mean number of per-slot offloaded jobs

jobs on average in comparison to S_{M1} and S_{M3}, as it gives less importance to the mean net revenue and, therefore, can afford to spend more money on offloading procedures. Another observation that can be done by looking at Fig. 4.11 is that offload is more used by RL-OL2, RL-OL3, RL-OL4 for their lack of flexibility in terms of modulation of the number of active CEs, but this causes an increase of costs for offloading with a consequent decrease in the reward function.

Finally, the non-monotonic trend of the service outage probability presented in Fig. 4.10 for S_M1 is due to the higher importance that this strategy gives to energy consumption, that determines a decreasing in the service outage probability when, due to the high values of q, delays and losses tend to increase. Indeed, when the MEC Domain is overloaded (high values of q), S_{M1} privileges energy saving, as it is also evident in Fig. 4.9, at expense of delays and losses, while S_{M2} and S_{M3} prefer to use more CEs to keep delays and losses low.

4.8.3 Overall performance from the Vehicular perspective

In this section, we analyze performance of the overall system, considering the point of view of the VPS at the Vehicular Domain. For this reason, we focus on a given strategy at the MEC Domain, specifically the S_{M2} one, and evaluate the impact of the strategy chosen at the Vehicular Domain. In particular, as said in Section IX.A, we compare two strategies at this domain, i.e. S_{V1} that weighs costs for vertical offload, delays and loss probabilities at the same way, and S_{V2} that gives more importance to performances in terms

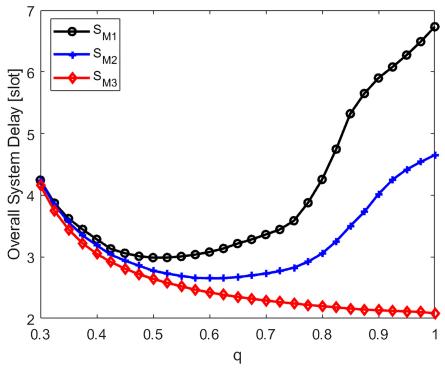


Figure 4.12: Overall System Delay, from the Vehicular Domain viewpoint

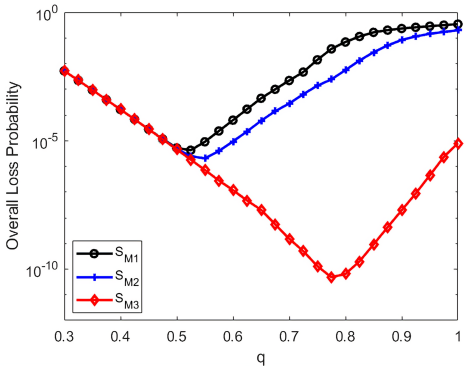


Figure 4.13: Overall System Loss Probability, from the Vehicular Domain viewpoint

q^*	S_{M1}	S_{M2}	S_{M3}
S_{V1}	0.325	0.4	0.425
S_{V2}	0.5	0.6	0.975

Table 4.3: Optimal Percentage of Offload to the MEC Domain

of delays and loss probability.

In Fig. 4.12 and 4.13, we have plotted the overall system performance in terms of mean delay, calculated as in 4.2, and loss probability, calculated as in 4.3, as seen from the Vehicular Domain viewpoint. Each of these figures presents three curves, one for each MEC Domain strategy, calculated against the amount of traffic that is offloaded to the MEC Domain. As expected, the overall performance is compliant with the requirements of each MEC strategy. Therefore, S_{M3} is the best strategy in terms of delays and losses, whilst S_{M1} the worst one. The non-monotonic trend depends on the fact that, when q increases, performance gets worse at the MEC Domain while improves at the Vehicular Domain.

The best working point depends on the specific strategy applied by the VPS at the Vehicular Domain. To this purpose, Fig. 4.14 compares the reward function at the Vehicular Domain obtained for each of the two strategies considered by the VPS, i.e. S_{V1} and S_{V2}. For each of these strategies, the curves are calculated for the three strategies considered at the MEC Domain. As we can see, each curve has a maximum value q^* that represents the optimal decision taken by the VPS at the Step V4 (see the VPS work description illustrated Fig. 4.4). The best values obtained from Fig. 4.14

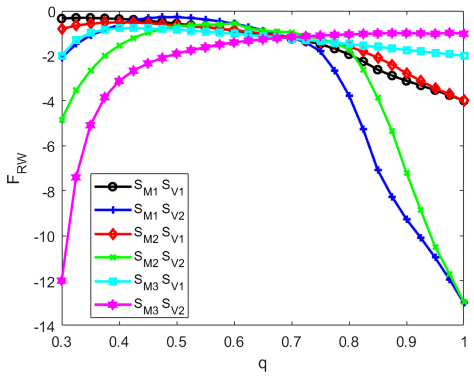
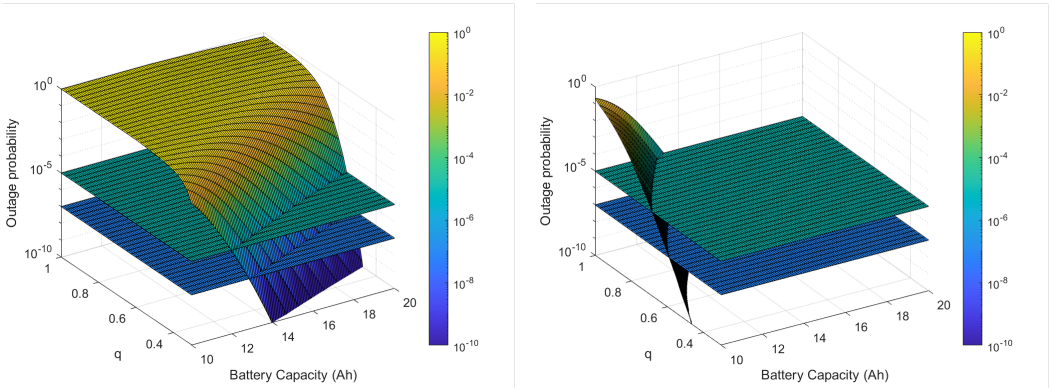


Figure 4.14: VPS Reward Function for the two Vehicular Domain Strategies for each strategy considered at the MEC Domain



(a) MEC Server 1 (b) MEC Server 2

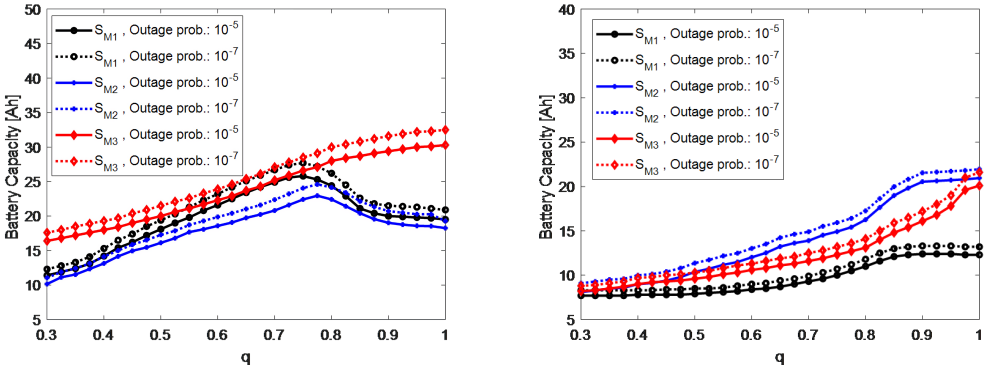
Figure 4.15: Service Outage Probability

are reported in Table 4.3. Each value in the table represents the percentage of traffic that each vehicle should offload to the MEC Domain to maximize the reward function at the Vehicular Domain.

4.8.4 An example of system design

Thanks to the fact that the behavior of the two entities MPS and VPS, whose target is decision making at MEC Domain and Vehicular Domain, is based on an analytical model, we can use this model to design the main parameters of the system. Let us note that design of the MEC Domain is independent of the design at the Vehicular Domain. Specifically, parameters of the Vehicular Domain are designed when the MEC Domain behavior is known.

At the MEC Domain, the main choices regard the strategy and the bat-



(a) MEC Server 1 (b) MEC Server 2

Figure 4.16: Feasibility Region

tery capacity. The choice of the parameters characterizing the strategy, as discussed so far, is determined by the importance that MPS needs to provide to the four components of the reward function, that is, energy consumption, net revenue, mean delay and loss probability. Availability of plots like the ones reported in Fig. 4.8, 4.9 and 4.10 allows the MEC designer to choose the best strategy.

In order to decide the battery capacity, the model allows the designer to draw three-dimensional figures representing the service outage probability as a function of battery capacity and offloading percentage q. A decision on the battery capacity limits the range Q_F of q (see Step V0 of the VPS work description illustrated Fig. 4.4). In order to show how this is done, in Fig. 4.15, as an example, we have plotted the service outage probability, calculated as in 4.21, for each MEC server and for the MEC strategy S_{M2}, and added two planes to calculate the intersection at 10^{-5} and 10^{-7}. The intersections for all the three strategies are plotted in Fig. 4.16 for both the MEC servers. This figure allows us to determine the feasibility region that limits the choice of the battery capacity. For example, if the strategy S_{M3} is used and a service outage probability of $10^{-}7$ is required, a battery installed on the MEC Server 1 with capacity of 25 Ah limits the range Q_F of q as $Q_F = \{\forall q \leq 0.65\}$. Instead, if we want to support all values of q, we need to install a battery with capacity of 33 Ah in the MEC Server 1 and of 23 Ah in the MEC Server 2.

Part II

Resource Allocation for AI

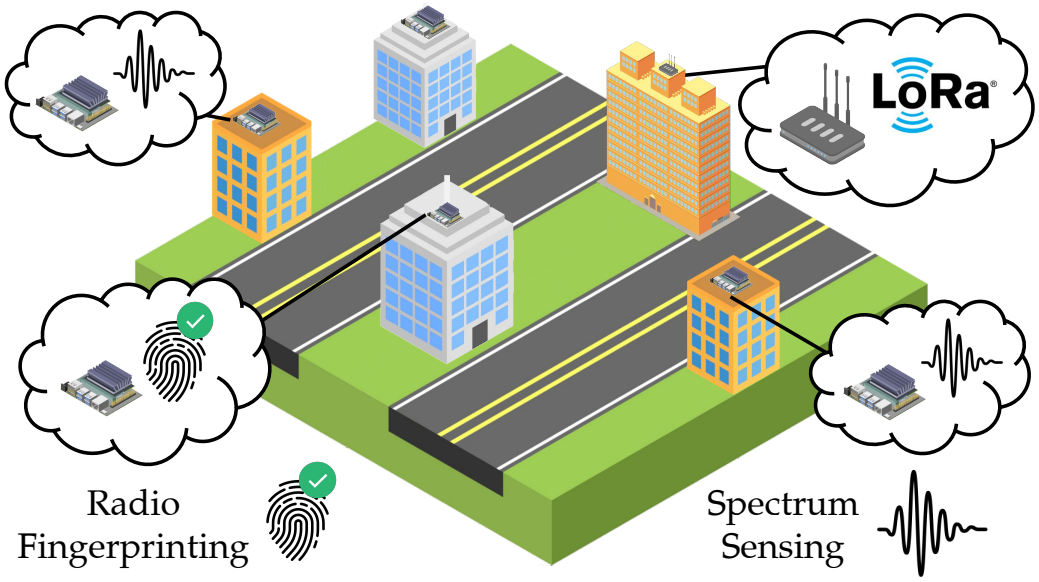


Figure 4.17: Federated ML for the IoT.

Introduction

Federated machine learning (FML) is being increasingly used to improve the robustness of locally-trained deep neural networks (DNNs). This is ultimately achieved by merging local DNNs into a globally-shared model [16], thus overcoming the non-i.i.d. nature of the local datasets [119, 120, 121, 122]. In the context of the Internet of Things (IoT), FML perfectly fits scenarios where a large number of "spectrum sensors" need to be deployed over a large geographical area – see Figure 4.17 – to alleviate spectrum congestion by performing operations such as DNN-based radio fingerprinting (RFP) [15, 123, 124] and DNN-based spectrum hole detection (SHD) [125, 13]. These operations are crucial to enable secure and effective dynamic spectrum access (DSA) [126, 127, 128, 129, 130]. Enabling fast and efficient FML in this context becomes fundamental since (i) spectrum sensors may be handled by different entities, which may not want to share their local dataset for privacy reasons; (ii) sending raw spectrum measurements for centralized training would be impractical, as the required data rate would go far beyond the capabilities of IoT sensors; (iii) the locally-collected spectrum datasets will be subject to different noise, interference, and fading regimes, which renders the local DNNs incapable to generalize to different spectrum environments [14].

The key challenge is that FML is based on the exchange of local DNN parameters to a central server (also called the *aggregator*). However, IoT protocols based on the low-power wide-area network (LPWAN) paradigm

have very limited data rates. For example, LoRa has a maximum data rate of 37.5 kbps due to duty cycle and other limitations [131]. Moreover, IoT nodes are severely power-constrained. Thus, on one hand, we want the nodes to send their DNN parameters to the aggregator as fast as possible, to ensure faster FML convergence. On the other hand, we want to keep energy consumption to a minimum to prolong the sensor's lifetime. Although existing literature – discussed in Section 2.3.2 – has investigated energy-aware LoRa optimization [78, 79, 80] and FML techniques for IoT [81, 83, 84], a series of assumptions (e.g., OFDM-based transmissions, zero interference, single-node network) make prior work not entirely applicable to real-world IoT contexts.

To address the above challenges, the work described in Part II makes the following novel contributions:

• First, we reverse-engineer the LoRa physical layer (PHY) layer functionalities, including the procedures of packet modulation, demodulation, and preamble detection. Accordingly, we design the first functional software implementation of the LoRa PHY layer for Software Defined Radio (SDR) from scratch, and evaluate its functionalities in two different settings: i) the Colosseum wireless channel emulator, and ii) a real testbed with off-the-shelf USRP radios and commercial devices. The results demonstrate both effectiveness of our implementation, and its interoperability with LoRa commercial device

• We demonstrate a fundamental theoretical result for FML schemes under lossy wireless communication channels. By aggregating all the gradient components retrieved from the local DNNs, it is possible to approximate the descendent gradient scheme and derive a *steady-state bound on the global model errors*. Importantly, this bound can be related to the communication error rate experienced by each sensor, and forced to zero in case the learning rate gradually decreases with the number of updating rounds.

• We leverage the theoretical result to design `FedLoRa`, an optimization framework for efficient large-scale federated learning in LoRa wireless networks, in which we optimize the Resource Allocation scheme for transmitting the local models trained by the sensors. We formalize a Resource Allocation Problem for LoRa (LoRa-RAP), where the communication resources (spreading factor and transmission power) assigned to each sensor are optimized so as to minimize the FML round time and keeping energy consumption into account. The key intuition behind this result is to apply a load balancing logic to the resource allocation procedure. In other words, the objective is fairly distribute the network load over the available Spreading Factors (SFs). Usually, nodes sharing the same SF must transmit in a sequential fashion to avoid any possible message collision. Thus, allocating most of the nodes

to the same SF results in a long FML round time. Conversely, we spread the nodes over different SFs, so that many simultaneous transmissions can happen at once. This leads to a more efficient usage of the network, and thus, to a shorter FML round time.

• We consider a state-of-the-art DNN for RFP [122], and leverage on our Long Range (LoRa) SDR implementation to investigate the `FedLoRa` performance with different RF configuration scenarios on Colosseum. Moreover, we also validated the effectiveness of `FedLoRa` with real-world measurements with a LoRa gateway and a LoRa node. Our results show that `FedLoRa` reduces the FML round time by up to about 35% with respect to baselines.

Chapter 5 focuses on the description and evaluation of our SDR LoRa implementation, while Chapter 6 describes the actual implementation of `FedLoRa`.

Chapter 5

Dissecting and Implementing LoRa on Software-Defined Radios from scratch

5.1 Background on LoRa

LoRa (short for *Long Range*) is a LPWAN proprietary protocol owned by Semtech. LoRa is based on the chirp spread spectrum (CSS) modulation technology, and supports reliable low data-rate transmissions over long distances, ranging from 1-2 to 10 (and possibly more) kilometers. The actual transmission range and data-rate strongly depend on the SF setting. The SF is an important spectral parameter, and roughly corresponds to the number of chips per bit transmitted. Higher SFs yield a longer transmission range, at the expense of a lower bit-rate, and vice-versa. LoRa specifies the PHY only. As such, it lacks link-layer and networking functionalities, which are instead defined by the LoRaWAN protocol from Semtech. Typical LoRa networks are arranged in a star-of-stars topology, where few LoRa *gateways* collect data transmitted by the LoRa *nodes*. The received data can eventually be forwarded to the Internet thanks to the networking functionalities implemented by LoRaWAN.

LoRa Regional Parameters. LoRa operates in the sub-GHz bands of the Industrial, Scientific, and Medical (ISM) spectrum, according to specific regional frequency plans: the **EU433** and the **EU863-870** bands for Europe, the **US902-928** band for US, and the **AS923** band for Asia. Another region-specific parameter is the supported bandwidth: European countries usually support a single bandwidth of 125 kHz, while US allow the usage of both 125 and 500 kHz. The maximum supported data-rate is influenced accordingly,

as a bigger bandwidth guarantees higher transmission rates.

LoRa Transmission Parameters. LoRa supports six different SFs ranging from 7 to 12. However, a SF equal to six is also allowed in some implementations. LoRa also supports up to three different bandwidth configurations of 125, 250, and 500 kHz, respectively. Both parameters can be set to reach the desired trade-off between data rate and reliability. Indeed, higher SFs and smaller bandwidths increase the sensitivity and robustness of the receiver, while lower SFs and bigger bandwidths maximize the transmission data-rate.

Finally, the robustness of LoRa communications is further boosted by the usage of Forward Error Correction (FEC) techniques. LoRa supports four different Coding Rate (CR) values, according to the formula $4/(4+n), n \in \{1,2,3,4\}$, where n is the number of redundant information bits. A bigger n increases the data protection, but negatively impacts the effective transmission rate.

5.1.1 Chirp Spread Spectrum (CSS) Modulation

LoRa implements a CSS modulation, which has been demonstrated to be very robust against in-band or out-band interference, which can be very critical when operating in ISM bands. In particular, LoRa employs an M-ary modulation scheme based on chirps [132]. Basic chirps are constant envelope signals whose frequency is linearly modulated sweeping from f_{min} to f_{max} (up-chirp), or from f_{max} to f_{min} (down-chirp). Chirps are cyclically-shifted to produce different symbols, and this cyclical shift carries the information. A symbol, whose length is divided in K equal time intervals called chips, can be cyclically shifted from 0 to $K-1$ positions. The reference position is given by the un-shifted (base) symbol at the beginning of the LoRa frame, which is also used for building the frame preamble.

For a given bandwidth $B = f_{max} - f_{min}$, the symbol time depends on the SF parameter, which defines two modulation features: (i) the time duration of each chirp (or, equivalently, the slope of the linear frequency sweep), which is given by 2^{SF} chip intervals; and (ii) the number of raw bits encoded by that symbol, equal to SF. The Data Rate (DR) thus depends on the bandwidth B in Hz, the SF and the Coding Rate (CR) as:

$$DR = SF \cdot \frac{B}{2^{SF}} \cdot CR \tag{5.1}$$

where $1/B$ is the chip interval, the factor $B/2^{SF}$ provides the symbol rate and the coding rate $CR = 4/(4+RDD)$ depends on the number of redundancy bits (RDD, from 1 to 4) used for Hamming code forward error correction.

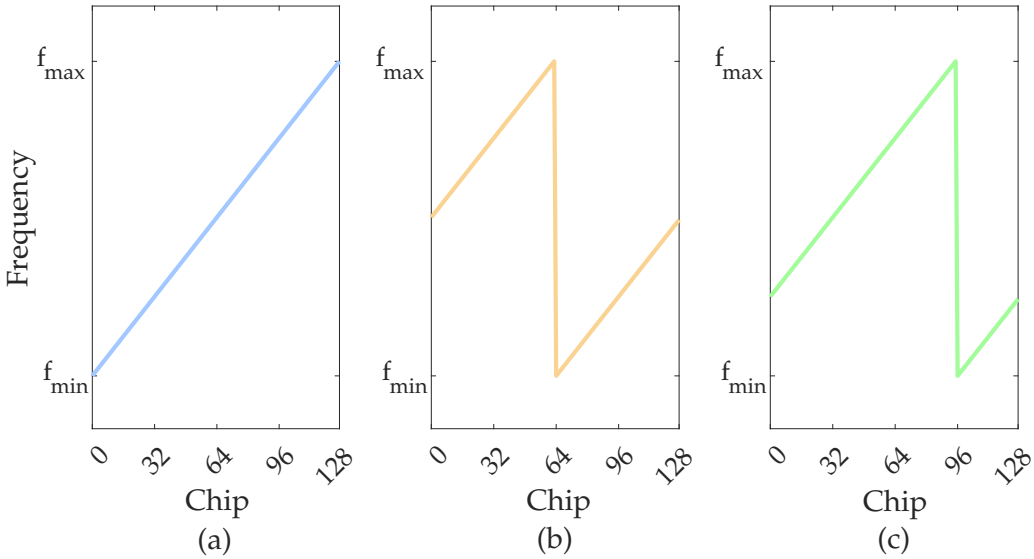


Figure 5.1: Instantaneous frequency of three upchirp signals for $SF = 7$.

The bandwidth can be configured as 125kHz, 250kHz and 500kHz (typically 125kHz is used in the 868MHz ISM band).

Fig. 5.1 shows the modulating signal used for (i.e. the instantaneous frequency of) a basic upchirp and two examples of circular shifts obtained for $SF = 7$: the basic upchirp can be shifted to represent up to 2^{SF} symbols, each encoding SF bits. The blue line (a) is the basic upchirp and encodes symbol $M = 0$; the orange line (b) encodes the symbol $M = 64$, while the green line (c) encodes the symbol $M = 96$. Moreover, the symbol time can be calculated as $T = 128T_c$.

The instantaneous frequency of an unmodulated (base) LoRa chirp can be written as:

$$f_i^{(0)}(t) = -\mu\frac{B}{2} + \mu\frac{B}{T}t \tag{5.2}$$

where $\mu = +1$ gives an *up*chirp and $\mu = -1$ a *down*chirp, $T = 2^{SF}T_c$ is the symbol time and $T_c = 1/B$ the chip duration, $0 \leq t < T$. There are $K = 2^{SF}$ possible symbols, each representing a cyclic shifted version of the base upchirp. The instantaneous frequency of symbol k is thus given by:

$$f_i^{(k)}(t) = \begin{cases} +\mu\frac{B}{2} + \mu\frac{B}{T}(t - kT_c) & 0 \leq t < kT_c \\ -\mu\frac{B}{2} + \mu\frac{B}{T}(t - kT_c) & kT_c \leq t < T. \end{cases} \tag{5.3}$$

The LoRa preamble starts with several repetitions of a base upchirp:

$$f_{pr}(t) = -\frac{B}{2} + B\left(\frac{t}{T} - \left\lfloor\frac{t}{T}\right\rfloor\right). \tag{5.4}$$

After several consecutive base upchirps, the preamble features two modulated symbols, called *sync* words, for network identification, and 2.25 downchirps which are useful for accurate synchronization. Overall, the preamble of a LoRa frame is constituted by a sequence of at least eight upchirps (including the two modulation sync words), followed by two and a quarter downchirps. Following the preamble, the payload header, the payload and an optional frame check sequence are transmitted by using the cyclically-shifted M-ary modulation.

5.1.2 Receiver Sensitivity

Another fundamental parameter in LoRa modulation is the receiver sensitivity, i.e. the minimum detectable signal strength. According to [133], the sensitivity for a LoRa receiver is:

$$S = -174 + 10\log_{10} BW + NF + SNR\,[dBm] \tag{5.5}$$

Where the first term is the thermal noise power in 1 Hz of bandwidth at a room temperature of 300K; BW is the transmission bandwidth; NF is the receiver noise figure, and is typically equal to 6 dB for popular transceivers models, such as Semtech SX1272 or Semtech SX1276 [132]. The last term is the Signal-to-Noise Ratio (SNR) value required at the receiver input for a successful demodulation, and depends, once again, on the receiver architecture, and on the SF, too. Typical values for SNR are reported in Table 5.1.

SF	7	8	9	10	11	12
$SNR[dB]$	-7.5	-10	-12.5	-15	-17.5	-20

Table 5.1: SNR thresholds for several SF values.

According to (5.5), the sensitivity is influenced by both the bandwidth and the SF settings. Hence, high SF values and small bandwidths achieve a high receiver sensitivity, at the expense of low data-rates. Conversely, faster LoRa communications are associated to low SFs, and larger bandwidths, but usually suffer of a low sensitivity. In other words, SF and BW can be properly tuned to achieve the desired trade-off between reliability and data rate.

5.2 SDR LoRa Transceiver

In this section, we present the design and implementation of our LoRa transceiver for SDR platforms, together with some preliminary simulation

results on the transceiver performance. All simulations have been run in Matlab. The receiver is based on three main modules: (i) the symbol detection module, which is responsible of identifying the symbols encoded in the received signal, once a new packet is correctly identified and synchronized; (ii) the synchronization module, which identifies the beginning of a new packet, and estimates the carrier and timing references used by the transmitter; (iii) the (optional) drift tracking module that compensates the clock drifts between transmitter and receiver for the correct demodulation of long frames. Apart from the building of LoRa symbols, we also implemented a pipeline of processing operations in the TX chain, including: parity check coding, whitening, shuffling and interleaving, and Gray coding.

5.2.1 Symbol Detection

LoRa demodulation can be implemented with very simple operations by mapping in each symbol the time interval at which the chirp jumps from f_{max} to f_{min} in a easily detectable frequency. In particular, our implementation works as follows. First, each received symbol is multiplied with the synchronized base down-chirp (at the same SF of the received signal). The result is a signal comprising only two frequencies (namely, $-k/T$ and $-B-k/T$, with T being the symbol time) which depend on the transmitted symbol k. Second, by down-sampling the signal at the rate B, both frequencies can be aliased to the same frequency $-k/T$. Finally, the signal is transformed in the frequency domain by means of an FFT. The symbol index k can be estimated by considering the position of the peak at the output of the FFT.

An interesting feature of LoRa is the quasi-orthogonality of signals transmitted at different SFs, as well as the robustness against external interference sources. Indeed, after the multiplication of the signal with the synchronized downchirp, the interfering signal is mapped into a noise over the whole frequency band of the signal, which prevents the correct identification of the symbol peak only for very low SIR (Signal-to-Interference Ratio) values. Conversely, when the interfering signal is a LoRa signal at the same SF, the receiver will observe multiple peaks at the output of the FFT: a maximum peak corresponding to the reference symbol, and two smaller peaks corresponding to two - partially overlapping - interference symbols. In such a case, if the reference signal is a few dB stronger than the interfering one, the symbol can be detected.

Fig. 5.2 shows the performance of our receiver in terms of Packet Error Rate (PER) in case of interference with an in-band sinusoidal (or narrowband) signal. The reference signal is given by the same packet with a payload of 20 bytes, which has been modulated at different SFs as indicated in the

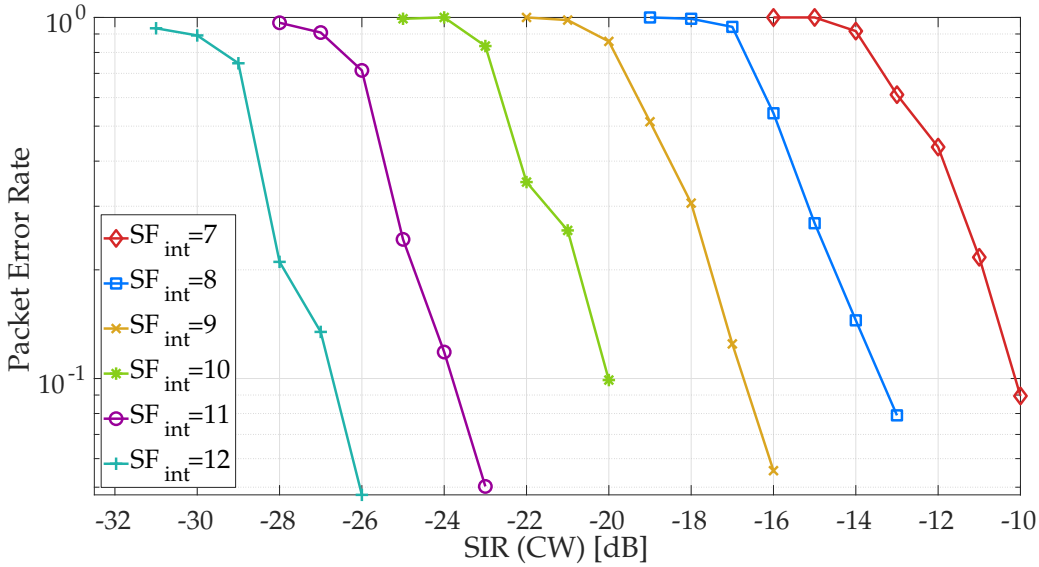


Figure 5.2: PER vs SIR in case of sinusoidal interference.

labels. From the figure, we observe that there are about 3dB of difference between the SIR values which guarantee a target PER for signals modulated at SF x and $x+1$. Fig. 5.3 quantifies the robustness of our receiver against interference sources generated by other LoRa signals. The figure depicts the PER versus the SIR values for a reference LoRa signal modulated at SF 7, with $B = 125$ KHz and a payload of 20 bytes, in case of interference with LoRa signals modulated under different SFs (as indicated in the labels). As expected, the interference due to a LoRa signal modulated at SF 7 is critical, because even a SIR value of 1 dB (often called *capture threshold*) can result in a PER equal to 0.1. For the other signals modulated at different SFs, it is evident that the signal orthogonality is not perfect: when the interfering signal is about 10dB stronger than the reference one, the PER is higher than 0.1. This SF-dependent threshold is often called *interference rejection* threshold. Obviously, the receiver has to correctly identify the SF used by the transmitter and the boundary of each symbol, as detailed in the next section.

5.2.2 Carrier and Time Synchronization

In order to recognize the symbol boundary, the receiver has to first identify the exact beginning of a preamble in time and in frequency, as well as the symbol duration corresponding to the correct SF.

Our synchronization mechanism is built by exploiting the preamble structure, which includes both upchirp and downchirp transmissions. The idea

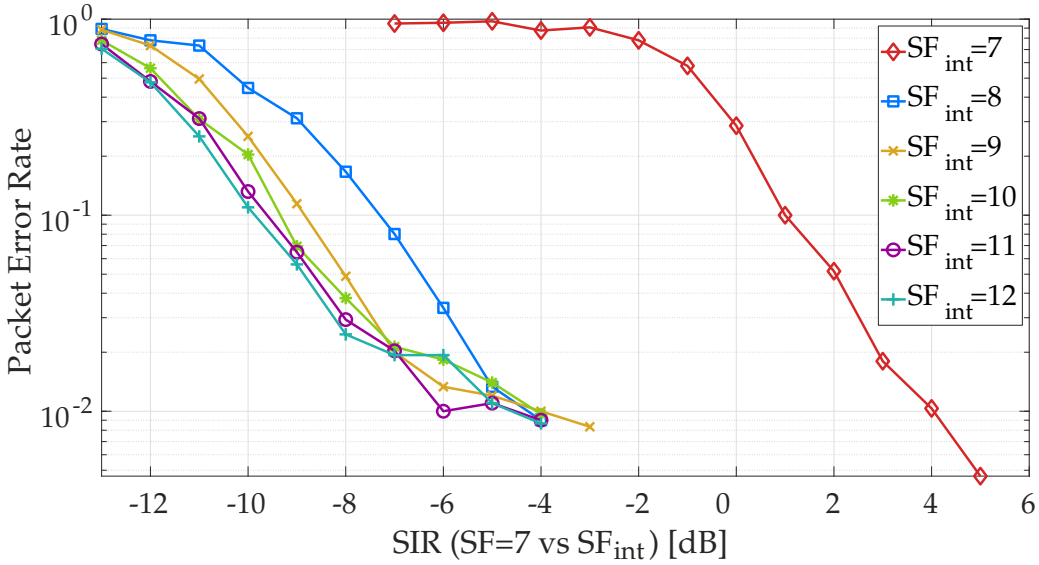


Figure 5.3: PER vs SNR values in case of interference with other LoRa modulated signals.

is mixing (i.e., multiplying) the received signal $f_{rx}(t)$ with the complex conjugate of a reference preamble upchirp $f_{pr}(t)$. Since the downchirp has the same absolute slope as the unsynchronized upchirp of the preamble, the frequency of the mixed signal $f_{mix,1}(t) = f_{rx}(t) - f_{pr}(t)$ changes over time as $\lfloor (t-\tau)/T \rfloor - \lfloor t/T \rfloor$, which is a square wave with values 0 and ± 1 and duty cycle $|\tau|/T$. It follows that the output of the mixer signal features tones at only two frequencies $\nu_1 = \text{CFO} - B\tau/T$ (when $t \geq \tau$) and $\nu_1 \pm B$ (when $t < \tau$). The same mixing mechanism can be applied to the last part of the preamble (constituted by downchirps), by multiplying the signal with base upchirps. The resulting signal has the same structure as the previous one with frequencies $\nu_2 = \text{CFO} + B\tau/T$ and $\nu_2 \pm B$. If ν_1 and ν_2 are available, the estimated carrier offset CFO_{est} can be computed as the average between ν_1 and ν_2, while the estimated timing offsets τ_{est} can be computed as $T/B \cdot (\nu_2 - \nu_1)/2$.

In order to identify a new preamble, the receiver:

1. samples the received signal $r(t)$ with a sampling frequency $f_s = B \cdot \text{OSF}$, i.e., Over Sampling Factor (OSF) times the nominal bandwidth of the signal, obtaining $r_n = r(n/f_s)$

2. multiplies (mixes) a window of $N = K \cdot \text{OSF}$ samples with a base downchirp, obtaining:

$$z_n = r_n \exp(\jmath\pi(n/\text{OSF} - n^2/(N \cdot \text{OSF}))) \tag{5.6}$$

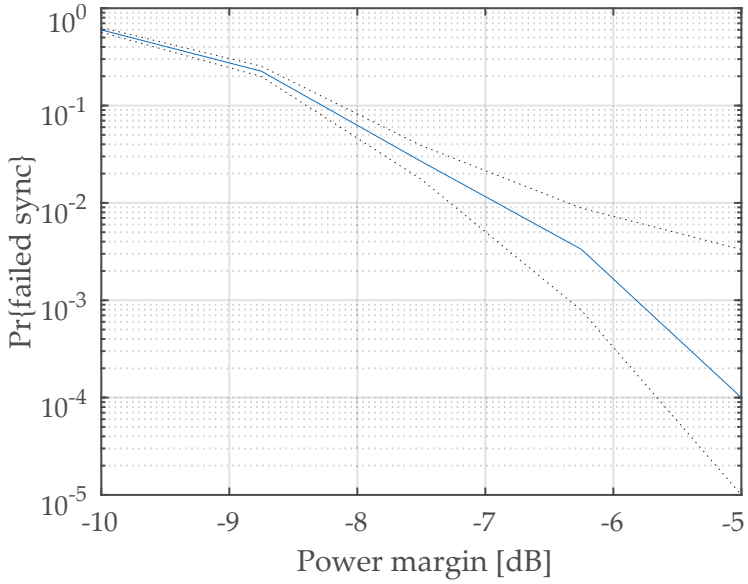


Figure 5.4: Probability of failing to detect a preamble.

3. computes the absolute value of the FFT of signal z_n:

$$\Gamma_k = \left| \sum_{n=0}^{N-1} z_n \exp(-\jmath 2\pi nk/N) \right| \tag{5.7}$$

4. estimates $\hat{k}$ as the position of the maximum in Γ_k;

Finally, if the estimated position $\hat{k}$ is detected continuously for a number of times (e.g., 3 consecutive windows), the receiver understands that an incoming preamble is received.

This procedure is executed continuously, for each possible SF, even when the demodulation of a frame is already in progress. This algorithm succeeds in detecting a preamble even several dBs *below* the sensitivity threshold. The probability of failing the detection of a preamble is shown in Fig. 5.4, when the power margin is computed as the difference between the received signal power and the receiver sensitivity. From the figure, it is evident that the receiver is able to detect a preamble even when the received power is below the sensitivity threshold of -121 dBm. Only when the margin is smaller than 7 dB, is the failure probability higher than 1%. Once the preamble is detected, fine estimates of ν_1 and ν_2 can be obtained as follows:

$$\hat{\nu} = \frac{B}{K} \left(\hat{k} + \frac{1}{2} \frac{\Gamma_{\hat{k}-1} - \Gamma_{\hat{k}+1}}{\Gamma_{\hat{k}-1} - 2\Gamma_{\hat{k}} + \Gamma_{\hat{k}+1}} \right) \tag{5.8}$$

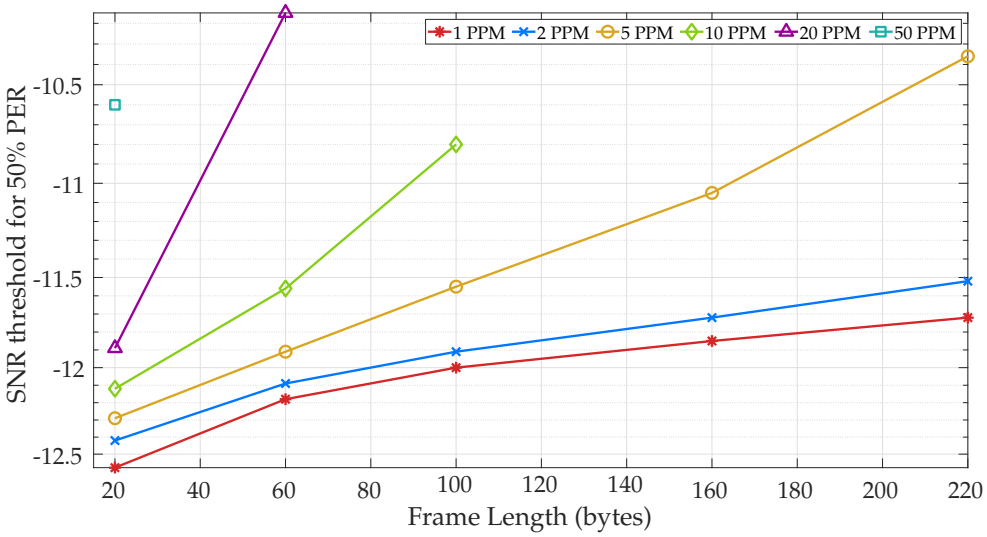


Figure 5.5: SNR values guaranteeing $PER \leq 0.5$.

where the values of $\hat{k}$ and Γ_k are obtained from the multiplication with the base downchirp in the first part of the preamble, and with its conjugate for the last 2.25 preamble symbols (for the last portion of the preamble, made of downchirps). Equation (5.8) is based on a parabolic interpolation around the maximum, and yields excellent results even for very low SNR values. For a reference signal transmitted at SF 7 and power margin of -10dB, the standard deviation of the CFO estimation error σ_{Cfo} over a bandwidth of 125 kHz is about $9 \cdot 10^{-4}$, while for a power margin of -5dB the ratio σ_{Cfo}/BW is reduced to $6.8 \cdot 10^{-4} \cdot BW$.

5.2.3 Impact of Clock Drifts

Once a preamble has been detected and initial estimates of CFO and τ have been performed, the LoRa receiver should periodically update these estimates for compensating the clock drifts between the transmitter and the receiver. Most SDR implementations have not addressed this issue, because SDR platforms usually rely on clocks whose errors are in the order of 1 PPM. However, off-the-shelf devices may experience inaccuracy can be as high as 17 PPM, due to low-cost crystal oscillator which may lead to synchronization problems (especially for long frames).

We quantified the SNR threshold leading to a PER lower than 0.5 for a receiver not implementing any clock tracking mechanism and for different packet transmission times. Specifically, Fig. 5.5 shows the SNR values as a function of the frame length, for a reference signal modulated at SF 7 and $B = 125$ kHz, for different clock stability values. Irrespective that the

reference signal is transmitted at SF 7 (and therefore the transmission times are accordingly minimized), packet reception is completely prevented for a clock frequency error higher than 10 PPM and a frame length higher than 60 bytes (PPM 50) or 100 bytes (PPM 20). Otherwise, packet reception works, but the errors are relevant.

A possible solution is extending the receiver architecture with the clock tracking module, such as the one we designed in [134]. Once a window of consecutive symbols is demodulated, the signal regenerated at the receiver is correlated with three different versions of the originally received signal: the one obtained considering a time offset equal to the initial estimate τ_{est}, and two other versions in which the offsets are equal to $\tau_{est} \pm 1/f_s$ (being f_s the sampling frequency), i.e., shifted by plus or minus one sample. The three correlation operations will result in three different maximum values, which will be interpolated using a quadratic function. Finally, the maximum of this parabolic interpolation will provide the time offset used in the current window for compensating the clock drift. In our implementation, we chose a time window of four symbols as a trade-off of accuracy and complexity.

5.2.4 Transmitter Implementation

At the transmitter side, we implemented parity check coding, whitening, shuffling, interleaving, and gray coding, to increase robustness towards synchronization errors or narrowband interference, which can be a serious issue for CSS-based modulations. While most of the transmitter side operations have been implemented as described in the LoRa modulation patent [133], a few implementation details required a low-level analysis of real signals, transmitted by LoRa commercial devices. Indeed, the patent leaves some ambiguities (for example about the row to column ordering in the interleaving, or the actual parity check matrix for coding at rate 4/5 and 4/8). Moreover, the initialization values for the CRC computation in the payload is not constant as specified in the patent, but rather seems to depend on the payload size. We found the initialization vectors for each possible packet size by exhaustive search. The resulting implementation has been demonstrated to be compatible with commercial devices.

5.3 Improving link reliability

Taking into account that each device experiences a specific Packet Delivery Rate (PDR) as a function of its channel and interference conditions, we designed and implemented two different solutions for improving link reliability:

i) the usage of on-demand retransmissions of corrupted frames, by means of an Automatic Repeat Request (ARQ) protocol, ii) the proactive transmission of additional frames, generated by means of rate-less coding schemes applied at a frame-level. Obviously, the first approach is effective in case a generic device is involved in sporadic transmissions, while the second one is useful when devices need to transmit a burst of multiple frames.

ARQ Mechanism. We implemented an optional Automatic Repeat Request (ARQ) reliability mechanism, to integrate and complement the FEC Hamming techniques natively implemented by the protocol. To support packet re-ordering and acknowledgment, as well as addressing, we reserve the first three bytes of the LoRa packet payload for the **Destination**, **Source**, and **Sequence Number** fields. Our protocol has several steps: (i) Transmission of the actual data, (ii) Response from the receiver with a NACK message, (iii) Re-transmission of the missing packets, if needed, (iv) Transmission of a final ACK from the receiver, and (v) Termination of the communication phase. To avoid any deadlock, during (ii), if no response is received from the transmitter for a specified timeout, the receiver re-sends the NACK packet.

Rateless Coding. Rateless codes such as those described in RFC6330 work by exploiting the possibility of generating as many encoded frames as needed from a burst of k frames at the transmitted side. Indeed, thanks to the coding scheme, the receiver is able to decode an exact copy of the entire burst of k frames from any subset of $k+\nu$ successful received (i.e. non-erased) encoded frames. The required reception overhead $\epsilon = \nu/k$ is usually in the order of a few percentage points. An interesting feature of this solution is that no downlink channel is required as a feedback to transmitter. Being downlink bandwidth limited in LoRaWAN networks, such an approach can be useful in many practical scenarios. We implemented a signalling mechanism from the network server to devices, for notifying the device-specific PDR. On the basis of this value and a target probability γ of successful delivery for a burst of k frames, each device involved in the transmission of data bursts computes the number k' of coded frames to be generated from each group of k frames, as:

$$k' : \sum_{l=k+\nu}^{k'} \binom{k'}{l} PDR^l \cdot (1 - PDR)^{k'-l} \geq \gamma \tag{5.9}$$

The expected number of delivered frames is $k' \cdot PDR$. The usage of rateless coding increases the load offered to the network and, consequently, the PDR experienced by other devices. However, when the number of devices involved in the transmission of data bursts is limited or when the main source of frame losses is the channel, the load increment has a minimal effect on the PDR variations and a single iteration suffices for finding a stable k'.

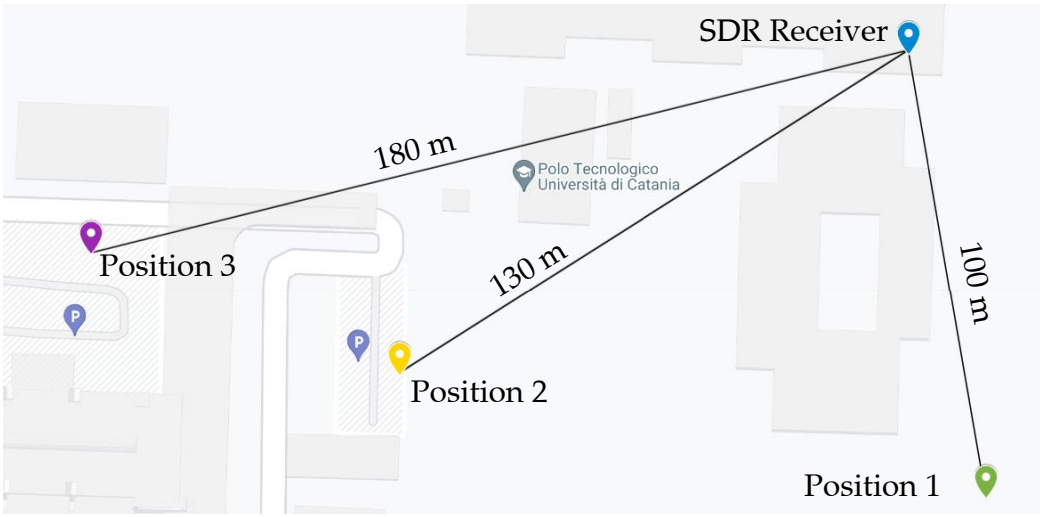


Figure 5.6: Map of the testbed locations.

5.4 Experimental results

We validated the implementation of our LoRa transceiver by running some compatibility tests with commercial LoRa devices. We set one SDR platform running the LoRa transceiver in our laboratory at the University of Catania, and deployed the devices in outdoor in locations characterized by partially obstructed links as depicted in Fig. 5.6. In position 1, where nodes were perfectly visible, we got a Packet Delivery Rate (PDR) equal to 1 for each available SF; in position 2, some errors were found at SF 7, while in position 3, the PDR was lower than 0.1 at SF 7 and about 0.5 at SF 10.

In order to study the performance of our prototype, we also integrated the LoRa transceiver on the Colosseum testbed, where multiple coordinated SDR platforms and channel emulators are available. An important feature of the testbed is the possibility of controlling the network scenarios, in terms of channel models between nodes, as well as the transmission patterns of coexisting devices. Indeed, the transmission attempts of independent SDR platforms can be synchronized or shifted of a desired time interval, for characterizing specific interference effects generated by multiple interference sources. Moreover, the testbed allows the analysis of low-level signals collected by independent receivers, enabling many experimental studies on receiver architectures.

5.4.1 Testbed Description

We evaluated SDR-LoRa on Colosseum, the world's largest network emulator [19]. Colosseum is a wireless emulator with 128 so-called Standard

Radio Nodes (SRNs), i.e. dedicated hardware nodes each equipped with a NI/Ettus USRP X310 SDR. Each SRN can host and run user-defined Linux Containers (LXCs), to offer a high degree of freedom in the customization and usage of the underlying hardware. The SRNs are all linked together by the Colosseum Massive Channel Emulator (MCHEM). The latter is made up of several FPGA modules and is thus able to process the radio signals through Finite Impulse Response (FIR) filters. The MCHEM can therefore emulate the effects of real-world wireless RF channels, such as attenuation, propagation delay, fading, and multipath. Another fundamental module of the Colosseum architecture is the RF scenario server. A Colosseum scenario is a collection of wireless links between several SRNs, where each link is defined by digital channel taps. When a scenario is activated on the emulation platform, these channel taps are fed to the MCHEM at run time. Our results are based on a custom LXC image. This container includes the developed SDR implementation, together with the libraries and system tools needed to run our code.

5.4.2 Results for Single Link

We run a first set of experiments with a single LoRa link, with two SDR nodes acting as a transmitter and receiver, connected by means of the MCHEM channel emulator. The selected RF scenario is characterized by a noise power with $\sigma_n = 3.5 * 10^{-8}$ and a tunable Path Loss.

Channel Attenuation	45 dB	50 dB	55 dB	60 dB
PDR	1	0.99	0.74	0.20

Table 5.2: PDR vs Channel Loss.

Table 5.2 shows the PDR at the receiver, as a function of different channel attenuation values[1], when the receiver transmits at SF 7, with a bandwidth of 125kHz, a payload size of 50 bytes, and a fixed normalized amplitude of the signal equal to 1. The link also suffers of additional power losses, due to the connectors between SDR nodes and the wired channel emulator. Since these losses cannot be easily quantified, we experimentally found the channel attenuation which results in an SNR lower than the reception threshold. Indeed, the PDR is lower than 1 (about 75%) for a channel attenuation of 55dB and lower than 15% when the channel attenuation is increased to 60dB.

For a channel attenuation value of 56dB (leading to an SNR value lower than the reception threshold at SF 7), we run further experiments at different

[1]This choice is useful to emulate several transmission distances. In fact, in free-space communications, a larger path-loss corresponds to a longer communication distance.

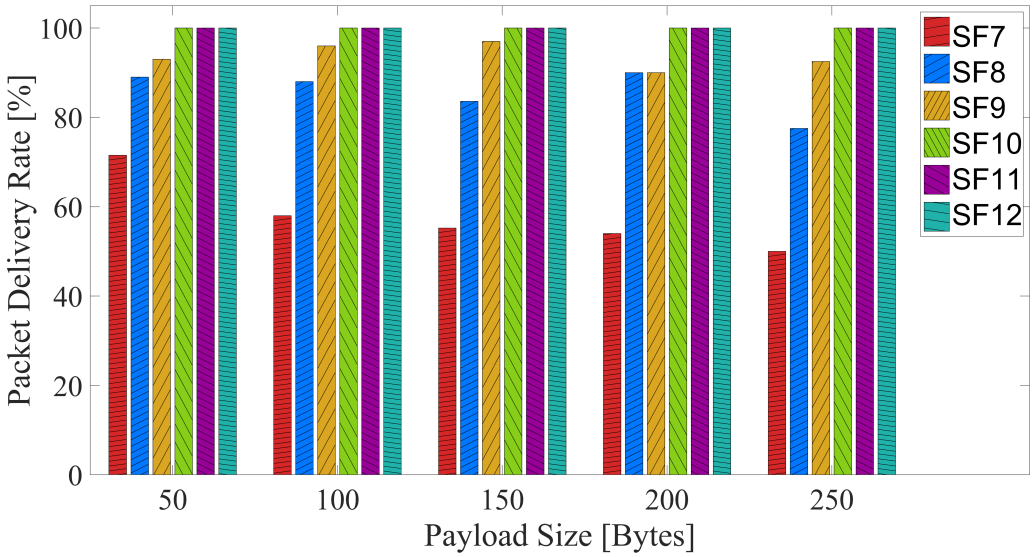


Figure 5.7: PDR vs the packet Payload Size.

SFs and for different payload sizes. We measured the PER when transmitting a fixed amount of data (namely, 10000 bytes), which corresponds to a different total number of frames. The results are summarized in Fig. 5.7. From the figure, it is evident that the SNR value at the receiver is higher than the minimum reception thresholds for SFs 10, 11 and 12. For the other SFs, for which the correct demodulation of the symbols cannot be guaranteed, the PDR is affected by the payload size, with a general degradation as the payload size increases (although there are a few exceptions due to the confidence interval of the results). However, the PDR does not decrease as an exponential function (with a base lower than 1) of the number of symbols in the frame, which are roughly proportional to the payload size. This suggests that synchronization problems can be the main cause of packet losses for the SNR values considered in the experiments.

On top of error-prone links, we also tested our schemes for improving link reliability. Fig. 5.8 quantifies the total amount of time needed to completely transfer 10000 bytes of data to the receiver, for a channel attenuation of 56 dB. The time takes into account the total interval required for transmitting all the frames, including selective retransmissions of corrupted frames. No duty cycle has been considered. Note how higher SFs exhibit a bigger transmission time despite the fact that the links are more robust: indeed, the higher is the SF, the higher is the number of chips in a single LoRa symbol, and thus, the bigger is the transmission time of each frame.

An alternative approach for improving the link reliability is exploiting rate-less coding. Table 5.3 summarizes the number of coded frames required

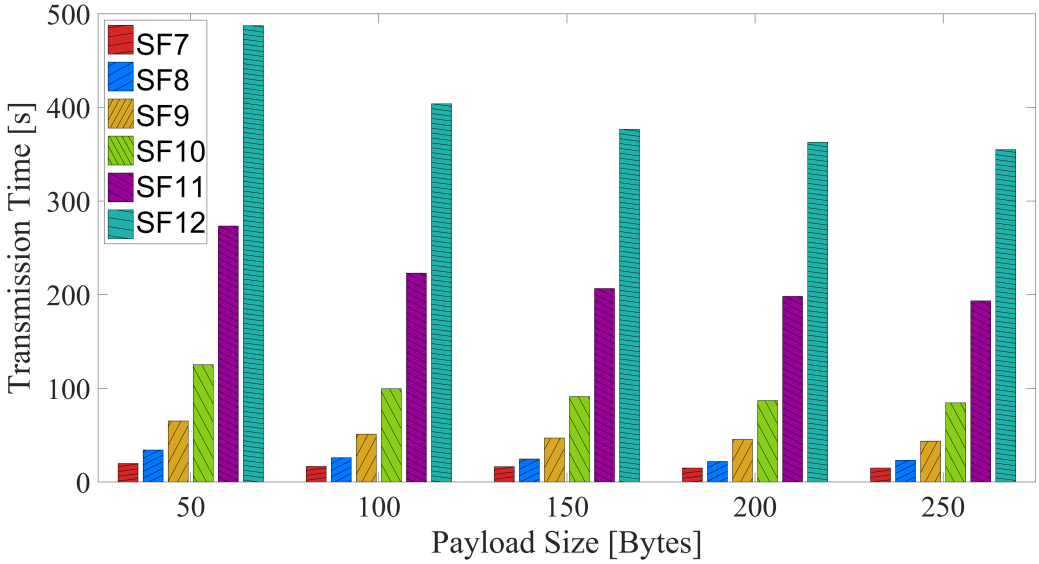


Figure 5.8: Transmission vs the packet Payload Size.

for guaranteeing a delivery probability $\gamma \leq 1-10^4$ and the relevant transmission time, for the same scenario of a data transfer of 10000 bytes, a channel attenuation of 56 dB, and a payload size equal to 100 or 250 bytes. The table refers to the usage of Raptor-Q codes, assuming to use $\nu = 5$ (which guarantees according to RFC6330 a probability lower than 10^{-10} that the frame decoding will fail). For a given PDR, the number of additional packets generated by the usage of rate-less coding is obviously greater than the one resulting from selective retransmissions, but the difference gets lower as the the total number of packets to be sent increases (for example, for 100 bytes and $SF = 9$ we transmit 125 packets, with a total number of expected transmissions equal to 108 packets).

	100 bytes			250 bytes		
SF	PDR	k'	Time	PDR	k'	Time
7	0.58	220	41.6 s	0.48	129	46.3 s
8	0.84	142	48.0 s	0.78	72	46.5 s
9	0.92	125	76.9 s	0.92	57	65.4 s

Table 5.3: Overheads of Raptor-Q coding

5.4.3 Results for Multiple Links

We run a second set of experiments for studying the impact of interference between multiple coexisting LoRa links. First, we analyzed the imperfect orthogonality of different SFs. We set-up a reference LoRa link working at SF 7 and transmitting frames with a payload of 50 bytes, and an interfering

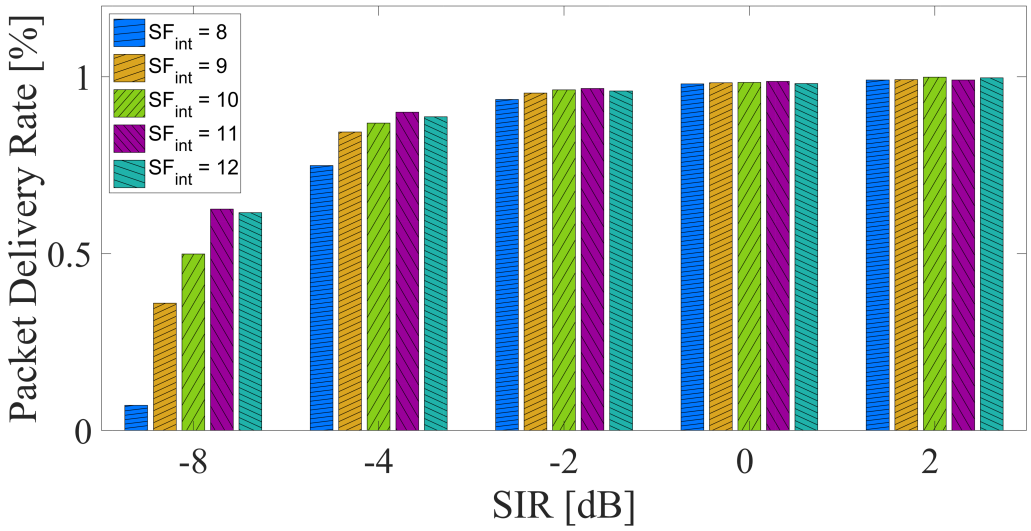


Figure 5.9: PDR vs SIR for a reference link at SF 7, in presence of collisions with different SFs.

node working at a SF different from SF 7. The interfering node has been configured for transmitting continuously, without any duty cycle, in order to guarantee that all the frames transmitted by the reference link overlap in time with the interfering node, thus leading to collisions. Fig. 5.9 shows the PDR measured in our experiments, when the transmitted frames collide with interfering signals at different SFs, as a function of the SIR. The limits of imperfect orthogonality are quite evident: for SF 8, it is enough a SIR value of -4 dB for damaging the frames and reducing the PDR to about 75%. Although the interference generated at SF 8 is the one reducing the PDR the most, we also notice that the SIR threshold which prevents a PDR equal to 1 is about the same for all the SFs used by interfering node.

The large rejection thresholds of the interfering signals is due to the receiver operation. When multiplying the received signal with the downchirp at SF 7 and computing the FFT, such an interfering power is spread on the whole bandwidth of 125kHz, while the power of the reference signal is seen as a narrow peaks whose position within the band corresponds to the coded symbols.

Obviously, a possible cause of symbol detection error due to a low SIR value is the presence of multiple interfering signals. Fig. 5.10 quantifies the PDR achieved by the reference link at SF 7, when multiple interfering nodes are active at SF 8 and for different normalized amplitude values of the reference signals. We let this amplitude value vary in the range 0.4-0.8, while the normalized amplitude of the interfering signals is fixed and equal

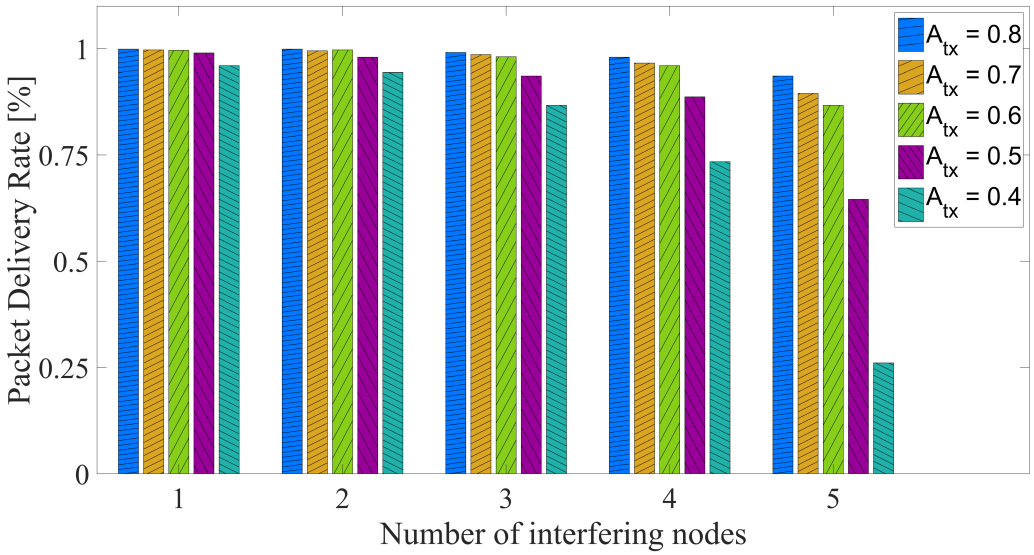


Figure 5.10: PDR of a reference link at SF 7 as a function of the number of interfering nodes at SF 8.

to 0.5.

We also considered the effects of collisions generated at SF 7. In principle, signals transmitted at the same SF are not orthogonal and should prevent the correct reception of the colliding frames. However, the capture thresholds of LoRa are very low: it is enough that the reference link has a SIR of a few dB (typically, 3 dB) to allow the correct demodulation of the frame. We repeated the experiment with multiple interfering nodes by configuring the interfering signal at SF 7. Fig. 5.11 shows the PDR observed as the number of interfering nodes increases and when the transmission power of the reference link is $6dB$ stronger than the interfering ones. From the figure we observe an interesting phenomenon: the PDR is almost one even in presence of 4 nodes, when the SIR is 0 dB, and slightly lower than 1 in case of 5 interfering nodes (for a negative SIR). This is due to the receiver operation, according to which the interference generated by multiple transmitters at SF 7 is not additive.

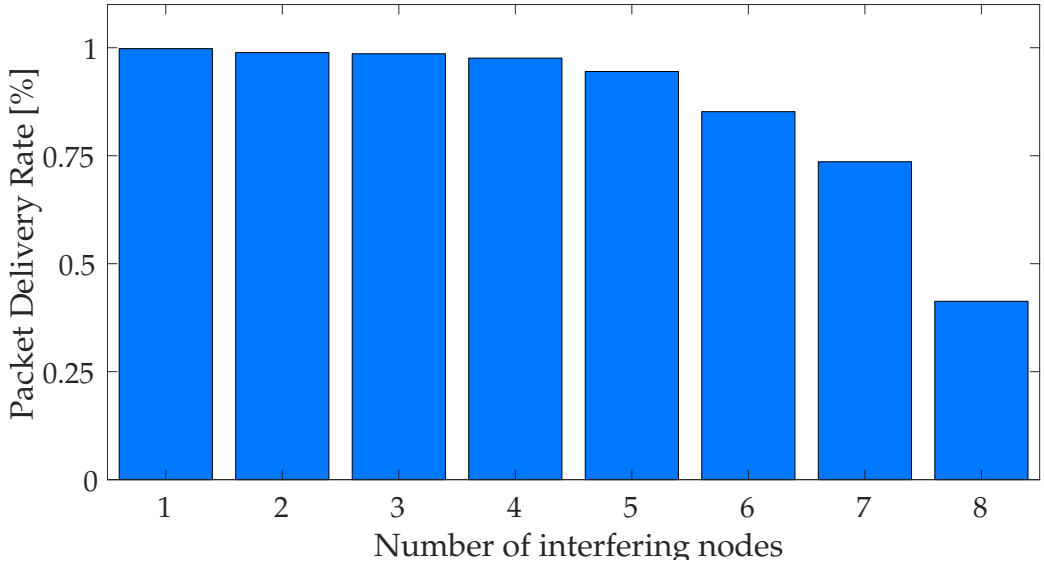


Figure 5.11: PDR of a reference link at SF 7 as a function of the number of interfering nodes on the same SF.

Chapter 6

Fast and Efficient Federated Learning in the Internet of Things Through LoRa Resource Optimization

6.1 Introduction to Federated Learning

In the following, we introduce a conceptual description of a Federated Learning problem. A typical Federated Learning system includes an aggregation server and several users. Each user can build its own local model starting from a local data set, and eventually send the result to the aggregation server, which, in turn, produces the final global model by averaging the local ones.

A Federated learning procedure unfolds across three different phases:

1. **Task definition**: During the first step, the central server defines the task, and, accordingly, the type of data and the relevant features to be taken into account. The server is also responsible of initializing the global model;

2. **Local Training**: In this phase, each device first receives the global model. At this point, each user makes use of its own data set to produce a local update of the global model. After the local training phase, the updates are eventually sent to the central server;

3. **Global Aggregation**: in the third and last phase, the server receives the new local models from the users and produces the global update through a weighed average of the local models. More specifically, each

local model is weighed by the local data set size; indeed, a larger number of examples produces more accurate local models. Accordingly, such models get more importance during the global aggregation.

To achieve the desired accuracy, phase 2 and 3 are repeated several times. The algorithm stops when convergence is reached or, alternatively, after a maximum number of iterations is executed.

6.2 The Federated Learning Scheme

We defined a federated learning scheme suitable for LoRa networks, in which communication links have limited bandwidth and are error prone. We assume that N sensors (also called clients) work each on a local data-set to be used for training a model characterized by a vector of d parameters $\boldsymbol{w} \in \mathbb{R}^d$. The goal of the training process is to find $\boldsymbol{w}_*$ able to minimize a loss function $f(\boldsymbol{w})$ on the whole data-set $D = \cup_{i=1}^{N} D_i$, being $D_i = \{x_{i,l}\}_{l=1}^{|D_i|}$ the data-set available to the i-th client, whose total number of data samples $x_{i,l}$ is equal to $|D_i|$.

Our approach is based on the federated averaging algorithm described in the previous section. The process is organized in a set of communication rounds, in which each client locally executes a single update epoch before sending back the local gradients to the central server. More into details, at a generic round k the process works as follows:

Broadcasting phase: the central server transmits in downlink the current vector $\boldsymbol{w}^k$ to all clients.

Local update: each client computes the gradient $\nabla f_i(\boldsymbol{w}^k)$ of the loss function on a random sub-set D_i' of its local data, being $f_i(\boldsymbol{w}^k) = \sum_{l=1}^{|D_i'|} F_i(\boldsymbol{w}^k, x_{i,l})$ and $F_i(\boldsymbol{w}^k, x_{i,l})$ the loss of the current model on the training sample $x_{i,l}$. The gradient vector is sent back to the central server by transmitting, in general, multiple LoRa frames. The frames are organized by including randomly ordered components, together with the relevant model index, in order to allow an exact identification of the gradient updates at the server.

Central update: The loss function on the complete data-set is given by $f(\boldsymbol{x}) = \sum_{i=1}^{N} f_i(\boldsymbol{x})$, whose gradient can be computed by summing the contributions sent by each client. However, at the end of each communication round, some coefficients of the gradient vectors $\nabla f_i(\boldsymbol{w}^k)$ may be lost. Since we assume that the gradient components are randomly ordered to form the packets, multiple losses due the corruption of one packet are uniformly spread on the total number of model dimensions d. Different options are available for dealing with these missed updates. For example, the server could completely

ignore the partial updates sent by the clients for which some coefficients are lost. However, this could bias the learning process by focusing on the data available at the clients experiencing the best communication links. A different approach is the utilization of all the data collected by the server. Let $\nabla f_i^0(\boldsymbol{w}^k)$ the gradient vector for a generic client i, where any missing coefficient $[\nabla f_i(\boldsymbol{w}^k)]_l$ is replaced by zero. The model update can be implemented as:

$$\boldsymbol{w}^{k+1} = \boldsymbol{w}^k - \eta_k \sum_{i=1}^{N} \nabla f_i^0(\boldsymbol{w}^k) \tag{6.1}$$

6.2.1 Bounding the Error

In this section we investigate the FML scheme in case of losses on the communication links. While discarding gradients with missing components can be seen as an optimization based on stochastic gradient, our scheme represents a perturbation of the descendent gradient under time-varying noise components. Although the vector resulting from the aggregation of the correctly received components will deviate from the true gradient along some dimensions, intuitively we can expect that the following updates will be able to compensate these loss-induced random deviations.

Let $\boldsymbol{\xi}^k = \sum_{i=1}^{N} \boldsymbol{\xi}_i^k = \sum_{i=1}^{N} \left[\nabla f_i(\boldsymbol{w}^k) - \nabla f_i^0(\boldsymbol{w}^k)\right]$ be the vector representing the sum of all the missing gradient components. We can express the model update rule as a perturbation of the true descendent gradient:

$$\begin{aligned} \boldsymbol{w}^{k+1} &= \boldsymbol{w}^k - \eta_k \sum_{i=1}^{N} \nabla f_i(\boldsymbol{w}^k) + \eta_k \boldsymbol{\xi}^k = \\ &= G_k(\boldsymbol{w}^k) + \eta_k \boldsymbol{\xi}^k \end{aligned} \tag{6.2}$$

where $G_k(\boldsymbol{w}^k)$ is the update due to the true gradient. Under the assumptions that the loss function is a smooth function, i.e. $\exists L > 0 : \forall \boldsymbol{x}, \forall \boldsymbol{y} \in \mathbb{R}^d, \|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\| \leq L\|\boldsymbol{x} - \boldsymbol{y}\|$, and $\eta_k \in]0, 2/L[\ \forall k$, we can derive some conditions for characterizing the updates of the global model over time [135]. First, in case the loss function is also strongly convex, there is a global minimum point $\boldsymbol{w}_*$ in which the gradient is null. For a smooth convex function, it can be shown that $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$:

$$\langle \nabla f(\boldsymbol{y}) - \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle \geq \frac{1}{L} \|\nabla f(\boldsymbol{y}) - \nabla f(\boldsymbol{x})\| \tag{6.3}$$

where $\langle \nabla f(\boldsymbol{y}) - \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle$ represents the inner product between the vectors $\nabla f(\boldsymbol{y}) - \nabla f(\boldsymbol{x})$ and $\boldsymbol{y} - \boldsymbol{x}$. Let $e_k = \|\boldsymbol{w}^k - \boldsymbol{w}_*\|^2$ the squared distance

between the current model vector and the point minimizing the loss function $\sum_{i=1}^{N} f_i(\boldsymbol{w})$[1]. For a Lipschitz loss function, the module of the difference between the loss function at $f(\boldsymbol{w}^k)$ and the minimum loss function at $f(\boldsymbol{w}_*)$ is proportional to the square root of e_k. By considering the inequality in 6.3, we can derive :

$$\begin{aligned}\|\boldsymbol{w}^{k+1} - \boldsymbol{w}_*\|^2 &= \|\boldsymbol{w}^k - \eta_k \nabla f(\boldsymbol{w}^k) + \eta_k \boldsymbol{\xi}^k - \boldsymbol{w}_*\|^2 = \\ &= \|G_k(\boldsymbol{w_k}) - \boldsymbol{w}_*\|^2 + \eta_k^2 \|\boldsymbol{\xi}^k\|^2 + \\ &- 2\eta_k \langle G_k(\boldsymbol{w}^k) - \boldsymbol{w}_*, \boldsymbol{\xi}^k \rangle \\ &\leq \|\boldsymbol{w}^k - \boldsymbol{w}_*\|^2 - 2\eta_k(1/L - \eta_k/2)\|\nabla f(\boldsymbol{w}^k)\|^2 + \\ &+ \eta_k^2 \|\boldsymbol{\xi}^k\|^2 - 2\eta_k \langle G_k(\boldsymbol{w}^k) - \boldsymbol{w_*}, \boldsymbol{\xi}^k \rangle\end{aligned} \tag{6.4}$$

from which the error sequence obey to:

$$e_{k+1} \leq \; e_k + \eta_k^2 \|\boldsymbol{\xi}^k\|^2 - 2\eta_k \langle G_k(\boldsymbol{w^k}) - \boldsymbol{w_*}, \boldsymbol{\xi}^k \rangle \tag{6.5}$$

In other words, the tracking error is bounded by a discrete time dynamical system, where the shift between the real and true gradient behaves as a disturbance input. The average value of the disturbance at each step k is given by $E[\eta_k^2 \|\boldsymbol{\xi}^k\|^2 - 2\eta_k \langle G_k(\boldsymbol{w^k}) - \boldsymbol{w_*}, \boldsymbol{\xi}^k \rangle]$, which is equal to $\eta_k^2 E[\|\boldsymbol{\xi}^k\|]^2$ under the assumption that $E[\boldsymbol{\xi}^k] = \mathbf{0}$. This assumption may be justified by recalling that the overall gradient around $\boldsymbol{w}_*$ becomes negligible, and this, in turn, is true for every partial gradient in case the data is i.i.d. among the clients. In case the data is not i.i.d., the $\boldsymbol{\xi}^k$ will not be negligible, but the assumption will be still valid if the packet losses are equally distributed between clients. It follows that the error admits a steady-state bound, but it does not decrease to zero in presence of communication losses. For η_k decaying with k, we can force the disturbance to vanish.

Note that the error bound critically depends on the norm of the missing gradient components, which in turns is affected by the loss probability on the communication channel. Specifically, if the coefficient error rate $\mathrm{CER_i}$ is small, i.e. lower than 30% so that the number of coincident missing coefficients in vector pairs is negligible, we can approximate the square norm of the gradient perturbation as $\|\boldsymbol{\xi}^k\|^2 \approx \sum_{i=1}^{N} \|\boldsymbol{\xi}_i^k\|^2$. The square norm of the i-th term, in turn, may be approximated by the expected number of missing components times the mean square value of the i-th gradient contribution.

[1]Note that, in general, models based on multi-layer perceptrons do not have a strong convex loss function and therefore more than one local minimum exists. However, the following derivation can be generalized for pseudoconvex functions with multiple minimizers.

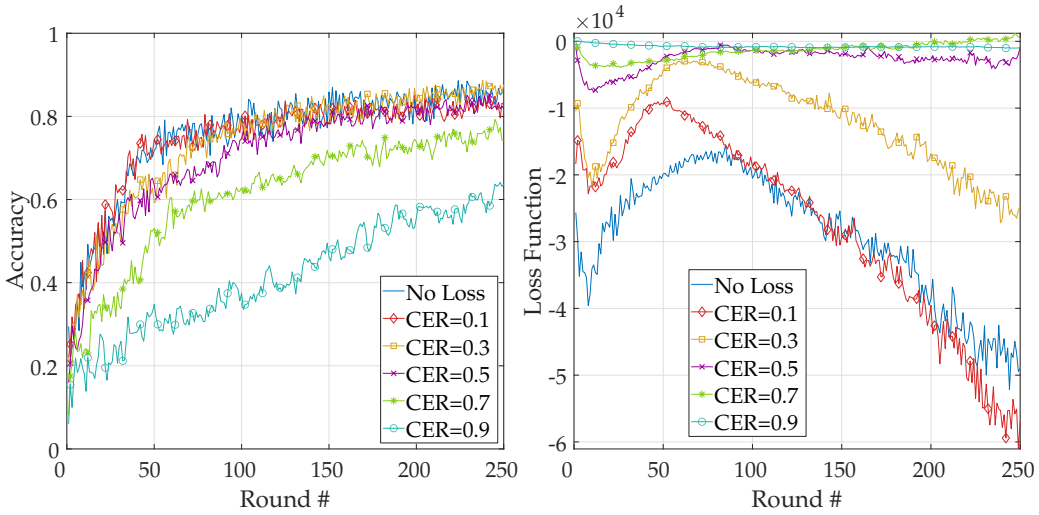


Figure 6.1: Accuracy and Loss of the proposed FML scheme vs CER for 9 nodes

This results in the following approximation for the perturbation term:

$$\begin{aligned} E[\|\boldsymbol{\xi}^k\|^2] &\approx \sum_{i=1}^{N} \|\boldsymbol{\xi}_i^k\|^2 \approx \\ &\approx \sum_{i=1}^{N} d \cdot \mathrm{CER}_i \cdot \frac{1}{d} \sum_{l=1}^{d} [\nabla f_i(\boldsymbol{w}^k)]_l^2 = \\ &= \sum_{i=1}^{N} \mathrm{CER}_i \|\nabla f_i(\boldsymbol{w}^k)\|^2 \end{aligned} \tag{6.6}$$

If we assume the gradient descent is convergent, we obtain the asymptotic approximate $E[\|\boldsymbol{\xi}^\infty\|^2] \approx \sum_{i=1}^{N} \mathrm{CER}_i \|\nabla f_i(\boldsymbol{w}_*)\|^2$, which goes to zero in case the data subset are i.i.d.

Figure 6.1 shows a numerical example of our federated learning scheme working with nine clients and under different coefficient error rates. The example refers to a scenario in which the model is given by 194,855 coefficients with not i.i.d data subsets. The model accuracy is minimally impacted by the CER up to 50% after 100 update rounds, while the convergence rate of the loss function, as expected by the previous argument, decreases as the CER gets higher.

6.3 The `FedLoRa` Framework

The main target of `FedLoRa` is to establish reliable, effective and energy-efficient federated learning in LoRa networks, taking into account the channel

and interference conditions of the sensors (i.e. the network level) and the acceptable errors on the global model over time (i.e. the learning level). We provide a walkthrough of `FedLoRa` below with the help of Figure 6.2. At the beginning, the architecture of the DL model $M = \boldsymbol{w}$ is shared with `FedLoRa` (**Step 1**). The DL model weights are then forwarded to the LoRa *nodes* by the LoRa *gateway* (**Step 2**). Then, the DL model size is fed to the LoRa Resource Allocation Problem (LoRa-RAP), formulated in Section 6.3.1. The LoRa-RAP takes as input some channel-related information, such as signal-to-noise ratio (SNR) and signal-to-interference (SIR) ratio, which are estimated experimentally through pilot transmissions (**Step 3**), and the desired CER on the model components. The PHY parameters are then sent to each node in the network through the LoRa *gateway* (**Step 4**). As regards the federated machine learning (FML) training, each node trains a local DL model in several subsequent *rounds*. Since nodes are not computationally powerful, the local model for node i, $M_i = \boldsymbol{w} - \eta \nabla f_i(\boldsymbol{w})$, is trained only with a sub-portion of the locally-available data D_i'. Each node then signals the weights of M_i by transmitting the gradient coefficients to the gateway. The gradients received from all the nodes are aggregated for updating the model weights, which are sent back to the nodes for the next round (**Step 5**).

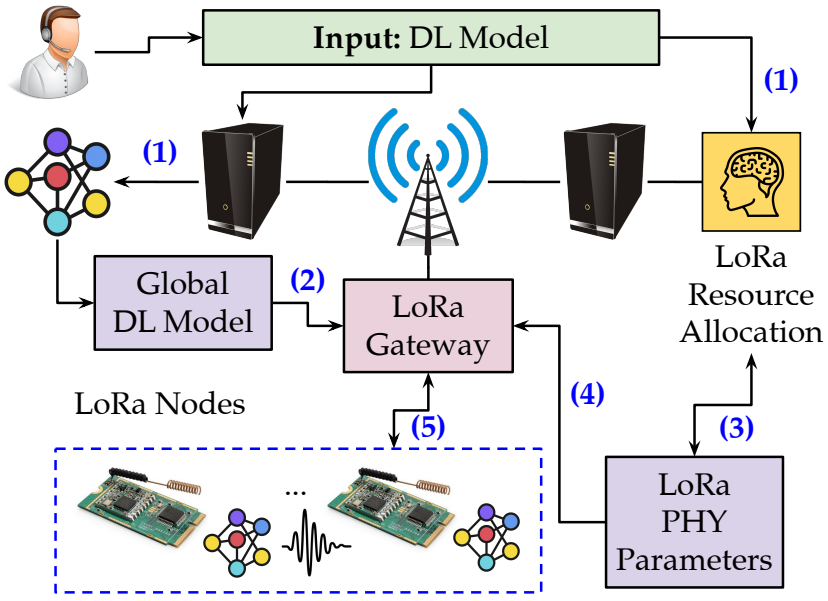


Figure 6.2: High-level overview of the `FedLoRa` framework.

6.3.1 LoRa-RAP Formulation

We assume that the network configuration can be optimized by dynamically adjusting the transmission power and Spreading Factor (SF) used by each sensor. Higher SFs are associated to more robust links, yet to lower data-rates (and, thus, to a longer transmission time). Moreover, higher values of transmission power guarantee better SNR at the receiver, but also imply a bigger energy consumption, and produce more interference to other nodes in the network. Other communication parameters such as the transmission bandwidth B and coding rate CR are not considered as network tunable parameters. In real LoRa applications the bandwidth is not configurable. For example, according to EU LoRa regulations, all SFs can only work on a bandwidth of 125 kHz (except for SF7). Moreover, We do not include the CR PHY parameter in the optimization as we assume a Line-of-Sight channel model (i.e. Rice fading), and a very limited effect of the coding gain employed in LoRa (1 dB at most, even when employing Soft-Input Soft-Output decoding [136]).

Since each sensor has to transmit a bulk of frames at each model update round, which results in a temporary congestion of the network, as a first optimization strategy we consider the possibility of polling sequentially the network nodes rather than using random access. Multiple polling rounds are executed in parallel on different SF, by exploiting the capability of LoRa modulation of rejecting signals modulated at different SF even for negative SIR values. By considering this access solution, we set the signal to noise (SNR) and interference (SIR) ratios which need to be provided to each node for guaranteeing the desired CER. For a reference node n, we assume that interfering signals are only due to other LoRa nodes transmitting at a SF different from SF_n (which are not perfectly orthogonal [137]) and that the channel is AWGN.

The goal of LoRa-RAP is to find the values of SF and transmission power for each node able to minimize the energy consumption, while also taking into account the aforementioned constraints. In the following, we only formulate the problem of energy minimization, and instead omit the problem of transmission time minimization, as the two problems are intertwined in LoRa. In fact, choosing the lowest possible value of SF achieves a shorter transmission time, which results in both an increased data rate and lower energy consumption.

We define $\mathcal{N}$ as the set of LoRa nodes in our reference scenario and $\mathbf{SF}$ as the set of available spreading factors. Moreover, ρ_n and SF_n are the transmission power and the spreading factor for a generic node n. We now model the energy consumption and the transmission time in the system.

First, the data-rate of a LoRa node n is equal to [132]

$$r_n = \mathrm{SF}_n \frac{B}{2^{\mathrm{SF}_n}} \frac{4}{4+\mathrm{CR}}$$

where SF_n is the SF assigned to node n. Accordingly, being s the size of the model coefficients (in bytes) to be transmitted to the central server, the energy consumption of node n is equal to:

$$E_n = \rho_n \frac{s}{r_n} = \rho_n s \frac{2^{\mathrm{SF}_n}}{B\mathrm{SF}_n} \frac{4+\mathrm{CR}}{4}$$

where we assume that the model size s is fixed for all the nodes[2]. We define the resource allocation problem as follows.

LoRa Resource Allocation Problem (LoRa-RAP)

$$\min_{\boldsymbol{\rho},\mathbf{SF}} \sum_{n\in\mathcal{N}} \rho_n s \cdot \frac{2^{\mathrm{SF}_n}}{B\mathrm{SF}_n} \frac{4+\mathrm{CR}}{4} \tag{6.7}$$

s.t.

$$\rho_{min} \leq \rho_n \leq \rho_{max} \ \forall n \in \mathcal{N} \tag{6.8}$$

$$\mathrm{SF}_n \in \{7,8,9,10,11,12\} \ \forall n \in \mathcal{N} \tag{6.9}$$

$$\mathrm{SIR}_n(\mathrm{SF}_n, \mathrm{SF}_{int}) \geq \mathrm{SIR}_{th}(\mathrm{SF}_n, \mathrm{SF}_{int}) \ \forall n \in \mathcal{N}, \\ \forall\, \mathrm{SF}_{int} \in \{7,8,9,10,11,12\}, \mathrm{SF}_{int} \neq \mathrm{SF}_n \tag{6.10}$$

$$\mathrm{SNR}_n \geq \mathrm{SNR}_{th}(\mathrm{SF}_n) \ \forall n \in \mathcal{N} \tag{6.11}$$

$$N_{\mathrm{SF},x} = N \frac{AT_{\mathrm{SF},x}}{\sum_{\mathrm{SF}\in\mathbf{SF}} AT_{\mathrm{SF}}} \forall n \in \mathbf{SF} \tag{6.12}$$

where:

- $\mathrm{SIR}_n(\mathrm{SF}_n, \mathrm{SF}_{int})$ is the Signal to Interference Ratio of node n w.r.t. an interfering spreading factor SF_{int} at the gateway.
- SNR_n is the signal to noise ratio of node n at the gateway.

[2] A further direction for optimizing the system could be the usage of compression schemes for sending the model coefficients, which could lead to heterogeneous model sizes s_n.

- $N_{\mathrm{SF},x}$ is the number of users assigned to SF x, while $AT_{\mathrm{SF},x}$ is the airtime of a LoRa packet with maximum payload on the same SF, x.

With reference to constraint (6.10), whenever SIR_n is below the threshold $\mathrm{SIR}_{min}(\mathrm{SF}_n, \mathrm{SF}_{int})$, decoding of frames sent from node n may fail in case of collisions with frames sent on SF_{int}. The reference threshold values are configured from the experimental work in [137].

For what concerns constraint (6.11), the threshold value SNR_{th} is properly set to guarantee a specific CER/FER on the incoming packets. We calculated a threshold value able to guarantee a CER equal to or less than 30 %, according to the study carried on in [138].

Finally, with constraint (6.12), we aim at balancing the load on all the available SFs. More specifically, the smaller is the air time, the bigger will be the portion of user assigned to the corresponding SF. The main reason to perform load balancing is maximization of parallel (non-interfering) transmissions and the consequent reduction of the FML round time.

The problem above formulated is non-linear and includes continuous variables ($\boldsymbol{\rho}$), integer variables (**SF**), as well as non-linear constraints (constraints 6.10 and 6.11), and therefore falls into the category of Mixed-Integer Non-linear Programming (MINLP) problems. By definition [139, 140], MINLP problems are **NP-Hard**, and must be solved through either numerical approximation or greedy algorithms.

6.3.2 A Greedy Algorithm for LoRa-RAP

We designed a greedy algorithm for resource allocation, whose pseudocode is shown in Algorithm 1. The algorithm is split in several phases: (i) Estimate the channel (Lines 1-4), (ii) Meet SNR constraints (Lines 6-15), (iii) Meet SIR constraints (Lines 17-34). During the first phase, the gateway node estimates the maximum SNR of each node (we assume the channel gain matrix and signal noise power to be known a priori). The maximum SNR can be easily estimated by assuming the nodes to transmit at the maximum available power. Nodes are accordingly ordered by SNR, in descending order. In the second phase, each node is assigned to the minimum SF possible, in compliance with the SNR constraint (6.11). However, if this choice violates the load-balancing constraint (6.12), the algorithm tries to assign the node to the next higher SF. This procedure is repeated until either constraint (6.12) is satisfied, or until the maximum SF value is reached. The third and last phase focuses on the SIR constraint (6.10).

For each node n in the scenario, the algorithm checks if the assigned transmission power is compliant with the SIR constraint for SF_n. If not,

Algorithm 1: Greedy Resource Allocation for FedLoRa

Data: LoRa Node Set $\mathcal{N}$, Channel Gain Matrix $H_{N\times N}$, Signal noise power σ_n^2

foreach *node* n *in* $\mathcal{N}$ **do**
  Set power ρ_n of node n to ρ_{max};
  Estimate SNR_n of node n from $H_{N\times N}$ and σ_n^2 ;
end
Sort nodes by SNR, in descending order;
foreach *node* n *in* $\mathcal{N}$ **do**
  Find minimum $\mathrm{SF}_{n,min}$, such that $\mathrm{SNR}_n > \mathrm{SNR}$;
  if $N_{\mathrm{SF}_{n,min}} + 1$ does not satisfy constraint (6.11) **then**
    do
      $\mathrm{SF}_{n,min} \leftarrow \mathrm{SF}_{n,min} + 1$
    while constraint (6.11) is satisfied **or** $\mathrm{SF}, x == \mathrm{SF}, max$;
  end
  $\mathrm{SF}_n \leftarrow \mathrm{SF}_{n,min}$;
  Find ρ'_{min} such that constraint (6.11) is satisfied ;
  $\rho_n \leftarrow \min(\rho'_{min}, \rho_{min})$;
end
do
  foreach *node* $n \in \mathcal{N}$ **do**
    if $\mathrm{SIR}_n(\mathrm{SF}_n, \mathrm{SF}_{int})$ satisfies constraint (6.10) **then**
      continue ;
    else
      foreach $\mathrm{SF}_{int} \in \{7, 8, 9, 10, 11, 12\}$ **do**
        if $\mathrm{SF}_{int} == \mathrm{SF}_n$ **then**
          continue;
        end
        $\Delta\rho \leftarrow \mathrm{SIR}_{th}(\mathrm{SF}_n, SF_{int}) - \mathrm{SIR}_n(\mathrm{SF}_n, \mathrm{SF}_{int})$;
        $\rho_n \leftarrow \rho_n + \Delta\rho$;
        if $\rho_n > \rho_{max}$ **then**
          Convergence Not Reachable, Exit;
        end
      end
    end
  end
while constraint (6.10) is satisfied **or** Problem Diverges;

the algorithm proceeds to increase the transmission power, until either the constraint is satisfied, or the assigned power exceeds the maximum power limit. In the latter case, no feasible solution can be found. Then, the algorithm starts over and repeats the check – since the nodes are examined in a sequential fashion, adjusting the transmission power of node i could lead to a violation of the SIR constraint for at least one of the $i-1$ previous nodes. Hence, the algorithm needs to repeat the check and exits if and only if the SIR constraint is still satisfied for all nodes. If not, the algorithm executes step 3 several times. If the algorithm converges, it yields a sub-optimal solution to the energy minimization problem formulated in (6.7).

6.3.3 `FedLoRa` Prototype

We prototyped and evaluated `FedLoRa` and LoRa-RAP on Colosseum, the world's largest network emulator [19]. Colosseum is a wireless emulator with 128 Standard Radio Nodes (SRNs). Each SRN is equipped with 48-core Intel Xeon E5-2650 CPUs and an NVIDIA Tesla K40m GPU, and with a NI/Ettus USRP X310 Software Defined Radio (SDR) as well. The SRNs are all linked together by a Massive Channel Emulator (MCHEM), which is responsible for the emulation of the wireless channels. Thanks to its FPGA modules, the MCHEM processes the radio signals through Finite Impulse Response (FIR) filters, and thus emulate the effects of a real radio channel, such as attenuation and propagation delay.

As part of the prototype, we implemented `FedLoRa` on Colosseum, starting from the LoRa PHY for SDR described in Chapter 5.

Polling Mechanism. Besides the LoRa PHY, we implemented for the first time a MAC protocol to establish reliable data exchange between the nodes and the gateway based on polling. Indeed, for transmitting the gradient components of the local model at each communication round, sensors generate a bulk of data frames. Our polling mechanism is intended to replace the ALOHA mechanism of standard LoRa networks, which has a very limited efficiency in case of greedy traffic sources.

With a polling mechanism, nodes assigned to the same SF transmit in a sequential way, and do not interfere with each other, while nodes working on different SFs can be polled in parallel. **For this reason, balancing the load in the network is of crucial importance** for maximizing parallel (non-interfering) transmissions, thus reducing the FML round time.

6.4 Colosseum results

Before presenting the results obtained through Colosseum, we first describe the baselines and the RF scenarios.

Baselines. We compare LoRa-RAP with the following baselines:

- *MinPower*: this algorithm first focuses on the minimization of the node transmission power. Then, it assigns the lowest SF possible, in compliance with LoRa SNR requirements.
- *BestSF*: as opposed to *MinPower*, the main goal of *BestSF* is to first find the minimum SF allowed by the SNR constraints, and, only then, to minimize the transmission power. No load balancing criteria are applied.

Both algorithms also deal with the SIR constraints in (6.10). The adopted procedure is identical to the one of Algorithm 1.

Scenario Description. We now describe the custom LoRa scenarios implemented in Colosseum to evaluate LoRa-RAP. A network scenario is easily defined in Colosseum as a collection of wireless links between several radio nodes. Each link is specified by digital channel taps, which are fed to the MCHEM at run time. As depicted in Figure 6.3, the scenario involves a maximum of 18 LoRa nodes and one LoRa gateway, scattered over a $400 \times 400m^2$ area. The color scheme is related to the SF value assigned to each node by LoRa-RAP in the full setting (as the number of considered nodes varies, the allocation of resources is accordingly different).

Channel attenuation is calculated by means of Friis propagation model, with a path loss exponent $\alpha = 2$ (i.e. the coefficient for free space path loss scenarios). Note that the choice to simulate a "small" area and to assume a free space propagation model could seem unsuitable for the emulation of LoRa-based communications. This choice, however, is due to the intrinsic limitations of Colosseum, and, more specifically, of the SDR hardware, which introduces a noise power estimated as high as $\sigma_n^2 \approx 3,5 * 10^{-8}$. This effect naturally reduces the maximum allowed simulated attenuation and, as a consequence, the maximum simulated communication distance.

Figure 6.3 depicts the virtual locations of the nodes in the emulated scenario, as well as the SF resource allocation performed by LoRa-RAP. Note how the farthest nodes are naturally associated to higher SFs. Moreover, the SF allocation reflects the load balancing criterion described in the previous sections.

Experiments. The experiments performed on the Colosseum channel emulator aimed at the evaluation of two main metrics: **energy consump-**

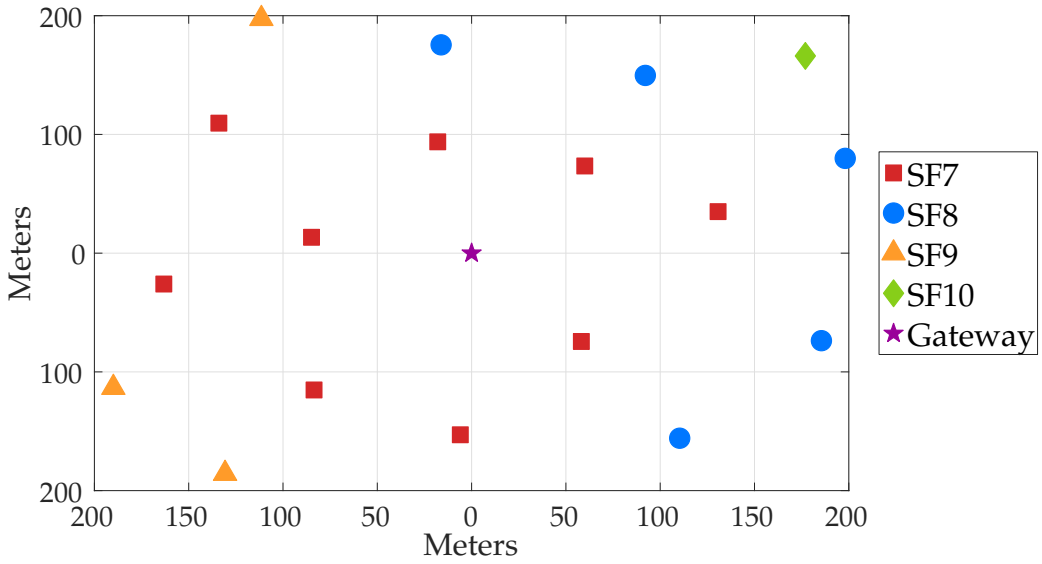


Figure 6.3: Colosseum Testbed for `FedLoRa`: Location of the LoRa Nodes and of the Gateway in the emulated scenario.

tion, and **FML round time**. Note how the former represents a *normalized* energy value, rather than a real energy value. Indeed, since SDRs are uncalibrated devices, calculating the actual output power is not possible. Instead, we calculate the energy consumption for each node from the digital signal amplitude, and from the model transmission time. The reference Machine Learning (ML) model for our experiments is the neural network from [122], whose specific aim is the fingerprinting of radio devices. The overall size of the network is 83.41 kBytes, and can thus suit the low data-rate capabilities of LoRa. The total number of weights in the network is 21,353. Since each weight is represented as a 4 byte float, a single LoRa packet can fit up to 62 weights. The experiments were run for several sub-sets of active nodes, ranging from just three LoRa nodes, to the full configuration of eighteen nodes. In each configuration, the chosen nodes were the closest to the gateway. Finally, the energy consumption and FML round time have been averaged on a total of five federated rounds per experiment.

Note how, in this particular scenario, the baseline *MinPower* allocates all the nodes to SF 9, while, for *BestSF*, every node is assigned to SF 7.

Figure 6.4 reports the average per-node energy consumption for LoRaRAP and the baselines. Significantly, all three strategies exhibit a similar performance for each node configuration, with a slight reduction in the energy consumption for the *BestSF* strategy. Note how a bigger number of active nodes results in a bigger average energy consumption. Indeed, the farthest

nodes either transmit on higher SFs and/or with high transmission power. Hence, the average energy consumption accordingly increases.

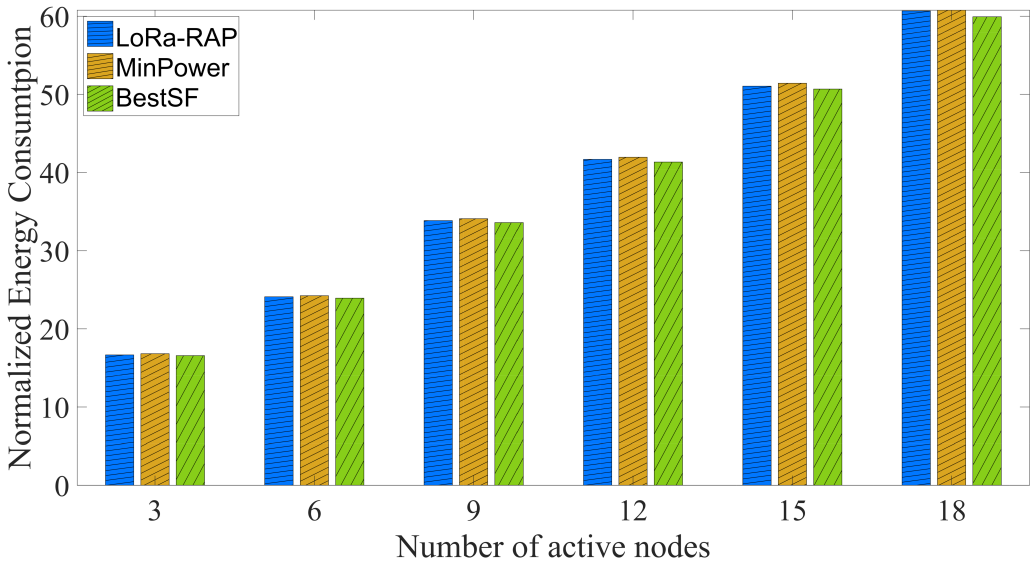


Figure 6.4: Colosseum - LoRa-RAP vs baselines: Average per-node Energy Consumption

Figure 6.5 depicts the average FML round time for LoRa-RAP and the baselines. The results show how our approach is able to reduce the average time for a FML round, and, hence, to finally reduce the convergence time of the FML procedure. In fact, while *BestSF* simply assigns the smallest SF possible to each node, LoRa-RAP balances the load among the available SFs, allowing for simultaneous transmissions on different SFs, thus reducing the FML round time by up to about 35%. Note how the baseline *MinPower* offers instead the worst performance: since the priority is the minimization of the transmission power, the nodes are accordingly allocated to higher SFs (in this specific case, SF 9), resulting in a high transmission time.

6.5 Testbed Results

To validate the performance of `FedLoRa` with real-world data on a larger scale, we have collected realistic SNR and RSSI values from real-world measurements through the testbed depicted in Figure 6.6. Specifically, we used (i) one Adafruit RFM95W LoRa radio transceiver breakout board operating at 915MHz, equipped with a Semtech SX1276 Engine with 127 dB Dynamic Range RSSI; (ii) a LoRaWAN-compliant RAK7268C WisGate Edge Lite 2 gateway from RAK Wireless; (iii) an NVIDIA Jetson Nano. We placed the

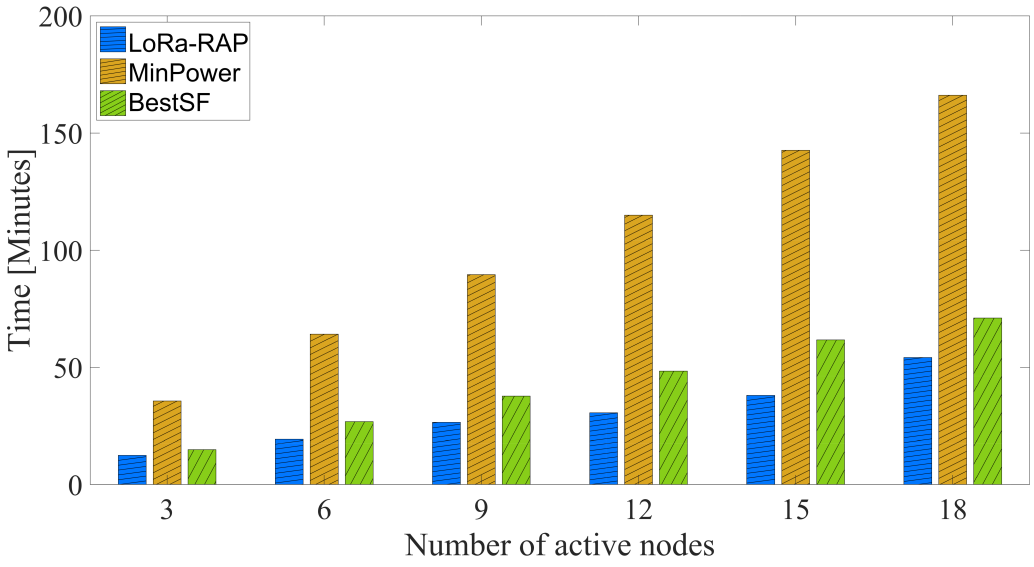


Figure 6.5: Colosseum - LoRa-RAP vs baselines: FML Round Time

LoRa gateway inside our laboratory and collected several SNR/RSSI measurements in several different locations, at a maximum distance of about 5 Km from the gateway, as shown in Figure 6.6. The SNR and RSSI values have been used to model the channel conditions between the nodes and the gateway, with an estimated noise power of about -105 *dbm*. All these data has been fed to LoRa-RAP and to the baseline allocation procedures. Since results from the previous section demonstrated the ineffectiveness of *MinPow*, we omit the performance results from this baseline. Instead, we introduce a variant of LoRa-RAP called LoRa-RAP Min SF. The difference is in the way the load balancing criterion is applied. While LoRa-RAP tries to distribute the nodes over all the available SFs (if possible), LoRa-RAP Min SF also aims at keeping the SFs as low as possible. Intuitevely, in small-scale scenarios, LoRa-RAP Min SF performs similarly to vanilla LoRa-RAP, if not worse. For this reason, we chose not to evaluate LoRa-RAP on Colosseum, and to instead test its performance over a larger, simulated scenario. Once again, the baselines are evaluated in terms of both average energy consumption, and FML round time. Similarly to the experiments run on Colosseum, we evaluate the allocation strategies over a variable number of nodes, ranging from 24 to 42 nodes. Figure 6.6 illustrates the SF allocation for LoRa-RAP over all the 42 evaluated positions.

Figure 6.7 reports the average energy consumption per-node. In line with the results from Colosseum, LoRa-RAP and LoRa-RAP Min SF exhibit a slightly higher energy consumption, as compared to *BestSF*. Once again, including more nodes, and, specifically, the farthest ones, results in an increased

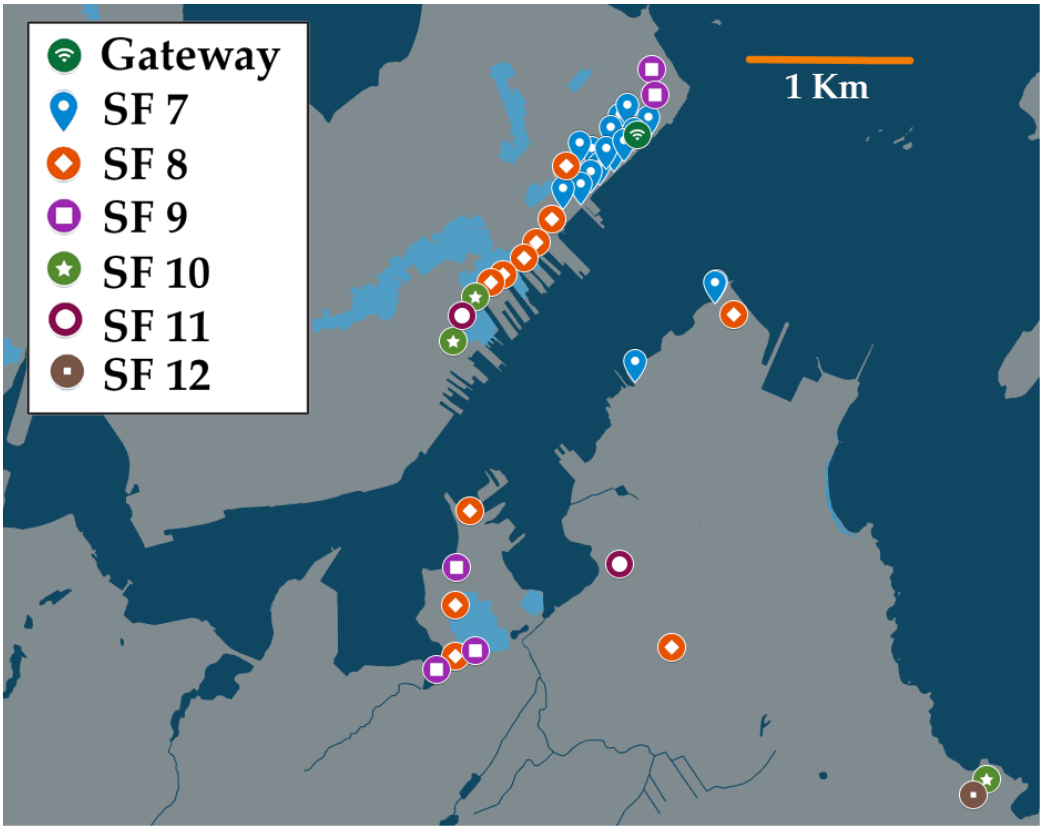


Figure 6.6: Physical locations of the nodes and gateways during data collection.

average energy consumption.

Figure 6.8 depicts the FML round time. LoRa RAP and LoRa-RAP Min SF achieve a relevant improvement in the FML round time. More specifically, LoRa-RAP is the best allocation strategy in most configurations, with an improvement of up to 50% over *BestSF*. However, in some specific cases, LoRa RAP Min SF introduces a slight improvement in the FML round time, as compared to its vanilla version. The reason is the following: in some specific cases, the node number can not exactly match the load balancing criteria. For instance, let us consider a network made up of 8 nodes. if the airtime over SF x is always double the airtime over SF $x+1$, i.e. $AT_{\mathrm{SF},x+1} = 2 * AT_{\mathrm{SF},x}$, then the load balancing condition is $\{4x, 2x, x\}$, i.e. 4 nodes assigned to SF 7, two nodes to SF 8, and one to SF 9. In this case, however, this condition can not be satisfied, and the resulting allocation is instead $\{4, 2, 2\}$. The maximum transmission time is therefore $2 * AT_{SF,9}$. Under proper channel conditions, one extra one could instead be allocated to SF 8, resulting in the distribution $\{4, 3, 1\}$, and in a smaller transmission time of $3 * AT_{\mathrm{SF},8} = 1.5 * AT_{\mathrm{SF},9}$.

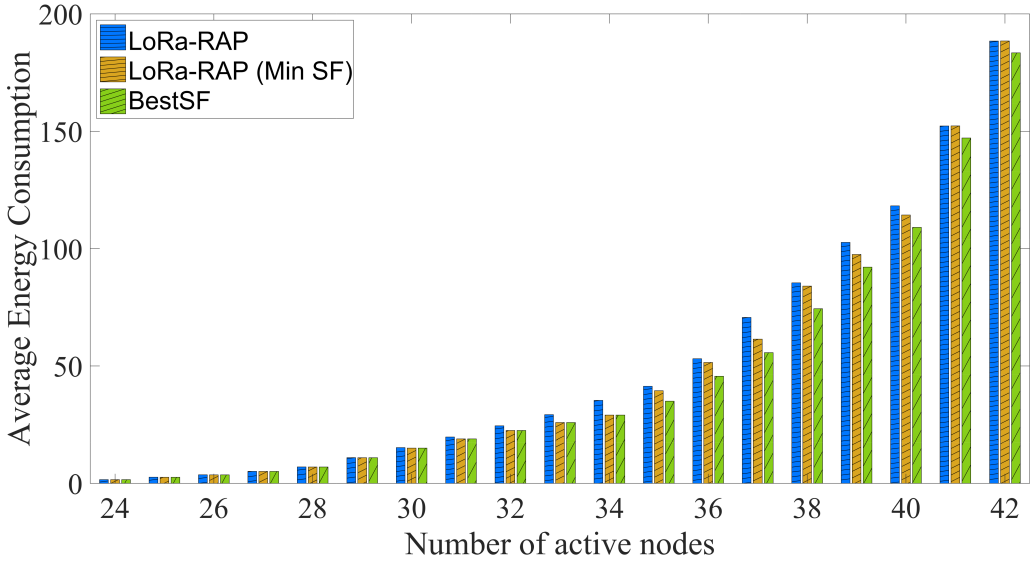


Figure 6.7: Testbed - LoRa-RAP vs baselines: Average per-node Energy Consumption

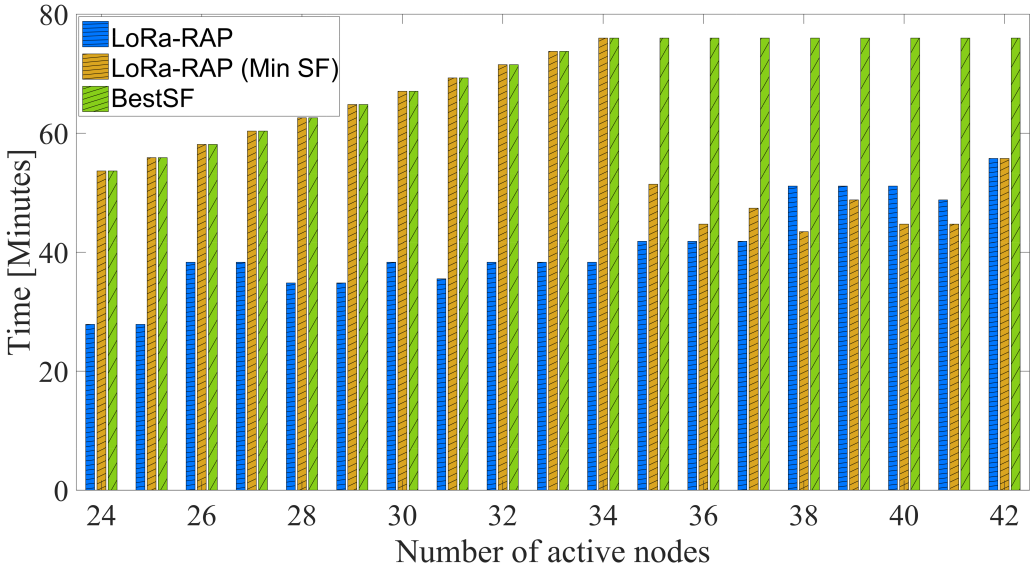


Figure 6.8: Testbed - LoRa-RAP vs baselines: FML Round Time

Chapter 7

Conclusions

This work described a two-fold paradigm at the network edge, where AI and Resource Allocation techniques are intertwined and can reciprocally benefit from each other. More specifically, this work first analyzed the usage of AI for Resource Allocation at the network edge, as opposed to traditional model-based optimization approaches. Moreover, the thesis has outlined two specific use cases, i.e. UAV cellular networks and vehicular networks. In both scenarios, AI has provably improved the management of the available resources, as well as the network performance.

Then, this thesis dealt with an opposite perspective, i.e. the usage of Resource Allocation frameworks to support AI in edge networks, with a specific focus on the IoT. Accordingly, this work first described the implementation of the LPWAN protocol over SDR. Such an implementation has proven crucial to design, experiment, and validate the effectiveness of `FedLoRa`, a resource allocation framework for LoRa to improve the performance of Federated Learning in the IoT. As expected, the proper allocation of communication resources improves the speed of the learning procedures, up to an impressive 35%, as compared to the baselines.

On the overall, this work of thesis has demonstrated why AI and ML will play a key role in 5G and beyond networks. These techniques will indeed enable a fully-distributed and model-free management of the few resources available at the network edge, to the benefit of the network latency, throughput, and lifetime. The importance of AI and ML will become crucial and will therefore stimulate a shift towards the paradigm of *AI-centric* networks. In such a view, not only AI and ML contribute to the improvement of the network efficiency, but, vice-versa, the network itself is programmed and managed to boost the performance of distributed learning schemes, such as Federated Learning.

Appendix A

Derivation of the Markov Decision Process Matrices of the VMEC-in-a-Box Framework

A.1 Transition probability matrix of the Markov Decision Process Σ

In order to evaluate the transition probability of the Markov Decision Process Σ, let us apply the total probability theorem to the number of possible arrivals from both the areas, i.e. λ_1 and λ_2. We have:

$$P^{(Q1,Q2|a)}_{[(s'_{Q1},s'_{Q2}),(s''_{Q1},s''_{Q2})]}(s''_{N1}, s''_{N2}) = \sum_{\substack{\forall\lambda_1 \in \Im^{(\Lambda 1)} \\ \forall\lambda_2 \in \Im^{(\Lambda 2)}}} B^{(\Lambda)}_{[(s''_{N1},s''_{N2}),\lambda_1,\lambda_2]} \cdot \Pr\left\{ \begin{matrix} S^{(Q1)}(n) = s''_{Q1} \\ S^{(Q2)}(n) = s''_{Q2} \end{matrix} \middle| \begin{matrix} S^{(Q1)}(n-1) = s'_{Q1}, \\ S^{(Q2)}(n-1) = s'_{Q2}, A(n) = a \\ \Lambda_1(n) = \lambda_1, \Lambda_2(n) = \lambda_2 \end{matrix} \right\} \tag{A.1}$$

The probability term of the previous equation can be evaluated by considering that, according to the choice made for the slot duration Δ, kept equal to the mean job service time on a MEC CE, b_1 jobs, if present, will be served in the queue Q_1, and b_2 jobs, if present, in the queue Q_2. The probability term in A.1 can be written as follows:

$$P^{(Q1,Q2|a)}_{[(s'_{Q1},s'_{Q2}),(s''_{Q1},s''_{Q2})]}(s''_{N1},s''_{N2}) = f^{(Q_1)}\begin{pmatrix} s'_{Q1}, s'_{Q2}, s''_{Q1}, \sigma, \\ b_1, b_2, \lambda_1, \lambda_2 \end{pmatrix} \cdot f^{(Q_2)}\begin{pmatrix} s'_{Q1}, s'_{Q2}, s''_{Q2}, \sigma, \\ b_1, b_2, \lambda_1, \lambda_2 \end{pmatrix} \tag{A.2}$$

where $f^{(Q_1)}$ and $f^{(Q_2)}$ are functions providing us the one-slot evolution probabilities of the two MEC queues. Let us define three Boolean variables to decide the MEC server that can perform offload, according to the comparison between the queue lengths:

$$I_{\text{No-OL}} = \left[s'_{Q2} = s'_{Q1}\right] \; I_{Q1\to Q2} = \left[s'_{Q2} < s'_{Q1}\right] \; I_{Q2\to Q1} = \left[s'_{Q2} > s'_{Q1}\right] \tag{A.3}$$

They are used as follows: if $I_{\text{No-OL}}$ is true, none of the two MEC servers performs offload; otherwise, offload can be done by the MEC Server 1 if $I_{Q1\to Q2}$ is true, or by the MEC Server 2 if $I_{Q2\to Q1}$ is true.

The first function in A.3 can be calculated taking into account that, according to the event sequence illustrated so far, the queue state at the beginning of the slot, s'_{Q1}, is decreased by the number of served jobs, b_1, and increased by the non-offloaded jobs, $\min\{\sigma, \lambda_1\}$. In case of queue overflow, it is truncated to the value K. So, we have:

$$f^{(Q_1)}\begin{pmatrix} s'_{Q1}, s'_{Q2}, s''_{Q1}, \sigma, \\ b_1, b_2, \lambda_1, \lambda_2 \end{pmatrix} =$$
$$\begin{cases} 1 \text{ if } (I_{\text{No-OL}} \text{ and } H^{(Q1)}_{\text{No-OL}}) \text{ or } (I_{Q1\to Q2} \text{ and } H^{(Q1)}_{Q1\to Q2}) \text{ or } (I_{Q2\to Q1} \text{ and } H^{(Q1)}_{Q2\to Q1}) \\ 0 \text{ otherwise} \end{cases} \tag{A.4}$$

The Boolean conditions $H^{(Q1)}_{\text{No-OL}}$, $H^{(Q1)}_{Q1\to Q2}$, and $H^{(Q1)}_{Q2\to Q1}$ consider the transitions of the MEC Server 1 queue when no offloads are done, when there are possible offloads from the MEC Server 1 to the MEC Server 2 and when there are possible offloads from the MEC Server 2 to the MEC Server 1, respectively. They can be obtained as:

$$\begin{aligned} H^{(Q1)}_{\text{No-OL}} &= \left[s''_{Q1} = \min\left\{\left(\max\left\{s'_{Q1} - b_1, 0\right\} + \lambda_1\right), K\right\}\right] \\ H^{(Q1)}_{Q1\to Q2} &= \left[s''_{Q1} = \min\left\{\left(\max\left\{s'_{Q1} - b_1, 0\right\} + \min\left\{\sigma, \lambda_1\right\}\right), K\right\}\right] \\ H^{(Q1)}_{Q2\to Q1} &= \left[s''_{Q1} = \min\left\{\left(\max\left\{s'_{Q1} - b_1, 0\right\} + \lambda_1 + (\lambda_2 - \min\left\{\sigma, \lambda_2\right\})\right), K\right\}\right] \end{aligned} \tag{A.5}$$

Dually, we can define $f^{(Q_2)}\big(s_{Q2}^{'}, s_{Q2}^{'}, s_{Q1}^{''}, \sigma, b_1, b_2, \lambda_1, \lambda_2\big)$ for the MEC Server 2.

A.2 Short-Term reward matrix of the Markov Decision Process Σ

In order to calculate the short-term reward matrix of the Markov Decision Process Σ, let us define the expected value of the immediate reward for a given transition from the state $\underline{s}_{\Sigma}^{'}$ to the state $\underline{s}_{\Sigma}^{''}$, and for a given action a for the state $\underline{s}_{\Sigma}^{'}$. It is a weighed sum of the four key performance parameters characterizing the MEC Domain behavior, that is, the mean power consumption, the mean net revenue, the mean delay for a job processed by the MEC Domain and its loss probability. More in deep, we define the immediate reward associated to that transition as follows:

$$\Psi_{[\underline{s}_{\Sigma}^{'}, \underline{s}_{\Sigma}^{''}]}^{(\Sigma|a)} = -c_1\overline{\xi}(a) - c_2\overline{\theta}(\underline{s}_{\Lambda}^{''}, a) - c_3\overline{\phi}_M(\underline{s}_{\Sigma}^{''}, a) - c_4\overline{\psi}_M(\underline{s}_{\Sigma}^{''}, a) \tag{A.6}$$

A.2.1 The first term in A.6: power consumption penalty

The first term regards the penalty (it becomes a reward thanks to the minus sign) received for power consumption due to the active CEs, when the action $a = (b_1, b_2, \sigma)$ is performed according to the starting state $\underline{s}_{\Sigma}^{'}$. It is given by:

$$\overline{\xi}(a) = (b_1 + b_2) \cdot \wp_{CE} \tag{A.7}$$

where $\wp_{CE}$ is the power consumption of each active CE. The terms b_1 and b_2 are the numbers of CEs that have been decided to be active, as part of the action a, in the current slot.

A.2.2 The second term in A.6: net revenue due to offload

The net revenue due to offload is constituted by two parts:

1. a revenue that is proportional to the amount of received jobs that have been offloaded by the Vehicular Domain, i.e. $E\{\Lambda_{V\to M}\}$, with a constant of proportionality, $\Theta_{V\to M}^{(OL)}$, representing the price applied by the MPS to process one job in the MEC Domain;

2. a penalty that is proportional to the mean number of offloaded jobs to another MEC server, $E\{\Phi_{M\to M}|a\}$, with a constant of proportionality, $\Theta^{(OL)}_{M\to M}$, representing the per-job offload cost from the MEC Server 1 to the MEC Server 2.

Therefore, we have

$$\overline{\theta}(\underline{s}^{''}_{\Lambda}, a) = -\Theta^{(OL)}_{V\to M} \cdot E\{\Lambda_{V\to M}\} + \Theta^{(OL)}_{M\to M} \cdot E\{\Phi_{M\to M}|a\} \tag{A.8}$$

The term $E\{\Lambda_{V\to M}\}$ can be derived by applying the total probability theorem to all the possible numbers of arrivals from the Areas 1 and 2 when the state of the underlying Markov chain of the bi-dimensional Markov modulated process is $\underline{s}^{''}_{\Lambda}$:

$$E\{\Lambda_{V\to M}\} = \sum_{\forall\lambda_1\in\Im^{(\Lambda 1)}} \sum_{\forall\lambda_2\in\Im^{(\Lambda 2)}} (\lambda_1+\lambda_2) \cdot B^{(\Lambda)}_{[\underline{s}^{''}_{\Lambda},\lambda_1,\lambda_2]} \tag{A.9}$$

Likewise, the $E\{\Phi_{M\to M}|a\}$ can be derived considering the MEC server that can perform some offload when the system state is $\underline{s}^{''}_{\Sigma}$:

$$\begin{aligned} &E\{\Phi_{M\to M}|a\} = \sum_{\forall\lambda_1\in\Im^{(\Lambda 1)}} \sum_{\forall\lambda_2\in\Im^{(\Lambda 2)}} B^{(\Lambda)}_{[\underline{s}^{''}_{\Lambda},\lambda_1,\lambda_2]} \cdot \\ &\left[(\lambda_1 - \min\{\sigma,\lambda_1\}) \cdot I_{Q1\to Q2} + (\lambda_2 - \min\{\sigma,\lambda_2\}) \cdot I_{Q2\to Q1}\right] \end{aligned} \tag{A.10}$$

A.2.3 The third term in A.6: mean delay in the MEC Domain

The third term in A.6 regards the delays suffered in the MEC Server 1 and MEC Server 2 queues. To this purpose, we indicate the number of jobs arriving from the Areas 1 and 2 and not lost, as $\widetilde{\lambda}_1$ and $\widetilde{\lambda}_2$, respectively. respectively. In order to calculate $\widetilde{\lambda}_1$ and $\widetilde{\lambda}_2$, we have to account that each queue state goes through two different steps before the possible arrival of the offloaded jobs coming from the other MEC server. For example, for the MEC Server 1, starting from the state $s^{'}_{Q1}$ and after the departures, it reaches the state $s^{''}_{Q1,INT1} = \max\{s^{'}_{Q1} - b_1, 0\}$; then, after the arrivals from the Area 1, we have two cases:

1. if offload from the MEC Server 1 is not allowed, all the λ_1 jobs arriving from the Area 1 are sent to Q_1, and therefore the queue state becomes $s^{''}_{Q1,INT2} = \min\{s^{''}_{Q1,INT1} + \lambda_1, K\}$;
2. if the MEC Server 1 is enabled to offload jobs to the other server, its queue state becomes $s^{''}_{Q1,INT2} = \min\{s^{''}_{Q1,INT1} + \min\{\sigma,\lambda_1\}, K\}$.

Therefore, the number of job arrivals from the Area 1 that are accommodated in Q_1 are $\widetilde{\lambda}_1 = s''_{Q1,INT2} - s''_{Q1,INT1}$. Likewise, the number of job arrivals from the Area 2 that are accommodated in Q_2 are $\widetilde{\lambda}_2 = s''_{Q2,INT2} - s''_{Q2,INT1}$. They will suffer a delay due to the queues they find in the MEC Servers 1 and 2, respectively. More specifically, the mean queue lengths they find are:

$$\overline{q}_{a1\to Q1} = \left[s''_{Q1,INT2} + \left(s''_{Q1,INT1} + 1\right)\right]/2 \tag{A.11}$$

$$\overline{q}_{a2\to Q2} = \left[s''_{Q2,INT2} + \left(s''_{Q2,INT1} + 1\right)\right]/2 \tag{A.12}$$

Instead, the number of offloaded jobs from the MEC server with the shortest queue to the other one is:

- $\widetilde{\lambda}_{OL} - \min\{\lambda_{OL}, K - s''_{Q2,INT}\}$, where $\lambda_{OL} = \lambda_1 - \min\{\sigma, \lambda_1\}$, if the offloading MEC server is the MEC Server 1;
- $\widetilde{\lambda}_{OL} - \min\{\lambda_{OL}, K - s''_{Q1,INT}\}$, where $\lambda_{OL} = \lambda_2 - \min\{\sigma, \lambda_2\}$, if the offloading MEC server is the MEC Server 2.

The mean queues the offloaded jobs find on the other server where they are offloaded are:

$$\overline{q}_{a1\to Q2} = I_{Q1\to Q2} \cdot \frac{s''_{Q2} + \left(s''_{Q2,INT2} + 1\right)}{2} \tag{A.13}$$

$$\overline{q}_{a2\to Q1} = I_{Q2\to Q1} \cdot \frac{s''_{Q1} + \left(s''_{Q1,INT2} + 1\right)}{2} \tag{A.14}$$

where the terms $I_{Q1\to Q2}$ and $I_{Q2\to Q1}$ indicate whether there is some offload from the MEC Server 1 to the MEC Server 2 or vice versa. Therefore, applying the total probability theorem to the number of jobs arriving from both the areas, the mean delay that the MPS estimates when the system is in the state $\underline{s}'_{\Sigma}$ and the action a is performed, is:

$$\begin{aligned}\overline{\phi}_M(\underline{s}''_{\Sigma}, a) = &\sum_{\forall\lambda_1\in\Im^{(\Lambda 1)}} \sum_{\forall\lambda_2\in\Im^{(\Lambda 2)}} B^{(\Lambda)}_{[\underline{s}''_{\Lambda},\lambda_1,\lambda_2]} \cdot \frac{1}{\widetilde{\lambda}_1 + \widetilde{\lambda}_2 + \widetilde{\lambda}_{OL}} \cdot \\ &\left[\frac{\widetilde{\lambda}_1}{b_1}\overline{q}_{a1\to Q1} + \frac{\widetilde{\lambda}_{OL}}{b_2}\overline{q}_{a1\to Q2} + \frac{\widetilde{\lambda}_2}{b_2}\overline{q}_{a2\to Q2} + \frac{\widetilde{\lambda}_{OL}}{b_1}\overline{q}_{a2\to Q1}\right]\end{aligned} \tag{A.15}$$

where we have divided the mean queue lengths experienced by the arrived jobs with the current queue service rates, and weighed them with the number of jobs that enter each queue.

A.2.4 The fourth term in A.6: loss probability in the MEC Domain

The loss probability suffered by jobs in the MEC Domain can be calculated as the ratio between the number of jobs lost in the two MEC servers, $L_{M1}+L_{M2}$, and the number of jobs arrived to the MEC Domain in the same slot, $\lambda_1+\lambda_2$:

$$\overline{\psi}_M(\underline{s}''_{\Sigma}, a) = \sum_{\forall \lambda_1 \in \Im^{(\Lambda 1)}} \sum_{\forall \lambda_2 \in \Im^{(\Lambda 2)}} B^{(\Lambda)}_{[\underline{s}''_{\Lambda}, \lambda_1, \lambda_2]} \cdot \frac{L_{M1} + L_{M2}}{\lambda_1 + \lambda_2} \tag{A.16}$$

The number L_{M1} of lost jobs in the MEC Server 1 is the sum of:

- L_{Q1}, representing the number of jobs arrived from the Area 1, which have not been offloaded and have not found any room in Q_1;
- L_{OF1}, representing the number of jobs that should be offloaded (according to the MPS decision) to Q_2, but are discarded by the OF_{MEC} block of the MEC Server 1 because they would not found any rooms in Q_2.

In order to calculate L_{Q1}, we consider that it only depends on the arrivals from the Area 1, since jobs from the Area 2 are offloaded by the OF_{MEC} block of the MEC Server 2 only if they are able to be accommodated in Q_1. Therefore, let us consider the Q_1 starting state, s'_{Q1}. It is decreased by the departure of b_1 jobs, and then increased by the arrival of either $\min\{\sigma, \lambda_1\}$ or λ_1 jobs (whether the MEC Server 1 is not enabled or not for offloading, respectively). The term L_{Q1} is not null only if the resulting queue length is higher than K after departures and local arrivals. Therefore, its value is:

$$L_{Q1} = \max\left\{ \left\lceil \begin{matrix} \max\{s'_{Q1} - b_1, 0\} + \\ (1 - I_{Q1\to Q2}) \cdot \lambda_1 + I_{Q1\to Q2} \cdot \min\{\sigma, \lambda_1\} - K \end{matrix} \right\rceil, 0 \right\} \tag{A.17}$$

The term L_{OF1}, on the other hand, is the difference between the number of jobs that the OF_{MEC} block has to offload to the MEC Server 2 and the number ρ_2 of available rooms in Q_2 after departures and local arrivals, that is:

$$L_{OF1} = \max\{I_{Q1\to Q2} \cdot min\{\sigma, \lambda_1\} - \rho_2, 0\} \tag{A.18}$$

where ρ_2 is the difference between the maximum queue size K and the Q_2 queue size after departures and local arrivals from the Area 2:

$$\rho_2 = \max\left\{ K - \left\lceil \begin{matrix} \max\{s'_{Q2} - b_2, 0\} + \\ (1 - I_{Q2\to Q1}) \cdot \lambda_2 + I_{Q2\to Q1} \cdot \min\{\sigma, \lambda_2\} \end{matrix} \right\rceil, 0 \right\} \tag{A.19}$$

The terms needed to calculate the number L_{M2} of lost jobs in the MEC Server 2 can be calculated at the same way.

A.2.5 Battery model to calculate the service outage probability of a MEC server

In this section, we derive the transition probability matrix of the battery behavior $\underline{S}^{(BT)(n)}$ of a MEC server, aimed at deriving its service outage probability. Its generic element can be written as the product of three terms:

$$P^{BT}_{[\underline{s}'_{BT},\underline{s}''_{BT}]} = P^{RG}_{[s'_{RG},s''_{RG}]} \cdot P^{CF}_{[b',b'']} \cdot P^{QB}_{[s'_{QB},s''_{QB}]}(s''_{RG}, b'') \tag{A.20}$$

The matrix P_{RG} is the transition probability matrix of the SBBP process $RG(n)$ describing the behavior of the microeolic power generator. This, together with the matrix $B^{(RG)}$ giving us the probability distribution of the amount of charge generated for each state of the wind, is an input of the problem. They are defined as follows:

$$\begin{aligned} P^{RG}_{[s'_{RG},s''_{RG}]} &= \Pr\Big\{ S^{(RG)}(n) = s''_{RG} \Big| S^{(RG)}(n-1) = \underline{s}'_{RG} \Big\} \text{ and} \\ B^{WG}_{[s''_{RG},\beta_{RG}]} &= \Pr\Big\{ (RG)(n) = \beta_{RG} \Big| S^{(RG)}(n) = \underline{s}''_{RG} \Big\} \end{aligned} \tag{A.21}$$

In order to calculate the matrix $P^{(CF)}$, let us define the subset $\Im^{(\Sigma,1)}_b$ of $\Im^{(\Sigma)}$ whose states are the ones that are characterized by an action that activates b CEs in the MEC Server 1 (it can be calculated in the same way for the MEC Server 2):

$$\Im^{(\Sigma,1)}_b = \left\{ \underline{s}_\Sigma \in \Im^{(\Sigma)} \text{ such that } a^*_{\underline{s}_\Sigma} = (b, b_2, \sigma), \forall b_2, \forall \sigma \right\} \tag{A.22}$$

where $a^*_{\underline{s}_\Sigma}$ a indicates the best action associated to the state $\underline{s}_\Sigma$.

Now, the probability of transition for the process $S^{(CF}(n)$, representing the number of active CEs in the MEC Server 1, from the generic state b' of the process $S^{(CF)}(n)$ (i.e. b' active CEs) to b'' can be easily calculated as follows:

$$\begin{aligned} P^{CF}_{[b',b'']} &= \Pr\Big\{ S^{(CF)}(n) = b'' \Big| S^{(CF)}(n-1) = b' \Big\} = \\ &\sum_{\forall \underline{s}'_\Sigma \in \Im^{(\Sigma,1)}_{b'}} \sum_{\forall \underline{s}''_\Sigma \in \Im^{(\Sigma,1)}_{b''}} P^{(\Sigma)}_{[\underline{s}'_\Sigma, \underline{s}''_\Sigma]} \cdot \frac{\pi^{(\Sigma)}_{[\underline{s}''_\Sigma]}}{\sum_{\forall \widetilde{\underline{s}}_\Sigma \in \Im^{(\Sigma,1)}_{b'}} \pi^{(\Sigma)}_{[\widetilde{\underline{s}}_\Sigma]}} \end{aligned} \tag{A.23}$$

where $\Im^{(\Sigma,1)}_{b'}$ and $\Im^{(\Sigma,1)}_{b''}$ are the subsets of the state space $\Im^{(\Sigma)}$ containing the states $\underline{s}_\Sigma$ characterized by an action that activates, in the MEC Server

1, $b^{'}$ and $b^{''}$ CEs, respectively. Finally, as far as the matrix $P^{QB}(s^{''}_{RG}, b^{''})$ in A.20 is concerned, it contains the transition probabilities of the battery charge level from the slot $n-1$ to the slot n. It depends on the state of the wind generator and the load (constituted by the active CEs of the considered MEC Server) in the arrival slot n. Let us indicate the number of QoCs that $b^{''}$ active CEs drain from the battery as ξ_{CF}. Since it may be not an integer, we approximate it to the closest integers with probabilities depending on the distance from them. More specifically, we will approximate ξ_{CF} to $\xi^{(-)}_{CF} = \lfloor \xi_{CF} \rfloor$ with probability $p^{(-)}_{CF}$, and to $\xi^{(+)}_{CF} = \lceil \xi_{CF} \rceil$ with probability $p^{(+)}_{CF}$, where $p^{(-)}_{CF} = \xi^{(+)}_{CF} - \xi_{CF}$ and $p^{(+)}_{CF} = \xi_{CF} - \xi^{(-)}_{CF}$. For example, if $\xi_{CF} = 3.7$, it will be rounded to $\xi^{+}_{CF} = 4.0$ with probability $p^{(+)}_{CF} = 0.7$, or to $\xi^{-}_{CF} = 3.0$ with probability $p^{(-)}_{CF} = 0.3$. Therefore, the generic element of $P^{(QB)}(s^{''}_{WG}, b^{''})$ can be calculated as follows:

$$
\begin{aligned}
&P^{QB}_{[s^{'}_{QB}, s^{''}_{QB}]}(s^{''}_{RG}, b^{''}) = \\
&\sum_{\forall \beta_{RG} \in \Psi^{(RG)}} B^{(WG)}_{[s^{''}_{RG}, \beta_{RG}]} \cdot \Pr\Big\{ S^{(QB)}(n) = s^{''}_{QB} \Big|^{S^{(QB)}(n-1) = s^{'}_{QB},\, S^{(RG)}(n) = s^{''}_{RG}}_{S^{(CF)}(n) = b^{''}} \Big\} = \\
&\sum_{\forall \beta_{RG} \in \Psi^{(RG)}} B^{(RG)}_{[s^{''}_{RG}, \beta_{RG}]} \cdot \left\{ \begin{matrix} B^{(WG)}_{[s^{''}_{RG}} \cdot p^{(+)}_{CF} \text{ if } C^{+}_{QB} \\ B^{(WG)}_{[s^{''}_{RG}} \cdot p^{(-)}_{CF} \text{ if } C^{-}_{QB} \\ 0 \text{ otherwise} \end{matrix} \right\}
\end{aligned}
\tag{A.24}
$$

where $C^{(+)}_{QB}$ and $C^{(-)}_{QB}$ are two Boolean conditions representing the transition of the battery charge level (expressed in QoBs) from the state QB $s^{'}_{QB}$ to the state $s^{''}_{QB}$ when β_{RG} QoBs arrived to the battery from the wind generator during the slot n, and either $\xi^{(+)}_{CF}$ or $\xi^{(-)}_{CF}$ QoBs have been drained by the active CEs during the same slot:

$$C^{(+)}_{QB} = \left[s^{''}_{QB} = \min\left\{ \max\left\{ s^{'}_{QB} + \beta_{RG} - \xi^{+}_{CF}, 0 \right\}, K_{QoB} \right\} \right] \tag{A.25}$$

$$C^{(-)}_{QB} = \left[s^{''}_{QB} = \min\left\{ \max\left\{ s^{'}_{QB} + \beta_{RG} - \xi^{-}_{CF}, 0 \right\}, K_{QoB} \right\} \right] \tag{A.26}$$

The maximum with 0 and the minimum with K_{QoB} avoid that $s^{''}_{QB}$ assumes negative values or values greater than K_{QoB}.

Bibliography

[1] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed, "Edge computing: A survey," *Future Generation Computer Systems*, vol. 97, pp. 219–235, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X18319903

[2] L. Wang, L. Jiao, T. He, J. Li, and M. Mühlhäuser, "Service entity placement for social virtual reality applications in edge computing," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 468–476.

[3] X. Yang, Z. Chen, K. Li, Y. Sun, N. Liu, W. Xie, and Y. Zhao, "Communication-constrained mobile edge computing systems for wireless virtual reality: Scheduling and tradeoff," *IEEE Access*, vol. 6, pp. 16 665–16 677, 2018.

[4] S. Safavat, N. N. Sapavath, and D. B. Rawat, "Recent advances in mobile edge computing and content caching," *Digital Communications and Networks*, vol. 6, no. 2, pp. 189–194, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352864819300227

[5] K. Zhang, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Cooperative content caching in 5g networks with mobile edge computing," *IEEE Wireless Communications*, vol. 25, no. 3, pp. 80–87, 2018.

[6] Z. Tang, J. Lou, F. Zhang, and W. Jia, "Dependent task offloading for multiple jobs in edge computing," in *2020 29th International Conference on Computer Communications and Networks (ICCCN)*, 2020, pp. 1–9.

[7] H. Wang, X. Chen, H. Xu, J. Liu, and L. Huang, "Joint job offloading and resource allocation for distributed deep learning in edge computing," in *2019 IEEE 21st International Conference on High*

Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2019, pp. 734–741.

[8] C.-K. Tham and R. Chattopadhyay, "A load balancing scheme for sensing and analytics on a mobile edge computing network," in *2017 IEEE 18th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2017, pp. 1–9.

[9] S. Li, G. Zhu, and S. Lin, "Joint radio and computation resource allocation with predictable channel in vehicular edge computing," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 3736–3741.

[10] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros, "Dynamic, latency-optimal vnf placement at the network edge," in *Ieee infocom 2018-ieee conference on computer communications*. IEEE, 2018, pp. 693–701.

[11] P. Jin, X. Fei, Q. Zhang, F. Liu, and B. Li, "Latency-aware vnf chain deployment with efficient resource reuse at network edge," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 267–276.

[12] H. Qi, X. Zhang, and Y. Gao, "Low-Complexity Subspace-Aided Compressive Spectrum Sensing Over Wideband Whitespace," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 12, pp. 11 762–11 777, 2019.

[13] C. Liu, J. Wang, X. Liu, and Y.-C. Liang, "Deep CM-CNN for Spectrum Sensing in Cognitive Radio," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2306–2321, 2019.

[14] A. Al-Shawabka, F. Restuccia, S. D'Oro, T. Jian, B. C. Rendon, N. Soltani, J. Dy, K. Chowdhury, S. Ioannidis, and T. Melodia, "Exposing the Fingerprint: Dissecting the Impact of the Wireless Channel on Radio Fingerprinting," *Proc. of IEEE Conference on Computer Communications (INFOCOM)*, 2020.

[15] A. Al-Shawabka, P. Pietraski, S. B. Pattar, F. Restuccia, and T. Melodia, "DeepLoRa: Fingerprinting LoRa Devices at Scale Through Deep Learning and Data Augmentation," in *Proceedings of*

the ACM International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc), 2021, pp. 251–260.

[16] J. Konečnỳ, B. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," *arXiv preprint arXiv:1511.03575*, 2015.

[17] Semtech, "What is LoRa?" https://www.semtech.com/lora/what-is-lora.

[18] Sigfox, "Sigfox technology," https://www.sigfox.com/en/what-sigfox/technology.

[19] L. Bonati, P. Johari, M. Polese, S. D'Oro, S. Mohanti, M. Tehrani-Moayyed, D. Villa, S. Shrivastava, C. Tassie, K. Yoder *et al.*, "Colosseum: Large-scale wireless experimentation through hardware-in-the-loop network emulation," in *2021 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*. IEEE, 2021, pp. 105–113.

[20] S. Morgenthaler, T. Braun, Z. Zhao, T. Staub, and M. Anwander, "Uavnet: A mobile wireless mesh network using unmanned aerial vehicles," in *2012 IEEE Globecom Workshops*, 2012, pp. 1603–1608.

[21] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Drone small cells in the clouds: Design, deployment and performance analysis," in *2015 IEEE Global Communications Conference (GLOBECOM)*, 2015, pp. 1–6.

[22] A. Merwaday and I. Guvenc, "Uav assisted heterogeneous networks for public safety communications," in *2015 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, 2015, pp. 329–334.

[23] Y. Zeng, R. Zhang, and T. J. Lim, "Wireless communications with unmanned aerial vehicles: Opportunities and challenges," *IEEE Communications Magazine*, vol. 54, no. 5, pp. 36–42, 2016.

[24] I. Ahmad, J. Kaur, H. T. Abbas, Q. H. Abbasi, A. Zoha, M. A. Imran, and S. Hussain, "Uav-assisted 5g networks for optimised coverage under dynamic traffic load," in *2022 IEEE International Symposium on Antennas and Propagation and USNC-URSI Radio Science Meeting (AP-S/URSI)*. IEEE, 2022, pp. 1692–1693.

[25] W. Qi, Q. Song, L. Guo, and A. Jamalipour, "Energy-efficient resource allocation for uav-assisted vehicular networks with spectrum sharing," *IEEE Transactions on Vehicular Technology*, 2022.

[26] J. Zhang, K. Kang, M. Yang, H. Zhu, and H. Qian, "Aoi-minimization in uav-assisted iot network with massive devices," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2022, pp. 1290–1295.

[27] V. Sharma, M. Bennis, and R. Kumar, "Uav-assisted heterogeneous networks for capacity enhancement," *IEEE Communications Letters*, vol. 20, no. 6, pp. 1207–1210, 2016.

[28] O. Andryeyev and A. Mitschele-Thiel, "Increasing the cellular network capacity using self-organized aerial base stations," in *Proceedings of the 3rd Workshop on Micro Aerial Vehicle Networks, Systems, and Applications*, ser. DroNet '17. New York, NY, USA: Association for Computing Machinery, 2017.

[29] A. Al-Hourani, S. Kandeepan, and S. Lardner, "Optimal lap altitude for maximum coverage," *IEEE Wireless Communications Letters*, vol. 3, no. 6, pp. 569–572, 2014.

[30] J. Lyu, Y. Zeng, R. Zhang, and T. J. Lim, "Placement optimization of uav-mounted mobile base stations," *IEEE Communications Letters*, vol. 21, no. 3, pp. 604–607, 2016.

[31] M.-A. Messous, S.-M. Senouci, H. Sedjelmaci, and S. Cherkaoui, "A game theory based efficient computation offloading in an uav network," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 4964–4974, 2019.

[32] A. Giagkos, M. S. Wilson, E. Tuci, and P. B. Charlesworth, "Comparing approaches for coordination of autonomous communications uavs," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2016, pp. 1131–1139.

[33] D. Athukoralage, I. Guvenc, W. Saad, and M. Bennis, "Regret based learning for uav assisted lte-u/wifi public safety networks," in *2016 IEEE Global Communications Conference (GLOBECOM)*, 2016, pp. 1–7.

[34] M. Rebato, M. Mezzavilla, S. Rangan, F. Boccardi, and M. Zorzi, "Understanding noise and interference regimes in 5g millimeter-wave

cellular networks," in *European Wireless 2016; 22th European Wireless Conference*, 2016, pp. 1–5.

[35] A. B. De Souza, P. A. L. Rego, T. Carneiro, J. D. C. Rodrigues, P. P. R. Filho, J. N. De Souza, V. Chamola, V. H. C. De Albuquerque, and B. Sikdar, "Computation offloading for vehicular environments: A survey," *IEEE Access*, vol. 8, pp. 198 214–198 243, 2020.

[36] B. Li, Y. Pei, H. Wu, Z. Liu, and H. Liu, "Computation offloading management for vehicular ad hoc cloud," in *International Conference on Algorithms and Architectures for Parallel Processing*. Springer, 2014, pp. 728–739.

[37] M. Y. Arslan, I. Singh, S. Singh, H. V. Madhyastha, K. Sundaresan, and S. V. Krishnamurthy, "Cwc: A distributed computing infrastructure using smartphones," *IEEE Transactions on Mobile Computing*, vol. 14, no. 8, pp. 1587–1600, 2014.

[38] T. Adhikary, A. K. Das, M. Razzaque, A. Almogren, M. Alrubaian, M. M. Hassan *et al.*, "Quality of service aware reliable task scheduling in vehicular cloud computing," *Mobile Networks and Applications*, vol. 21, no. 3, pp. 482–493, 2016.

[39] Q. Shen, B.-J. Hu, and E. Xia, "Dependency-aware task offloading and service caching in vehicular edge computing," *IEEE Transactions on Vehicular Technology*, 2022.

[40] M. Ferens, D. Hortelano, I. de Miguel, R. J. D. Barroso, J. C. Aguado, L. Ruiz, N. Merayo, P. Fernández, R. M. Lorenzo, and E. J. Abril, "Deep reinforcement learning applied to computation offloading of vehicular applications: A comparison," in *2022 International Balkan Conference on Communications and Networking (BalkanCom)*. IEEE, 2022, pp. 31–35.

[41] K. Zheng, H. Meng, P. Chatzimisios, L. Lei, and X. Shen, "An smdp-based resource allocation in vehicular cloud computing systems," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 12, pp. 7920–7928, 2015.

[42] T. Zhang, E. Robson, and A. Boukerche, "Design and analysis of stochastic traffic flow models for vehicular clouds," *Ad Hoc Networks*, vol. 52, pp. 39–49, 2016.

[43] C. Ren, G. Zhang, X. Gu, and Y. Li, "Computing offloading in vehicular edge computing networks: Full or partial offloading?" in *2022 IEEE 6th Information Technology and Mechatronics Engineering Conference (ITOEC)*, vol. 6. IEEE, 2022, pp. 693–698.

[44] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task offloading in vehicular edge computing networks: A load-balancing solution," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2092–2104, 2019.

[45] S. Islam, S. Badsha, S. Sengupta, H. La, I. Khalil, and M. Atiquzzaman, "Blockchain-enabled intelligent vehicular edge computing," *IEEE Network*, vol. 35, no. 3, pp. 125–131, 2021.

[46] S. Tang, Z. Gu, S. Fu, and Q. Yang, "Vehicular edge computing for multi-vehicle perception," in *2021 Fourth International Conference on Connected and Autonomous Driving (MetroCAD)*. IEEE, 2021, pp. 9–16.

[47] Y. Hui, N. Cheng, Y. Huang, R. Chen, X. Xiao, C. Li, and G. Mao, "Personalized vehicular edge computing in 6g," *IEEE Network*, vol. 35, no. 6, pp. 278–284, 2021.

[48] S. Buda, S. Guleng, C. Wu, J. Zhang, K.-L. A. Yau, and Y. Ji, "Collaborative vehicular edge computing towards greener its," *IEEE Access*, vol. 8, pp. 63 935–63 944, 2020.

[49] M. S. Bute, P. Fan, G. Liu, F. Abbas, and Z. Ding, "A collaborative task offloading scheme in vehicular edge computing," in *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*. IEEE, 2021, pp. 1–5.

[50] A. Boukerche and V. Soto, "Computation offloading and retrieval for vehicular edge computing: Algorithms, models, and classification," *ACM Comput. Surv.*, vol. 53, no. 4, aug 2020. [Online]. Available: https://doi.org/10.1145/3392064

[51] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4377–4387, 2018.

[52] X. Zhang and S. Debroy, "Energy efficient task offloading for compute-intensive mobile edge applications," in *ICC 2020-2020 IEEE*

International Conference on Communications (ICC). IEEE, 2020, pp. 1–6.

[53] H. Lu, C. Gu, F. Luo, W. Ding, S. Zheng, and Y. Shen, "Optimization of task offloading strategy for mobile edge computing based on multi-agent deep reinforcement learning," *IEEE Access*, vol. 8, pp. 202 573–202 584, 2020.

[54] M. Zhao, J.-J. Yu, W.-T. Li, D. Liu, S. Yao, W. Feng, C. She, and T. Q. Quek, "Energy-aware task offloading and resource allocation for time-sensitive services in mobile edge computing systems," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 10, pp. 10 925–10 940, 2021.

[55] J. Zhang, Q. Cui, X. Zhang, W. Ni, X. Lyu, M. Pan, and X. Tao, "Online optimization of energy-efficient user association and workload offloading for mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 2, pp. 1974–1988, 2021.

[56] H. Guo and J. Liu, "Collaborative computation offloading for multiaccess edge computing over fiber–wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 4514–4526, 2018.

[57] Z. Zhou, P. Liu, Z. Chang, C. Xu, and Y. Zhang, "Energy-efficient workload offloading and power control in vehicular edge computing," in *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. IEEE, 2018, pp. 191–196.

[58] Z. Zhou, J. Feng, Z. Chang, and X. Shen, "Energy-efficient edge computing service provisioning for vehicular networks: A consensus admm approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 5087–5099, 2019.

[59] Y. Mao, J. Zhang, S. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 16, no. 9, pp. 5994–6009, 2017.

[60] C. Tang, C. Zhu, X. Wei, Q. Li, and J. J. Rodrigues, "Task caching in vehicular edge computing," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2021, pp. 1–6.

[61] K. Zhang, Y. Mao, S. Leng, S. Maharjan, and Y. Zhang, "Optimal delay constrained offloading for vehicular edge computing networks," in *2017 IEEE International Conference on Communications (ICC)*. IEEE, 2017, pp. 1–6.

[62] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 2, pp. 89–103, 2015.

[63] C. Yang, Y. Liu, X. Chen, W. Zhong, and S. Xie, "Efficient mobility-aware task offloading for vehicular edge computing networks," *IEEE Access*, vol. 7, pp. 26 652–26 664, 2019.

[64] Y. Wu, J. Wu, L. Chen, J. Yan, and Y. Han, "Load balance guaranteed vehicle-to-vehicle computation offloading for min-max fairness in vanets," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[65] J. Xu, B. Yang, and C. Chen, "Task offloading based on edge computing considering overhead and load balancing in industrial internet of things," in *2020 IEEE 18th International Conference on Industrial Informatics (INDIN)*, vol. 1. IEEE, 2020, pp. 53–59.

[66] Q. Ye, W. Shi, K. Qu, H. He, W. Zhuang, and X. S. Shen, "Learning-based computing task offloading for autonomous driving: A load balancing perspective," in *ICC 2021-IEEE International Conference on Communications*. IEEE, 2021, pp. 1–6.

[67] Y. Wang, J. Fu, and Y. Zhang, "Multi-edge server load balancing offloading method based on ant colony algorithm," in *2021 40th Chinese Control Conference (CCC)*. IEEE, 2021, pp. 1826–1831.

[68] W. Ayoub, A. E. Samhat, F. Nouvel, M. Mroue, and J.-C. Prévotet, "Internet of Mobile Things: Overview of LoRaWAN, DASH7, and NB-IoT in LPWANs Standards and Supported Mobility," *IEEE Communications Surveys Tutorials*, vol. 21, no. 2, pp. 1561–1581, 2019.

[69] J. P. Shanmuga Sundaram, W. Du, and Z. Zhao, "A Survey on LoRa Networking: Research Problems, Current Solutions, and Open Issues," *IEEE Communications Surveys Tutorials*, vol. 22, no. 1, pp. 371–388, 2020.

[70] T. Chen, D. Eager, and D. Makaroff, "Efficient image transmission using lora technology in agricultural monitoring iot systems," in *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2019, pp. 937–944.

[71] Z. Yuan, J. Jin, L. Sun, K.-W. Chin, and G.-M. Muntean, "Ultra-Reliable IoT Communications with UAVs: A Swarm Use Case," *IEEE Communications Magazine*, vol. 56, no. 12, pp. 90–96, 2018.

[72] K.-H. Lam, C.-C. Cheung, and W.-C. Lee, "RSSI-Based LoRa Localization Systems for Large-Scale Indoor and Outdoor Environments," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 12, pp. 11 778–11 791, 2019.

[73] C. Bernier *et al.*, "Low Complexity LoRa Frame Synchronization for Ultra-Low Power Software-Defined Radios," *IEEE Transactions on Communications*, vol. 68, no. 5, pp. 3140–3152, 2020.

[74] M. Knight *et al.*, "Decoding LoRa: Realizing a modern LPWAN with SDR," in *Proceedings of the GNU Radio Conference*, vol. 1, no. 1, 2016.

[75] A. Marquet *et al.*, "Towards an SDR implementation of LoRa: Reverse-engineering, demodulation strategies and assessment over Rayleigh channel," *Computer Communications*, vol. 153, pp. 595–605, 2020.

[76] P. Robyns *et al.*, "gr-lora: An efficient LoRa decoder for GNU Radio," https://github.com/rpp0/gr-lora, 2017.

[77] J. Tapparel *et al.*, "An Open-Source LoRa Physical Layer Prototype on GNU Radio," in *2020 IEEE SPAWC*, 2020, pp. 1–5.

[78] R. S. Benatti, C. P. de Souza, and O. Baiocchi, "An Optimization Method based on LoRa Parameters for Energy Consumption Reduction," in *2021 5th International Symposium on Instrumentation Systems, Circuits and Transducers (INSCIT)*, 2021, pp. 1–5.

[79] B. Su, Z. Qin, and Q. Ni, "Energy Efficient Resource Allocation for Uplink LoRa Networks," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–7.

[80] X. Liu, Z. Qin, Y. Gao, and J. A. McCann, "Resource Allocation in Wireless Powered IoT Networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4935–4945, 2019.

[81] R. Jin, X. He, and H. Dai, "On the design of communication efficient federated learning over wireless networks," 2020.

[82] ABIResearch, "LoRaWAN and NB-IoT : competitors or complementary?" 2019. [Online]. Available: https://lora-alliance.org/resource_hub/lorawan-and-nb-iot-competitors-or-complementary/

[83] V. D. Nguyen, S. K. Sharma, T. X. Vu, S. Chatzinotas, and B. Ottersten, "Efficient federated learning algorithm for resource allocation in wireless iot networks," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[84] J. Yao and N. Ansari, "Enhancing Federated Learning in Fog-Aided IoT by CPU Frequency and Wireless Power Control," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[85] [Online]. Available: https://gsacom.com/paper/lte-to-5g-june-2021-global-update/

[86] [Online]. Available: https://gsacom.com/paper/5g-ecosystem-member-report-july-2022/

[87] Y. Yazid, I. Ez-Zazi, A. Guerrero-González, A. El Oualkadi, and M. Arioua, "UAV-Enabled Mobile Edge-Computing for IoT based on AI: A comprehensive review," *Drones*, vol. 5, no. 4, p. 148, 2021.

[88] M. Mozaffari, W. Saad, M. Bennis, Y.-H. Nam, and M. Debbah, "A tutorial on UAVs for wireless networks: Applications, challenges, and open problems," *IEEE communications surveys & tutorials*, vol. 21, no. 3, pp. 2334–2360, 2019.

[89] W. Chen, B. Liu, H. Huang, S. Guo, and Z. Zheng, "When UAV swarm meets edge-cloud computing: The QoS perspective," *IEEE Network*, vol. 33, no. 2, pp. 36–43, 2019.

[90] N. Cheng, W. Xu, W. Shi, Y. Zhou, N. Lu, H. Zhou, and X. Shen, "Air-ground integrated Mobile Edge networks: Architecture, challenges, and opportunities," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 26–32, 2018.

[91] J. Liu, A. Zhou, C. Liu, T. Zhang, L. Qi, S. Wang, and R. Buyya, "Reliability-enhanced task offloading in Mobile Edge Computing environments," *IEEE Internet of Things Journal*, 2021.

[92] W. Zhang, L. Li, N. Zhang, T. Han, and S. Wang, "Air-ground integrated Mobile Edge Networks: A survey," *IEEE Access*, vol. 8, pp. 125 998–126 018, 2020.

[93] N. H. Motlagh, M. Bagaa, and T. Taleb, "UAV-based IoT platform: A crowd surveillance use case," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 128–134, 2017.

[94] A. Mas-Colell, M. D. Whinston, J. R. Green *et al.*, *Microeconomic theory*. Oxford university press New York, 1995, vol. 1.

[95] A. Bosch-Domènech and N. J. Vriend, "Imitation of successful behaviour in cournot markets*," *The Economic Journal*, vol. 113, no. 487, pp. 495–524, 2003.

[96] G. Nan, Z. Mao, M. Yu, M. Li, H. Wang, and Y. Zhang, "Stackelberg game for bandwidth allocation in cloud-based wireless live-streaming social networks," *Systems Journal, IEEE*, vol. 8, no. 1, pp. 256–267, March 2014.

[97] J. Elias, F. Martignon, A. Capone, and E. Altman, "Non-cooperative spectrum access in cognitive radio networks: a game theoretical model," *Computer Networks*, vol. 55, no. 17, pp. 3832–3846, 2011.

[98] H. Tembine, E. Altman, R. El-Azouzi, and Y. Hayel, "Evolutionary games in wireless networks," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 40, no. 3, pp. 634–646, 2010.

[99] P. D. Taylor and L. B. Jonker, "Evolutionary stable strategies and game dynamics," *Mathematical biosciences*, vol. 40, no. 1, pp. 145–156, 1978.

[100] S. D'Oro, L. Galluccio, S. Palazzo, and G. Schembra, "A game theoretic approach for distributed resource allocation and orchestration of softwarized networks," *IEEE Journal on Selected Areas in Communications, Special Issue on Game Theory for Networks*, vol. 35, no. 3, 2017.

[101] J. B. Rosen, "Existence and uniqueness of equilibrium points for concave n-person games," *Econometrica: Journal of the Econometric Society*, pp. 520–534, 1965.

[102] S. D'Oro, P. Mertikopoulos, A. Moustakas, and S. Palazzo, "Interference-based pricing for opportunistic multi-carrier cognitive radio systems," *IEEE Transactions on Wireless Communications*, vol. 14, no. 12, December 2015.

[103] P. Mertikopoulos and A. L. Moustakas, "The emergence of rational behavior in the presence of stochastic perturbations," *The Annals of Applied Probability*, vol. 20, no. 4, pp. 1359–1388, 2010.

[104] [Online]. Available: https://www.ettus.com/wp-content/uploads/2019/01/USRP_B200mini_Data_Sheet.pdf

[105] [Online]. Available: https://www.dji.com/matrice100/info

[106] G. Slawomir, "Next generation its implementation aspects in 5g wireless communication network," in *2017 15th International Conference on ITS Telecommunications (ITST)*, 2017, pp. 1–7.

[107] S. A. Rizzo and G. Scelba, "A hybrid global mppt searching method for fast variable shading conditions," *Journal of Cleaner Production*, vol. 298, p. 126775, 2021.

[108] K. Zheng, Q. Zheng, P. Chatzimisios, W. Xiang, and Y. Zhou, "Heterogeneous vehicular networking: A survey on architecture, challenges, and solutions," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2377–2396, 2015.

[109] Z. Kljaić, P. Škorput, and N. Amin, "The challenge of cellular cooperative its services based on 5g communications technology," in *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2016, pp. 587–594.

[110] T. Meisling, "Discrete-time queuing theory," *Operations Research*, vol. 6, no. 1, pp. 96–105, 1958.

[111] A. Lombardo, G. Morabito, and G. Schembra, "Modeling intramedia and intermedia relationships in multimedia network analysis through multiple timescale statistics," *IEEE Transactions on Multimedia*, vol. 6, no. 1, pp. 142–157, 2004.

[112] ETSI, "Multi-access Edge Computing (MEC); Study on MEC support for alternative virtualization technologies," https://www.etsi.org/deliver/etsi_gr/MEC/001_099/027/02.01.01_60/gr_mec027v020101p.pdf, 2019.

[113] M. R. Haider and M. M. Dongre, "Vehicular communication using 5g," in *2015 International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, 2015, pp. 263–266.

[114] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[115] F. Busacca, G. Faraci, C. Grasso, S. Palazzo, and G. Schembra, "Designing a multi-layer edge-computing platform for energy-efficient and delay-aware offloading in vehicular networks," *Computer Networks*, vol. 198, p. 108330, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128621003315

[116] S. Mohod and M. V. Aware, "Micro wind power generator with battery energy storage for critical load," *IEEE Systems Journal*, vol. 6, no. 1, pp. 118–125, 2012.

[117] S. Conti, G. Faraci, R. Nicolosi, S. A. Rizzo, and G. Schembra, "Battery management in a green fog-computing node: a reinforcement-learning approach," *IEEE Access*, vol. 5, pp. 21 126–21 138, 2017.

[118] https://github.com/FI-lab2021/Traffic_statistics/blob/main/traffic_statistics_CT.rar.

[119] M. M. Amiri and D. Gündüz, "Federated Learning over Wireless Fading Channels," *IEEE Transactions on Wireless Communications*, vol. 19, no. 5, pp. 3546–3557, 2020.

[120] S. Niknam, H. S. Dhillon, and J. H. Reed, "Federated Learning for Wireless Communications: Motivation, Opportunities, and Challenges," *IEEE Communications Magazine*, vol. 58, no. 6, pp. 46–51, 2020.

[121] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated Learning via Over-the-air Computation," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 2022–2035, 2020.

[122] M. Piva, G. Maselli, and F. Restuccia, "The Tags Are Alright: Robust Large-Scale RFID Clone Detection Through Federated Data-Augmented Radio Fingerprinting," in *Proceedings of the ACM International Symposium on Theory, Algorithmic Foundations, and*

Protocol Design for Mobile Networks and Mobile Computing (MobiHoc), 2021, pp. 41–50.

[123] S. D'Oro, F. Restuccia, and T. Melodia, "Can you fix my neural network? real-time adaptive waveform synthesis for resilient wireless signal classification," in *Proc. of IEEE Intl. Conf. on Computer Communications (INFOCOM)*, Vancouver, BC, Canada, May 2021.

[124] F. Restuccia, S. D'Oro, A. Al-Shawabka, M. Belgiovine, L. Angioloni, S. Ioannidis, K. Chowdhury, and T. Melodia, "DeepRadioID: Real-Time Channel-Resilient Optimization of Deep Learning-based Radio Fingerprinting Algorithms," *Proc. of ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2019.

[125] D. Uvaydov, S. D'Oro, F. Restuccia, and T. Melodia, "DeepSense: Fast wideband spectrum sensing through real-time in-the-loop deep learning," in *Proc. of IEEE Intl. Conf. on Computer Communications (INFOCOM)*, Vancouver, BC, Canada, May 2021.

[126] H.-H. Chang, H. Song, Y. Yi, J. Zhang, H. He, and L. Liu, "Distributive Dynamic Spectrum Access through Deep Reinforcement Learning: A Reservoir Computing Based Approach," *IEEE Internet of Things Journal*, 2018.

[127] B. Hamdaoui, B. Khalfi, and M. Guizani, "Compressed Wideband Spectrum Sensing: Concept, Challenges, and Enablers," *IEEE Communications Magazine*, vol. 56, no. 4, pp. 136–141, 2018.

[128] X. Jin, J. Sun, R. Zhang, Y. Zhang, and C. Zhang, "SpecGuard: Spectrum Misuse Detection in Dynamic Spectrum Access Systems," *IEEE Transactions on Mobile Computing*, vol. 17, no. 12, pp. 2925–2938, 2018.

[129] M. Li, D. Yang, J. Lin, and J. Tang, "SpecWatch: A Framework for Adversarial Spectrum Monitoring with Unknown Statistics," *Computer Networks*, vol. 143, pp. 176–190, 2018.

[130] S. Wang, H. Liu, P. H. Gomes, and B. Krishnamachari, "Deep Reinforcement Learning for Dynamic Multichannel Access in Wireless Networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 2, pp. 257–265, 2018.

[131] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the Limits of LoRaWAN," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 34–40, 2017.

[132] Semtech Corporation, "LoRa™ Modulation Basics," Camarillo, CA, USA, Tech. Rep., 2015.

[133] Semtech, "SX1272/3/6/7/8: LoRa Modem - Designer's Guide," may 2013.

[134] D. Garlisi *et al.*, "Interference cancellation for lora gateways and impact on network capacity," *IEEE Access*, vol. 9, pp. 128 133–128 146, 2021.

[135] D. P. Bertsekas, *Convex optimization algorithms*. Athena Scientific, 2015.

[136] T. Elshabrawy and J. Robert, "Evaluation of the BER Performance of LoRa Communication using BICM Decoding," in *2019 IEEE 9th International Conference on Consumer Electronics (ICCE-Berlin)*, 2019, pp. 162–167.

[137] D. Croce, M. Gucciardo, S. Mangione, G. Santaromita, and I. Tinnirello, "LoRa Technology Demystified: From Link Behavior to Cell-Level Performance," *IEEE Transactions on Wireless Communications*, vol. 19, no. 2, pp. 822–834, 2020.

[138] O. Afisiadis, A. Burg, and A. Balatsoukas-Stimming, "Coded lora frame error rate analysis," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[139] T. Berthold, *Heuristic algorithms in global MINLP solvers*. Verlag Dr. Hut, 2015.

[140] L. Liberti, "Undecidability and hardness in mixed-integer nonlinear programming," *RAIRO-Operations Research*, vol. 53, no. 1, pp. 81–109, 2019.