UNIVERSITÀ DEGLI STUDI DI PALERMO

Information And Communication Technologies
ING-INF/03-Telecomunicazioni
Dipartimento di Ingegneria

Challenges and opportunities in emerging high-density wireless networks

Ph.D. Candidate

Ing. Michele GUCCIARDO

Tutor

Prof. Ilenia TINNIRELLO

Coordinator

Prof. Alessandro BUSACCA

Internship Tutors

Prof. Vincenzo MANCUSO

Ing. Gian Michele DELL'AERA

XXXII CYCLE - ACADEMIC YEAR 2018-2019

Abstract

Wireless technologies have had an impressive growth in the last two decades. The latest evolutions of cellular and wireless local area networks provide Internet access to billions of users with connection speed and realiability comparable to wired networks. Besides, sensor networks and the emerging Internet of Things will make future wireless network scenarios highly dense in terms of both deployed access points and connected devices. Indeed, the ever-growing, wireless-enabled pervasiveness of smart devices in the most diverse spaces gave birth to concepts like smart city, smart health and smart industry. Such ecosystems leverage the massive amount of data gathered from the devices in order to provide a stratified set of applications: i) local environment monitoring and sensing; ii) raw data processing to generate broader systems insights; iii) real-time control of mission-critical processes. In order to meet the challenges of such complex scenarios and to seize the opportunities of the latest application domains, a shift towards flexible, context-aware and autonomously adaptive network architectures is needed. In this context, 5G ambition to integrate networking, computing and storage resources into one unified infrastructure is a key aspect. This vision can be enabled following three research directions. First, optimizing the deployment of *heterogeneous radio access technologies*, devised to respond to very different application requirements. Second, softwarizing the radio access networks for improving flexibility and enabling *centralized coordination schemes*. Third, employing *novel computing architectures* at the edge of the access networks for supporting ultra-reactive services.

In this thesis work we cover the road-map outlined by the 5G vision. In Chapter 2, we focus on the emerging Long Range (LoRa) access technology designed for low cost, low energy wireless devices to deploy in IoT scenarios. We first analyze LoRa modulation, both numerically and experimentally to demonstrate that collisions between packets modulated with the same Spreading Factor (SF) usually lead to channel captures, while different SFs can indeed cause packet loss if the interference power is strong enough. Then,

we model the effect of such findings to quantify the capacity in a typical LoRa cell. We show that high SFs can be severely affected by inter-SF interference and that different SF allocations within the cell may lead to significantly different results. In Chapter 3, we shift our attention to softwarization of cellular radio access networks, proposing a centralized Software Defined Networking (SDN) platform devised to control indoor femto-cells for supporting multiple network-wide optimizations and applications. In particular, we focus on an example localization application in order to enlighten the main potentialities of the approach. First, we demonstrate that the platform can be exploited for re-configuring operational procedures at the programmable femto-cells, thus enabling customized logics for collecting measurement reports from mobile devices. Second, we experimentally validate the measurement collection mechanism and the possibility to build indoor radio-maps. Finally we compare a k-nearest neighbor classifier and a neural network for supporting fingerprint-based localization, showing that the approach can lead to accurate to accurate positioning results. In Chapter 4, we present a network marketplace for the exchange of computing resources between wireless devices and different stakeholders such as Cloud computing service providers and Telecom operators. Indeed, the forecast diffusion of novel computing architectures deployed at the edge of the network, like Multi-Access Edge Computing (MEC) and Fog computing, can be exploited in order to support heavy computation applications with low latency requirements. Such a scenario is enabled by the presence of an innovative, centralized network element, referred as Broker, that couples autonomously demand and supply of computing resources. We explore the potentials of the proposed offloading architecture through a custom-made fluid network simulator. We compare basic decision policies and show the importance of assigning computing tasks according to the load distribution of network producers. We also evaluate the importance of keeping memory in the decision-making process of the Broker.

The work has been partially supported by the Italian Minister of Education, University and Research, within the research program *PON Ricerca e Innovazione*, in cooperation with the Italian cellular operator TIM and with the Spanish Research Institute IMDEA Networks.

Published Content

This thesis is based on the following published papers:

P1. D. Croce, D. Garlisi, **M. Gucciardo**, S. Mangione, G. Santaromita, I. Tinnirello, "*Impact of Spreading Factor Imperfect Orthogonality in LoRa Communications*", in Proceedings of Tyrrhenian International Workshop in Digital Communications, Mondello (PA), ITALY, Sept 18-20 2017.

- The authors are arranged alphabetically;
- This work is partially included and its content is reported in Chapter 2;
- All the authors equally contributed to the definition of the paper problem and approach. The author played a major role in the analysis of LoRa modulation.

P2. **M. Gucciardo**, I. Tinnirello, D. Garlisi, "*Demo: A Cell-level Traffic Generator for LoRa Networks*", in Proceedings of MobiCom 2017, Snowbird (UT), USA, Oct 16-20, 2017.

- The authors are arranged according to their contribution;
- The tool developed in this work has been used for the experimental evaluation of the content of Chapter 2;
- The author played a co-leader's role in the design and development of the traffic generator and in the writing of the paper as well.

P3. D. Croce, **M. Gucciardo**, S. Mangione, G. Santaromita, I. Tinnirello, "*Impact of LoRa Imperfect Orthogonality: Analysis of Link-level Performance*", in IEEE Communications Letters, vol. 22, no. 4, pp. 796-799, April 2018.

- The authors are arranged alphabetically;
- This work is fully included and its content is reported in Chapter 2;

- All the authors equally contributed to the definition of the paper problem and approach. The author wrote the modulation section of the paper.

P5. **M. Gucciardo**, I. Tinnirello, G. M. Dell'Aera and M. Caretti, "*A Flexible 4G/5G Control Platform for Fingerprint-based Indoor Localization*", IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Paris, France, 2019, pp. 744-749.

- The authors are arranged according to their contribution;
- This work if fully included and its content is reported in Chapter 3
- In this work the author played a leader's role in the design and development of the control platform, the applications and the testbed and a co-leader's role in the development of the soft-node, in the machine learning analysis and in the writing of the paper as well.

P5. D. Croce, **M. Gucciardo**, S. Mangione, G. Santaromita, I. Tinnirello, "*LoRa Technology Demystified: from Link Behavior to Cell-level Performance*", accepted for publication in IEEE Transactions on Wireless Communications, 2020.

- The authors are arranged alphabetically;
- This work is fully included and its content is reported in Chapter 2;
- All the authors equally contributed to the definition of the paper problem and approach. The author played a co-leader's role in the analysis of the LoRa cell capacity.

P6. D. Croce, **M. Gucciardo**, S. Mangione, G. Santaromita, I. Tinnirello, "*Performance of LoRa Technology: Link-level and Cell-level Performance*", accepted for publication in a chapter of the book LPWAN Technologies for IoT and M2M Applications, 2020.

- The authors are arranged alphabetically;
- This work is partially included and its content is reported in Chapter 2;
- All the authors equally contributed to the definition of the paper problem and approach. The author played a co-leader's role in the study of the LoRa cell capacity.

Contents

List of Tables

List of Figures

We are what we do, especially we are what we do for changing what we are.

[E. Galeano]

Never say never, because limits, like fears, are often just an illusion.

[M. Jordan]

Chapter 1

Introduction

Nowadays information and communication technologies are everywhere. Billions of devices are connected to the Internet through wired and wireless networks, getting closer and closer to Weiser's ubiquitous computing vision [1]. In particular, wireless communications have had an explosive growth in the last two decades. Cellular technologies and Wireless Local Area Networks (WLANs) have evolved to provide Internet radio access to mobile devices with improved connection speed and reliability. Besides, Wireless Personal Area Networks (WPANs) provide connectivity to sensors meant to collect environment data. Most recently, Low Power Wide Area Networks (LPWANs) have been designed to provide wide connectivity coverage to low cost, low energy wireless devices equipped with sensors. The advent of the Internet of Things (IoT) boosted the diffusion of a plethora of smart devices (whose number is expected to reach 18 billions by 2022 [2]) fueling concepts like smart city, smart health and smart industry. The simplest idea behind such smart ecosystems is to leverage the high density of both wireless devices and access networks in order to provide monitoring and sensing applications. Besides, the heterogeneous context data collected in different vertical/local systems can be further elaborated for producing enriched insights about horizontal/broader systems. Finally, more advanced use cases aim at real-time control over complex processes where mission-critical information must be elaborated and provided in ultra reliable and low latency fashion.

The fifth generation of cellular networks (5G) will play a central role in facing the challenges of such a scenario. 5G vision is to integrate networking, computing and storage resources into one programmable and unified infrastructure [3]. This ambitious target will be pursued following three main strategic directions:

- An heterogeneous set of radio access technologies will be employed in order to cover more application domains and to provide faster and more reliable connectivity. This will lead to more dense base station deployments with small and femto cells used in combination with macro cells.
- The softwarization of upper layers through Software Defined Networking (SDN) and Network Function Virtualization (NFV) for coping with the complexity of such a diversified access layer. These paradigms are meant to fluidify network operations, blurring the differences between wired and wireless networks [4].
- The native support of emerging computing architectures like Multi-access Edge Computing (MEC) and Fog computing in order to bring artificial intelligence closer to the devices.

In this thesis work, we cover some of the most important aspects of all three 5G research directions. In particular, we focus on scenarios characterized by an high-density of devices, base stations (or access points) or both. We believe that the challenge of designing novel and effective solutions for such complex scenarios represents also an opportunity to take a step forward towards the development of networks that autonomously adapt to context.

In Chapter 2, we consider Long Range (LoRa) technology, one of the most promising LPWAN technologies. LoRa is designed for IoT deployments with low cost devices with low energy and data rate requirements. Typical LoRa scenarios are characterized by large cells and heterogeneous application domains, which may lead to extremely high numbers of end devices. We provide two main contributions: a link-level characterization of LoRa modulation and then, exploiting such link-level properties, we provide a complete cell model study of multi-link LoRa systems. Regarding the first aspect, we characterize LoRa modulation both numerically and experimentally. We modified the software transceiver presented in [11] to generate synthesized LoRa modulated packets and transmit them through the well-known USRP software-defined-radio (SDR) platform. This transceiver is used to emulate collisions produced by different devices. Despite LoRa pseudo-orthogonal Spreading Factors (SF) are commonly considered as orthogonal, we show that collisions between packets of different SFs can indeed cause packet loss. The second main contribution of the Chapter is the development of a simple yet accurate analytical framework (built as a generalization of LoRa's Aloha channel access model) to model the performance of LoRa cells. We derive the aggregated capacity and data extraction rate of a LoRa cell working on a single frequency with one or multiple gateways by taking into account the heterogeneous probabilities of intra-SF and inter-SF collisions. This Chapter has been done in cooperation with the Ph.D. candidate Giuseppe Santaromita and it is also included in his thesis.

In Chapter 3, we consider 5G indoor scenarios characterized by a very high number of wireless devices. We envision that the increasing density of devices is due to not only mobile devices like smartphones or smart wearables, but also to the presence in the environment of smart appliances (e.g., televisions, electric plugs, thermostats) in fixed positions. On one side, the high-density of devices is a challenge that can be be faced increasing the number of femto-cells deployed in indoor spaces. On the other side, such a pervasive distribution of smart objects represents an opportunity to build radio-maps of the environment. In this scenario, we argue that a centralized localization system, able to collect measurement reports from several observation points at regular time intervals, can be very promising for implementing localization functions based on radio fingerprinting. We provide two main contributions: a SDN control platform for indoor femto-cells and a localization application based on machine learning. The platform has both the capability to gather directly from LTE's Access Stratum (AS) protocols a number of network status parameters and to actively modify some aspects of the soft-nodes in real-time.

In Chapter 4, we focus on novel methodologies for computational offloading. The scenario envisioned is characterized by an high-density of wireless devices that need computing resources in order to run heavy-computation applications. Besides, we consider the wide variety of offloading services already offered by Cloud service providers and the last frontiers represented MEC and Fog computing. In this context, we propose a network marketplace where computing resources can be bought from different network players able to produce such resources (e.g., telecom operators, over-the-top companies, etc.). The resources can be exchanged between network nodes, in the form of tasks with specific requirements and constraints. Such a market is enabled by an intelligent, centralized network element that acts as a broker between resource consumers (i.e., the wireless devices) and resource producers. We explore the potentials of the proposed offloading architecture through a fluid simulator that we developed in Python. We compare basic decision policies and show the importance of assigning tasks according to the load distribution of network producers.

Chapter 2

Long Range technology for the IoT

2.1 Introduction

In recent years, we have assisted to an impressive proliferation of wireless technologies and mobile-generated traffic, which is now the highest portion of the total internet traffic [5]. Such proliferation has been characterized by high-density deployments of base stations (based on heterogeneous technologies, such as 4G cellular base stations and WiFi Access Points), and a pervasive diffusion of wireless devices, not limited to traditional user terminals. Indeed, with the advent of Internet-of-Things (IoT) applications, many smart objects, such as domestic appliances, cameras, monitoring sensors, etc., are equipped with a wireless technology.

In this Chapter we consider the emerging Long Range (LoRa) technology, which represents a critical example of wireless technology working in high-density scenarios. Indeed, LoRa has been conceived for Low Power Wide Area Networks (LPWANs), characterized by low data rate requirements per single device, large cells and heterogeneous application domains, which may lead to extremely high numbers of end devices (EDs) coexisting in the same cell. For this reason, LoRa provides different possibilities to orthogonalize transmissions as much as possible – Carrier Frequency (CF), Spreading Factor (SF), Bandwidth (BW), Coding Rate (CR) – and provides simultaneous collision free communications. However, despite the robustness of the LoRa PHY [6] patented by Semtech, in WAN scenarios where multiple gateways can be installed, the scalability of this technology is still under investigation

[7]. Current studies are mostly based on the assumption that using multiple transmission channels and spreading factors leads to a system that can be considered as the simple super-position of independent (single channel, single SF) sub-systems [8]. This is actually a strong simplification, especially because the SFs adopted by LoRa are pseudo-orthogonal [9] and therefore, in near-far conditions, collisions can prevent the correct reception of the overlapping transmissions using different SFs.

For characterizing these phenomena, we provide two main contributions: a link-level characterization of LoRa modulation (based on our previous work [10]) and then, exploiting such link-level properties, we provide a complete cell model study of multi-link LoRa systems. Regarding the first aspect, we characterize LoRa modulation experimentally, showing that collisions between packets of different SFs can indeed cause packet loss. We modified the software transceiver presented in [11] to generate synthesized LoRa modulated packets and transmit them through the well-known USRP software-defined-radio (SDR) platform. This transceiver is used to emulate, in a controlled and repeatable manner, collisions produced by different devices: the modulated LoRa signals are first generated in software, then summed together (with tunable power level difference) to replicate a given superposition of LoRa signals and finally the obtained combined radio signal is transmitted over the air. We use this traffic generator to experimentally characterize the performance of a commercial LoRa device, under intra-SF and inter-SF collisions caused by multiple simultaneously active LoRa links. We quantify the power difference for which capture effects and packet loss occur, for all combinations of SFs. Our experimental results show that the co-channel rejection thresholds are on average an order of magnitude higher than the theoretical ones presented in [13], with values as high as -8 dB. These poor co-channel rejection thresholds might be insufficient in common LoRa application scenarios (the received power of two radio signals can easily differ by tens of dB), thus contradicting the common belief that pseudo-orthogonal SFs can be considered as orthogonal in practice.

The second main contribution of the Section regards the capacity analysis of a LoRa cell under realistic link behaviors: we propose a simple yet accurate analytical framework to model the performance of LoRa cells, deriving the aggregated capacity and data extraction rate of a LoRa cell working on a single frequency with one or multiple gateways. The framework has been built as a generalization of the Aloha model (the channel access protocol used in LoRa), by taking into account the heterogeneous probabilities of intra-SF and inter-SF collisions, due to the specific position of the target

ED (which translates in a specific received power at the gateway). The models provide excellent results, closely following the simulations obtained with LoRaSim [14] and with our own custom Matlab simulator. Our analysis demonstrates that capture effects and imperfect orthogonality of SFs can significantly affect the cell capacity. In particular, we show that more robust SFs, usually envisioned for EDs experiencing strong channel attenuations, can be severely affected by inter-SF interference and therefore, their usage could consume a large fraction of cell resources without real benefits. Also, we show that power control and packet fragmentation can be counterproductive. Finally, we quantify the performance increase obtained by deploying multiple gateways and we show that it might be best to deploy them at the edge of the cell more than on a regular grid.

The rest of the Chapter is organized as follows. After a brief review of literature work about LoRa performance evaluation in Section 4.2, we provide a background description of LoRa modulation and a characterization of link-level performance in Section 2.3. The analysis of cell capacity is presented in Section 2.4, while in Section 2.5 we extend our model in case of non-uniform allocation of SFs, power control and packet fragmentation. Finally, we analyze the capacity improvements achievable with multiple gateways and the performance impact of topology in Section 2.6 and conclude our Section in Section 3.5.

2.2 Related Work

LoRa Technology. Since LoRa is quite a recent technology, apart from general descriptions of LoRa applications [15] and implications for IoT scenarios, relatively few works have already been published on the evaluation of LoRa link-level performance or cell capacity. Link-level studies are mostly based on the experimental characterization of coverage and interference rejection capabilities of the LoRa PHY, patented by Semtech [16]. LoRa modulation is presented in [17], together with an implementation of a software LoRa transceiver, called *gr-lora*, based on Software-Defined-Radio. The Section in [13] quantifies the power reception thresholds for different modulation formats and the Signal-to-Interference-Ratio (SIR) required for rejecting interfering LoRa signals, modulated with different spreading factors. However, no justification about the derivation of these numbers is provided and, as we will show, their theoretical results are very different from our experimental ones. In [9] link-level performance of LoRa are compared with ultra narrowband (Sigfox-like) networks, concluding that ultra narrowband has a larger coverage but LoRa networks are less sensitive to interference.

Capacity results of LoRa cells are based on the characterization of link behaviors. In [7], the authors experimentally demonstrate the capture phenomena between colliding LoRa frames, and then use these results for quantifying the cell capacity in simulation [14], in terms of maximum number of devices that can be served with a desired data extraction rate. The simulator assumes that it is enough a power ratio of 6dB between colliding frames to correctly demodulate the one received at the highest power, and that multiple transmissions with different spreading factor can be considered as perfectly orthogonal [8]. This last assumption is a strong simplification, because the spreading factors adopted by LoRa are pseudo-orthogonal [9] and, as we will show, inter-SF collisions can indeed be an issue in near-far conditions, when the interferer is close to the receiver. Despite this simplification, in [7] the authors show that a typical deployment can support only 120 EDs per 3.8 ha, although performance can be improved by using multiple gateways (i.e. by increasing the probability of capture events). The cell capacity is compared with pure Aloha networks, while no model is provided for characterizing the impact of channel captures. In [18], authors use a LoRa cell performance model in a scenario with high number of devices and propose solutions to improve its performance, but no detail is provided about the model.

An important aspect affecting the cell capacity is the SF allocations within the cell. LoRa technology defines a default Adaptive Data Rate (ADR) scheme for deciding the SF to be used by each device, based on the selection of the minimum SF compatible with the available link budget. Such a choice guarantees that each device achieves the highest possible data rate with local decisions. Another approach proposed in [19] assigns the SFs by also considering load balancing between different SFs: this improves the cell capacity, although the authors ignore inter-SF interference. The work in [20] presents a SF allocation scheme which jointly works on SF and power allocation in a single cell scenario, for achieving the optimal device distribution presented in [21]. Also in this case, the scheme acts without considering the capture effect and by assuming orthogonal SFs. Finally, the use of multi-channel MAC protocols could be used to improve the cell performance, as proposed in [27]. The results presented in this paper are valid also in case multiple channels are used, for analyzing the performance of the single channels separately.

Capture models. Capture effects can significantly boost the performance of random access schemes, because some collision events may result in the

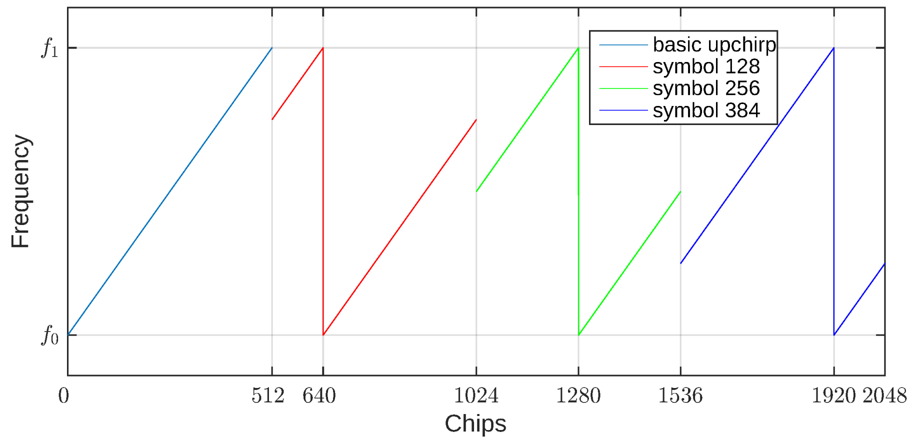


Figure 2.1. Modulating signal with $SF = 9$ for one basic upchirp and three symbols: 128, 256 and 384.

correct reception of the strongest signal. For modeling this phenomena, different approaches have been considered so far, based on evaluation of the interfering power, colliding times, channel fades, etc. The characterization of the interfering power, which in turns depends on the number of interfering signals, can be used for estimating channel captures when the SIR of a target frame is higher than a capture threshold [22]. Alternatively, it is possible to map the highest possible interference level into a vulnerability circle, out of which interfering signals do not affect packet reception [23]. Despite its simplicity, this approach leads to good results when the network works in stable conditions. Therefore, we chose to follow this approach, by generalizing the concept of vulnerability circle to both intra-SF and inter-SF interfering signals.

2.3 LoRa link-level behavior

2.3.1 LoRa modulation and demodulation

LoRa modulation is derived from *Chirp Spread Spectrum (CSS)*, which makes use of *chirp signals*, i.e. frequency-modulated signals obtained when the modulating signal varies linearly in the range $[f_0, f_1]$ (upchirp) or $[f_1, f_0]$ (downchirp) in a symbol time T. Binary modulations, mapping 0/1 information bits in upchirps/downchirps, have been demonstrated to be very robust against in-band or out-band interference [6]. LoRa employs a M-ary modulation scheme based on chirps, in which symbols are obtained by considering different circular shifts of the basic upchirp signal. The temporal shifts, characterizing each symbol, are slotted into multiples of time $T_{chip} = 1/BW$, called chip, being $BW = f_1 - f_0$ the bandwidth of the signal. It results that the modulating signal for a generic n-th LoRa symbol can be expressed as:

$$f(t) = \begin{cases} f_1 + k(t - n \cdot T_{chip}) & \text{for } 0 \leqslant t \leqslant n \cdot T_{chip} \\ f_0 + k(t - n \cdot T_{chip}) & \text{for } n \cdot T_{chip} < t \leqslant T \end{cases}$$

where $k = (f_1 - f_0)/T$ is the slope of the frequency variations. The total number of symbols (coding i information bits) is chosen equal to 2^i, where i is called spreading factor. The symbol duration T required for representing any possible shift is $T = 2^i \cdot T_{chip} = 2^i/BW$. It follows that, for a fixed bandwidth, the symbol period and the temporal occupancy of the signal increase with larger SFs. Fig. 2.1 shows the modulating signal used for a basic upchirp and three examples of circular shifts obtained for a SF equal to 9: the symbol time is 512 T_{chip}, while the three exemplary shifts code the symbols 128, 256 and 384.

The preamble of any LoRa frame is obtained by sending a sequence of at least eight upchirps followed by two coded upchirps, used for network identification (sync word), and two and a quarter base downchirps. Payload data are then sent by using the M-ary modulation symbols. LoRa provides three BW settings (125, 250 or 500 kHz) and seven different SF values (from 6 to 12). In general, a larger bandwidth translates in a data rate increase and a receiver sensitivity deterioration. Conversely, higher spreading factors can be used for improving the link robustness at the cost of a lower data rate. For demodulation, the received LoRa signal is synchronously multiplied to the base downchirp. This results in a signal comprising only two frequencies: $f_n = -kn \cdot T_{chip}$ and $f_n - BW = -(f_1 - f_0) - kn \cdot T_{chip}$. Both frequencies will be aliased to the same frequency f_n by downsampling at the rate BW. The estimated symbol index $\hat{n}$ corresponds to the position of the peak at the output of an iFFT, as described in [13].

An interesting feature of LoRa modulation is the orthogonality of signals modulated under different spreading factors, which can be exploited for enabling multiple concurrent transmissions. Although perfect orthogonality is guaranteed only in case of exact synchronization of the transmitters, the cross-energy between two signals modulated with different spreading factors is almost zero, regardless of the starting of the symbol times. Then the cross-energy is not exactly equal to zero although it reaches very low values. In general, considering two LoRa signals modulated with different spreading

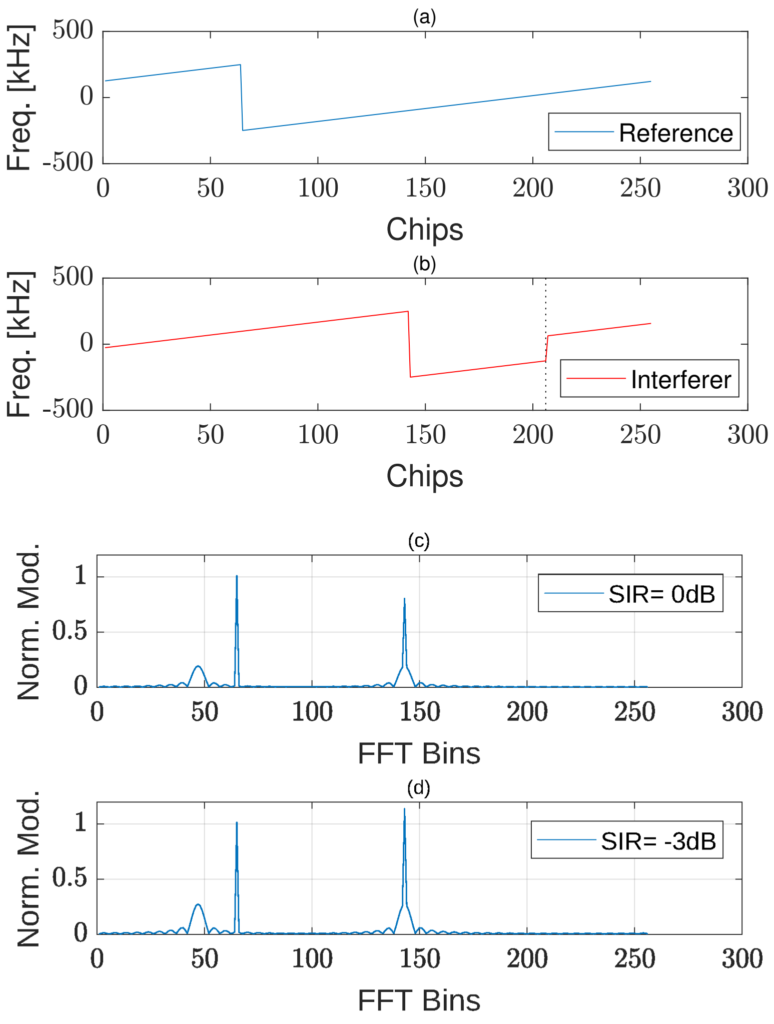


Figure 2.2. An example of capture effect within signals modulated with same SF 8. A LoRa reference symbol (a) and two partially overlapping interfering symbols (b) are received at different SIR levels. The iFFT output after multiplication with the base SF 8 downchirp and downsampling shows the highest peak for the perfectly synchronized reference symbol and two lower peaks for the partially overlapping symbols (c) but a SIR of -3 dB is enough to overcome the reference signal (d).

factors, say s_1 and s_2, the cross-energy between them is not equal to zero:

$$E_{s_1,s_2}(\tau) = \int_0^T s_1(t) \cdot s_2(t-\tau)^* dt \simeq 0 \qquad (2.1)$$

where T is the symbol period of the signal with the highest spreading factor.

In case of collisions with other LoRa symbols, we can distinguish two different scenarios, depending on the interfering spreading factor SF_{int}. First, if the SF_{int} is the same as the one the receiver is listening for, the above receiver will observe multiple peaks at the output of the iFFT. Indeed, assuming that the two transmissions are received at the same power and that the reference signal is perfectly synchronized with the receiver, the iFFT will show a maximum peak corresponding to the reference symbol and two smaller peaks corresponding to two partially overlapping interference symbols, with different height depending on the transmitted symbols and on the offset with the receiver. For example, Fig. 2.2 shows two signals modulated with same SF 8 and bandwidth 500 kHz: the reference symbol (Fig. 2.2-a) and two partially overlapping interfering symbols (Fig. 2.2-b). As depicted in Fig. 2.2-c, when the signals are received with the same power, the iFFT output after multiplication with the base downchirp and downsampling, shows the highest peak for the synchronized reference symbol (index $\hat{n} = 64$) and two lower peaks for the partially overlapping symbols (index $\hat{n_1} = 96$ and $\hat{n_2} = 192$, with shift of $0.2T$ – i.e. 51.2). However, a Signal to Interference Ratio (SIR) of -3dB is enough for the interfering signal to overcome the reference signal and "capture" the channel (Fig. 2.2-d). This means that LoRa exhibits a very high capture probability within the same SF.

Second, when the SF_{int} is different from the one the receiver is interested in, after multiplication with the base downchirp and downsampling, the interfering signal will still be a chirped waveform, resulting in a wide-band spectrum with low spectral density. An example is illustrated in Fig. 2.3, where one signal modulated with SF equal to 9 (Fig. 2.3-a) is overlapped to two symbols modulated with SF 8 (Fig. 2.3-b), circularly shifted to desynchronize them with the reference symbol (the dotted lines represent the boundaries of the symbols). At the receiver, when the two signals are received with same power, the iFFT output after multiplication with the base SF 9 downchirp and downsampling shows a clear peak corresponding to the reference SF 9 symbol index $\hat{n} = 128$ (Fig. 2.3-c), while this is not the case when the SIR is too low because of the non perfect SF orthogonality (Fig. 2.3-d). In this scenario, the co-channel rejection is much higher (≈ -20dB in the figure).

2.3.2 LoRa PHY coding

Up to now, we have neglected the impact of bit coding schemes. Indeed, the patented LoRa PHY includes several mechanisms to make the system more robust to interference. After transmitting the preamble, both header and

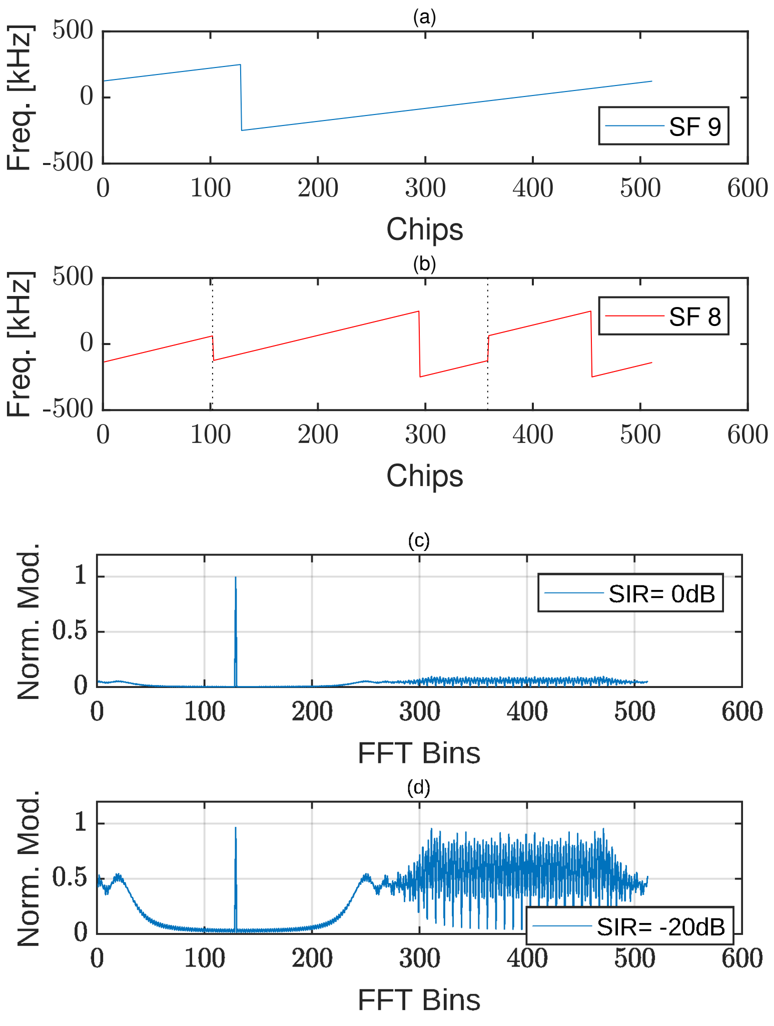


Figure 2.3. An example of collision between signals modulated with different SF. A LoRa symbol modulated with SF equal to 9 (a) and two overlapping and circularly shifted interfering symbols with SF 8 (b) are simultaneously received at different SIR levels. The iFFT output after multiplication with the base SF 9 downchirp and downsampling shows a clear peak when the two signals have the same power (c) while this is not the case when the SIR is too low (d).

payload bits of LoRa frames are mapped to symbols by a pipeline of processing operations, which include: Hamming coding[1], whitening, shuffling

[1]The Hamming codes used in LoRa have a coding rate between 4/5 and 4/8, and can reveal or correct at most one error on the AWGN channel.

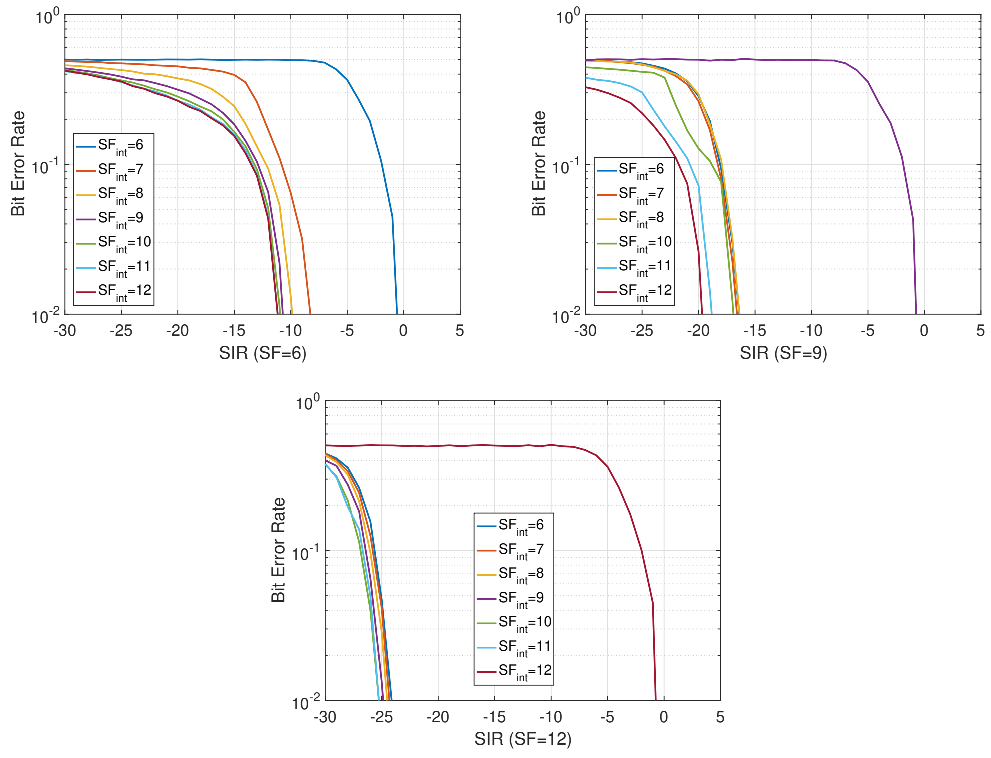


Figure 2.4. BER of three different spreading factors in function of the SIR.

& interleaving, and gray coding. These operations have been specifically designed for increasing robustness towards synchronization errors or narrowband interference, which can be a serious issue for CSS-based modulations. In fact, in case of synchronization errors or narrowband interference, the receiver described in the previous section will most probably mistake the transmitted symbol, mapped to frequency f_n after the iFFT, for one of the immediately adjacent symbols. Since gray coding ensures that adjacent symbols are mapped to bit patterns differing in one position only, the receiver is able to identify the less reliable bits (at most two bits) of each received symbol. The purpose of the LoRa interleaver is spreading unreliable bits among several codewords, thus enabling even the 4/5 Hamming code (consisting in a simple parity check) in exhibiting a significant channel coding gain.

In order to understand if Gray coding has an impact also on inter-SF interference, we tried to characterize the distance between the transmitted symbol and the decoded one in presence of inter-SF collisions. To this purpose, we extended our MATLAB implementation with Gray encoding and quantified such distance in our simulation. For example, Fig. 2.5 shows the histogram

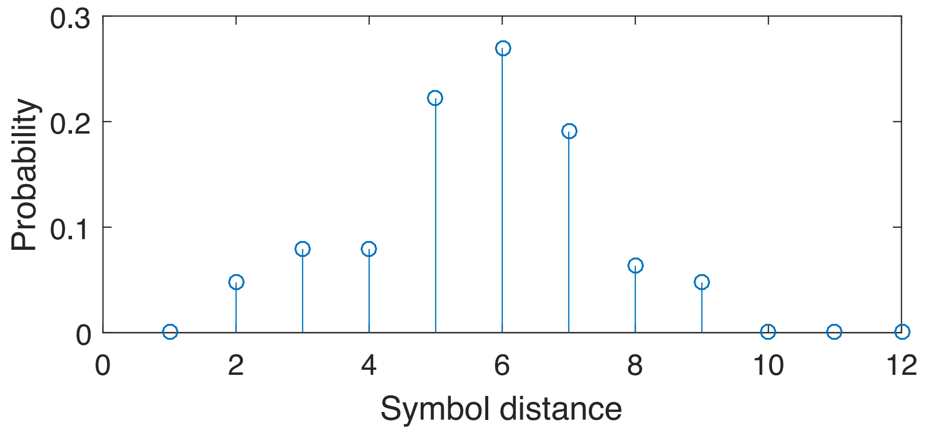


Figure 2.5. Histogram of the error distance when a LoRa transmission at $SF = 12$ (with Gray encoding enabled) is interfered by another transmission with different $SF = 8$ at SIR=-25dB.

of the Hamming distance of the decoded symbol from the transmitted one, when a LoRa transmission at SF equal to 12 (with Gray encoding enabled) is interfered by another transmission with SF equal to 8 and SIR=-25dB. From the figure it is clear that the error distance probability approximates a Binomial distribution (and is not concentrated around the adjacent symbol). Thus, LoRa PHY coding mechanisms cannot protect from collisions.

2.3.3 Imperfect orthogonality analysis

MATLAB simulations

To quantify the co-channel rejection, including the impact of PHY coding, we implemented a LoRa modulator and demodulator in MATLAB, based on [12] and [13]. We performed a number of simulations for testing the reception of two overlapping transmissions modulated with different SFs, after Hamming coding at rate 4/7, interleaving and Gray encoding. Our goal is identifying a SIR threshold below which the demodulation of the received frame is affected by errors. In each simulation run, we created an overlapped signal by summing the reference frame, modulated with a reference spreading factor SF_{ref}, with a number of random interfering symbols, modulated with a different spreading factor SF_{int} (with an equivalent time on air). We assumed the transmitter to be perfectly synchronized with the receiver, while the interference frame is randomly shifted in time for de-synchronizing the interfering symbols. The amplitude A_{ref} of the reference signal is set to one,

Table 2.1. SIR thresholds in MATLAB simulations.

SF_{ref} \ SF_{int}	7	8	9	10	11	12
7	0	-11	-13	-14	-14	-14
8	-13	0	-14	-16	-17	-17
9	-17	-16	0	-17	-19	-20
10	-19	-19	-19	0	-20	-22
11	-22	-22	-22	-22	0	-23
12	-24	-24	-25	-25	-25	0

whereas the amplitude A_{int} of the interferer is a tunable value depending on the SIR, i.e. $A_{\text{int}} = \sqrt{10^{-SIR/10}} \cdot A_{\text{ref}}$. The resulting combined signal has been then processed by the MATLAB demodulator, in absence of noise on the channel. For each simulation run, we randomly generated interfered packets until the occurrence of 100 total error events. Packets are transmitted with SF_{ref} and include 20 Bytes of data and a zero padding up to an integer number of interleaving blocks. This signal is interfered by a random LoRa-like signal modulated with SF_{int}, with a random time-offset and a SIR increasing from -30dB with 1dB steps. A Bit Error Rate (BER) statistic has then been obtained by comparing the demodulated bits with the modulated ones.

Fig. 2.4 shows the results of these simulations for three different SF_{ref} values as an example. The curves represent the error probability for one selected SF_{ref} against all the possible SF_{int}. From the figure, it can be easily recognized that for each SF_{ref} there exists a minimum SIR threshold below which the success probability starts degrading (high BER). Furthermore, the smaller the interfering SF, the higher the SIR threshold required for obtaining an acceptable BER.

Table 2.1 summarizes the SIR thresholds leading to a BER of approximately 1%. In the table, we also consider the case when the interfering signal has the same SF of the reference signal. As also documented in the Semtech specifications, LoRa modulations achieve a very high probability of capture effects even with low SIR values (0dB for the different SFs in our simulations, versus 6dB specified in [24]). In other words, it is very likely that in case of collisions between two signals modulated with the same SF, the strongest signal can be correctly demodulated. Note that, as the BER curves are very steep, the corresponding Packet Error Rate (PER) thresholds result very similar.

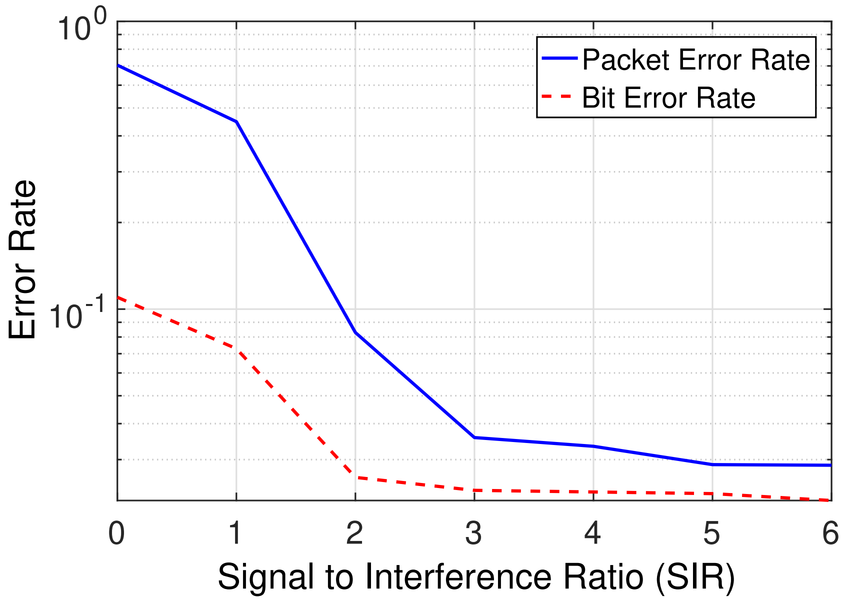


Figure 2.6. Error rate of the SX1272 transceiver for reference and interferer streams modulated with $SF = 7$.

USRP experiments

For validating the thresholds found with the MATLAB simulator, we performed a number of experiments on real LoRa links. To this purpose, we used a Semtech SX1272 transceiver, controlled by an Arduino Yun, for characterizing the behavior of a commercial receiver in presence of collisions. We implemented a LoRa synthesizer able to encode, modulate and generate the I/Q samples of a real LoRa packet, which can be easily transmitted over the air with a USRP B210 board through GNU radio. With this LoRa synthesizer, we generated two traces (one for the interferer and one for the reference LoRa link) for each combination of SFs, composed of a stream of 20 byte-long packets (for the reference SF) and adjusting the payload length of the interfering SF to match the length of the reference signals. The offset of each interfering packet, overlapped in time to the packets of the reference link, has been randomly selected within a window which guarantees that the two packets collide for at least one symbol. We filled the payload of all frames with randomly generated bytes, except for the two bytes that specify the destination address and the payload length. In particular, we assigned the destination address of the SX1272 receiver only to the packets of the reference link. This allows the receiver to discard the interfering packets when they are modulated with the same SF of the reference ones. Finally, we scaled the amplitude of the interfering packet stream to achieve the desired SIR and added it to the reference stream. This correctly models the channel effects when both the reference and interfering transmitters are experiencing a LOS propagation (or NLOS with only one resolvable path), with minimal (or negligible) frequency selective fading[2]. For each couple of SF_{ref} and SF_{int}, the resulting combined stream was transmitted through the USRP towards the SX1272, thus emulating the traffic generated by two different transmitters.

Fig. 2.6 shows the error rate of the receiver when both the interferer and the reference packets are modulated with SF equal to 7. We can observe that, if the power of the reference stream is at least 3 dB higher than the interferer, the PER is below 2%. The BER, instead, is very low also when the interferer and the reference packets have equal power. Furthermore, we can observe that the PER cannot be simply obtained as $1-(1-BER)^{P\cdot 8}$, being P the number of transmitted bytes, because only a sub-set of symbols are corrupted by the overlapping interfering signal due to the random overlapping of packet transmissions. The results of the experiments are summarized in table 2.2, for a subset of reference and interfering SF combinations. The table shows that the SIR thresholds for correct demodulation are similar to the ones obtained in MATLAB simulations and very different (over 10 dB – an orders of magnitude) lower than the ones in [13], with values as low as -8 dB[3]. Such power difference between two radio signals can easily appear in common LoRa application scenarios, thus contradicting the common belief that different SFs can be considered as orthogonal in practice.

Table 2.2. SIR thresholds with SX1272 transceiver.

SF_{ref} \ SF_{int}	7	8	9	10	11	12
7	1	-8	-9	-9	-9	-9
8	-11	1	-11	-12	-13	-13
9	-15	-13	1	-13	-14	-15
10	-19	-18	-17	1	-17	-18
11	-22	-22	-21	-20	1	-20
12	-25	-25	-25	-24	-23	1

[2]In this scenario we verified that, by emulating a multipath channel, selective fading has an impact on the SIR thresholds of about 1 or 2 dB.

[3]In table 1 of [13], the lower triangular part follows the law $10 \cdot \log_{10}(2^{SF_{\text{ref}}})$, i.e. the SIR thresholds are equal to the spreading gain of a matched filter receiver over an AWGN channel. However, this result is unrealistic, because the receiver does not work by comparing the mean squares of the signals and the interfering signal is not a white process (see for example figure 2).

2.4 Analysis of Single Cell Capacity

The brief description of the LoRa PHY presented in section 2.3 enlightens two important aspects that have to be considered for studying the real capacity of LoRa cells: i) the possibility of correctly receiving a packet, in case of collision with other packets modulated with the same SF; ii) the possibility that multiple SFs are not exactly orthogonal and therefore do not work as independent multiple channels. In this section we show that both these aspects have a strong effect on the uplink cell capacity, because of the simple access scheme used in LoRa, which is basically a non-slotted Aloha mechanism (without carrier sense). We derive some simple expressions for predicting LoRa uplink capacity in presence of a single gateway, in terms of average throughput, by generalizing the classical Aloha results in presence of channel captures and imperfect orthogonality between SFs. We also compare our capacity models with simulation results obtained by using a custom Matlab simulator, which we also validated against the LoRaSim simulator [14] used by the authors of [7] (which we warmly thank for publishing the source code). Unless specified otherwise, the parameters used for configuring the reference LoRa cell are summarized in table 2.3.

2.4.1 LoRa cell capacity with ideal links.

LoRa cells work as non-slotted Aloha systems. Under Poisson packet arrivals, the throughput of an ideal non-slotted Aloha cell is $G \cdot e^{-2G}$, being G the normalized load offered in the cell, i.e. the amount of data transmitted in the unit time by the EDs over nominal channel capacity. The Data Extraction Rate (DER), i.e. the probability of correctly receiving a packet transmission – a typical parameter for characterizing LoRa systems–, is given by e^{-2G}. In ideal conditions, since different SFs are available, the system works as the super-position of multiple coexisting (but independent) Aloha systems, each one experiencing the load due to the EDs employing a given SF equal to i (with $i \in \{7, 12\}$).

Let $G_i = \lambda_i \cdot ToA_i$ be the load offered in the cell sub-system working with SF i, which depends on the packet arrival rate λ_i and packet transmission time ToA_i (also called Time on Air or airtime). The ToA_i values change significantly from one SF to another. Indeed, the time interval required for transmitting a packet is given by the sum of the preamble time, which lasts n_{ph} symbol times T, and the payload transmission time. Since each symbol codes i bits and a channel coding with rate $CR = 4/(4 + RDD)$ is

Table 2.3. Simulation parameters.

Parameter	**Value**
Carrier Frequency	868.0 MHz
Transmission Power	14 dBm
Bandwidth	500 kHz
Code Rate	4/5
Message size	20 Bytes
Message Period	90000 ms
Number of EDs	[50-2000]
Path loss attenuation exponent	4
Simulation time	9000 s

applied (with redundancy bits $RDD = 1, ..., 4$), the time ToA_i required for transmitting a frame long P bytes with SF i can be expressed as $(n_{ph} + \lceil \frac{P\cdot 8}{i\cdot 4} \rceil \cdot (4+RDD))\cdot T$. Thus, the total uplink capacity results equal to $\sum_{i=7}^{12} G_i e^{-2G_i}$ and can be dramatically reduced (down to zero) as the loads G_i increase (up to infinity).

Obviously, in a real cell the number n_i of EDs working on a given SF i is generally high but finite, and λ_i can be evaluated as $n_i \cdot s$, being s the source rate of each ED (which we assume to be constant for all devices). In absence of capture effects and inter-SF interference, the cell capacity is affected solely by the number of EDs configured on each SF, but it does not depend on the spatial distribution of the EDs within the cell (provided that all EDs are in the coverage range of the gateway).

2.4.2 Impact of channel captures

Consider first the case when collisions are only due to frames using the same SF (i.e. different SFs are perfectly orthogonal) and the spatial distribution of the devices is uniform in the whole cell area. As discussed in Section 2.3, LoRa modulation is very robust to interfering signals, and therefore it is very likely that the frames colliding at a given gateway result in the correct reception of the strongest one. For quantifying the performance improvements due to these events, we assume that in most practical cases a target ED can be interfered by a single colliding signal at a time. As we will show, this assumption is reasonable when the cell works in stable, non-congested conditions.

Capture effects have been observed when the reception power of the colliding signal is sufficiently lower than the power of the target ED (i.e. the

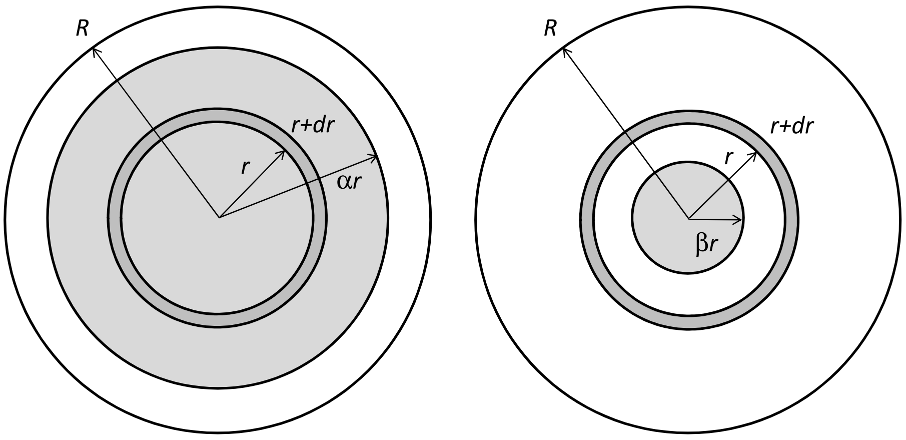
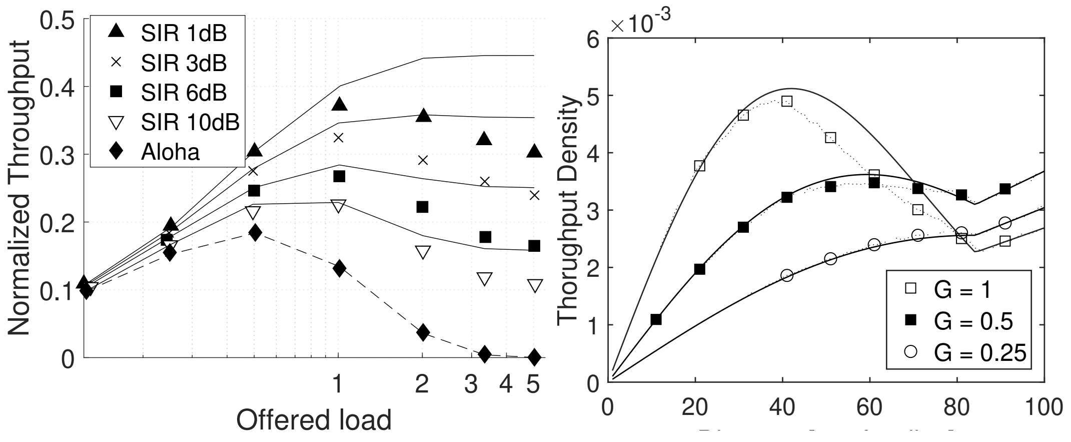


Figure 2.7. Traffic competing with receivers placed in the circular ring between r and $r + dr$ (dark gray area): distribution of intra-SF (left cell) and inter-SF (right cell) competing load.

SIR of the target ED in dB is higher than a positive threshold, which we experimentally found equal to 1 dB, as shown in table 2.2). For a transmitter located at distance r from the gateway, neglecting the effect of random fading and assuming uniform transmission power among EDs and an attenuation law of type $r^{-\eta}$, this capture condition can be mapped into the placement of the interfering ED in a circular ring delimited by a distance αr, with $\alpha = 10^{SIR/10\eta} > 1$, and the cell radius R. An exemplary representation of this area in which the interfering ED does not prevent the reception of the target ED is shown in the left-most cell of Fig. 2.7 in white. Obviously, when αr is higher than R, such a region does not exist, because the transmitter is too far from the gateway and captures cannot occur. It follows that a target ED employing a given SF i is actually competing with a fraction of the total load G_i, which corresponds to the ratio between the area of the circle of radius $\min(\alpha r, R)$ and the total area of the cell. The smaller the α coefficient, the smaller the real competing load is. Note that in case of severe attenuation, higher values of η translate to lower values of α and higher chances of channel captures. Without loss of generality, in our numerical experiments we use $\eta = 4$.

To model the performance of LoRa in presence of channel captures, we extend the basic non-slotted Aloha model exploiting the above considerations: assuming for simplicity that all frames have fixed size with transmission time ToA_i, the throughput $S_c(G_i)$ in presence of captures obtained using SF i can be quantified by considering that the load offered in the circular ring between



(a) (b)

Figure 2.8. Impact of single packet interference approximation on throughput with channel captures: theoretical model (solid lines) and simulation results including multiple packet interference (markers).

distance r and $r + dr$ will compete with the fraction $\min(\alpha^2 \cdot r^2/R^2, 1)$ of G_i:

$$S_c(G_i) = 2\pi \int_0^R \delta_i e^{-2\min(\frac{\alpha^2 r^2}{R^2},1)\cdot G_i} r \cdot dr \tag{2.2}$$

where $\delta_i = G_i/(\pi R^2)$ is the density of load offered to SF i. It results:

$$S_c(G_i) = \frac{1}{2\alpha^2}\left(1 - e^{-2\cdot G_i}\right) + G_i\left(1 - \frac{1}{\alpha^2}\right)e^{-2\cdot G_i}$$

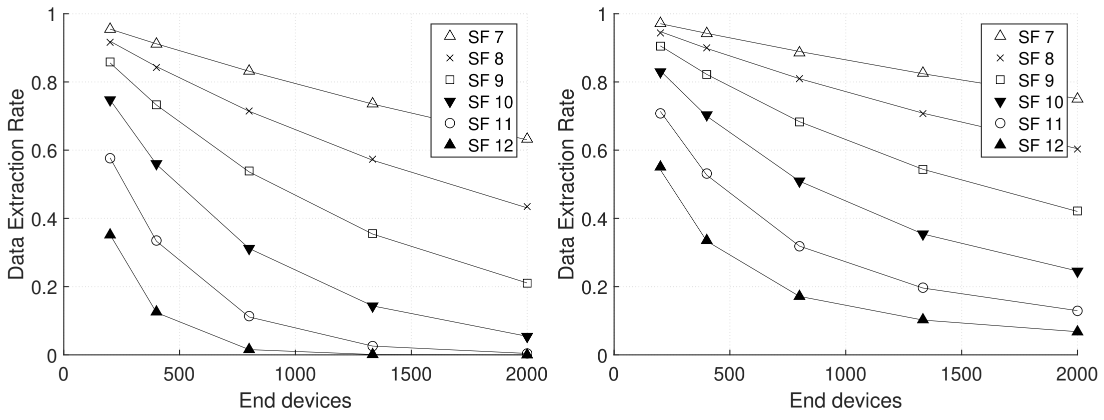
and the DER can then be obtained as $S_c(G_i)/G_i$.

In the previous derivation, we generically refer to a uniform load density δ_i, while in real cells we have a finite number n_i of EDs usually placed at fixed positions. However, we can generally consider that n_i is sufficiently high and the throughput derivation can refer to the average results obtained in different realizations of node placements.

Fig. 2.8-a shows the throughput curves obtained by equation 2.2 as a function of the offered load G_i, for several SIR values (i.e. α values). In the figure, the lines correspond to the analytical model (theoretical results), while the marker points represent the simulation results obtained with our Matlab simulator, in a simulation run of 9000 seconds. For deriving the average throughput results in simulation, we varied the offered load by adjusting the source rate of 1000 EDs and we randomly generated the position of each ED at each transmission attempt. The figure clearly shows that capture effects can significantly increase the maximum Aloha efficiency (up to about 300% for a capture SIR of 1dB when the offered load G_i is 1). Note that, our model works well in non-congested operating scenarios ($G_i < 1$), while it diverges from simulations in highly congested conditions, in which collisions involve multiple transmitted frames. Indeed, the asymptotic capacity of real systems tends to zero with the increase of the traffic load, while the model asymptotic value is different from zero (namely, it is equal to $1/2\alpha^2$).

To better visualize the effects of the load on the capacity approximation, Fig.2.8-b shows the throughput density (i.e. the integral argument in equation 2.2) achieved by EDs uniformly placed within the cell, as a function of the distance from the gateway (from 0 to 100% of the cell radius). Different curves refer to different G_i values: simulation results are shown with a dotted line and points, while our model results are plotted with solid lines. When the distance is higher than R/α and no capture effect is possible, we can easily recognize that the throughput density follows a linear distribution, because all EDs have the same success probability and the number of EDs considered in the integral grows proportionally to the distance. This last segments of the curve coincide with the throughput density of Aloha whose complete distribution is a linear distribution of the same slope in the whole interval $[0, R]$. For smaller distances, the capture effects can significantly increase the throughput density of Aloha. The figure also shows how our model overestimates the capture probability as the normalized load G_i increases: indeed, for $G_i = 0.25$ the points are perfectly overlapped with the solid line, while for $G_i = 1$ there is a region of the cell in which the real throughput is smaller than the one predicted.

Finally, we evaluated the DER achieved in a cell where all the nodes are configured on the same SF with and without the capture effects (for a capture SIR of 1 dB). Fig. 2.9 shows the DER results as a function of the number n of EDs configured on each SF, ignoring inter-SF interference. Each curve refers to an independent cell sub-system with the same number of EDs but different load conditions. The results demonstrate that the DER can increase significantly thanks to channel captures. For example for $n = 800$ EDs, using SF 10 the DER increases from about 0.32 in case of pure Aloha to about 0.52 in presence of captures. Obviously, for a fixed number of EDs n, lower SFs have the best DER because the resulting offered load is lower.



(a) Aloha (no capture) (b) Capture

Figure 2.9. Simulation (markers) and analytical model (lines) results for channel capture effect.

2.4.3 Impact of imperfect orthogonality

In order to quantify the impact of imperfect orthogonality among SFs on the cell capacity, we reasoned similarly to the previous section, by considering a single interfering signal at time. Because of imperfect orthogonality, a target ED working on SF i at a generic distance r will compete not only with the load G_i offered to the same SF, but also with a fraction of the load G_{-i} working with a SF different from i. Such a fraction corresponds to the EDs closer to the gateway, which generate an interfering signal whose power is much higher than the desired signal and exceeds the rejection capability of the LoRa receiver. Our experimental results showed that the SIR threshold under which interference rejection does not work is almost independent on the SF used by the interfering signal. Therefore, the minimum required SIR value can be mapped into the placement of the interfering ED in a cell sub-region delimited by a radius $\beta_i \cdot r$, with $\beta_i = 10^{SIR/10\eta} < 1$, as shown in Fig. 2.7 (right).

The analysis of collisions between frames transmitted with different SFs requires taking into account that frames have heterogeneous transmission times, even if they have a fixed size P. Consider first a simple scenario in which channel captures with frames transmitted with the same SFs are not possible. In this case, interfering signals are given by the totality of transmissions performed with SF i and with the fraction $\beta_i r^2/R^2$ of transmissions performed with other SFs. The success probability of a target ED employing

SF i depends on the probability of finding the channel free from other interfering signals when starting frame transmission and during the following time interval ToA_i. The probability Pr_s of finding the channel idle at the starting time of frame transmission is given by the probability that no packet arrival at SF i is originated within a previous interval lasting ToA_i, while a fraction $\beta_i r^2/R^2$ of other packets employing a different SF $k \neq i$ has not started a transmission in an interval corresponding to the relevant frame transmission time ToA_k. Such a probability can be expressed as:

$$Pr_s(r) = e^{-\lambda_i ToA_i - \frac{\beta_i r^2}{R^2}\sum_{k\neq i}\lambda_k ToA_k} = e^{-G_i} \cdot e^{-\frac{\beta_i r^2}{R^2}G_{-i}}$$

being $G_{-i} = \sum_{k\neq i} G_k$ the total load offered by SFs different from i. The probability Pr_e that no other interfering signal is started until the end of the transmission time ToA_i can be expressed as:

$$Pr_e(r) = e^{-(\lambda_i + \frac{\beta_i r^2}{R^2}\sum_{k\neq i}\lambda_k)ToA_i} = e^{-G_i} \cdot e^{-\frac{\beta_i r^2}{R^2}G^*_{-i}}$$

with $G^*_{-i} = \sum_{k\neq i}\lambda_k \cdot ToA_i = \lambda_{-i}ToA_i$. The total throughput obtained on sub-channel i with imperfect orthogonality between SFs can be computed by integrating the success probability experienced at each distance r over all possible distances as:

$$S_{qo}(G_i, G_{-i}) = 2\pi \int_0^R \delta_i Pr_s(r) \cdot Pr_e(r) r \cdot dr =$$

$$= e^{-2G_i} \cdot 2\pi \int_0^R \delta_i e^{-\frac{\beta_i^2 r^2}{R^2}(G_{-i}+G^*_{-i})} r \cdot dr \qquad (2.3)$$

It results:

$$S_{qo}(G_i, G_{-i}) = G_i e^{-2\cdot G_i} \frac{1 - e^{-\beta_i^2 (G_{-i}+G^*_{-i})}}{\beta_i^2 \cdot (G_{-i} + G^*_{-i})} \qquad (2.4)$$

which is obviously smaller than the ideal orthogonal case $S_o(G_i, G_{-i}) = G_i e^{-2\cdot G_i}$.

Finally, if we want to take into account both the capture effects and the imperfect orthogonality of SFs, we can follow the same approach discussed so far and specify that the competing load for each target ED working on SF i is the sum of a fraction $l = \min(\alpha^2 r^2/R^2, 1)$ of the intra-SF load and a fraction $\beta_i^2 r^2/R^2$ of the inter-SF load:

$$S_{qo,c}(G_i, G_{-i}) = 2\pi \int_0^R \delta_i e^{-\frac{\beta_i^2 r^2}{R^2}(G_{-i}+G^*_{-i})} e^{-2lG_i} r \cdot dr$$

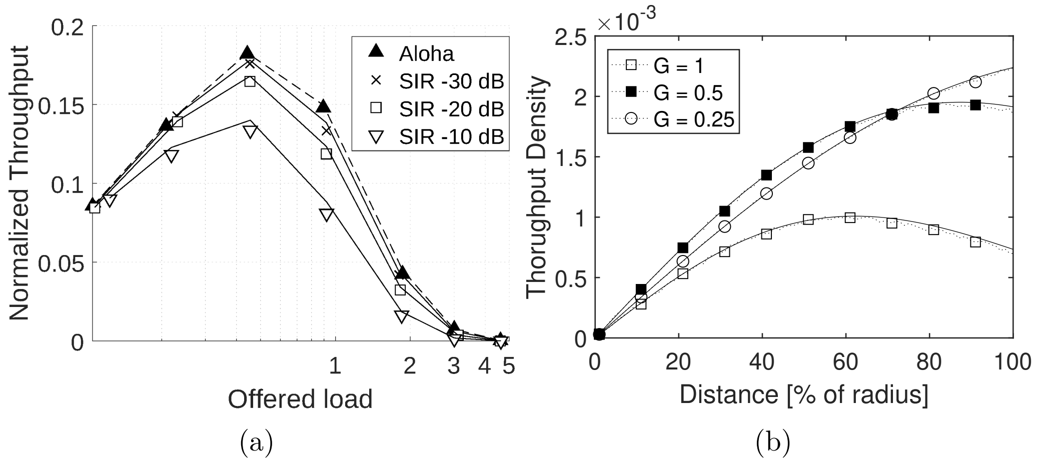


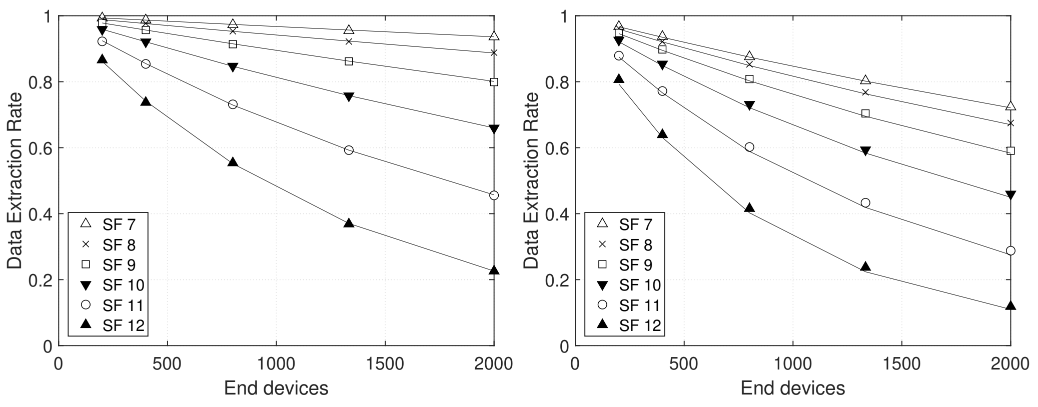
Figure 2.10. Impact of single packet interference approximation on throughput with non orthogonal SFs: model approximation (lines) and simulation results including multiple packet interference (markers).

which leads to:

$$S_{qo,c}(G_i, G_{-i}) = \frac{G_i e^{-2G_i}\left(1 - e^{-2G_i-(G_{-i}+G^*_{-i})\alpha^2/\beta_i^2}\right)}{2\alpha^2 G_i + \beta_i^2(G_{-i}+G^*_{-i})} + \frac{G_i e^{-2G_i}}{\beta_i^2(G_{-i}+G^*_{-i})}\left(e^{(G_{-i}+G^*_{-i})\alpha^2/\beta_i^2} - e^{(G_{-i}+G^*_{-i})\beta_i^2}\right) \tag{2.5}$$

The average DER can then be computed by dividing the throughput with the total offered load.

To show the impact of inter-SF collisions, fig. 2.10-a shows the theoretical throughput curves (lines) obtained by equation 2.4 (no captures) for different values of the rejection SIR (i.e. β_i values), together with simulation results (points), when two different SFs coexist in the same cell (namely, SF 7 and SF 9). The curves refer to the throughput of EDs configured on SF 7. Also in this case, for deriving the average throughput results in simulation, we considered a fixed number of 1000 EDs (half configured on SF 7 and half on SF 9), varying the ED positions at each transmission attempt and tuning the source rate with increasing load. As shown in the figure, despite the pseudo-orthogonality of the SFs, the throughput can indeed be severely affected compared to the ideal Aloha without inter-SF collisions (almost 50% reduction in case of congested scenarios, i.e. $G_i \approx 1$). Fig. 2.10-b shows the throughput density (i.e. the integral argument in equation 2.3) achieved by



(a) Orthogonal SFs (b) Interfering SFs

Figure 2.11. Simulation (markers) and analytical model (lines) results for interfering SFs.

EDs using 2 SFs with a rejection SIR of -10 dB and uniformly placed within the cell, as a function of the distance from the gateway (from 0 to 100% of the cell radius). Different curves refer to different G_i values: for low load conditions and small distances, the throughput density follows an almost linear distribution typical of Aloha with uniform EDs. However, for higher load conditions the curves can significantly deviate from a linear function, especially at high distances where inter-SF interference can be more critical.

Fig. 2.11 shows the DER results as a function of the total number N of EDs active in the cell, under the assumption that such a number is equally shared between different SFs (i.e. each SF is assigned to $n_i \approx N/6$ EDs) and same source rate s for all EDs. Again, markers represent simulations and lines the analytical results, which are still remarkably close to the simulations. From the figure, it is clear that the impact of non-orthogonality can be severe. The performance deteriorates quickly as the number of EDs increases.

2.5 Cell configurations

In this section, we discuss the impact of different cell configurations that can be considered for optimizing the cell capacity when multiple SFs are available. For sake of presentation, we consider two SFs only, namely SF $i <$ SF j, but generalizations to multiple SFs are straightforward. From our previous considerations, it is evident that LoRa cell performance is a function of the arrival rates of packets working on SF i and SF j, but also of their placement within the cell, because the inter-SF and intra-SF interference power experienced in case of collisions also depends on the relative position of the EDs.

2.5.1 Load balancing

Assuming that a total arrival rate of Λ pk/s is uniformly distributed within the cell, the arrival rates λ_i and λ_j experienced in each SF (with $\lambda_i + \lambda_j = \Lambda$) depend on the SF selected for each packet transmission. The selection of the certain SF may be a local decision, such as the selection of the highest possible rate compatible with the link budget available at a given spatial position, or may be extended with load balancing considerations. If the link budget at the cell border is enough for transmitting at the highest rate corresponding to SF i, the first approach would lead to $\lambda_i = \Lambda$ and $\lambda_j = 0$. Load balancing can significantly improve the overall cell capacity by reducing the load on SF i and by exploiting the additional capacity available on SF j.

A balancing solution devised to provide fair performance to devices working on different SFs is equalizing the offered load $G_i = G_j$. Note that this is different from equalizing the number of transmissions performed at each SF, because packet transmission times vary as a function of the employed SF. Specifically, load balancing is achieved for $\lambda_i \cdot ToA_i = \lambda_j \cdot ToA_j$, i.e. $\lambda_i = \Lambda \cdot \frac{ToA_j}{ToA_j + ToA_i}$. This implies that in a real cell with a finite number N of EDs, if all EDs work with uniform source rates, the same proportion is applied for deriving n_i as $N \cdot \frac{ToA_j}{ToA_j + ToA_i}$. However, perfect load balancing is not always feasible, because some EDs distant from the gateway can be forced to work on the most robust SF for guaranteeing that the received power is above the reception threshold.

Fig. 2.12-a compares the DER obtained in a cell with a total number N of EDs and two SFs available (namely, SF 7 and SF 9). Simulation results are shown with markers and model results with lines. The figure shows results for different criteria on SF allocations: equally sharing the number of EDs between SF 7 and SF 9 (dashed lines) or allocating a number of EDs proportionally to the relevant airtimes, with n_7 about four times of n_9 (solid lines). From the figure, it is evident that the second choice, i.e. load balancing, can be an effective solution for providing a similar DER to all devices, regardless of the allocated SF. However, DER performance are not exactly the same because inter-SF interference is not symmetrical due to different rejection thresholds (i.e. β_i and β_j coefficients) and transmission times (which result in $G^*_{-i} < G^*_{-j}$ even in case of load balancing). Overall,

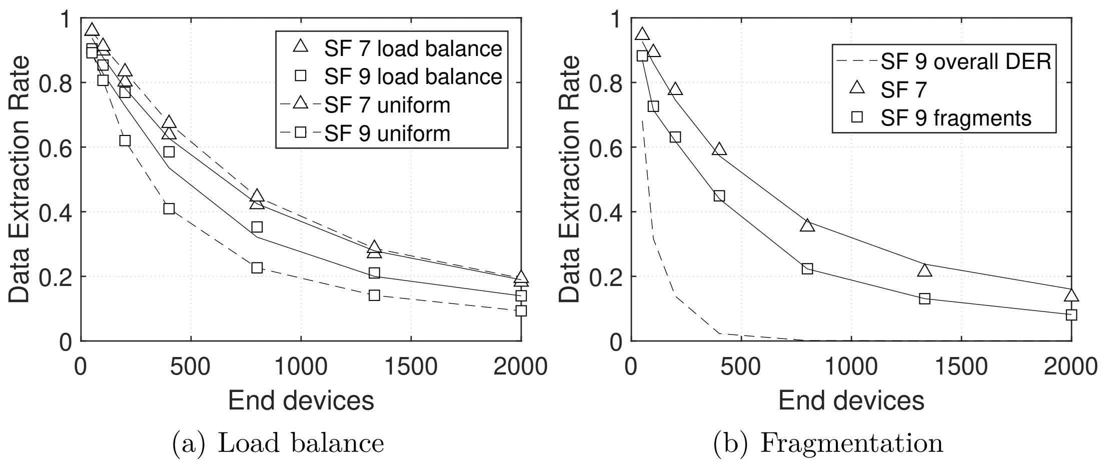


Figure 2.12. Impact of load balancing (left) and fragmentation (right) to increase fairness among the SFs.

the most robust SF j suffers an higher inter-SF competing load.

An additional mechanism to be considered for improving the fairness of the system could be the use of fragmentation. The idea is to equalize the airtimes of packets transmitted at different SFs. Obviously, in such a case, the arrival rates of fragments grow proportionally to the number of per-packet fragments. Fig. 2.12-b shows the performance results in the same scenario described for Fig. 2.12-a, with n_7 about four times n_9, and 4 fragments are used for packets transmitted with SF 9. From the figure, it is clear that fragmentation is not effective for equalizing the performance of the EDs, due to the additional overhead added to each fragment. Additionally, we have to consider that now four fragments are required for reassembling a single packet at SF 9. Thus, the overall DER (dashed line) is worse than the previous case. This is also due to the fact that LoRa technology does not easily support selective re-transmissions of corrupted fragments, because the downlink channel from the gateway to the EDs would result congested by the transmission of the feedback messages. Therefore, fragmentation without selective re-transmissions does not bring benefits to the network.

2.5.2 Spatial allocations

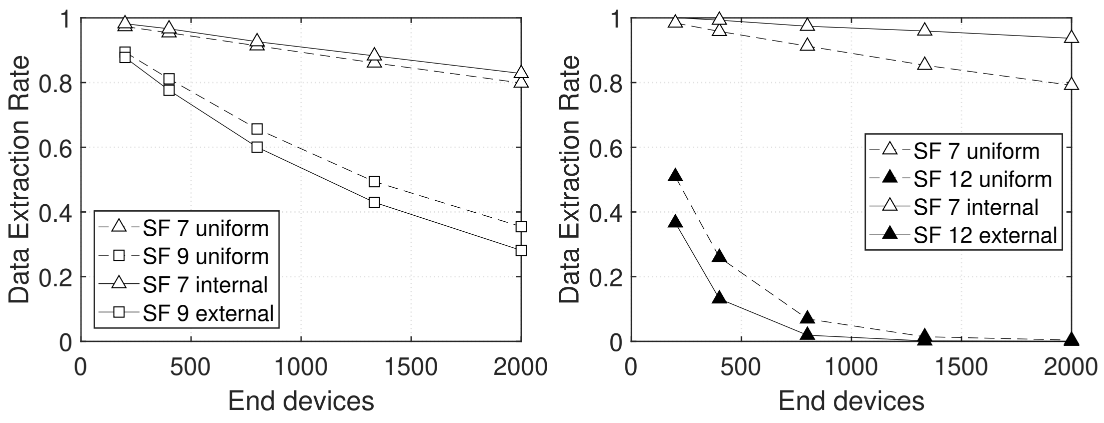
For a fixed number of EDs n_i (or n_j) to be configured on SF i (or SF j), different allocation choices are possible. EDs working on the same SF can be selected uniformly within the whole cell area or can be restricted to a specific area of the cell. According to the position of the selected nodes, each allocation policy can be mapped into the opportunistic definitions of the $\delta_i(r)$ and $\delta_j(r)$ functions. In order to predict the cell capacity resulting from a specific allocation, we can generalize previous throughput derivations for dealing with non uniform load density functions.

As an illustrative example of model extension under generic $\delta_i(r)$ functions, we consider the case in which nodes employing different SFs are placed in different sub-regions of the cell, rather than being uniformly spread in the whole area. A choice could be allocating the most robust SF to far users, placed in a circular ring between distance $d > 0$ and R, and the less robust SF to users closer to the gateway within a maximum distance d, in order to maximize the reception margin of each ED. The distance d can be chosen for achieving the desired load balancing. In such a case, the density functions of nodes employing different SFs can be defined as: $\delta_i(r) = G_i/(\pi d^2)$ when $r \leq d$ (and 0 otherwise), and $\delta_j(r) = G_j/(\pi R^2 - \pi d^2)$ when $r < d$, and 0 otherwise.

Allocating EDs in circular rings have different implications for both the inter-SF and the intra-SF interference. Regarding the first aspect, by neglecting the fading effects, it never happens that users employing spreading factor SF i are interfered by users employing SF j with a higher interfering power (being these users deterministically located at higher distances). In other words, the throughput achieved on spreading factor SF i is given by $S_i = G_i \cdot e^{-2 \cdot G_i}$, i.e. the density of potentially interfering signals is equal to zero as in case of perfect orthogonality. Conversely, users employing spreading factor SF j are more likely affected by interference generated by users employing spreading factor SF i, because in case of collisions the interfering signals are concentrated in the cell area closer to the gateway, which results in a interference density higher than the previous case. Being $\delta_{-i}(r) = \sum_{j \neq i} \delta_j(r)$ and $\delta^*_{-i}(r) = \sum_{j \neq i} \delta_j(r) \frac{ToA_i}{ToA_j}$, the throughput in absence of channel captures becomes:

$$\begin{aligned} S_{qo}(i) &= e^{-2\pi \int_0^R 2\delta_i(r) r \cdot dr} \cdot \\ &\cdot 2\pi \int_0^R \delta_i(r) \left[e^{-2\pi \int_0^{\beta_i r} (\delta_{-i}(t) + \delta^*_{-i}(t)) t \cdot dt} \right] r \cdot dr \end{aligned} \tag{2.6}$$

Figures 2.13-a and 2.13-b compare the performance of SF 7 when competing with SF 9 and SF 12 respectively, in scenarios with uniform allocation of SFs in the cell (dashed lines) or when far distance EDs are using the highest SF 9 and SF 12, while SF 7 is used for EDs close to the gateway. The figures



(a) (b)

Figure 2.13. Performance of SF9 (a) and SF12 (b) when competing with SF7. Comparison between uniform distribution of EDs (dashed lines) and when higher SFs are allocated to far away EDs (solid lines). The inter-SF SIR threshold is 10 dB and $\eta = 4$.

show that the performance of the higher SFs is deteriorated when allocating them to faraway EDs, while SF 7 performance improves because of the absence of inter-SF collisions. This, demonstrates that allocating higher SFs to far distance EDs is detrimental more than beneficial, because the performances are highly affected by closer devices while fading has a much lower impact. In other words, although higher SFs improve the robustness to fading and allow longer distances, this comes at the cost of an increased airtime of the transmitted frames and, since LoRa uses the Aloha access protocol, collisions arise quickly when increasing the offered load. Clearly, using low SFs for long distances might cause unnecessary retries, or even no packet delivery at all in case the link budget is not sufficient.

SF allocation has also an impact on the capture probability. For a given number n_i of EDs working on SF i, the highest capture probability is achieved when nodes are spread in the whole cell (rather than in a circular ring), because this choice corresponds to the spreading of the RSSI values of potentially colliding signals (which may result in the correct reception of the strongest signal). In the limit case in which all nodes working on SF i are at the same distance from the gateway, no capture effect can occur.

2.5.3 Power control

Another important configuration parameter of LoRa cells is the transmission power of EDs, which can be tuned by means of specific control messages sent by the gateway. The message specifies the power to be used in terms of a reduction margin to be applied to the maximum possible power (which may vary in different countries); the reduction margin is coded in steps of -2dB from 0 to -14dB. The tuning of the transmission power can be considered for reducing the energy consumption of devices which are close to the gateway, but also for mitigating the impact of inter-SF interference, Indeed, orthogonality of different SFs can be guaranteed in case the difference between the reception powers of EDs working at different SFs is lower than the minimum margin in table I (about -8dB).

To model the impact of power control on the LoRa cell performance, we can consider that each power reduction applied to a specific ED is equivalent to moving the device at an higher distance from the gateway. If η is the propagation loss coefficient, every step of -2dB corresponds to a distance increment of a factor equal to $\gamma = 10^{2/10\eta}$. In case power control is used for equalizing the reception power of the EDs, taking into account that only -14dB are available at steps of -2dB and the distance r_{min} of the closer ED, we could equivalently consider that no ED is placed at a distance lower than $d = 10^{14/10\eta} \cdot r_{min}$ and that EDs originally placed within the circular area of radius d are moved in the circular ring between d and $d \cdot \gamma$. In other words, power control can be evaluated again by working on the definition of the load density function $\delta_i(r)$ as follows: 0 when $r < d$, $G_i/(\pi R^2) \cdot [1 + 1/(\gamma^2 - 1)]$ when $r \in [d, d \cdot \gamma]$, and $G_i/\pi R^2$ when $r > d \cdot \gamma$. Equivalently, we can define $\delta_j(r)$ and derive the cell capacity S_{qo} of each SF applying equation 2.6.

As an illustrative example, we consider a cell using only SF 7 and SF 9, with half of the EDs using SF 7 are placed close to the gateway and the others using SF 9 are far away from the gateway. For clearness of presentation, Fig. 2.14 reports separately in two plots the DER achieved by the two SFs, although in the experiment both SFs are used as explained above. From the figure, it is evident that when equalizing the received power, the DER of SF 9 is close to the Aloha model (no channel captures) and much lower than the DER obtained without power control, while for SF 7 there is almost no change. This means that tuning the transmitted power of the EDs in order to equalize the received power at the gateway is detrimental more than beneficial for performance because, in case of collision between packets of the same SF, it reduces the probability of capturing the channel.

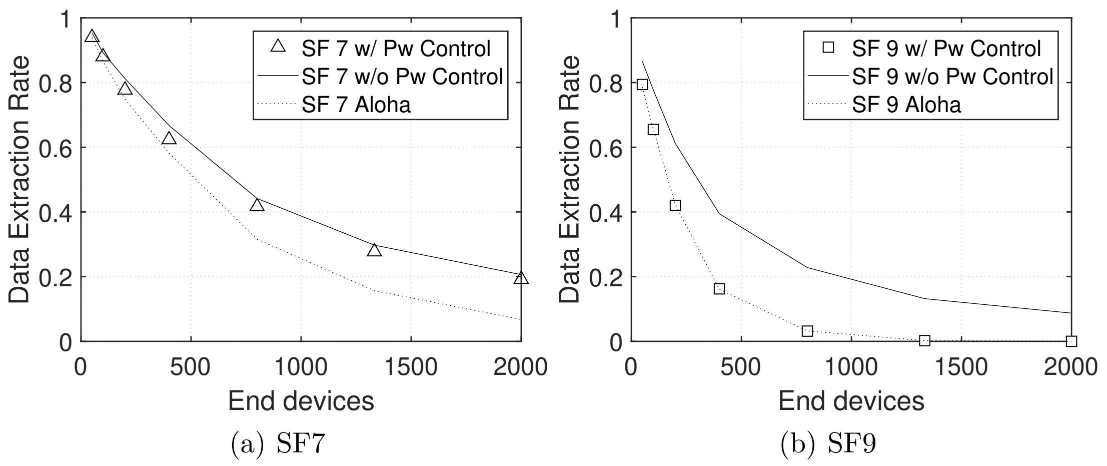


Figure 2.14. DER with two interfering SFs with and without using transmission power control.

2.6 Cell Capacity with Multiple Gateways

Thanks to the capture effect, the capacity of a LoRa cell can be increased by deploying multiple gateways. Indeed, each gateway sees a given ED with a different distance and therefore, in case of collisions, experiences a different power ratio between the strongest received packet and the interfering signals. When the power ratios are higher than the capture threshold, the collisions can result in the correct reception of a number of packets equal at most to the number of gateways. Obviously, it may also happen that the same packet is correctly received by multiple gateways, but all the packets are forwarded to a common network server, which discards duplicated packets.

For deriving the average cell capacity in presence of multiple gateways, it is necessary to specify the positions of the gateways within the cell, because the capture probability depends on the relative distance between the potential receivers (i.e. the gateways) and the transmitters. Differently from the single gateway case, even under the assumption that only a colliding signal is experienced at a given time, the EDs competing with a target transmitter are different for each gateway and distributed in regions which cannot be modeled as simple circle areas. For example, Fig. 2.15 shows the scenario of a cell with two gateways, $\alpha = 1$ and uniform distribution of EDs: the target transmitter in position C experiences different competing loads at each gateway, which are proportional to the intersection area between a circle of radius αr_m and the cell, being r_m the distance between the ED and a generic

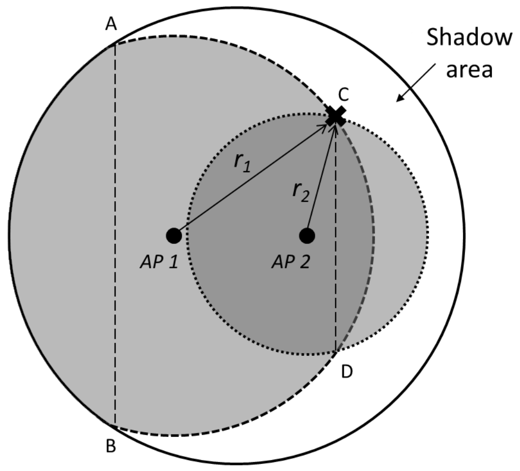
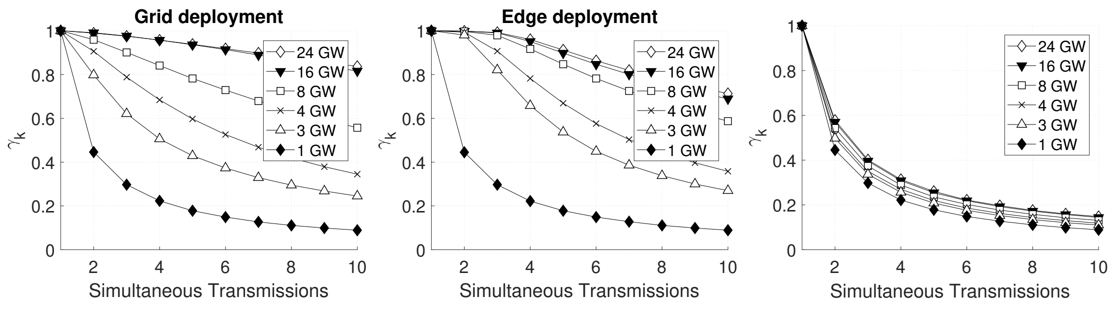


Figure 2.15. Competing area (gray) in presence of multiple gateways and "shadow area" created by the ED in position C.

gateway $m = 1{,}2$. Although the areas can be computed as circular areas delimited by the chords AB and CD, it is not easy to generalize the approach proposed in Section 2.4, because the performance of the target transmitter cannot be averaged as a function of the distance from the cell center, being also affected by the direction (i.e. by the specific position within the cell).

For a specific placement of the gateways, we modeled the geometry of the cell under uniform distribution of the EDs and complete coverage (i.e. assuming that EDs can transmit to a gateway placed at distance $2 \cdot R$) by numerically evaluating the average probability γ_k that a target ED successfully transmits its packet to at least one gateway, in presence of $k \geq 0$ interfering EDs. By considering one interfering signal at time as in the single gateway derivation, such a probability has been computed by averaging (on all possible transmitter positions) the probability that, for at least one gateway, k interfering EDs are at distances higher than α times the distance of the target transmitter. The analysis of the capture probability in presence of k interfering EDs allows us the decoupling between the geometric effects (due to the specific gateway placement) and the interference probability (due to the cell load). In other words, assuming that the γ_k coefficients are known, the cell capacity in presence of multiple gateways can be computed for any possible load G_i as:

$$S_c(i) = G_i \cdot \left[\sum_{k=0}^{\infty} \left(\gamma_k \frac{(2G_i)^k \, e^{-2G_i}}{k!} \right) \right]. \tag{2.7}$$



(a) Grid (b) Edge (c) Center

Figure 2.16. Impact of gateway deployment: values of γ_k with $\eta = 4$, SIR= 1dB and different number of gateways on a grid deployment (a), at the cell edge (b), or when the gateways are all concentrated close to the center of the cell(c).

Note that, when $SIR = 0$ dB (i.e. $\alpha = 1$) and a single gateway is placed in the center of the cell, then $\gamma_k = 1/(k+1)$, because for any placement of $k+1$ EDs, only one ED will result closer to the single gateway (assuming negligible the probability of extracting two EDs on the same distance to the gateway). In this condition, it is easy to see that equation (2.7) gives the same result of equation (2.2).

Obviously, the γ_k coefficients depend not only on the number of gateways but also on their specific position within the cell. Since a closed form derivation is not generally possible, we evaluated the γ_k coefficients numerically for specific gateway positions. To this purpose, we randomly generated $k+1$ EDs uniformly distributed in the cell, quantified the ratio of EDs whose distance r from at least one gateway was α times smaller than the distance between the other k EDs and the same gateway, and averaged results over multiple random placements. In our evaluations, we analyzed different deployment strategies: placing the gateways at a regular grid, on the cell edge, or concentrated very close to the cell center. For the grid deployments, we kept a central symmetry towards the cell center, by equally spacing the gateways in the two cell dimensions; for the edge deployment, we equally spaced the gateways along the cell circumference, while for the last setting we created a small grid of size $R/10$.

Capture probability. Fig. 2.16 shows the γ_k results numerically obtained by placing a varying number of gateways with the three deployment strategies explained above: from the figure, it is easy to see how γ_k coefficients are strongly dependent on the gateway placements. This is particularly evident

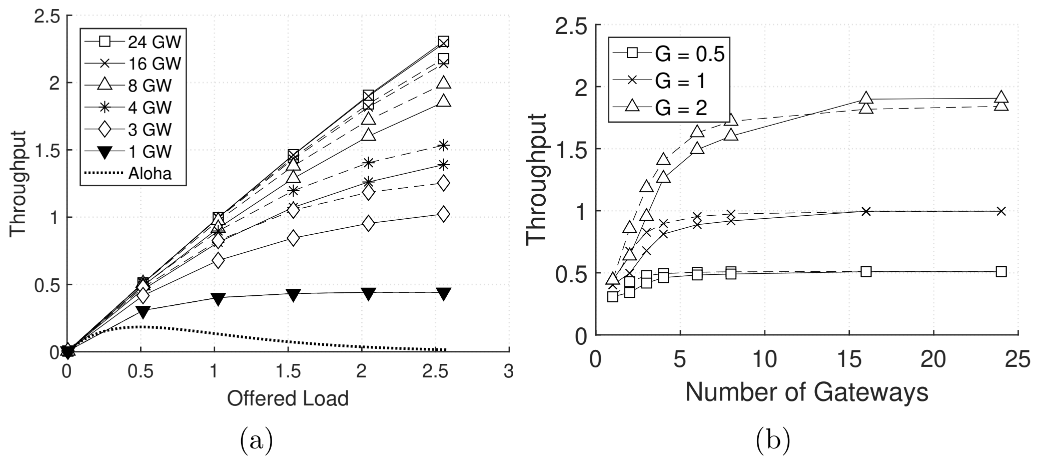


Figure 2.17. Grid deployment (solid) vs. edge deployment (dashed) when varying the offered load or the number of gateways.

in the last case: when the gateways are too close to each other, the performance does not improve in comparison with the single gateway cell. Indeed, the space diversity between gateways is poor and offers little opportunities for increasing the channel captures. For the other deployment solutions, we also observe that coefficients result generally higher when the gateways are placed on the cell edge rather than on a regular grid. Only when the number of gateways is high (e.g. 16 or 24) the grid topology is better, although the improvement on the capture probability is marginal. Although not intuitive, this result can be justified by considering that (as long as coverage is guaranteed) the more the gateways are sparse, the more channel captures can be achieved. For example, for $k = 1$ (one interferer, 2 simultaneous transmissions) and 3 gateways, the average number of transmissions correctly received by at least one gateway is 0.999 for gateways deployed on the cell edge and 0.804 for gateways placed in a regular grid. As depicted in Fig. 2.15, on a grid topology a transmitting ED can create a "shadow area" towards the cell edge, impeding any other ED in this area to successfully capture the channel.

Throughput. Fig. 2.17 quantifies the cell throughput by using equation 2.7 and the γ_k coefficients derived in our numerical evaluations, for the grid (solide lines) and edge (dashed lines) deployment strategies. The figures clearly show that increasing the number of gateways allows to achieve a throughput almost equal to the offered load. Obviously, higher capture probabilities are mapped into higher throughput results. For example, with

3 gateways the throughput achieved with the edge deployment is about 25% higher than the one achieved with the regular grid, while for a number of gateways equal or higher than 16 the throughput gain of the grid deployment is about 5%. We can expect that these capacity differences under different gateway deployments can be reduced in case of multi-cell systems.

In order to validate our model, Fig. 2.18 compares our throughput predictions with simulation results. For space reason, we only show results for the regular grid case (but conclusions are similar for the edge deployment case). The figures have been obtained by considering the availability of one SF only (namely, SF7) and three different capture thresholds (0, 1 and 3 dB). The figure allows to draw some interesting conclusions. First, despite of the simplification assumptions used for modeling the capture effects, the model is in agreement with simulations in all the considered scenarios, thus validating equation 2.7 and the γ_k derivation. Second, when the capture effects are very common (i.e. for capture threshold of 0 dB), a cell with M gateways and a number of devices equal to N provides similar performance of M independent systems loaded with N/M devices. For example, the DER achieved with 4 gateways and 2000 EDs results slightly less than 0.8 and comparable with the one achieved by a single gateway with 500 EDs or 8 gateways and 4000 EDs. Third, the capture threshold has a much lower impact than in a single gateway case, thanks to the increased capture possibilities provided by the spatial diversity of the gateways. Finally, although results always improve as the number of gateways increases, at a given point the improvements are marginal (DER results with 16 or 24 gateways are almost comparable).

Fairness. The position of the gateways has also a significant impact not only on the average cell performance, but also on the fairness of the network, i.e. on the spreading of the DER results achieved by different EDs. Fig. 2.19 shows the success probability histogram of the ED's transmissions in various scenarios characterized by *same* DER of 0.6 but different number of gateways, EDs and deployment strategy. In particular, the figure shows what happens in a cell with 1 gateway in the center or with 4 gateways, in a regular grid or at the cell edge. To obtain the same DER of 0.6, the number of EDs was 1000, 4000 and 4900 respectively and the corresponding offered load was 0.57, 2.29 and 2.81 (these values were obtained experimentally to equalize the capacity and achieve equal DER for all scenarios). From the figure, it is clear that with only one gateway most of the EDs have low performances, while in a grid topology the distribution is bimodal, with a group of EDs that transmit with very high success probability (higher than 80%) and another group which suffer low performance (around 30% successful frames). Instead,

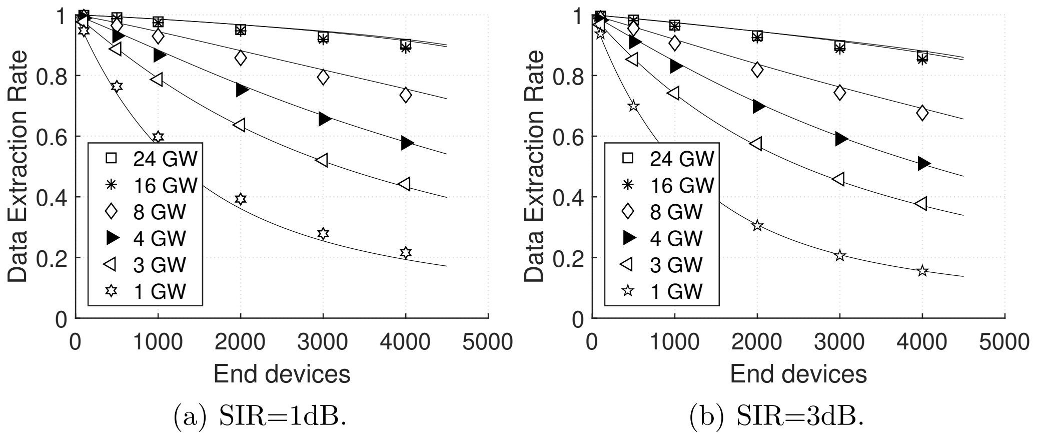


Figure 2.18. DER using multiple gateways in a grid topology: model (lines) and simulation results (points) for SIR values of 1 and 3 dB.

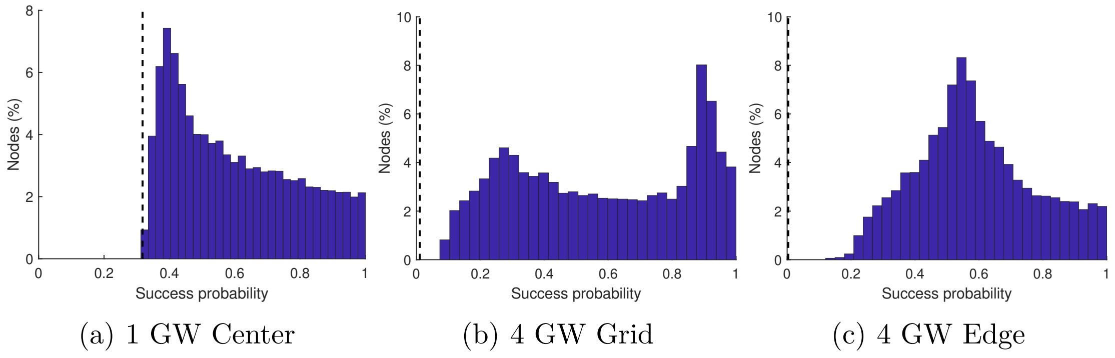


Figure 2.19. Impact of gateway deployment on fairness: distribution of the success probability among EDs in a cell with 1 gateway in the center (a) and with 4 gateways in a regular grid (b) or at the cell edge (c).

the edge deployment tends to be more fair, in the sense that most of the EDs experience a success probability around 0.6, corresponding to the cell average DER. The figure also shows the minimum theoretical success probability of the pure Aloha system (given by e^{-2G} and marked by the dashed lines), which is obviously different for the three scenarios because of the different load levels. Note that the capture effects can improve the Aloha results for some devices, without penalizing the ones with low capture probabilities.

2.7 Conclusion

In this research work we have studied the impact of two peculiar characteristics of LoRa modulations, i.e. the high capture probability and the imperfect orthogonality between different SFs, on the overall cell capacity. Although parallel reception of multiple overlapping frames is generally possible, we showed that the link-level performance of LoRa is deeply influenced by capture effects and by inter-SF collisions which can indeed cause loss if the interference power is strong enough.

We then exploited this link-level analysis to model analytically the achievable network capacity in a typical LoRa cell. We showed that high SFs are severely affected by inter-SF interference and that the use of power control and packet fragmentation to compensate such problem may be counterproductive. Although deploying multiple gateways can mitigate the capacity loss and boost the occurrence of channel captures, the overall capacity increase becomes negligible after 16-24 gateways. Finally, when only a handful of gateways are present, the deployment should be as distant from the center as possible and not on a regular grid.

We believe that the results presented in this work provide important insights on LoRa technology and can pave the way to new accurate guidelines for the correct design of future LoRa networks.

Chapter 3

A 4G/5G Control Platform for fingerprint-based indoor localization

3.1 Introduction

In the emerging scenario of cellular network evolution, key aspects such as increased mobile terminal densities, smaller cells, multiple access technologies and access point reconfigurability can be beneficial for localization (especially in indoor environments). Apart from the smartphones of the users, we can envision that the increasing density of mobile devices in indoor spaces is due to the proliferation of smart wearable devices (e.g., bracelets, watches, glasses), smart appliances (e.g., televisions, electric plugs, thermostats), proximity sensors, etc. We consider two different implications due to the proliferation of these devices: on one side, in order to cope with the increasing demand of capacity, several overlapping small cells will be deployed in indoor spaces, thus offering multiple reference signals for localization; on the other side, some of these devices, such as the proximity sensors or some desktop gadgets, will be located at fixed known positions, thus offering a pervasive set of observation points for building radio-maps of the environment. In such a scenario, we argue that a centralized localization system, able to collect measurement reports from high-density objects at regular time intervals, can be very promising for implementing localization functions based on radio fingerprinting. Indeed, exploiting smart objects for building radio-maps allows automatic calibrations and time-varying corrections of the maps in presence of moving obstacles (including people). Moreover, the current paradigm of Software-Defined-Networking (SDN) can speed-up the real implementation of the envisioned system, because programmable access points allow the customization of the signalling mechanisms required for building a global view of the network, on top of which different network-wide optimizations and applications can be defined.

We propose a SDN control framework for indoor femto-cells, including both the design and implementation of programmable access points, or evolved-Node-B (eNB), and global orchestrator. Two are the most important features of the platform: the first one is the capability to gather directly from LTE's Access Stratum (AS) protocols a number of network status parameters, which are in themselves very valuable from the operator's perspective; the second one is the possibility to actively modify some aspects of the eNB behavior in real-time, like the bandwidth allocation or the measurements report periodicity. By exploiting this framework, we also define a localization application, devised to functionally validate our implementation and prove the technical feasibility and potentialities of fingerprint-based localization supported by high-density devices.

The rest of the Chapter is organized as follows. In Section 4.2 we discuss the related work on the topics of wireless SDN and indoor localization. In Section 3.3 we describe the architecture of the platform and an orchestration app for building an indoor radio-map. In Section 3.4 we compare two machine learning approaches able to localize a device in different scenarios with very high accuracy exploiting the radio-maps. Finally, conclusions are drawn in Section 3.5.

3.2 Related work

Control Framework. In the last years, there was a clear trend towards the softwarization of cellular networks, especially in the core network where mature technologies like virtual machines and Linux containers can be successfully employed to realize virtualized network functions. Although software-defined Radio Access Networks (RAN) are very promising, in particular for coordinating high density cells and terminals, there are additional challenges to be considered because the unique nature of wireless resources makes more

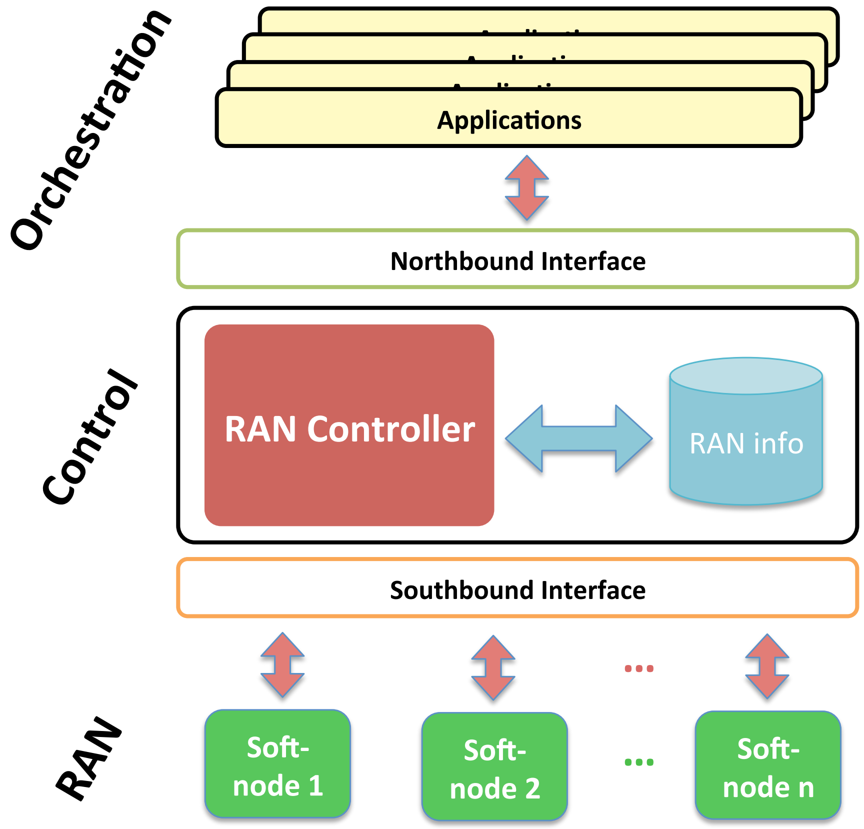


Figure 3.1. The platform architecture develops on there levels: RAN layer, Control layer, Orchestration layer.

difficult the definition of resource abstractions. Control frameworks have been proposed for managing heterogenous wireless networks coexisting in ISM bands [28] and high density WiFi networks [29]. From LTE perspective, SoftRAN [30] is among the earliest works on this topic, with the idea of base station abstractions aimed at improving the management of dense networks. Conceptually, both fully-centralized [31] and hierarchical controllers [32, 33] have been proposed, but so far real implementations have some limitations, either in terms of support of data plane programmability and dynamic functional split, or in terms of node scalability and integration of legacy user terminals. For filling this gap, similarly to FlexRAN [34], in this work we propose a flexible and programmable platform for RAN control, which works on software-defined base stations and legacy user terminals. More into details, software base stations have been built on top of OpenAirInterface (OAI)

[35], connected to a real core network, and controlled by our control platform.

Fingerprint-based Cellular Localization. As an example network-wide application, we developed a fingerprint-based localization scheme. Fingerprinting localization is an indirect localization method which is based on the collection of RF received signal measurements, in a set of reference points, for building a radio-map of the environment. It has been widely adopted for indoor localization, by exploiting different technologies such as WiFi [36], but also RFID technologies [37] or cellular networks [38]. Fingerprinting solutions for cellular networks can exploit power measurements of the received signals and channel quality indicators (as considered in this work), but also observed time of arrival or time difference of arrivals [39]. Multi-radio can be exploited for further extending the features of the radio-maps [40].

3.3 Platform Architecture

In this section we present our programmable platform for RAN control, devised to extend the SDN benefits to the wireless access segment of cellular networks, thus supporting advanced optimization schemes. The main challenges of a RAN SDN platform can be summarized in the following aspects [34]:

- to separate the parts of LTE's Access Stratum (AS) protocols that *make decisions* about elements or resources of the RAN and the parts that *apply those decisions*;
- to enable the capability to react to the monitored network conditions executing contex-specific virtual functions;
- to achieve the capability of implementing orchestration applications.

These challenges are faced by our architecture depicted in Fig. 3.1. The architecture is composed of three layers, from the bottom to the top: RAN layer, Control layer and Orchestration layer. The network nodes that populates the RAN layer are LTE's release 10 compliant eNBs that we call soft-nodes, as they can be configured via software at run-time. The RAN layer exposes the southbound Application Programming Interface (API) to the Control layer, which is responsible of monitoring the status of the network; the information about the soft-nodes and the devices attached to can be used in real-time or stored in a database to be further elaborated. At the top of the architecture we have the Orchestration layer, where a number of applications (including our localization application) can be developed to fulfill different operator's requirements. The applications interact with the Control layer by means of the northbound API.

3.3.1 Soft-node

An essential part of the architecture are the programmable base stations called soft-nodes. A soft-node is composed of a USRP B210 Software-Defined-Radio (SDR), which acts as the radio front-end, connected to a host machine where two software elements run. The first element is a customized version of the well-known framework OpenAirInterface Radio Access Network (OAI-RAN) that implements LTE's Radio Resource Control (RRC), Radio Link Control (RLC), Packet Data Convergence Protocol (PDCP) and Medium Access Control (MAC) protocols. We modified some parts of OAI-RAN at MAC and RRC level for making accessible to the controller those network parameters that we thought valuable for the operator; moreover, we enabled the possibility of dynamically uploading these software modules for further functional extensions.

At the MAC level, we are able to access the following parameters: Cell ID (CID); Channel Quality Indicator (CQI); Cell Radio Network Temporary Identifier (C-RNTI). Furthermore, we can set the resource blocks (RBs) that the soft-node allocates in downlink for a certain user; as OAI-RAN assigns the radio resources in group of consecutive RBs, this is achieved specifying for that user the number of RBs and the position within the radio frame of the first RB.

At RRC level instead, we enabled the possibility of loading customized control logics for handling the standard LTE signalling mechanisms. In particular, for our localization application, we added two new procedures for activating or de-activating a periodic reconfiguration of user connections. Indeed, a reconfiguration request sent by the base station has the effect of triggering the transmission of a measurement report by the corresponding user terminal. While these requests are usually sent during handover, the periodic scheduling of these requests can force user terminals to notify information on the RF signal measurements, regardless of the status of the link. Measurement reports sent by the users specify the Reference Signal Received Power (RSRP) and Reference Signal Received Quality (RSRQ) of the serving soft-node, as well as the same parameters relevant to the neighbor soft-nodes. This information can be used for updating the radio-maps of the environment. In addition we can read the Temporary Mobile Subscriber

Table 3.1. Modified OAI-RAN functions

Function	Parameters	Operation
`eNB_dlsch_ulsch_scheduler`	CQI, C-RNTI	read
`decode_guti_eps_mobile_identity`	TMSI	read
`eNB_dlsch_ulsch_preprocessor`	Resource Blocks	write
`eNB_dlsch_ulsch_preprocessor_allocate`	Resource Blocks	write
`rrc_eNB_generate_RRCConnectionReconfiguration`	RSRP, RSRQ, QCI	read

Identity (TMSI), for locating the user within the network, and QoS Class Identifier (QCI) for assessing the service requirements of the user.

Table 3.1 shows the full list of OAI functions that we modified and the parameters that we are able to read or write. We can get (or change in some cases) these parameters at run-time, by means of two parallel threads that we created in the MAC layer of OAI-RAN. These threads maintain a C structure that holds the status of the sensible parameters and communicate via sockets with an external program.

The external program is the second software element that composes the soft-node and it is called OAI Agent. Fig. 3.2 also shows the communication between the Agent and OAI-RAN inside the soft-node. We chose to develop the Agent in Python as a separate process in order to not increase the workload of OAI-RAN, thus making the soft-node and the overall architecture more stable. In addition, with this design we can further develop the Agent and OAI-RAN separately in the future. The main responsibility of OAI Agent is the exposition of the southbound API to the Controller, in order to abstract the low level operations to the upper layers. In other words, the Agent is a run-time interface through which the Controller can handle the soft-node.

Table 3.2 summarizes the methods of the southbound API which are used for monitoring and managing soft-nodes and devices of the RAN. In one direction the Agent parses the configuration commands sent by the Controller in JSON format and sends to OAI-RAN a lightweight bytes structure that enforces the command. In the other one, it parses periodically from OAI-RAN a bytes structure with the node status parameters, serializes and sends the data in JSON format to the Controller. The time interval that occurs between two consecutive arrivals is also configurable. In addition, OAI Agent has also some intelligence, as it can turn on or restart OAI-RAN.

Table 3.2. Southbound API

Method	Parameters	Returned values	Operation
`get_enb_users`	cid	tmsi_list	read
`get_user_cqi`	cid, tmsi	cqi	read
`get_measures`	cid, tmsi	rsrp_vect, rsrq_vect	read
`start_measures`	cid, tmsi	None	write
`stop_measures`	cid, tmsi	None	write
`detach_user`	cid, tmsi	None	write
`set_user_rbs`	cid, tmsi, start_rb, end_rb	None	write

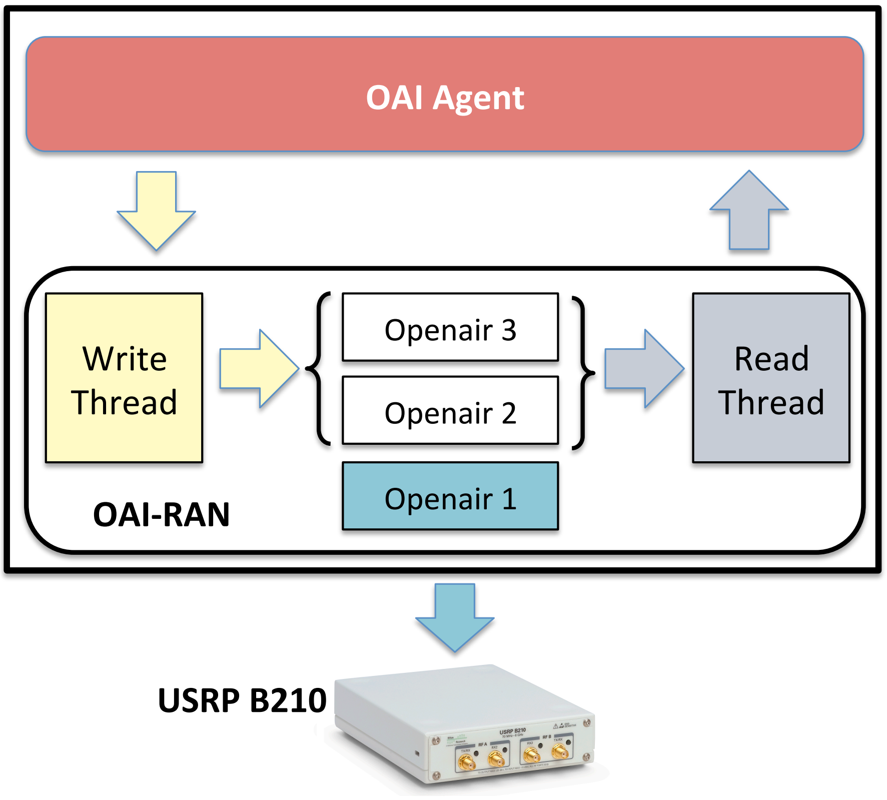


Figure 3.2. The soft-node is built on top of OAI-RAN: two threads interface LTE's protocols with the external OAI-Agent.

3.3.2 RAN Monitoring and Management

Two levels of network monitoring and management are built upon the southbound API. The first one, the Control layer, is definitely the core element

Table 3.3. Northbound API

Method	**Parameters**	**Returned values**	**Operation**
`get_ran_enbs`	None	cid_list	read
`get_ran_users`	None	cid_list, tmsi_list	read
`get_enb_users`	CID	tmsi_list	read
`find_user`	tmsi	CID, c-rnti	read
`get_user_cqi`	tmsi	cqi	read
`map_area`	tmsi, duration, step	None	write
`get_measures`	tmsi, date, area	rsrq_vect, rsrp_vect	read

of the architecture as it controls the soft-nodes by means of the OAI-agents. The RAN Controller is equipped with a database in order to store, in a structured manner, the low-level network parameters gathered during routine monitoring operations. In this way, the platform is able to track the evolution of the network status and aggregate low-level information into a global network view, thus enabling the possibility of analyzing the RAN operation conditions over time. For example, one of the Controller activities could be the automation of LTE's Minimization of Drive Tests (MDT) feature, which can be implemented for monitoring the signal strength indicators of the devices (e.g. RSRP and RSRQ) and discovering inefficiencies in the radio network planning. The Orchestration layer, in our platform, is a set of stand-alone, high level applications, implementing intelligent decisions on the RAN elements.

Table 3.3 provides a list of some methods in the northbound API that the Orchestrator can use for implementing high-level network management procedures.

3.3.3 Applications for orchestration

As an example of the intelligent communication between the platform's elements, we consider a simple application that we developed for mapping the signal strength measurements collected from three soft-nodes deployed in our laboratory. We used this application for building the training dataset used in the experimental use case described in the next section. The Orchestration app invokes the method `map_area` of the northbound API, asking the

Controller to gather RSRP and RSRQ from a device in a selected area. The app specifies the duration of the operation and the time interval between two consecutive measurements. As shown in Algorithm 1, the Controller makes use of the southbound API for fulfilling the app request. First of all, it has to find which soft-node the user is attached to. Then it tells the device to start sending measurements and it makes sure that the measurements are correctly sent. At this point, the Controller can get the RSRP and RSRQ values from the device for the time period specified by the app. When the job is done, the application tells the device to stop sending measurements.

Algorithm 1: Controller building a piece of the radio-map

```
def map_area(tmsi, duration, step):
    cid, crnti = find_user(tmsi)
    active = False
    while not active:
        start_measures(cid, crnti)
        active = check_measures(cid, crnti)
    meas_num = duration/step
    for i in range(0, meas_num):
        rsrp, rsrq = get_measures(cid, crnti)
        write_db(rsrp, rsrq)
        sleep(step)
    stop_measures(cid, crnti)
```

3.4 Fingerprint-based localization

We experimentally validated the functionalities of our platform in an experimental network setup inside the TIM Wireless Innovation Lab in Turin.

Fig. 3.3 shows a map of the lab: three soft-nodes have been deployed at the borders of a room (which is 5 meters wide and 11 meters long) for emulating an indoor space with high-density cells. Indeed, as discussed in the introduction, network densification is one the paradigm envisioned for coping with the requirements of 5G networks and we assume that a similar set-up will be normal in the near future, similarly to the current high-density deployment of WiFi access points. Although each soft-node in principle can cover the whole room with good connectivity performance, we decreased the transmission gain of the USRP boards from 90 to 70 dB, in order to reduce the cell overlapping and energy consumption of the system. We experimentally found a coverage radius of about 15 meters for all the cells, despite the fact that for logistic reasons only one cell has been placed at the ceiling of the room (at an height of about 2.7 meters above the ground), while the other two cells have been placed at 1.8 meters.

The soft-nodes have been interfaced to a core network by Amarisoft, in order to support end-to-end connectivity with legacy terminals, and with a RAN controller (by means of the OAI agents) running on a data center together with the Orchestrator.

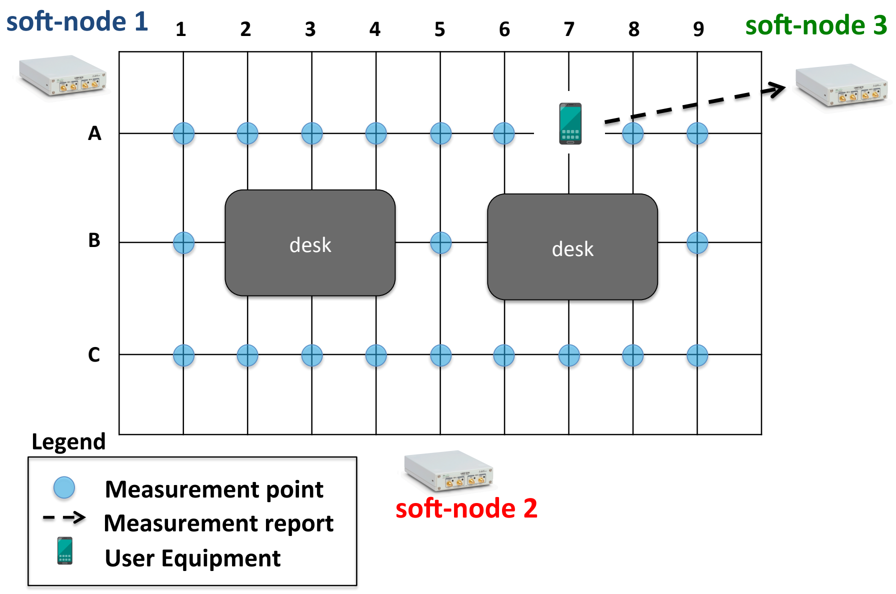


Figure 3.3. The experimental deployment in TIM labs: a device in position A7 send a measurement report to its serving soft-node (in green).

3.4.1 Building the radio-map

In order to characterize the radio environment with a good spatial resolution, we assumed that at least one smart object or mobile terminal is available in each of the 21 regions in which the room is partitioned. Although in real deployments such regions could not be regular (depending on the known positions of the reference devices), in our case the room has been divided in three equal-size rows (indicated by letters A, B and C) and nine equal-size columns (indicated by a progressive number). A legacy mobile device, acting as a reference point, is placed at the center of the region. For implementation issues, rather than working with 21 different terminals, we used at most three terminals at the same time and moved the terminals at different positions for collecting a whole set of measurements in all the reference points. For example, in Fig. 3.3 it is shown a device placed in position A7, which sends a measurement report to the soft-node to which it is associated. The idea is exploiting the devices that in the future will be integrated into the room furnitures, walls, paves at known positions for collecting periodically radio-maps with a good spatial resolution.

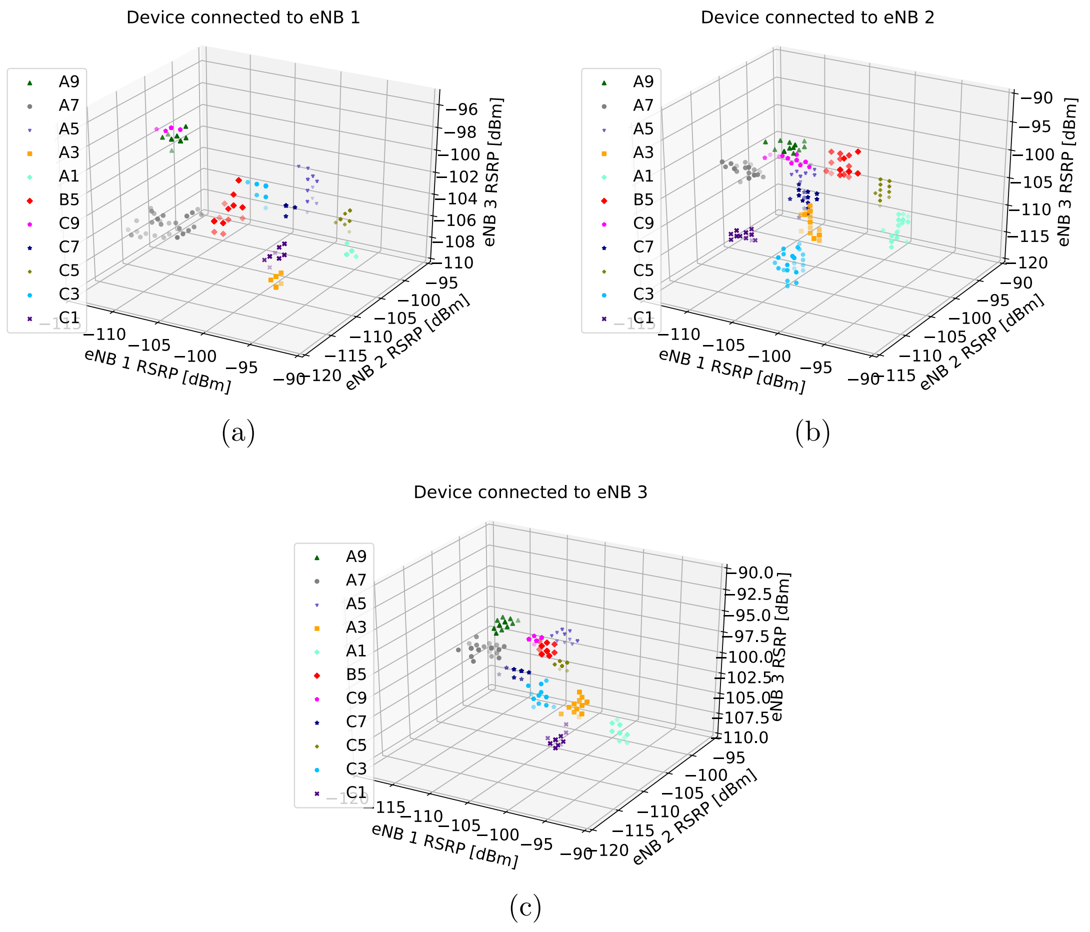


Figure 3.4. RSRP points in the 3D space of some of the 21 positions of the room in three different scenarios: a) the device is connected to soft-node 1; b) the device is connected to soft-node 2; c) the device is connected to soft-node 3.

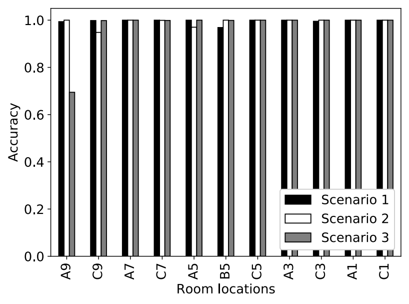
We performed several measurements rounds in different times of the day and under different crowding conditions of the room. Moreover, since the serving base station can have an effect on the measurement reports, we built multiple radio-maps by alternating the serving base stations of the terminals placed at different positions. Obviously, the RSRP measurements are affected by the obstacles in the room and therefore the people moving inside the room have an impact on the overall radio-map results. We configured the localization application for triggering a measurement report by mobile terminals every 5 seconds for a continous measurement round of 36 consecutive reports (before stopping). In principle, whenever the reference terminals provide the reports in parallel, a radio-map can be built in 3 minutes only. Radio-maps updates can be decided at regular time intervals (e.g. every hour) or in case the orchestrator is able to predict an environmental change from other information.

In Fig. 3.4 are represented the RSRP values measured by the devices when connected to each deployed base station, in three different measurement rounds at different hours of the day, (leading to three different radio-maps). Each measurement report sent by the mobile terminal at a given position is identified by a point with the same color and shape in the 3D space, where each dimension represents the RSRP value measured from the same base station.

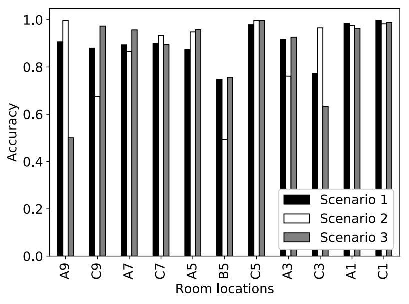
Two things are worth noting observing the groups of points of the majority of the positions: i) in different measurement rounds, the points that identifies one position can move considerably in the space; ii) within the same measurement round, the points are stable enough as they can be clustered in well defined regions of the RSRP space. Long-term variations of radio-maps can be due to different positions of people within the room or variations in the interference conditions; conversely, in short-terms the radio-maps can be reliable enough for localizing user terminals.

3.4.2 Analysis and results

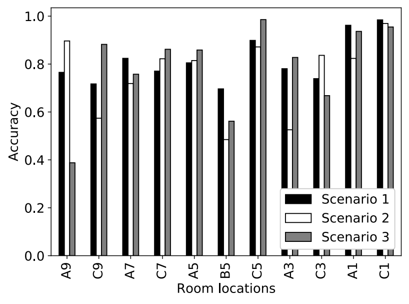
We analyzed the radio-maps built during the measurement campaign with the goal of finding a supervised learning algorithm able to classify, with high accuracy and acceptable computational complexity, the position in which the device was placed for a given point in the 3D space. For each scenario we used 75% of the dataset for training and 25% for testing. We initially used the K-Nearest Neighbor (KNN) algorithm because it has no need of training, thus it is suitable for running in real-time. We implemented a KNN classifier with $k = 3$, 3 input layers, one for each soft-node active



(a)



(b)



(c)

Figure 3.5. Average accuracy performances of the three learning algorithms when the classifier is trained and tested with the same dataset scenario: a) KNN; b) MLP with 2×200 neurons; c) MLP with 2×70 neurons.

during the experiments, and 21 output layers. We tried the following three different search implementations of KNN in order to assess their classification performances and computational costs: Brute-force, KD-tree and Ball-tree. From the performances point of view, we calculated the average accuracy of each approach in one hundred runs with different random splits for each scenario and found that they have only some decimal points of difference. The KNN classifier identified correctly about 99.6% of the measurement made when the device is connected to soft-node 1 and about 98.1 and 96.9 when is connected to soft-node 2 and 3 respectively. On the other hand, Ball tree and KD-tree have a better computational cost as they scale with $O(F \cdot S \cdot \log S)$ whereas brute-force approach scales with $O(F \cdot S^2)$, where $F = 3$ is the number of features and $S = 36 * 21 = 756$ is the number of samples.

Considering that the computational complexity of KNN can grow fast as the number of samples increases, we developed also a Multi-layer Perceptron

Table 3.4. Classifiers accuracy comparison

	Test Scenario	KNN [%]	MLP 200 [%]	MLP 70 [%]
Classifier scenario 1	1	**99.6**	**91.2**	**83.6**
	2	12.7	10.4	10.2
	3	8.0	12.5	13.1
Classifier scenario 2	1	8.8	10.4	10.6
	2	**98.1**	**87.4**	**76.3**
	3	15.1	12.6	12.8
Classifier scenario 3	1	24.3	13.1	12.2
	2	6.2	7.6	9.8
	3	**96.9**	**88.0**	**80.2**

(MLP) classifier with the idea of reducing further the computational cost. The neural networks that we implemented have 3 input layers, 2 hidden layers, 21 output layers and it uses the sigmoid neuron as activation function and the Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) as optimization algorithm, which has shown to perform better of the well-known Gradient Descent algorithm when in presence of small datasets. We also set the maximum number of iterations to 3000 in order to avoid overfitting and to have a fast training time for all the scenarios. As in neural networks with two hidden layers the complexity scales with $O(F \cdot N^2)$ to be competitive against KNN the number of neurons N should be approximatively lower than 70. Unfortunately, with 70 neurons the network reached only 83.6% in the best scenario. Table 3.4 summarizes the accuracy results obtained from our analysis: the left-most column indicates with which scenario dataset the classifier has been trained whereas the second indicates the scenario against with it has been tested. The performances are always very good when the classifiers are trained and tested with the same scenario dataset but they are poor in the other cases, showing that the training must be repeated periodically in order to capture variations in the radio environment. It is worth nothing that the first scenario shows the best score with all the classifiers; this is probably due to the fact that soft-node 1 has an height advantage over the other two.

Fig. 3.5 shows the average accuracy of the classifiers in some of the 21 positions of the room for the three scenarios. KNN shows an impressive score in almost every location of the room. The two MLP classifiers are instead clearly worst, especially in position B5. It is worth noting that B5 is between two desks and maybe the device suffered a severe multi-path caused by the objects over the desks.

3.5 Conclusion

In this Chapter we presented a flexible platform, implemented on top of OpenAirInterface (OAI), able to control, monitor and build radio-maps of indoor environments that can be used for localization. We proposed an architecture that develops on three levels and tackles all the challenges of the emerging Software-Defined-Networking (SDN) paradigm. We developed a centralized controller able to monitor the network status and make context-dependent decisions which are enforced by the soft-nodes via the OAI-Agents. Furthermore, we defined a northbound API through which an orchestration app can build a radio-map of the environment. Last but not least, we compared the performances obtained by three different classifiers trained with the radio-maps collected. Our analysis shows that K-Nearest Neighbors (KNN) has the best performances both in terms of accuracy and computational cost with respect to the two neural networks. Indeed, it is is possible to localize a device in high-density femtocells indoor deployments updating regularly the maps. This research work is currently active as we plan to enrich the southbound API in order to deeply customize the soft-nodes. Then, we want to implement a network slicing use case that exploits user's position information in order to allocate radio resources to different users according to their service requirements.

Chapter 4

A network Broker for computational offloading in wireless networks

4.1 Introduction

In the last years, Cloud computing services offered by tech giants like Amazon, Google and Microsoft, have had a great commercial success. An ever growing number of companies buys Infrastructure, Platform or Software as a Service (IaaS, PaaS, SaaS) solutions [54] instead of developing their own IT systems. Nowadays, novel computing architectures like MEC and Fog [55, 56] are extending the reach of computational offloading to resource constrained wireless devices. Emerging technologies like IoT, augmented and virtual reality, vehicular communications (just to cite a few) would enormously benefit from having extra computing capabilities in relative proximity to the devices. The offloading benefits can be expressed in terms of task completion time, monetary cost, device energy savings. These benefits can be achieved deciding as best as possible what, when, where and how to offload [57].

In this context there is an open discussion about what is the best architectural solution among Cloud, Edge and Fog computing paradigms. Our idea is to consider computational offloading as a service by itself, decoupled by the specific application scenario where it is needed. We envision a network marketplace where computing resources can be bought from different players (telecom operators, over-the-top companies and generic public or private entities) at different costs. On one side, in such a marketplace we have wireless devices with limited computing and battery resources. Such devices are resource consumers as they need processing power in order to run resource-hungry applications. We believe that, with the emerging high-density of smart objects, the diffusion of AR/VR applications and vehicles-to-vechicles communications (e.g., autonomous driving, platooning and AR driving assistance), the demand of computing resources will grow considerably in the next years. On the other side, network architectures are evolving towards the deployment of multiple network nodes, at different hierarchical levels, which can supply such a computing power. We generically refer to these nodes as producers. Examples of these nodes are: i) Cloud nodes in the internet, ii) Multi-Access Edge Computing (MEC) nodes, managed by network operators and also iii) base stations equipped with computing capabilities, on the basis of the Fog computing paradigm. The resources can be exchanged between network nodes in the form of tasks that consists of: i) processing a certain amount of operations; ii) transferring a number of bits between consumers and producers. The tasks have also constraints that should be met in order to guarantee the consumer satisfaction. In principle, we can envision two types of constraints: i) a deadline by which the task has to be completed, in order to provide the best user experience; ii) a monetary cost that the consumer may not want to exceed.

In order to enable such a scenario, we introduce an intelligent element that acts as a broker between consumers and producers. The Broker represents an innovative network element that is capable of coupling autonomously supply and demand of offloading tasks. The Broker's decision-making process is shown in Figure 4.1. It first collects task requests coming from the consumers and analyze the task demands and constraints. Then, it looks for producers that can provide the computing power necessary to complete the task within the deadline. Besides, it checks that on the path between the producer found and the consumer there is enough free capacity to complete the bits transfer within the deadline, as shown in Figure 4.1(b). It also checks for monetary cost restriction if any. In case it finds one or more suitable producer candidates it accepts the request, otherwise it refuses it (meaning that the task will be performed by the consumer itself, if possible). Finally, as shown in Figure 4.1(c), the Broker chooses one of the producers in the candidate list to perform the task. In this Chapter, we explore the potentials of the proposed offloading architecture through a fluid simulator that we developed using Python. We compare basic decision policies and show the importance of assigning tasks according to the load distribution of network producers

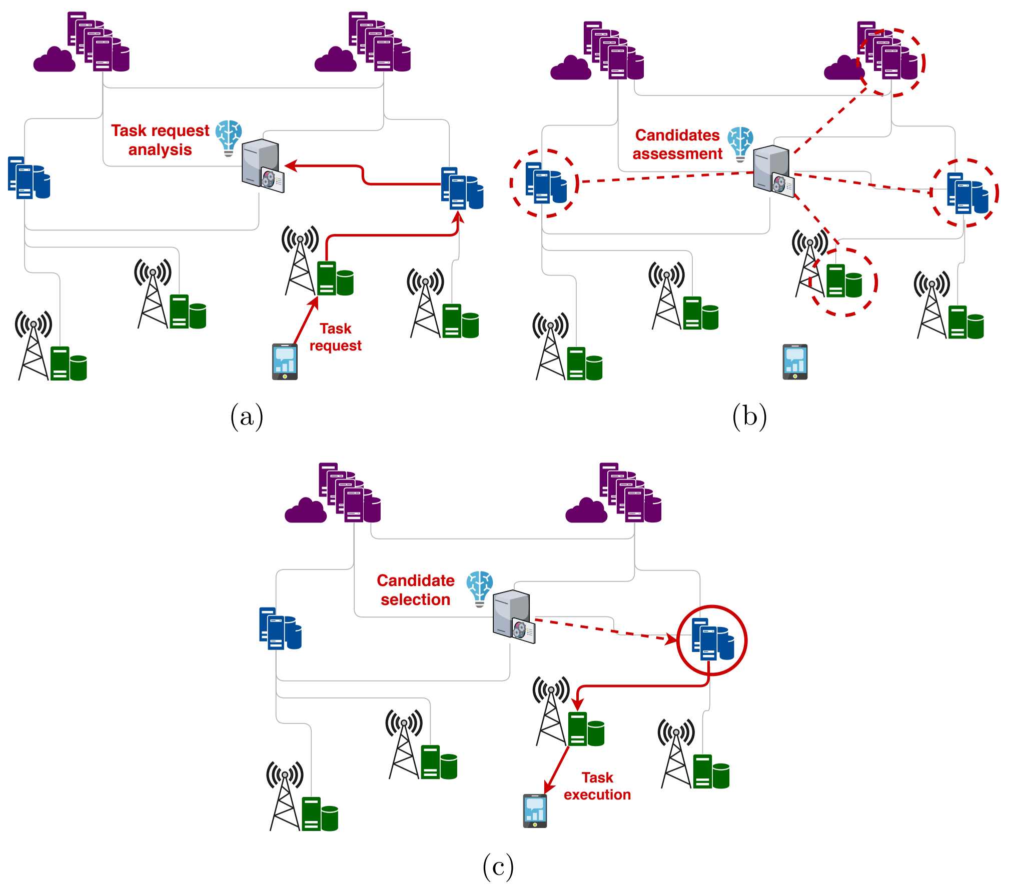


Figure 4.1. Broker's decision making process in a network scenario with Cloud, MEC and Fog nodes (colored in purple, blue and green respectively): a) the Broker receives a task request and analyzes requirements and constraints; b) the Broker finds all the producers in the network that can complete the task; c) the Broker selects one of the candidates to carry out the task.

and the gain that is achieved in terms of tasks completed successfully. For each policy we also show the importance of keeping memory in the decision-making process of the Broker and the impact of errors in the bandwidth estimation. The rest of the Chapter is organized as follows: in Section 4.3.1 we describe the simulator that we developed; in Section 4.3.2 we explain how we modeled the task request process; in Section 4.3.3 an augmented-reality use case is presented and analyzed; Section 4.4 concludes the Chapter describing how we want to develop this work in the future.

4.2 Related work

Computational offloading has been studied extensively in the last two decades, in particular in the wired domain. Recently, the focus has been posed on architectural solutions, commonly referred as edge computing, that put computing resources next to mobile networks, in order to support mobile, heavy-computation and delay-sensitive applications. According to MEC computing paradigm, small data-centers are deployed in proximity to groups of base stations [42]. In Fog computing, instead, computation resources can be directly coupled to access points or other networking hardware [41]. Finally, Cloudlets can be rapidly instantiated on demand on pools of virtual machines deployed in trusted wireless LANs [43].

In such a context, where multiple egde servers with possibly finite processing resources can be reached experiencing different delays, two decisions have to be made when a task is generated by a device: i) where to offload it, which is a *dispatching* problem and ii) when to execute it in case other tasks have been destined to the same server, which is a *scheduling* problem. Furthermore, these two problems are related to each other when the goal is to enhance the user experience minimizing the system delay. In literature, due to the complexity of the combination of these two problems, some works focused on the dispatching problem [44, 45, 46, 47, 49, 50, 51], assuming a simple First-Come-First-Serve scheduling, and other works dealt contemporary with both problems [52, 53].

Among the first works in this area we find ThinkAir [44], a software framework that allows to parallelize computing tasks on virtual machines showing that both execution time and power consumption can be greatly reduced by using offloading strategies. In [45], the authors design the network edge as a tree hierarchy of geo-distributed servers and propose an algorithm to place the mobile users workload and to allocate the computing capacity of the tiers of the hierarchy. They evaluate the performance of their solution with small-scale experimentation and large-scale simulations. In [46], the focus is on the cooperation between fog and cloud nodes in order to achieve the best tradeoff between power consumptions and transmission delay. The authors formulate the workload allocation problem and show with numerical results the potentialities of cloud-fog interplay. In [47], an hybrid scheme is proposed where tasks can be executed locally or offloaded to cloud or to D2D devices. The problem of minimizing the energy consumption is recast as a minimum weight matching problem over a three-layer graph developed for capturing the choice space. In [49], the authors investigate the dual dispatching problem of optimal cloudlet placement and user to cloudlet allocation. They propose Heaviest-AP-First (HAF) and Density-Based-Clustering (DBC) for cloudlet placement with the latter performing better. They also show the importance of using a Relative-Distance (RD) scheme instead of allocating users to the nearest cloudlet. In [50], the competition between IoT users for cloud and node resources is modeled with a computation offloading game. The authors prove the existence of a pure Nash equilibrium and show through simulations that the proposed mechanism significantly reduces the computation delay. In [51], the authors propose and edge network orchestrator to enable fast and accurate object analytics for mobile augmented reality. An analytical model is developed based on measurements in order to characterize the tradeoff between service latency and accuracy and it is validated through an experimental setup.

The most recent works tackle the offloading task dispatching and scheduling problem together. In [52], the authors propose a Highest Residual Density First (HRDF) scheduling algorithm that prioritize the task with the highest weight to processing time ratio combined to a dispatching mechanism that aims at minimizing the total weighted response time. Through simulations, based on real-world traces, they show that the proposed algorithms outperforms both First Come First Serve (FCFS) and Weighted Round Robin (WRR) scheduling algorithms. A similar approach is used in [53], where an online algorithm decide the schedule trying first to maximize the number of task that meets deadlines and, if some tasks cannot make it, minimizing the total completion time. Simulations with real data traces and testbed experimentation is used to show a significant performance improvement in terms of accomplished tasks with respect to state-of-the-art-methods.

4.3 Network Broker fluid simulator

4.3.1 Overview

We developed a discrete-event network simulator in Python, following the object-oriented programming paradigm, that consists of more than 1200 lines of code. The software can simulate network scenarios with heterogeneous nodes interconnected at different levels: i) at the bottom level we have wireless devices associated to generic radio access networks represented by base station nodes; ii) at the middle level we can find MEC nodes serving groups of base stations; iii) at the top level we have Cloud nodes. The computing capabilities of the nodes increases as we move from the bottom to the top level. In this scenario the Broker is a special node that has no computing capabilities, as it acts exclusively as an intermediary between parties. It is directly connected to MEC and Cloud nodes and indirectly to base stations and devices.

The simulator that we developed is fluid, in the sense that it is protocol and technology agnostic: no protocols are implemented in the communication between network nodes and no specific technologies are taken into account for the radio access network. In the communication between nodes, we only consider bit streams passing through the links and eventually calculate the delays caused by saturation of one or more links in the path. This simplification is needed in order to reduce the burden of simulating such a complex network with very different nodes and processes involved. Nevertheless, a fluid implementation can surely capture trends and gains in the user experience enabled by the Broker's decision-making.

4.3.2 Task requests modeling

Modeling the task requests made by the consumers was the first problem that we faced in the simulator development. A task consists of two subtasks: i) the data processing necessary for producing a content and ii) the transfer of that content from the producer to the consumer. We want to clarify that also the first sub-task involves content delivery, as we envision that the data to process are retrieved by the producer via caching systems. From the consumer perspective, a task request can be seen as a content request as he/she basically expects to receive data processed (and retrieved) by someone else.

For these reasons we chose to model the task request process with the Poisson Shot-Noise Model (SNM) [58] that takes into account the temporal dynamics of content's popularity in the context of Content Delivery Networks (CDN). Today's Internet contents popularity changes over time, as new contents are constantly produced and distributed. This aspect is perhaps even more pronounced in case of wireless devices. In order to capture the temporal variation of content's popularity , the model considers that the requests arrive in shots. A shot is a period of time in which the traffic volume, i.e., the requests arrival rate, is constant. Besides, different shots have different request rates as they are statistically independent. This dynamic in the request rate is intended to reproduce the content popularity dynamic.

We modeled the activity of each device in the network with a shot. A Poisson process with constant rate λ determines the time instants when a

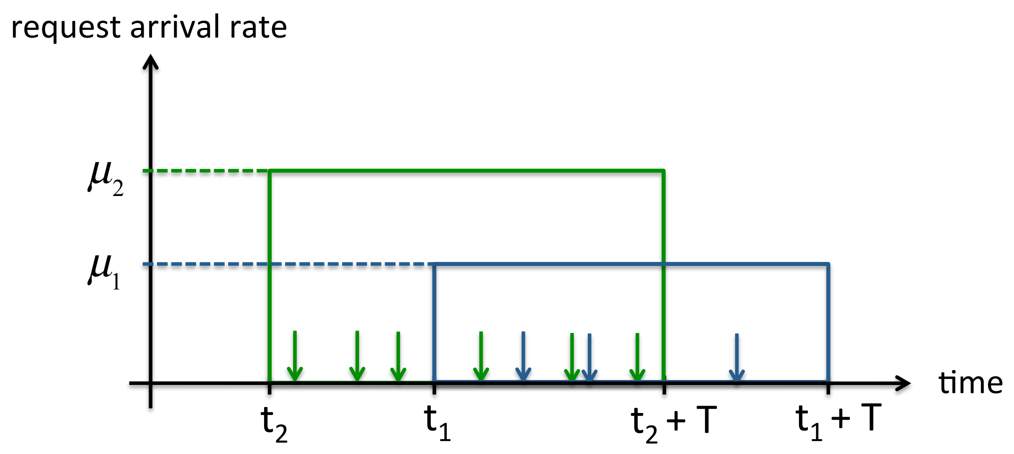


Figure 4.2. Example of two task request shots with different rates; the arrows represent the arrival instants of the requests.

device joins a base station node. Each device i is active for a period of time T during which it makes requests with a rate μ_i that is drawn from a power-law distribution with mean $\bar{\mu}$ and exponent α. If X_i is a random variable uniformly distributed in $[0,1]$, then the request rate μ_i for each device i can be calculated as

$$\mu_i = \frac{X_i^{-\alpha}}{\int_0^1 x_i^{-\alpha} dx}\bar{\mu} = X_i^{-\alpha}\bar{\mu}(1-\alpha)$$

In Figure 4.2 the requests generated by two devices with rates $\mu_1, \mu_2 : \mu_1 < \mu_2$ are represented as rectangular shots that start in different time instants.

4.3.3 Augmented reality use case

As a use case we chose augmented reality (AR) as it is a resource-hungry technology that has a wide variety of application fields like entertainment, gaming, driving, healthcare, tourism and social networking. All the applications have a common basis that is the processing of camera frames in order to provide a richer image to the user that represents an augmented reality. In order to have the final image it is first necessary to recognize objects of interest in the original frames. The main steps for object recognition are: i) the detection of objects in the original frames without labeling; ii) the extraction of notable features from the objects detected; iii) the labeling of objects through machine learning software that compares the features with large datasets. According to [59], while the third step of this pipeline is certainly the more demanding in terms of computation, the first two are lighter and can be performed locally by the device. Thus two strategies are possible, to offload the whole pipeline or only the object labeling step. We chose to implement the case in which only labeling is offloaded. This choice allows to neglect the uplink traffic for each task request, as only feature points will be uploaded by the devices.

Topology and scenario

In this augmented reality scenario we implemented a topology with two Cloud nodes, two MEC nodes and four base station nodes (two for each MEC), as shown in Figure 4.3.

The computational capabilities of the producer nodes are sized in such a way that the task processing time allows users to have an experience as much immersive as possible [60]. We envision that the producer processors have the following number of cores of 3.4 GHz (with each core with 4 virtual cores): 128 for base stations; 512 MECs; 1024 for Clouds. In order to dimension the connections between the nodes of this scenario, we refer to the latest high-capacity and low-latency communication technologies for supporting augmented reality applications [61]. For the wired section we envision the use of 10G and 100G Passive Optical Newtork (PON) (the latter scheduled for 2020). The radio access network is instead in line with 5G Key Performance Indicators (KPIs). Capacities and latencies values are summarized in Table 4.1.

Furthermore, we take into account the channel conditions in the radio access network. For a base station with $B_{max} = 100$ MHz bandwidth and $C_{max} = 1$ Gbps capacity, we assign to each device associated a spectral efficiency value uniformly distributed in the interval $[1, \eta_{max}]$, where $\eta_{max} = C_{max}/B_{max} = 10$ is the maximum spectral efficiency for a base station.

As described in section 4.3.2, a Poisson process models the joining of the devices to the base stations. For each base station the device arrival rate is $\lambda = 50$ devices/second. Each device stays in the network for a time $T = 0.004$ seconds during which it makes, on average, one request. Thus, the average task request rate is $\bar{\mu} = 1/T = 250$ requests/second. Similarly to the task request rate, demands and constraints of each task are drawn from power-law distributions. Table 4.2 summarizes the average values of the processor's cycles for data computing, the bits to transfer from the producer to the consumer and the deadline for completing both the sub-tasks. At this stage

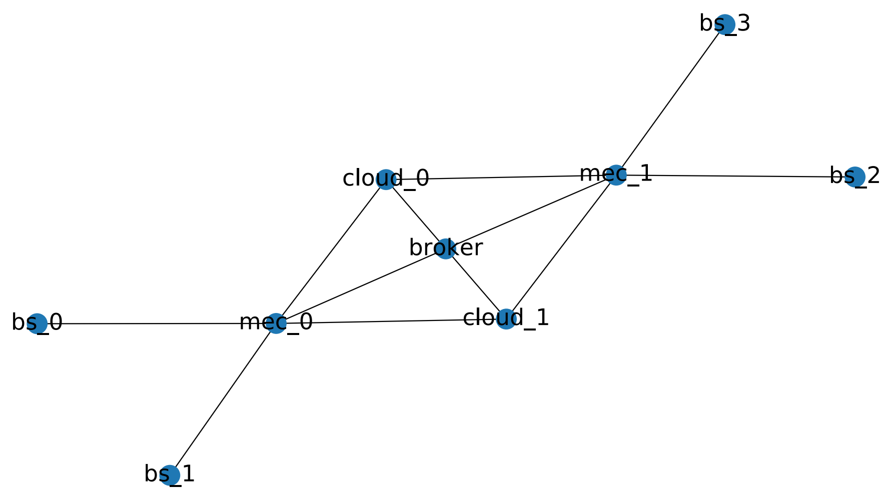


Figure 4.3. Network topology with two Cloud nodes, two MEC nodes and two base stations per MEC.

Table 4.1. Bandwidth and latencies of the links.

Link type	Bandwidth capacity	Latency
`Base station to device`	1 Gbps	1 ms
`Base station to MEC`	10 Gbps	1 ms
`MEC to Broker`	10 Gbps	10 ms
`MEC to Cloud`	100 Gbps	15 ms
`Cloud to Broker`	100 Gbps	15 ms

of research we did not apply monetary cost constraints.

Table 4.2. Power-law distributions parameters.

Object	Mean	Alpha
`Task request`	4 ms	0.8
`Computing demand`	10^9 cycles	0.48
`Bandwidth demand`	1 Mbits	0.48
`Time constraint`	100 ms	0.48

Decision-making process

As the devices populate the network, they send task requests that propagate toward the Broker. In the decision-making process the Broker takes three actions: i) search for candidates; ii) assign the task to one of the candidates according to a policy; iii) remember (or forget) the choice made.

In the candidates search phase, the Broker processes each request and looks for nodes in the network that can satisfy the demands taking into account the time constraint. Given the transfer and processing demands, respectively d_{tx} and d_{pro}, and the deadline t_{dead} the Broker calculates the minimum computing power $P_{min} = d_{pro}/t_{dead}$ and transfer rate $R_{min} = d_{tx}/t_{dead}$ required to complete the task within the deadline. By doing so, we assume that the two sub-tasks are made in parallel. Indeed, while the producer elaborates the content, it also starts the transfer toward the consumer at the same time. All the nodes that can provide at least P_{min} GHz of computing power and at least R_{min} bps in the downlink path toward the consumer are taken into consideration as candidates.

We want to remark here that P_{min} and R_{min} must be compared respectively with the computing and bandwidth capacity estimated by the Broker. We assume that the computing capacity estimation is very precise, as the producers communicate their free computing capacity instantly every time there is a level change. The producers also communicate the bandwidth occupied by the streams that are passing through them. Then for a downlink transfer with a given R_{min}, the Broker must check for each link l in the path if $C^l_{max} <= C^l_{occ} + R_{min}$, where $C^l_{occ} = \sum_n R_n$ is the bandwidth occupied in the link and R_n is the rate of the n-th stream. If the check is true for all the links in the path and the producer has enough computing capacity, then the producer can be considered as a candidate, otherwise it is discarded. If the candidates list is not empty the task is accepted, otherwise it is refused and it will be performed locally by the device itself if it has the resources to do it. We want to clarify that for each link C^l_{occ} is exact in the time instant in which the producer communicates it to the Broker. Nevertheless, when the task reaches the producer both computing and bandwidth capacity levels could be changed. To understand this aspect, we consider a case in which the Broker assigns task 1 to the serving base station of device 1 that made the request. The minimum bandwidth required for task 1 R_{min} is such that the bandwidth of the serving base station will be $C_{occ} + R^1_{min} = C_{max}$, i.e., it will reach the maximum capacity when the task will reach the producer after the propagation delay. After a negligible amount of time (considerably before the instant in which task 1 will be performed), the Broker must assign task 2 to device 2 served by the same base station. The Broker then checks again the bandwidth of the base station that is such that $C_{occ} + R^2_{min} <= C_{max}$ and assign the task to the base station. This is clearly an error because when the Broker takes the second decision the bandwidth of the base station is virtually saturated. For this reason the capacity information sent by the nodes can be considered only as an estimate. If the Broker finds more than one candidate, it applies a decision policy to choose one of them.

In the first policy the Broker just chooses randomly one of the possible candidates. In the second policy the candidates are first sorted according to the free capacity. For each candidate i two temporal values are calculated : i) the time that the candidate would take to finish the processing using all its free computing capacity $T^i_{pro} = d_{pro}/P^i_{free}$; ii) the time needed to finish the bit transfer using the minimum free capacity in the path $T^i_{tx} = d_{tx}/\arg\min_j(C^j_{free})$ where C^j_{free} is the free capacity of the $j - th$ link. With this values the Broker calculates the temporal budget left for each candidate as

$$T^i_{budget} = t_{dead} - t^i_{latencies} - T^i_{max}$$

where $t^i_{latencies}$ takes into account all the propagation delays and $T^i_{max} = \max(T^i_{pro}, T^i_{tx})$ is the maximum time expected for finishing the task without saturation delays. Finally, the Broker chooses the candidate that as the maximum temporal budget left.

After the decision is made the Broker can apply two strategies: it can forget the decision or remember it. If it forgets it, then it will take the next decision only on the basis of the capacity information communicated by the nodes, that we saw that can lead to errors. If the Broker remembers the decision made instead, it will build progressively its own personal capacity map. This map will be exact as it will take into account the decisions already made in order to prevent congestions.

Numerical results

In this section we present the results of the analysis of the augmented reality scenario. The topology, the link features and the power-law parameters for the task request are shown in Figure 4.3, Table 4.1 and Table 4.2 respectively. The producers have the following computing capacities: 64 x 4 x 3.4 GHz for base stations; 256 x 4 x 3.4 GHz for MECs; 512 x 4 x 3.4 GHz for Clouds.

In our analysis we measured the average number of tasks accepted in case of random and sorted decision by the Broker. This metric indicates if the network has enough computing and bandwidth capacity to offload the tasks of the devices. It is also an indicator of the Broker's ability to handle the resources offered by the producers in order to meet consumer demands. For each policy we evaluated the same metric in case the Broker makes decisions with or without memory. Finally we considered an ideal case in which offloading traffic is the only traffic in the network and a case with background traffic; in the latter case the Broker underestimates the links occupied bandwidth of a percentage random and uniformly distributed in the interval $[0, 10\%]$.

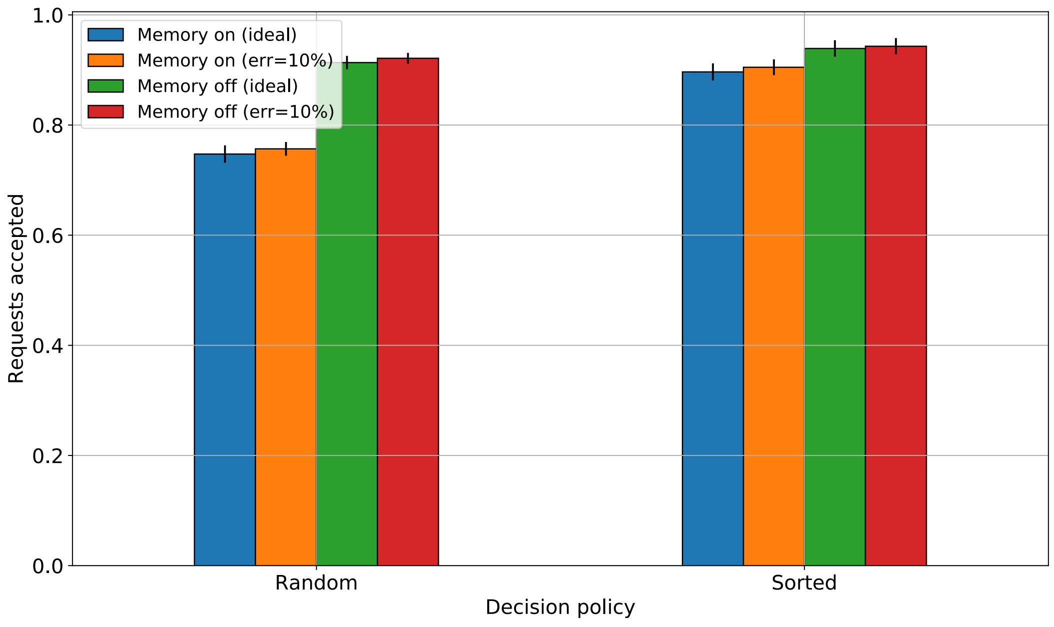
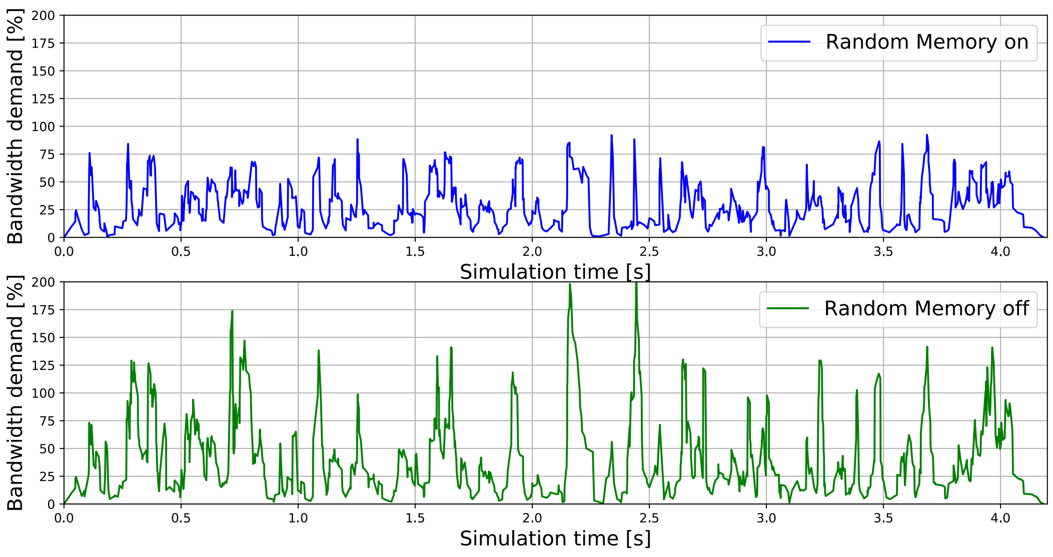


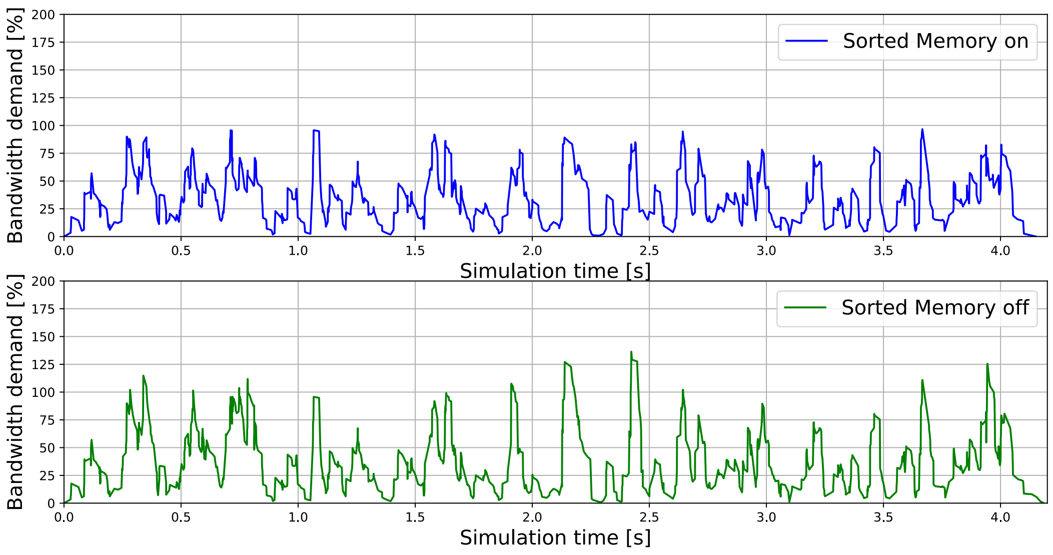
Figure 4.4. Normalized number of task requests accepted in case of random and sorted decision. Each policy is analyzed in ideal conditions or with a random estimation error and with or without memory.

Figure 4.4 shows the average number of requests accepted by the Broker in case of random or sorted decision. Different colors in the two groups identifies the cases of decision with or without memory and with or without background traffic. First of all, if we compare bars with the same color, we can see that the Broker accepts on average more requests in the sorted case. This is counterintuitive as the selection policy is applied after a candidate list is available, i.e., when there is at least one candidate. We could expect that the number of requests accepted would be similar in the two cases. The simulation results show instead a notable difference. This effect is exacerbated in the case of memory on, as it is a more conservative strategy that virtually reserves the bandwidth for the task right after the decision made. The conservative impact of memory on accepted tasks can be observed also comparing bars of different colors in the same policy group. Finally, we can see that Broker's decision-making process is robust to estimation errors due to background traffic, as it has a tiny impact on the number of accepted tasks.

In order to try to explain the differences in the number of task accepted, we analyzed the variation of bandwidth demand in the base stations in case of sorted and random decision, with and without memory. Figure 4.5 shows the bandwidth demand temporal trend in the same base station during the same simulation realization (i.e., with the same random seed). Very similar trends can be observed in the other three base stations. Given that in each time instant the base station is transferring N tasks, each point of the timelines represents the sum $\sum_{n=1}^{N} R_{min}^{(n)}$, where $R_{min}^{(n)}$ is the bandwidth required to complete within the deadline the n-th task. The total bandwidth demand is normalized to the maximum bandwidth capacity of the base station. We clarify that when the total demand exceeds the maximum capacity two effects take place: i) the Broker cannot accept other task requests from devices associated to the base station; ii) all the tasks that are being transferred by the base station will experience saturation delay (and virtually will be completed beyond the deadline). We analyzed random and sorted policies in Figures 4.5(a) and 4.5(b) respectively. First of all we can observe the impact of memory in both cases looking at the blue curves. When the memory is on the total demand never exceeds the maximum bandwidth of the base station. On the contrary, when the memory is off we can see the total demand going beyond the 100%. These results confirm what we observed in Figure 4.4, as the presence of bottlenecks reduce the number of requests accepted. Then we can compare the two policies to each other when the memory is off. If we look at the green curves we can see that the random policy is worst by far as it reaches 200% in some cases while the sorted policy reaches the 125% at maximum. Besides, if we compare the blue curves to each other the sorted case seems to go more often beyond the 75% level. It seems that in case of memory on, the random policy more often under utilizes the base station's bandwidth. Summarizing, on one hand it is clear the beneficial effect of the memory in the resources utilization. On the other hand, it results that in the random case network's resources are under-utilized when the memory is on and over-utilized when the memory is off. Thus, we can say that the sorted



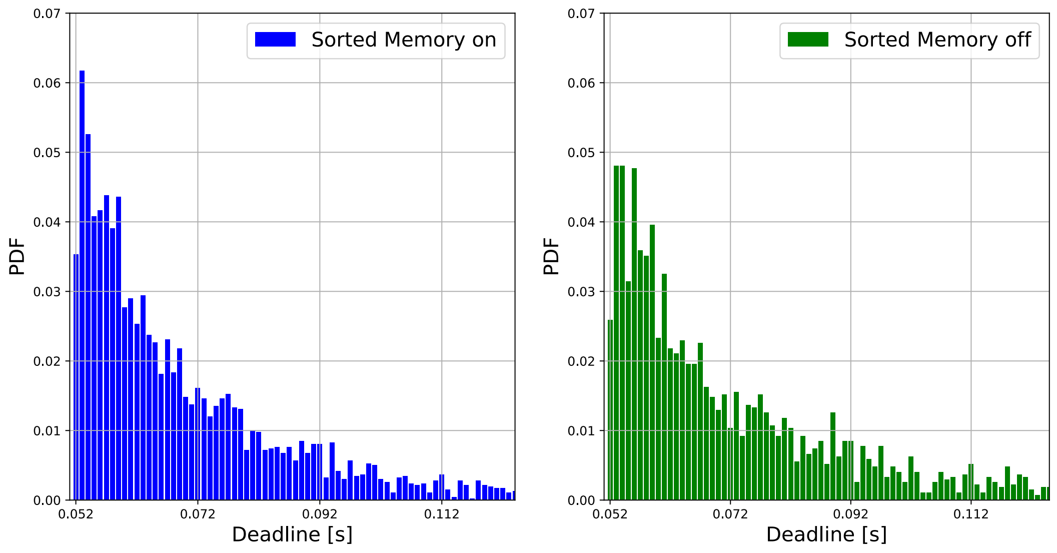
(a)



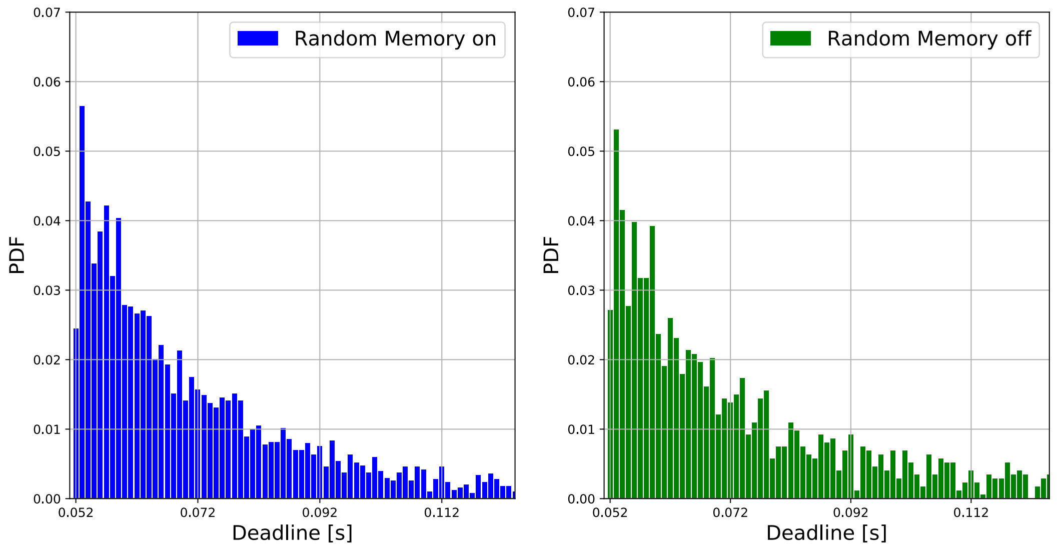
(b)

Figure 4.5. Bandwidth demand variation on the same base station in the same simulation realization in four different cases: Random or Sorted and with or without memory in ideal conditions.

policy makes a better use of network's resources, although a deeper analysis that correlates bandwidth usage with accepted tasks is needed to prove this



(a)



(b)

Figure 4.6. Distribution of the deadline of the refused requests in four different cases: Random or Sorted and with or without memory in ideal conditions.

point.

In Figure 4.6 we show the distribution of deadlines of refused requests in different cases. We compared the random and sorted policies in ideal conditions, i.e., the blue and green bars of Figure 4.4. We can see that the distributions are very similar and there is no characteristic pattern as they have the shape of the power-law (although they are truncated in the figures to hide the long tail). In particular we can see that in all cases there is a clear shift of refused requests towards lower deadlines. This was expected as tasks with more stringent deadlines have higher P_{min} and R_{min}, as described in the previous subsection, and then are more difficult to serve. Finally, we can observe that the shift toward left is less pronounced in case of memory off (i.e., the green curves).

Accepted tasks can be completed on time or late, in case they find saturated links. Figure 4.7 shows the average percentage of requests accepted and completed on time. We consider this metric as a quality of service indicator, as tasks completed on time lead to a better user experience. Comparing bars of the same color we can see that the sorted policy leads to a significant performance gain with respect to the random one, about 15% and 20% more in case of ideal conditions with memory and without memory respectively. Comparing the bars of the same policy instead we can observe the huge impact of memory. For the random policy, we have about the 25% more of tasks on time in ideal conditions if we use memory with respect to if we do not. For the sorted policy, we have instead a gain of tasks on time of about 22% if we use memory in ideal conditions. Besides, these results match well with the demand variation analysis, as the high demand peaks shown in the green curves of Figure 4.5 leads to bottlenecks and then to an high number of refused requests. We can observe that also in this case the Broker's decision-making is robust to the bandwidth estimation error.

4.4 Conclusion and future work

In the context of computational offloading for wireless devices, we envisioned a network marketplace where computing resources can be exchanged by resource producers and consumers. The central and innovative element of the marketplace is represented by a Broker that autonomously matches supply and demand of resources acting as an intermediary between consumers and producers. In order to assess the feasibility of such a vision, we developed a discrete-event, object-oriented, fluid network simulator capable of analyzing different scenarios where computational task requests are modeled with Poisson SNM. We then presented and analyzed an augmented reality use case, where tasks requested by wireless devices are offloaded to producers in

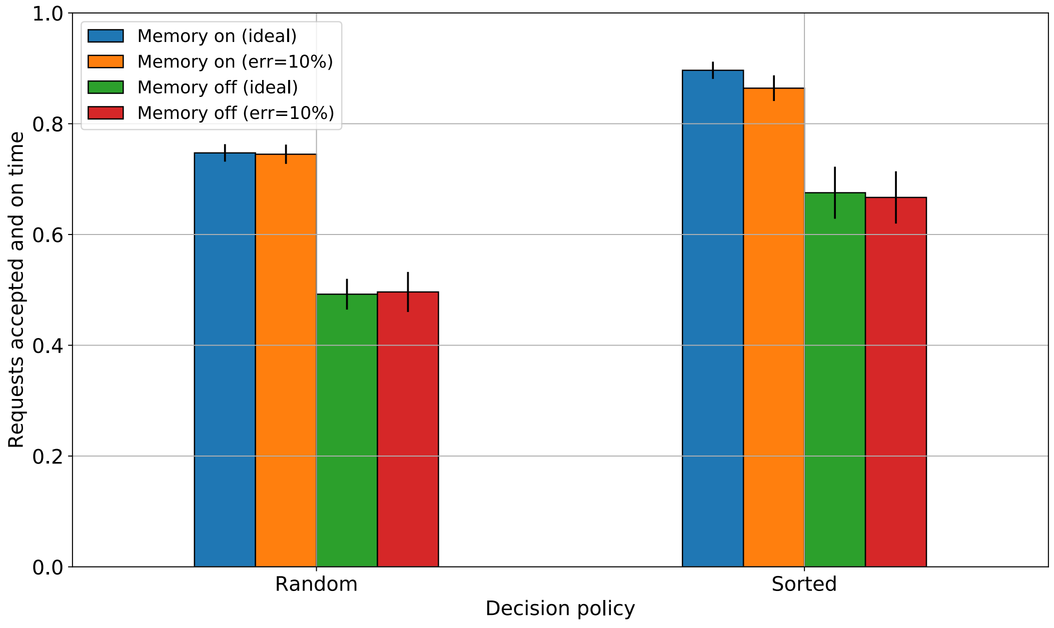


Figure 4.7. Normalized number of task requests accepted and completed on time in case of random and sorted decision. Each policy is analyzed in ideal conditions or with a random estimation error and with or without memory.

a topology with two Cloud nodes, two MEC nodes and four bases stations with different computing capabilities. In this scenario, the Broker takes the following actions: i) finds producer candidates that can meet the task demands within the deadline; ii) selects one the candidates using a random or sorted decision policy; iii) reminds or forgets the decision made. In our analysis we compared the two policies and evaluated the impact of the memory and of errors in the bandwidth estimation due to background traffic. Our results show that the sorted policy makes a better usage of network's resources with respect to the random policy, as the latter leads to more bottlenecks. Besides, the sorted policy improves considerably the quality of experience as it shows by far the greatest number of task accepted and completed on time. Lastly, we observed that Broker's decision-making process is robust to estimation errors.

We plan to complete and extend this work in many directions. The first target is to complete the analysis of the augmented reality scenario: finding a correlation between the resource usage and the accepted (and refused) tasks; testing other networks configurations both in terms of computing and bandwidth capacities; applying monetary cost constraints. Then, we want to evaluate other decision policies. For example a geographic policy that gives priority to the nearest producer candidate. More importantly, we want to assess a policy that takes into account the monetary cost of computation in the different type of producers of the network. We also plan to apply reinforcement learning to the Broker's decision-making process, introducing a feedback mechanism that assigns a reward for each task completed on time.

Chapter 5

Conclusions

In this thesis work we met some of the most important challenges of emerging wireless networks, focusing on scenarios characterized by an high density of devices, base stations or both. We used the 5G vision as a guideline, due to its ambitious plan of integrating the latest networking and computing technologies in one programmable and unified infrastructure. Our activity followed the three main research directions outlined by 5G vision:

- novel radio access technologies for heterogeneous application domains such as IoT and AR/VR;
- softwarization of the radio access network through SDN technology in order handle lower layers complexity, fluidify networks operations and provide context data;
- employment of the latest computing architectures for bringing artificial intelligence as close as possible to users.

These three research topics are covered in chapters 2, 3 and 4. In chapter 2 we analyzed the impact of two peculiar characteristics of LoRa modulations, i.e. the high capture probability and the imperfect orthogonality between different SFs, on the overall cell capacity. Although parallel reception of multiple overlapping frames is generally possible, we showed that the link-level performance of LoRa is deeply influenced by capture effects and by inter-SF collisions which can indeed cause loss if the interference power is strong enough.

In this research work we have studied the impact of two peculiar characteristics of LoRa modulations, i.e. the high capture probability and the imperfect orthogonality between different SFs, on the overall cell capacity.

We then exploited this link-level analysis to model analytically the achievable network capacity in a typical LoRa cell. We showed that high SFs are severely affected by inter-SF interference and that the use of power control and packet fragmentation to compensate such problem may be counterproductive. Although deploying multiple gateways can mitigate the capacity loss and boost the occurrence of channel captures, the overall capacity increase becomes negligible after 16-24 gateways. Finally, when only a handful of gateways are present, the deployment should be as distant from the center as possible and not on a regular grid. We believe that the results presented in this work provide important insights on LoRa technology and can pave the way to new accurate guidelines for the correct design of future LoRa networks. In chapter 3, we presented a flexible platform, implemented on top of OpenAirInterface (OAI), able to control, monitor and build radio-maps of indoor environments, characterized by high-density of devices, that can be used for localization. We proposed an architecture that develops on three levels and tackles all the challenges of the emerging Software-Defined-Networking (SDN) paradigm. We developed a centralized controller able to monitor the network status and make context-dependent decisions which are enforced by the soft-nodes via the OAI-Agents. Furthermore, we defined a northbound API through which an orchestration app can build a radio-map of the environment. We compared the performances obtained by three different classifiers trained with the radio-maps collected. Our analysis shows that K-Nearest Neighbors (KNN) has the best performances both in terms of accuracy and computational cost with respect to the two neural networks. Indeed, it is is possible to localize a device in high-density femtocells indoor deployments updating regularly the maps. This research work is currently active as we plan to enrich the southbound API in order to deeply customize the soft-nodes. Then, we want to implement a network slicing use case that exploits user's position information in order to allocate radio resources to different users according to their service requirements. In chapter 4, we proposed a network marketplace where wireless devices can buy computing resources from different network players at different costs. The central and innovative element of the marketplace is represented by a Broker that autonomously matches supply and demand of resources acting as an intermediary between consumers and producers. We showed with a discrete-event, object-oriented, fluid simulator that we developed in Python the potentialities of our architecture comparing a random and a sorted decision policies. We also evaluated the impact of keeping memory of decisions and of errors in the bandwidth estimation due to background traffic. Our results show that the sorted policy makes a better usage of network's resources with respect to the random policy, as the latter leads to more bottlenecks. Besides, the sorted policy improves considerably the quality of experience as it shows by far the greatest number of task accepted and completed on time. Lastly, we observed that Broker's decision-making process is robust to estimation errors. We plan to complete and extend this work in many directions. The first target is to complete the analysis of the augmented reality scenario: finding a correlation between the resource usage and the accepted (and refused) tasks; testing other networks configurations both in terms of computing and bandwidth capacities; applying monetary cost constraints. Then, we want to evaluate other decision policies. For example a geographic policy that gives priority to the nearest producer candidate. More importantly, we want to assess a policy that takes into account the monetary cost of computation in the different type of producers of the network. We also plan to apply reinforcement learning to the Broker's decision-making process, introducing a feedback mechanism that assigns a reward for each task completed on time.

Appendix A

Developed tools

A.1 Soft-nodes general controller

```
import socket
import sys
import threading
import json
import time

def receive_from_terminal():
  # 1. Wait for command line instructions
  print("Select␣the␣Node␣to␣connect:␣(from␣0␣to␣3).")
  print("Type␣ESC␣to␣close␣the␣program")
  command = input()

  if command == "ESC":
    print("Exit␣the␣program.")
    sys.exit()

  else:
    cell_id = int(command)
    if 0 <= cell_id <= 3:
      print(
        "\n\nSelect␣Your␣action:\n\n1␣-␣SEND␣RRC_MEASUREMENT␣\n\n2␣-␣"
        "SEND␣SINGLE␣RRC_MEASURMENTS␣AND␣STOP␣RRC_MEASUREMENT\n\n"
        "3␣-␣DISCONNECT␣UE␣FROM␣NODE\n\n4␣-␣Update␣Information\n\n")
      action = input()

      local_selection = None
      while local_selection is None:
        print("Waiting␣for␣users...")
        time.sleep(1)
        lock.acquire()
        # copy attached_users to temp var inside lock (thread-safe)
        attached_users_tmp = attached_users
        lock.release()
        if len(attached_users_tmp[cell_id]) > 0:
          print("-----␣List␣of␣attached␣users␣to␣cell␣with␣ID␣{}-----\n".format(
            cell_id))
          # counter = 0

          num_of_users_in_this_cell = len(attached_users_tmp[cell_id])
          for users in range(0, num_of_users_in_this_cell):
            print("User␣{}␣crnti:␣{}\n".format(users + 1,
                                              attached_users_tmp[cell_id][
                                                users]))

          print("\n\nSelect␣one␣of␣the␣listed␣users:")
          attached_user_idx = input()
          local_selection = (cell_id, int(action), attached_users_tmp[cell_id][
            int(attached_user_idx) - 1])

        else:
          print("No␣users␣attached.\n")
          break

    else:
      print("Invalid␣Cell␣ID.␣Range␣is␣from␣0␣to␣3.")
      local_selection = None

    return local_selection


def oai_controller():
  sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

  while True:
    # Receives instructions from command line
    selection = receive_from_terminal()

    if selection is not None:
      cell_id = selection[0]
      interface_address = (
      ip_address, port)  # The message is transmitted to the splitFunction

      message2cell = {cell_id: 0}
      messageInformation = {'crnti': selection[2], 'starting_rgb': 0,
                            'num_of_rgb': 12, 'message_flag': 0,
                            'message_rrc': 0, 'reset_idx': -1}

      if 1 <= selection[1] <= 4:
        action = selection[1]
        if action == 1:
          messageInformation['message_rrc'] = 1

        elif action == 2:
          messageInformation['message_rrc'] = 2

        elif action == 3:
          messageInformation['message_rrc'] = 3

        elif action == 4:
          messageInformation['message_rrc'] = 0

        message2cell.update({cell_id: messageInformation})
        message = json.dumps(message2cell, indent=2)
        sock.sendto(message.encode(), interface_address)
      print(
        "Sent␣{}␣bytes␣to␣{}.\n".format(len(message), interface_address[0]))

    else:
      print("Invalid␣selection.␣Range␣is␣from␣1␣to␣4")


def receive_from_oai():
  sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
  sock.bind(controller_rx_addr)

  while True:
    data, address = sock.recvfrom(4096)  # receive data from oai_controller_rx
    received = json.loads(data.decode())  # type dict
    lock.acquire()
    for cell_id in received:
      attached_users[int(cell_id)].clear()
      counter = 0
      for users in received[cell_id]:
        # put crnti of user in attached_users list
        attached_users[int(cell_id)].append(received[cell_id][users]['id'])
        counter = counter + 1

    lock.release()


if __name__ == "__main__":
  # send and receive data always from splitFunction
  port = 5000
  ip_address = "163.162.89.28"

  splitFunction_addr = (ip_address, 5000)

  controller_rx_addr = ('163.162.97.33', 6001)

  cell0_users = list()
  cell1_users = list()
  cell2_users = list()
  cell3_users = list()
  attached_users = list([cell0_users, cell1_users, cell2_users, cell3_users])
  lock = threading.Lock()  # Lock

  to_oai_thread = threading.Thread(target=oai_controller)
  from_oai_thread = threading.Thread(target=receive_from_oai)

  to_oai_thread.start()
  from_oai_thread.start()
```

A.2 Soft-nodes measurements campaign controller

```
import socket
import sys
import threading
import json
import oai_interface as oai_ifc
import oai_database as oai_db
import oai_controller as oai_ctrl
import pymongo
import time


def user_options():
  valid = False

  while not valid:

    print("\nSELECT␣ONE␣OF␣THE␣FOLLOWING␣OPTIONS:\n")
    print("1␣-␣START␣EXPERIMENT")
    print("2␣-␣RECOVER␣EXPERIMENT")
    print("3␣-␣EXIT")

    choice = input("\n-->\n")

    if choice == '1' or choice == '2' or choice == '3':
      valid = True
    else:
      print("\nYour␣choice␣MUST␣be␣'1'␣or␣'2'␣or␣'3'")

  return choice


def get_squares(mongo_id):
  search = {'_id': mongo_id}
  topo = oai_db.search_db(exp_collection_ref, search, 'topology')
  # db_search returns a list
  completed_squares = oai_db.search_db(exp_collection_ref, search,
                                        'completed_squares')

  topo = topo[0]
  completed_squares = completed_squares[0]  # get the first element only

  print("This␣is␣the␣topology␣of␣the␣experiment:\n{}\n".format(topo))

  to_be_completed = []
  if not completed_squares:
    print("\nNo␣squares␣are␣still␣completed.\n\n")
  else:
    print("\nThe␣following␣squares␣have␣been␣completed:\n␣{}".format(
      completed_squares))

  for row in topo:
    for square in row:
      if square not in completed_squares:
        to_be_completed.append(square)

  return topo, to_be_completed


def select_square(topo, squares):
  final = False
  while final is False:

    print("Select␣one␣of␣the␣following␣uncompleted␣squares:␣{}".format(squares))
    selected = input()

    topo_loc = []
    for rows in topo:
      for sqr in rows:
        topo_loc.append(sqr)

    if selected in topo_loc:

      if selected not in squares:

        print(
          "WARNING:␣You␣selected␣a␣square␣which␣has␣been␣completed␣already.\n"
          "Do␣you␣want␣to␣overwrite?")

        final_selection = input("(y/N)")
        if final_selection == "y":

          print(
            "\n------------------␣You␣will␣overwrite␣the␣measurements␣in␣{}␣----------------"
            "-----".format(selected))
          final = True

        else:
          print("Aborted.")
      else:

        print(
          "\n------------------␣Square␣{}␣selected␣------------------\n".format(selected))
        final = True

    else:

      print("Selection␣not␣valid.\n")

  return selected


def receive_from_oai(receiver, net_info, net_info_lock, store_data,
                     user_info=None, meas_num=None, db_info=None):
  sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
  sock.bind(receiver)

  do_run = True
  meas_count = 0
  while do_run:

    data, address = sock.recvfrom(4096)  # receive data from the network
    node_info = json.loads(data.decode())
    cell_id = next(iter(node_info))

    if not store_data:

      net_info_lock.acquire()
      # update network info
      net_users_num = oai_ctrl.update_attached_users(node_info, net_info,
                                                     cell_id)
      net_info_lock.release()
else:

      if meas_count == 0:

        c_id = user_info[0]
        crnti = user_info[1]
        tmsi = user_info[2]
        square = user_info[3]

        exp_coll_ref = db_info[0]
        data_coll_ref = db_info[1]
        mongo_id = db_info[2]

      else:
        pass

      if cell_id == c_id:

        for users in node_info[c_id]:

          if node_info[c_id][users]['id'] == crnti == tmsi or \
                  node_info[c_id][users]['tmsi'] == tmsi:  # soft check

            oai_db.dump_oai_data(data_coll_ref, mongo_id, node_info, square)
            meas_count += 1
            print("\n(((MEASURES))) --- {} ({} remaining)".format(meas_count,
                                                            meas_num - meas_count))

      # When reach num of measures needed end the thread
      if meas_count == meas_num:

        completed_sq = oai_db.search_db(exp_coll_ref, {'_id': mongo_id},
                                         'completed_squares')
        completed_sq[0].append(square)
        exp_coll_ref.update_one({'_id': mongo_id}, {
          '$set': {'completed_squares': completed_sq[0]}})
        do_run = False
        print("\n(((Message))) --- Measures finished!\n\n")


if __name__ == "__main__":

  port = 5000
  ip_address = (
  "163.162.97.5", "163.162.97.72", "163.162.97.170", "163.162.97.43")

  receiver1_addr = ('0.0.0.0', 6001)
  receiver2_addr = ('0.0.0.0', 9998)

  letters = ('A', 'B', 'C', 'D', 'E', 'F', 'G')

  # mongo_host = ('163.162.89.28', 27017)          # TIM Lab
  # db_name = 'tim_experiment_test'
  db_name = 'TIM_experiments_last'
  # mongo_host = ('192.168.1.151', 27017)        # Strummer
  mongo_host = ('10.8.20.7', 27017)  # Strummer
  # db_name = 'tim_experiment_test_copy'
```

```
exp_collection_name = 'experiment_info_test'
meas_collection_name = 'oai_data_test'

mongo_client = pymongo.MongoClient(mongo_host[0], mongo_host[1])

exp_collection_ref = mongo_client[db_name][exp_collection_name]
data_collection_ref = mongo_client[db_name][meas_collection_name]

cell0_users = list()
cell1_users = list()
cell2_users = list()
cell3_users = list()
attached_users = list([cell0_users, cell1_users, cell2_users, cell3_users])

experiment_info = {'author': None, 'description': None, 'duration': None,
                   'time_resolution': None,
                   'topology': None, 'bs_location': [], 'bs_power': [],
                   'completed_squares': []}

lock = threading.Lock()  # Lock

tx_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
lower_fun_addr = ('163.162.89.28', 5000)

from_oai_thread = threading.Thread(target=receive_from_oai, args=(
receiver2_addr, attached_users, lock, False))
from_oai_thread.start()

while True:

  experiment = {}

  choice = user_options()

  if choice is '1':

    experiment.update(oai_db.get_experiment_data(exp_collection_ref))

    """ Create experiment collection in MongoDB """
    mongo_experiment_id = oai_db.create_experiment(exp_collection_ref,
                                                   experiment)

  elif choice is '2':

    mongo_experiment_id, experiment = oai_db.get_experiment_id(
      exp_collection_ref)

  elif choice is '3':

    print('\n---␣Goodbye␣---')
    sys.exit()

  topology, todo_squares = get_squares(mongo_experiment_id)

  selected_square = select_square(topology, todo_squares)

  # Put user in square
```

```
print(
  "\n———— Put the phone in {} and turn off airplane mode ————".format(
    selected_square))
print("———— Press any key when you have done ————")
input("——>\n")

# Wait for users attach to the network
attached = False

while not attached:

  lock.acquire()
  net_users_num = oai_ctrl.count_attached_users(attached_users)
  lock.release()

  if net_users_num is not 0:

    lock.acquire()
    attached, selection = oai_ctrl.select_user(attached_users, None,
                                               net_users_num)
    lock.release()
    cell_id = selection[0]
    crnti = selection[1]
    tmsi = selection[2]

  else:

    print(
      "\n(((Message))) ——— No users attached ——>"
      "Checking again in 4 seconds...")
    time.sleep(4)

lock.acquire()
net_info = attached_users.copy()

lock.release()

# Check if measurements are active once
rsrq_serving, new_crnti, activated, tmsi_is_valid = \
  oai_ctrl.check_measures_activation(net_info, None, tmsi)
final_decision = False
while not final_decision:

  if activated:

    print(
      "\n\n(((Message))) ——— User {} (crnti {}) is attached to == {} =="
        .format(tmsi, crnti, cell_id))
    print("\n(((Message))) ——— Measures are active yet!")
    print("\n\n———— Press 'R' if you want to RESET the measures ————")
    print("\n———— Press 'C' if you want to CONTINUE ————")
    decision = input("\n——>\n")

    if decision is 'R':

      print("\n(((Message))) ——— Measures will be reset!")
      activated = False
      final_decision = False
```

```
        elif decision is 'C':

          print("\n(((Message)))␣−−−␣Measures␣are␣going␣to␣be␣stored␣in␣DB...")
          final_decision = True

    else:

      # Activate measurements in UE
      message = oai_ifc.build_msg_for_agent(1, cell_id, crnti)
      tx_socket.sendto(message.encode(), lower_fun_addr)

      time.sleep(2)
      activated = False
      final_decision = True

      while not activated:

        lock.acquire()
        net_info = attached_users.copy()
        lock.release()

        rsrq_serving, new_crnti, activated, tmsi_is_valid = \
          oai_ctrl.check_measures_activation(net_info, None, tmsi)

        if not activated:
          time.sleep(2)

      print("\n(((Message)))␣−−−␣Measurements␣have␣been␣activated!")
      print(
        "(((Message)))␣−−−␣User␣{}␣(crnti␣{})␣is␣attached␣to␣==␣{}␣=="
          .format(tmsi, new_crnti, cell_id))
      print("(((Message)))␣−−−␣Measures␣are␣going␣to␣be␣stored␣in␣DB...")

  # Get time_resolution and exp duration
  time_res = oai_db.search_db(exp_collection_ref,
                              {'_id': mongo_experiment_id}, 'time_resolution')
  duration = oai_db.search_db(exp_collection_ref,
                              {'_id': mongo_experiment_id}, 'duration')

  # number of measurements to gather for each square
  meas_num = int(duration[0]) / int(time_res[0])
  user_info = (cell_id, new_crnti, tmsi, selected_square)
  db_info = (exp_collection_ref, data_collection_ref, mongo_experiment_id)

  # Start thread that stores measures into DB
  store_measure_thread = threading.Thread(target=receive_from_oai,
                                          args=(
                                          receiver1_addr, attached_users,
                                          lock, True, user_info, meas_num,
                                          db_info))
  store_measure_thread.start()
  store_measure_thread.join()  # Wait here until measures finish
```

A.3 Fluid simulator Broker Class

The network simulator presented in Chapter 4 has seven classes. In this Section we show the Broker Class that contains the Broker's decision-making process.

```
import networkx as nx
from scipy import stats
import simpy
import matplotlib

matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import time
import sys
import getopt

class Broker:

  def __init__(self, sim_env, network):
    self.sim_env = sim_env
    self.network = network  # Broker knows everything about the network
    self.rand_count = 100

  def find_direction(self, src, dst):
    if (src[0:2] == 'de' and dst[0:2] == 'bs') or (
          src[0:2] == 'bs' and dst[0:2] == 'me') \
          or (src[0:2] == 'me' and dst[0:2] == 'cl') or (
          src[0:2] == 'me' and dst[0:2] == 'br') \
          or (src[0:2] == 'br' and dst[0:2] == 'cl'):
      direction = 'uplink'

    elif (src[0:2] == 'bs' and dst[0:2] == 'de') or (
          src[0:2] == 'me' and dst[0:2] == 'bs') \
          or (src[0:2] == 'cl' and dst[0:2] == 'me') or (
          src[0:2] == 'br' and dst[0:2] == 'me') \
          or (src[0:2] == 'cl' and dst[0:2] == 'br'):
      direction = 'downlink'

    elif src[0:2] == dst[0:2]:
      if src[src.index('_') + 1:] < dst[dst.index('_') + 1:]:
        direction = 'downlink'
      else:
        direction = 'uplink'

    else:
      pass

    return direction

  def check_path_rate_old(self, path, rate_req, spec_eff):
    isok = True
    if self.network.broker_error == True:
      error = round(int(stats.randint(0, 9).rvs(size=1,
                                random_state=self.rand_count)) * 0.01, 2)
    else:
      error = 0
    isok = True
    """
    A. Check if there is enough bandwidth on the base station
    """
    B = self.network.topology.node[path[-2]]['downlink']['streams_table'].loc[
      self.network.topology.node[path[-2]]['downlink']['streams_table'][
        'active'] == True, 'bw_req'].sum()
    Bestimate = B * (1 - error)

    if round(rate_req / spec_eff, 8) > \
          self.network.topology.node[path[-2]]['downlink'][
            'Bmax'] - Bestimate:
      isok = False
    else:
      pass

    """
    B. Check if there is enough bandwidth on the other edges of the path
    """
    if isok:
      for hh in range(0, len(path) - 2):
        edge = (path[hh], path[hh + 1])
        direction = self.find_direction(path[hh], path[hh + 1])
        if len(self.network.topology.edges[edge][direction]['streams_table'][
                  'active']) > 0:
          C = self.network.topology.edges[edge][direction]['streams_table'].loc[
            self.network.topology.edges[edge][direction]['streams_table'][
              'active'] == True, 'rate_req'].sum()
          Cestimate = C * (1 - error)

          if rate_req > self.network.topology.edges[edge][direction][
            'Cmax'] - Cestimate:
            isok = False
            break
          else:
            pass
        else:
          pass

    else:
      pass

    return isok

  def check_path_rate_new(self, path, rate_req, spec_eff):
    isok = True
    if self.network.broker_error == True:
      error = round(int(stats.randint(0, 9).rvs(size=1,
                                    random_state=self.rand_count)) * 0.01, 2)
    else:
      error = 0
    """
    A. Check if there is enough bandwidth on the base station
    """
    Bestimate = self.network.topology.node[path[-2]]['downlink']['Bmax'] - (
          self.network.topology.node[path[-2]]['downlink']['Bmax'] - \
            self.network.topology.node[path[-2]]['downlink']['bw_available'])\
                  * (1 - error)
    if (round(rate_req / spec_eff, 8) >
            self.network.topology.node[path[-2]]['downlink'][
              'Bmax']) or \
              (Bestimate - round(rate_req / spec_eff, 8) < 0):
        isok = False
    else:
        pass
    """
    B. Check if there is enough bandwidth on the other edges of the path
    """
    if isok:
        for hh in range(0, len(path) - 2):
            edge = (path[hh], path[hh + 1])
            direction = self.find_direction(path[hh], path[hh + 1])
            Cestimate = self.network.topology.edges[edge][direction]['Cmax'] - (
                    self.network.topology.edges[edge][direction]['Cmax'] - \
                    self.network.topology.edges[edge][direction]['bw_available']) \
                          * (1 - error)
            if (rate_req > self.network.topology.edges[edge][direction][
                'Cmax']) or (
                    Cestimate - rate_req < 0):
                isok = False
                break
            else:
                pass

    else:
        pass

    return isok

def reserve_bw(self, req):
    self.network.topology.node[req.dl_path[-2]]['downlink']['bw_available'] \
      -= round(req.rate_req / req.spec_eff, 8)
    for idx, edge in zip(range(0, len(req.dl_edges_new)),
                         req.dl_edges_new[:-1]):
        self.network.topology.edges[edge['name']][edge['direction']][
          'bw_available'] -= req.rate_req

def select_random_seller(self, req):

    seller_candidates = pd.DataFrame(
        columns=['t_deliver', 'comp_req', 'rate_req', 'comp_cost', 't_tx',
                 't_com',
                 'budget_left'])
    # 1. calculate Tdeliver for all the nodes in the network
    for seller in self.network.allnodes:
        dl_delay = self.network.find_path_latency(
            nx.shortest_path(self.network.topology, seller, req.dev_id))
        toseller_delay = self.network.find_path_latency(
            nx.shortest_path(self.network.topology, 'broker', seller))
        Tdeliver = req.deadline - req.ul_delay - toseller_delay - dl_delay
        Tdeliver = round(Tdeliver, 8)

        """ for each candidate calculate rate, computing and price demand """
        # 2. eliminate nodes with Tdeliver<=0
    if Tdeliver > 0 and dl_delay != 0:
      # 3. calculate rate and computing demands for Tdeliver
      rate_req = round(req.bits_dem / Tdeliver, 8)
      comp_req = round(req.comp_dem / Tdeliver, 8)
      comp_cost = self.network.allnodes[seller].computing_price
      seller_candidates.loc[seller] = [Tdeliver, comp_req, rate_req,
                                       comp_cost, None, None, None]
    else:
      pass

  # 4. eliminate nodes with insufficient computing
  if len(seller_candidates) != 0:

    for candidate in seller_candidates.index:
      if self.network.allnodes[candidate].computing_available < \
            seller_candidates.loc[candidate]['comp_req']:
        seller_candidates = seller_candidates.drop([candidate], axis=0)
      else:
        pass

    # 5. eliminate nodes with insufficient rate in the path
    if len(
          seller_candidates) != 0:

      for candidate in seller_candidates.index:
        """ MEMORY OR NOT? """
        if self.network.memory == True:
          path_isok = self.check_path_rate_new(
            nx.shortest_path(self.network.topology, candidate,
                             req.dev_id),
            seller_candidates.loc[candidate]['rate_req'],
            req.spec_eff)
        else:
          path_isok = self.check_path_rate_old(
            nx.shortest_path(self.network.topology, candidate,
                             req.dev_id),
            seller_candidates.loc[candidate]['rate_req'],
            req.spec_eff)

        if not path_isok:
          seller_candidates = seller_candidates.drop([candidate], axis=0)
        else:
          pass

    else:
      pass

  else:
    pass

  # 6. choose randomly one of the nodes remained
  if len(seller_candidates) != 0:
    """ RANDOM OR SORTED? """
    if self.network.broker_choice == 'random':
      thewinneris = seller_candidates.iloc[
        int(stats.randint(0, len(seller_candidates)).
            rvs(size=1,
                random_state=self.rand_count))].name
      elif self.network.broker_choice == 'sorted_1':
        thewinneris = self.choose_least_loaded(req, seller_candidates,
                                               parallel_tx=True)

      elif self.network.broker_choice == 'sorted_2':
        thewinneris = self.choose_least_loaded(req, seller_candidates,
                                               parallel_tx=False)

      else:
        pass

      self.rand_count += 1
      req.seller_id = thewinneris
      req.Tdeliver = seller_candidates.loc[thewinneris]['t_deliver']
      req.comp_req = seller_candidates.loc[thewinneris]['comp_req']
      req.rate_req = seller_candidates.loc[thewinneris]['rate_req']
      req.comp_cost = seller_candidates.loc[thewinneris]['comp_cost']
      req.toseller_path = nx.shortest_path(self.network.topology, 'broker',
                                           thewinneris)
      req.toseller_delay = self.network.find_path_latency(req.toseller_path)
      req.dl_path = nx.shortest_path(self.network.topology, thewinneris,
                                     req.dev_id)
      req.dl_delay = self.network.find_path_latency(req.dl_path)
      for h in range(1, len(req.dl_path)):
        edge = (req.dl_path[h - 1], req.dl_path[h])
        direction = self.find_direction(req.dl_path[h - 1], req.dl_path[h])
        req.dl_edges_new.append({'name': edge, 'direction': direction})

      req.dl_edges = [(req.dl_path[h - 1], req.dl_path[h]) for h in
                      range(1, len(req.dl_path))]
      self.reserve_bw(req)
      self.network.req_accepted += 1

      return thewinneris

    else:
      # There are no candidates for the task :(
      req.refused = True
      self.network.req_refused += 1

      return None

  def choose_least_loaded(self, req, candidates, parallel_tx=False):

    for candidate in candidates.index:
      candidates.loc[candidate, 't_com'] = req.comp_dem / self.network.allnodes[
        candidate].computing_available

      # calc theorical minimum processing time
      dl_path = nx.shortest_path(self.network.topology, candidate, req.dev_id)
      dl_edges = []
      bws = []
      for h in range(1, len(dl_path)):
        edge = (dl_path[h - 1], dl_path[h])
        direction = self.find_direction(dl_path[h - 1], dl_path[h])

        if h != len(dl_path) - 1:
          edge_ref = self.network.topology.edges[edge][direction]
```

```
            bw_available = edge_ref['Cmax'] - edge_ref['streams_table'].loc[
              edge_ref['streams_table']['active'] == True, 'rate_req'].sum()
          else:
            edge_ref = self.network.topology.nodes[edge[0]][direction]
            bw_available = edge_ref['Bmax'] - edge_ref['streams_table'].loc[
              edge_ref['streams_table']['active'] == True, 'bw_req'].sum()
            bw_available = req.spec_eff * bw_available

          bws.append(bw_available)

        candidates.loc[candidate, 't_tx'] = req.bits_dem / min(bws)

        if parallel_tx:
          candidates.loc[candidate, 'budget_left'] = candidates.loc[
            candidate, 't_deliver'] - max(candidates.loc[candidate, 't_tx'],
                                      candidates.loc[candidate, 't_com'])
        else:
          candidates.loc[candidate, 'budget_left'] = candidates.loc[
            candidate, 't_deliver'] - (candidates.loc[candidate, 't_tx'] +
                                      candidates.loc[candidate, 't_com'])

      candidates_left = candidates.loc[candidates.loc[:, 'budget_left'] ==
                                    candidates.loc[:, 'budget_left'].max()]

      if len(candidates_left) > 1:
        hired = candidates_left.iloc[
          int(stats.randint(0, len(candidates_left)).rvs(size=1,
                                        random_state=self.rand_count))].name
      else:
        hired = candidates_left.index[0]
      return hired
```

Bibliography

[1] Mark Weiser, "The Computer of the 21st Century", Scientific American, September 1991

[2] IoT connected devices forecast, www.ericsson.com

[3] 5G vision brochure v1, 5g-ppp.eu

[4] Marco Ajmone Marsan, Giuseppe Bianchi, Nicola Blefari Melazzi, "Living and Fluid Networks: The way ahead?", Computer Communications, Volume 131, 2018, Pages 46-50, ISSN 0140-3664, https://doi.org/10.1016/j.comcom.2018.07.002.

[5] Worldwide connected devices forecast www.statista.com

[6] Semtech. LoRa Modulation Basics. AN1200.22, Revision 2, May 2015. www.semtech.com

[7] M. C. Bor, U. Roedig, T. Voigt, and J. M. Alonso, "Do LoRa Low-Power Wide-Area Networks Scale?", In Proc. of MSWiM 2016. ACM, New York.

[8] B. Reynders and S. Pollin, "Chirp spread spectrum as a modulation technique for long range communication", In SCVT 2016, Mons.

[9] B. Reynders, W. Meert and S. Pollin, "Range and coexistence analysis of long range unlicensed communication", In ICT 2016, Thessaloniki.

[10] D. Croce, M. Gucciardo, S. Mangione, G. Santaromita and I. Tinnirello, "Impact of LoRa Imperfect Orthogonality: Analysis of Link-Level Performance," in IEEE Communications Letters, vol. 22, no. 4, pp. 796-799, April 2018.

[11] M. Gucciardo, I. Tinnirello, and D. Garlisi. Demo: A Cell-level Traffic Generator for LoRa Networks. In Proc. of the 23rd Conference on Mobile Computing and Networking (MobiCom '17), October 2017, Snowbird, UT, USA.

[12] O. Bernard, A. Seller, N. Sornin, "Low power long range transmitter", European Patent Application EP 2763321 A1 by Semtech Corp., 2014

[13] C. Goursaud, J.M. Gorce, "Dedicated networks for IoT: PHY / MAC state of the art and challenges", in EAI endorsed transactions on Internet of Things, 2015.

[14] Available at http://www.lancaster.ac.uk/scc/sites/lora/

[15] L. Vangelista, A. Zanella, M. Zorzi, "Long-Range IoT Technologies: The Dawn of LoRa", in Future Access Enablers of Ubiquitous and Intelligent Infrastructures, pp. 51–58, Springer, 2015.

[16] A. Augustin, J. Yi, T. Clausen, W. Townsley, "A Study of LoRa: Long Range & Low Power Networks for the Internet of Things", in *Senors*, vol. 16, no. 9, p. 1466, 2016.

[17] M. Knight, B. Seeber. Decoding LoRa: Realizing a Modern LPWAN with SDR. Proceedings of the GNU Radio Conference, [S.l.], v. 1, n. 1, sep. 2016.

[18] D. Bankov, E. Khorov and A. Lyakhov, "On the Limits of LoRaWAN Channel Access", 2016 International Conference on Engineering and Telecommunication (EnT), Moscow, 2016, pp. 10-14.

[19] F. Cuomo, M. Campo, A. Caponi, G. Bianchi, G. Rossini, and P. Pisani, "Explora: Extending the performance of lora by suitable spreading factor allocations," in 2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Oct 2017, pp. 8.

[20] K. Q. Abdelfadeel, V. Cionca, and D. Pesch, "Fair adaptive data rate allocation and power control in lorawan," in 2018 IEEE 19th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM), June 2018, pp. 1415.

[21] B. Reynders, W. Meert, and S. Pollin, "Power and spreading factor control in low power wide area networks" in 2017 IEEE International Conference on Communications (ICC), May 2017, pp. 16.

[22] M. Zorzi and R. R. Rao. 2006. Capture and retransmission control in mobile radio. IEEE J.Sel. A. Commun. 12, 8 (September 2006), 1289-1298.

[23] D. J. Goodman and A. A. M. Saleh. The near/far effect in local ALOHA radio communications. In IEEE Trans. on Vehicular Technology, vol. 36, no. 1, pp. 19-27, Feb 1987.

[24] Semtech Corporation, "LoRa SX1272/73 Transceiver Datasheet", 2015.

[25] LoRaWAN Regional Parameters v1.1rB. Available at `https://lora-alliance.org/resource-hub/lorawanr-regional-parameters-v11rb`

[26] "Distributed MAC Protocol Design for Full-Duplex Cognitive Radio Networks," in 2015 IEEE Global Communications Conference (IEEE GLOBECOM 2015), San Diego, CA, USA, December, 2015.

[27] "Multi-Channel MAC Protocol for Full-Duplex Cognitive Radio Networks with Optimized Access Control and Load Balancing," in 2016 IEEE International Conference on Communications (IEEE ICC 2016), Kuala Lumpur, Malaysia, May, 2016.

[28] P. Ruckebusch, S. Giannoulis, D. Garlisi, P.Gallo, P. Gawlowicz, A. Zubow, M. Chwalisz, E. De Poorter, I. Moerman, I. Tinnirello, L. DaSilva "WiSHFUL: Enabling Coordination Solutions for Managing Heterogeneous Wireless Networks", in IEEE Communications Magazine, vol. 55, no. 9, pp. 118-125, Sept. 2017.

[29] P. Gallo, K. Kosek-Szott, S. Szott and I. Tinnirello "CADWAN: A Control Architecture for Dense WiFi Access Networks", in IEEE Communications Magazine, vol. 56, no. 1, pp. 194-201, Jan. 2018.

[30] A. Gudipati, D. Perry, L.E. Li, S. Katti "SoftRAN: Software defined radio access network", in Proc. of HotSDN '13, Honk Kong 2013.

[31] I. F. Akyildiz, P. Wang, SC Lin "SoftAir: A software defined networking architecture for 5G wireless systems", Computer Networks, 2015.

[32] H. Ali-Ahmad, C. Cicconetti, A. de la Oliva, M Draxler, R. Gupta, V. Mancuso, L. Roullet, V. Sciancalepore "CROWD: an SDN approach for DenseNets", in Proc. of EWSDN '13, pages 25-31. IEEE 2013.

[33] T. Chen, H. Zhang, X Chen, O. Tirkkonen "SoftMobile: Control Evolution for Future Heterogeneous Mobile Networks", Wireless Communications, IEEE, 21(6):70-78, 2014.

[34] X. Foukas, N. Nikaein, M. Kassem, M. Marina, K. Kontovasilis "FlexRAN: A flexible and programmable platform for software-defined radio access networks". In Proc. of CoNEXT '16. ACM, 2016.

[35] N. Nikaein, M. Marina, S. Manickam, A. Dawson, R. Knopp, C. Bonnet "OpenAirInterface: A flexible platform for 5G research", ACM SIGCOMM CCR, 44(5):33-38, 2014.

[36] S He, SHG Chan, "Wi-Fi fingerprint-based indoor positioning: Recent advances and comparisons", IEEE Commun. Surv. Tutorials. 18 (2016).

[37] Z Belhadi, L Fergani, "Fingerprinting methods for RFID tag indoor localization", in Proc. of ICMCS '14, Marrakech, Morocco, April 14-16. (IEEE, 2014), pp. 717-722.

[38] T Wigren, "LTE fingerprinting localization with altitude", in Proc VTC '12, Quebec City, QC, Canada, September 3-6. (IEEE, 2012), pp. 1-5.

[39] M Driusso, C Marshall, M. Sabathy, F. Knutti, H. Mathis, F. Babich "Indoor positioning using LTE signals" in Proc. of IPIN '16, Alcala de Henares, Spain. (IEEE, 2016), pp. 1-8.

[40] T Wigren, Y. Jading, I. Siomina, A. Kangas, C. Tidestav "Enhanced WCDMA fingerprinting localization using OTDOA positioning measurements from LTE", in Proc. of VTC '12, Quebec City, QC, Canada.

[41] M. Mukherjee, L. Shu and D. Wang, "Survey of Fog Computing: Fundamental, Network Applications, and Research Challenges" in IEEE Communications Surveys and Tutorials, vol. 20, no. 3, pp. 1826-1857, thirdquarter 2018. doi: 10.1109/COMST.2018.2814571

[42] P. Mach and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," in IEEE Communications Surveys and Tutorials, vol. 19, no. 3, pp. 1628-1656, thirdquarter 2017. doi: 10.1109/COMST.2017.2682318

[43] M. Satyanarayanan, P. Bahl, R. Caceres and N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing" in IEEE Pervasive Computing, vol. 8, no. 4, pp. 14-23, Oct.-Dec. 2009. doi: 10.1109/MPRV.2009.82

[44] S. Kosta, A. Aucinas, Pan Hui, R. Mortier and Xinwen Zhang, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," 2012 Proceedings IEEE INFOCOM, Orlando, FL, 2012, pp. 945-953. doi: 10.1109/INFCOM.2012.6195845

[45] L. Tong, Y. Li and W. Gao, "A hierarchical edge cloud architecture for mobile computing," IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications, San Francisco, CA, 2016, pp. 1-9. doi: 10.1109/INFOCOM.2016.7524340

[46] R. Deng, R. Lu, C. Lai, T. H. Luan and H. Liang, "Optimal Workload Allocation in Fog-Cloud Computing Toward Balanced Delay and Power Consumption," in IEEE Internet of Things Journal, vol. 3, no. 6, pp. 1171-1181, Dec. 2016. doi: 10.1109/JIOT.2016.2565516

[47] X. Chen and J. Zhang, “When D2D meets cloud: Hybrid mobile task offloadings in fog computing,” 2017 IEEE International Conference on Communications (ICC), Paris, 2017, pp. 1-6. doi: 10.1109/ICC.2017.7996590

[48] H. Shah-Mansouri, V. W. S. Wong and R. Schober, “Joint Optimal Pricing and Task Scheduling in Mobile Cloud Computing Systems,” in IEEE Transactions on Wireless Communications, vol. 16, no. 8, pp. 5218-5232, Aug. 2017. doi: 10.1109/TWC.2017.2707084

[49] M. Jia, J. Cao and W. Liang, “Optimal Cloudlet Placement and User to Cloudlet Allocation in Wireless Metropolitan Area Networks,” in IEEE Transactions on Cloud Computing, vol. 5, no. 4, pp. 725-737, 1 Oct.-Dec. 2017. doi: 10.1109/TCC.2015.2449834

[50] H. Shah-Mansouri and V. W. S. Wong, “Hierarchical Fog-Cloud Computing for IoT Systems: A Computation Offloading Game,” in IEEE Internet of Things Journal, vol. 5, no. 4, pp. 3246-3257, Aug. 2018. doi: 10.1109/JIOT.2018.2838022

[51] Q. Liu, S. Huang, J. Opadere and T. Han, “An Edge Network Orchestrator for Mobile Augmented Reality,” IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, Honolulu, HI, 2018, pp. 756-764. doi: 10.1109/INFOCOM.2018.8486241

[52] Z. Han, H. Tan, X. Li, S. H. -. Jiang, Y. Li and F. C. M. Lau, “OnDisc: Online Latency-Sensitive Job Dispatching and Scheduling in Heterogeneous Edge-Clouds,” in IEEE/ACM Transactions on Networking, vol. 27, no. 6, pp. 2472-2485, Dec. 2019. doi: 10.1109/TNET.2019.2953806

[53] J. Meng, H. Tan, C. Xu, W. Cao, L. Liu and B. Li, "Dedas: Online Task Dispatching and Scheduling with Bandwidth Constraint in Edge Computing," IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, Paris, France, 2019, pp. 2287-2295. doi: 10.1109/INFOCOM.2019.8737577

[54] Peter Mell, Tim Grance, et al. The nist definition of cloud computing, 2011.

[55] Sami Kekki, Walter Featherstone, Yonggang Fang, Pekka Kuure, Alice Li, Anurag Ranjan, Debashish Purkayastha, Feng Jianping, Danny Frydman, Gianluca Verin, Kuo-Wer Wen, Kwihoon Kim, Rohit Arora, Andy Odgers, Luis M. Contreras, Salvatore Scarpina “MEX in 5G networks”, ETSI White Paper no. 28, June 2018.

[56] Industrial Internet Consortium “OpenFog Reference Architecture for Fog Computing”, Available at `https://www.iiconsortium.org/pdf/OpenFog_Reference_Architecture_2_09_17.pdf`, September 2017.

[57] H. Wu, “Multi-Objective Decision-Making for Mobile Cloud Offloading: A Survey” in IEEE Access, vol. 6, pp. 3962-3976, 2018. doi: 10.1109/ACCESS.2018.2791504

[58] Stefano Traverso, Mohamed Ahmed, Michele Garetto, Paolo Giaccone, Emilio Leonardi, and Saverio Niccolini. 2013. “Temporal locality in today’s content caching: why it matters and how to model it”. SIGCOMM Comput. Commun. Rev. 43, 5 (November 2013), 5-12. DOI=http://dx.doi.org/10.1145/2541468.2541470

[59] Wenxiao Zhang, Bo Han, and Pan Hui. 2017. “On the Networking Challenges of Mobile Augmented Reality”. In Proceedings of the Workshop on Virtual Reality and Augmented Reality Network (VR/AR Network ’17). ACM, New York, NY, USA, 24-29. DOI: https://doi.org/10.1145/3097895.3097900

[60] E. Bastug, M. Bennis, M. Medard and M. Debbah, “Toward Interconnected Virtual Reality: Opportunities, Challenges, and Enablers” in IEEE Communications Magazine, vol. 55, no. 6, pp. 110-117, June 2017. doi: 10.1109/MCOM.2017.1601089

[61] Virtual Reality/Augmented Reality White Paper, www.huawei.com