Joint communication and computation resources allocation for cloud-empowered future wireless networks

Jessica Oueis

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Informatique**

Arrêté ministérial : 7 août 2006

Présentée par

Jessica Oueis

Thèse dirigée par **Andrzej Duda**
et codirigée par **Sergio Barbarossa**

préparée au sein **CEA -LETI/DSIS/STCS/LESC - et UMR 5217 - LIG - Laboratoire d'Informatique de Grenoble**
et de **MSTII**

Gestion conjointe de ressources de communication et de calcul pour les réseaux sans fils à base de cloud

Thèse soutenue publiquement le **12 Février 2016**,
devant le jury composé de :

M., Josep Vidal
UPC Barcelona, Rapporteur, Président
M., Kei Sakaguchi
Fraunhofer HHI, Tokyo Institute of Technology, Rapporteur
M., Valerio Frascolla
Intel, Examinateur
M., Andrzej Duda
Grenoble INP-Ensimag, Directeur de thèse
M., Sergio Barbarossa
Université de Rome, La Sapienza, Co-Directeur de thèse
M., Emilio Calvanese Strinati
CEA-LETI, Grenoble, Co-Encadrant de thèse

Abstract

Mobile Edge Cloud brings the cloud closer to mobile users by moving the cloud computational efforts from the internet to the mobile edge. We adopt a local mobile edge cloud computing architecture, where small cells are empowered with computational and storage capacities. Mobile users' offloaded computational tasks are executed at the cloud-enabled small cells. We propose the concept of small cells clustering for mobile edge computing, where small cells cooperate in order to execute offloaded computational tasks. A first contribution of this thesis is the design of a multi-parameter computation offloading decision algorithm, SM-POD. The proposed algorithm consists of a series of low complexity successive and nested classifications of computational tasks at the mobile side, which leads to an offloading decision to each of the tasks. The tasks are either computed locally using the handset resources, or offloaded to the cloud. To reach the offloading decision, SM-POD jointly considers computational tasks, handsets, and communication channel parameters. In the second part of this thesis, we tackle the problem of small cell clusters set up for mobile edge cloud computing for both single-user and multi-user cases. The clustering problem is formulated as an optimization that jointly optimizes the computational and communication resource allocation, and the computational load distribution on the small cells participating in the computation cluster. We propose a cluster sparsification strategy, where we trade cluster latency for higher system energy efficiency. In the multi-user case, the optimization problem is not convex. In order to compute a clustering solution, we propose a convex reformulation of the problem, and we prove that both problems are equivalent. With the goal of finding a lower complexity clustering solution, we propose two heuristic small cells clustering algorithms. The first algorithm is based on resource allocation on the serving small cells where tasks are received, as a first step. Then, in a second step, unserved tasks are sent to a small cell managing unit (SCM) that sets up computational clusters for the execution of these tasks. The main idea of this algorithm is task scheduling at both serving small cells, and SCM sides for higher resource allocation efficiency. The second proposed heuristic is an iterative approach in which serving small cells compute their desired clusters, without considering the presence of other users, and send their cluster parameters to the SCM. SCM then checks for excess of resource allocation at any of the network small cells. SCM reports any load excess to serving small cells that re-distribute this load on less loaded small cells. When no small cell is overloaded, the SCM validates the clusters set up accordingly. In the final part of this thesis, we propose the concept of computation caching for edge cloud computing. With the aim of reducing the edge cloud computing latency and energy consumption, we propose caching popular computational tasks for preventing their re-execution. Our contribution here is two-fold: first, we propose a caching algorithm that is based on requests popularity, computation size, required computational capacity, and small cells connectivity. This algorithm identifies requests that, if cached and downloaded instead of being re-computed, will increase the computation caching energy and latency savings. Second, we propose a method for setting up a search small cells cluster for finding a cached copy of the requests computation. The clustering policy exploits the relationship between tasks popularity and their probability of being cached, in order to identify

possible locations of the cached copy. The proposed method reduces the search cluster size while guaranteeing a minimum cache hit probability.

Keywords

Mobile cloud computing, Local cloud, Edge cloud, Small cells cluster, Resource allocation, Computation offloading, Computation caching, Load distribution

List of Publications

Journal Papers

[J1] J. Oueis, E. Calvanese Strinati, S. Sardellitti, and S. Barbarossa, "Joint Computation and Communication Resource Allocation in Edge Cloud Computing Clusters"*to be submitted, 2015.*

Conference Papers

[C1] Oueis, J.; Strinati, E.C.; Barbarossa, S., "Multi-parameter decision algorithm for mobile computation offloading," in Wireless Communications and Networking Conference (WCNC), 2014 IEEE , vol., no., pp.3005-3010, 6-9 April 2014

[C2] Oueis, J.; Calvanese-Strinati, E.; De Domenico, A.; Barbarossa, S., "On the impact of backhaul network on distributed cloud computing," in Wireless Communications and Networking Conference Workshops (WCNCW), 2014 IEEE , vol., no., pp.12-17, 6-9 April 2014

[C3] Oueis, J.; Strinati, E.C.; Barbarossa, S., "Small cell clustering for efficient distributed cloud computing," in Personal, Indoor, and Mobile Radio Communication (PIMRC), 2014 IEEE 25th Annual International Symposium on , vol., no., pp.1474-1479, 2-5 Sept. 2014

[C4] Oueis, J.; Strinati, E.C.; Barbarossa, S., "The Fog Balancing: Load Distribution for Small Cell Cloud Computing," in Vehicular Technology Conference (VTC Spring), 2015 IEEE 81st , vol., no., pp.1-6, 11-14 May 2015

[C5] Oueis, J.; Strinati, E.C.; Sardellitti, S.; Barbarossa, S., "Small Cell Clustering for Efficient Distributed Fog Computing: A Multi-user Case," in Vehicular Technology Conference (VTC Fall), 2015 IEEE 82nd , vol., no., pp., Sept. 2015

[C6] Oueis, J.; Strinati, E.C.; Barbarossa, S., "Distributed Mobile Cloud Computing: A Multi-user Clustering Solution," submitted to the International Conference on Communications (ICC), 2016 IEEE , 23-27 May, 2016

[C7] Oueis, J.; Strinati, E.C.; Barbarossa, S., "Energy Aware Computation caching on the Edge cloud," *to be submitted, 2015.*

[C8] Oueis, J.; Strinati, E.C.; Barbarossa, S., "Uplink Traffic in future mobile networks: pulling the alarm," submitted to CROWNCOM 2016.

Patents

[P1] J. Oueis, E. Calvanese Strinati, and A. De Domenico, “Method for Modifying The State of Local Access Points in a Cellular Network,” US20140220994, EP2763469 A1.

[P2] J. Oueis, E. Calvanese Strinati, and S. Barbarossa, “Method for Offloading the Execution of Computation Tasks of a Wireless Equipment,” DD14763SP.

[P3] J. Oueis, and E. Calvanese Strinati, “Method, Devices and System for Cloud Constitution With Signal Response Time,” DD15918SP.

[P4] J. Oueis, and E. Calvanese Strinati, “Method for distributed Local Cloud Cluster Formation with Multiple joint Communication and Computing UE requests,” DD16683SP.

[P5] J. Oueis, and E. Calvanese Strinati, “Caching and Fog Distributed Clustering Offloading,” DD16782SP.

Contents

List of Figures

List of Tables

Abbreviations and Acronyms

ACK	Acknowledgment
AMC	Adaptive Modulation and Coding
ANN	Artificial Neural Network
BB	Baseband
BBU	Baseband Unit
BER	Bit Error Rate
BFD	Best Fid Decreasing
BS	Base Station
CAC	Call Admission Control
CAPEX	Capital Expenditure
CoMP	Coordinated Multi-Point
CRAM	Cloud Resource Allocations for Mobile applications
C-RAN	Cloud (Centralized) Radio Access Network
CRE	Coverage Range Extension
CSI	Channel State Information
DHT	Distributed Hash Table
DL	Downlink
D2D	Device to Device
EDF	Earliest Deadline First
EDGE	Enhanced Data rates for GSM Evolution
EE	Energy Efficiency
EHF	Extremely High Frequency
EMF	Electro Magnetic Field
eICIC	enhanced Inter-Cell Interference Coordination
EPI	Energy Per Instruction
FD	Full Duplex
FFD	First Fit Decreasing
FFT	Fast Fourier Transform
FIFO	First In First Out
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Global System for Mobile communications
HD	Half Duplex
Het-Nets	Heterogeneous Networks
HSC	Helper Small Cell
HSPA	High Speed Packet Access
IaaS	Infrastructure as a Service

ICN	Information Centric Networking
ICP	Internet Cache Protocol
IFFT	Inverse Fast Fourier Transform
IoT	Internet of Things
IP	Internet Protocol
ISP	Internet Service Provider
LAN	Local Area Network
LRU	Least Recently Used
LSA	Licensed Shared Access
LTE	Long Term Evolution
LoS	Line of Sight
MAC	Media Access Control
MCC	Mobile Cloud Computing
MDP	Markov Decision Process
MEC	Mobile Edge Computing
MILP	Multiple Integer Linear Programming
MIMO	Multiple Input Multiple Output
MIPS	Million Instructions Per Second
mmW	millimeter Waves
MTC	Machine Type Communications
MUE	Mobile User Equipment
NACK	Negative Acknowledgment
NFN	Named Function Networking
NLoS	Non Line of Sight
OPEX	Operational Expenditure
OTA	Over The Air
PaaS	Platform as a Service
PER	Packet Error Rate
PF	Proportional Fairness
QoE	Quality of Experience
QoS	Quality of Service
RAN	Radio Access Network
RF	Radio Frequency
RRH	Radio Remote Head
RRM	Radio Resource Management
SaaS	Software as a Service
SAR	Specific Absorption Rate
SC	Small Cell
SCC	Small Cell Cloud
SCM	Small Cell Manager
SHF	Super High Frequency
SINR	Signal to Interference and Noise Ratio
SLA	Service Level Agreement
SSC	Serving Small Cell
SNR	Signal to Noise Ratio
SIR	Signal to Interference Ratio
TDD	Time Division Duplex
TTL	Time To Live

UL	Uplink
VM	Virtual Machine
WAN	Wide Area Network
WSN	Wireless Sensor Networks

Introduction and Thesis Outline

The imminent emergence of mobile cloud computing to wireless networks is today a reality. Mobile equipments are henceforth a platform that consumers are using to replace their desktop applications. Mobile equipment offers nowadays not only communication services, but also sensing, computing and storage. Mobile communicating devices are becoming ubiquitous, from smartphones in the hands of billions of persons, to wireless sensors, to connected things (vehicles, machines, cities, etc.). As a result, there is an increasing proliferation of various domains in the wireless communication such as medicine, health monitoring, business, banking, image processing, and home monitoring. It becomes more and more obvious that wireless devices need to have greater capabilities in terms of communication resources, computational capacities, storage space, and most importantly great autonomy. As new functionalities have been integrated in mobile connected devices over the last couple of decades, devices have grown in features, capacities, capabilities, and sometimes in size. However, battery technology has not been able to cope with the development of mobile devices and their increasing demand of energy. Indeed, sensing, computing and communicating through mobile equipment increase the energy consumption and thus decrease the devices battery lifetime. Mobile Cloud Computing is one of the most powerful solutions for allowing mobile devices to do more while consuming less. Mobile cloud computing is an attempt, that is proving success, at extracting the functionalities that have been added to mobile equipment without depriving mobile equipment users from accessing the offered service. In other terms, mobile devices will always have the result of a certain computation without necessarily calculating it on their own. Mobile cloud computing is a 'mobile delivery service' that is offered to mobile devices at a lower cost. The base of a mobile cloud computing service is computation offloading from mobile devices to the available cloud.

In addition to empowering mobile users' devices, in the IoT paradigm, computation offloading is also important to empower simple devices, like e.g. tiny sensors, with computational capacities that they could have otherwise. In addition, moving computations from base stations to the cloud is a form of computation offloading that is the central issue of centralized-Radio Access Network (RAN), which can be efficiently implemented as Cloud-RAN. In this case, computation refers to the various blocks in the radio chain, such as de-multiplexing, decoding, etc. In this particular case, computation offloading is primarily for reducing the computational load of base stations, rather than mobile devices.

The focus in this thesis is put on the mobile edge cloud architecture where cloud functionalities are driven to proximity of mobile users. Mobile edge cloud could take place in small cells base stations, in connected routers, in network switches, or in any connected entity that can be equipped with computational and storage capabilities. First, aiming at minimizing mobile equipment power consumption, but also, and more importantly, guaranteeing a good quality of experience, we shall examine computation offloading decision algorithm design for mobile cloud computing. Indeed, one of the most important ingredients of the success of mobile cloud computing process is to have the right offloading strategy.

Second, we shall look into the computation services on the edge cloud. We consider edge cloud clustering possibility where several computing entities participate in the computation of users' tasks by forming a computation cluster. Therefore, intra-cluster resource management should be well orchestrated in order to achieve high quality of experience. To this end, it should be interesting to combine communication and computation resource allocation inside every computation cluster. Computation load distribution and/or balancing, communication resources allocation, and computational capacities association at each of the cluster participants should be studied for a reliable computational cluster performance and high service quality delivery. Computational tasks are mostly subject to latency constraints and computation capacity requirements. In a cluster-based computation on edge cloud, overall perceived service latency incorporates transmission delays and

computation time. Transmission delays result from intra-cluster communication where computing entities exchange computational data and results. Respecting latency constraints, couples the allocation of communication and computation resources, as well as load distribution. Finally, we shall investigate possible mechanisms and novel paradigms that could be considered for reducing computational clusters power consumption. Since mobile edge cloud infrastructure is distributed, scalable, adaptable, and mostly based on user deployed network nodes, power consumption is an important issue in this kind of architectures. Reducing the edge cloud power consumption makes mobile edge cloud computing an interesting solution, especially in the green networking framework.

The challenge of this thesis is to investigate how we can jointly optimize communication and computation resources in a mobile edge cloud cluster while improving the delivered quality of service and guaranteeing mobile users' requests satisfaction. The outline of the thesis is as follows:

- In Chapter 1 we discuss the proliferation of a new services and applications ecosystem in the mobile networks. We show how future mobile cellular networks are based on new technical breakthrough that will allow the mobile network to maintain the capability of serving the increasing number of mobile users and their generated traffic. We present a quick review on future 5G networks and their requirements set to be in line with the network traffic explosion. 5G networks are required to support massive devices connectivity but also massive system capacity. Higher data rates are at the top of the requirements list along with very low latency, which will enable more real time services and applications. To achieve the set of 5G requirements, a number of enabling technologies are being widely discussed. A main enabler technology is massive MIMO, which has proved to achieve higher data rates through the equipment of communicating devices and nodes with multiple antennas. We also discuss the use of wider spectrum and the introduction of millimeter waves and full duplex radio. Furthermore, we focus on the emergence of ultra-dense heterogeneous networks, and the integration of cloud services. These two technologies are the key enablers for the mobile edge cloud architecture. Then, we zoom on cloud technologies evolution in mobile networks, and show how the cloud integrated wireless communication systems through evolving and various architectures. We start by describing the Cloud-RAN architecture as one of the first cloud based networks, and we continue with mobile cloud computing platforms like cloudlet based mobile cloud, edge cloud, and fog. We show the advantages and characteristics of each of the architecture in terms of availability, reliability, experienced latency, mobility support and proximity to users. The final part of the chapter discusses the upcoming uplink traffic explosion, which is mainly caused by the increase of mobile cloud computing traffic. We explain why it is just a matter of time before an uplink traffic explosion hits the mobile network. Mobile networks which have always been designed and scaled according to downlink traffic requirements, have always accorded lower attention, and lower capacities to the uplink direction. While this was supported by justified reasons, the traffic patterns are now changing, and uplink is not less important than downlink anymore. In this chapter we discuss several factors that contribute in the uplink data traffic increase such as the growing number of mobile subscribers, the number of connected devices, and the evolution of mobile networks. The emergence of cloud technologies is one of the main factors of the uplink traffic increase, not to forget the convergence of IT and wireless communication systems, and the integration of the Internet of things, wireless sensor networks, and machine to machine communication to the mobile networks. Furthermore, this chapter presents a set of trade-offs that are faced in the mobile edge computing framework. A part of the material of this chapter is reported in conference paper [C8].

- Chapter 2 describes the adopted mobile cloud computing architecture in this thesis. We present the advantages and characteristics of the adopted architecture, while pointing out the assumptions that are made within this thesis. We discuss the limitations, optimization parameters, and objectives that we tackle in this thesis. We explain why a joint communication and computation resource allocation, and load distribution is needed in the edge cloud computing architecture based on small cells computing clusters. A study over the existing trade-offs in the adopted architecture is presented. The proposed architecture is based on heterogeneous cellular networks. Therefore, we give a preliminary overview on the communication-related trade-offs of heterogeneous networks. We extend the study to incorporate computation-related parameters, brought by the edge cloud integration of mobile networks. We discuss a series of trade-offs showing that the edge cloud shifts the heterogeneous networks optimization paradigm to a whole new level by adding additional resources to optimize. Energy efficiency from both mobile and system perspective, computation and communication resource allocation efficiency, users quality of experience, and network deployment efficiency are some of the trade-offs actuators that are studied in this chapter. In a cluster environment, an important bottleneck of future wireless networks is backhaul. In the last part of this chapter we present a comparative study on the impact of backhaul technology and topology on small cell clusters, in terms of latency and power consumption. We compare the full mesh, tree, and ring topologies for fiber, microwave, and wireless LTE technologies. Wireless backhaul use is important to complement the standard wired backhaul, especially in dense small cell networks.

  A part of the material of this chapter is reported in conference paper [C2].

- In Chapter 3, we review some of the offloading decision mechanisms and algorithms presented in literature for the context of mobile cloud computing. We show that the majority of these mechanisms are based on the energy trade-off between local computation on mobile handsets, and computation offloading to the cloud. While mobile handsets energy consumption is an important parameter to consider for deciding on computational tasks offloading, it is not the only parameter that affects the decision. Users' quality of service should be the main parameter that contributes to the decision process. The trade-off of energy consumption and quality of service should then be studied and optimized in order to enable mobile handsets to do more while consuming less. Furthermore, computational tasks are mostly constrained by latency limits and memory requirements. These constraints impose a minimum computational capacity that should be allocated to the tasks. Few are the existing offloading decision strategies that consider the totality of these parameters. Most of the existing algorithms are based on mathematical optimization, multi-criteria utility functions, and solved using linear programming, integer (or multi-integer) linear programming. Hence, the complexity of these algorithms increase with the number of considered parameters.

  The material of this chapter is reported in conference paper [C1]

- A mobile user may decide to run its applications locally, if energy and time are not an issue, or at the nearest cloud-enhanced fixed device, or in a cluster of federated devices, depending on energy consumption and latency constraints. In Chapter 4 we focus on edge cloud clustering for computational purposes. We propose and study optimization formulations of computation cells clustering. Our models are based on joint optimization of communication and computation resources inside the computation clusters. Each cluster is responsible for fulfilling a computational request. The optimized solution identifies the cells to include in the cluster. Furthermore, it distributes the computation load on the participating cells.

Indeed, each of the cluster participants computes a part of the computational load, with a computational capacity that is also defined by the derived solution. As computational tasks are received at first at one cell, the load is distributed, and necessary information is transmitted to helping cells in the clusters. The transmit power used for sending the necessary data between cluster participants is the third degree of freedom defined by the cluster optimization solution. The three parameters (transmit power, load distribution, and computational capacity allocation) define the total edge cloud computing process latency. The total perceived latency includes both transmission and computation delay. The optimization solution is constrained to always respect the latency constraints imposed by the computational tasks.

We start by studying the single user case where only one task has to be computed. We propose four different cluster optimization strategies. The first has the objective of minimizing the cluster perceived latency. The second is based on the latency vs power consumption trade-off, and searches to sparsify the cluster by excluding some of the participating cell and increasing the load of others. The two remaining strategies aim at reducing the cluster power consumption from a whole cluster and single participant point of view, respectively. Then, in the second part of the chapter, we evolve the considered scenario to a multi-user situation where several clusters are to be formed at once, using the same pool of resources. The optimization problem that is presented has an objective of minimizing the overall clusters power consumption, while guaranteeing users' quality of service by respecting the delay constraints imposed by each task. As the formulated problem is not convex, we propose an equivalent convex problem that leads to efficient implementation of the solution. We evaluate all of the proposed clustering solutions in an indoor small cells deployment, where mobile edge computing is offered by the set of active small cells. We compare the proposed solutions, and show their effectiveness in terms of users' satisfaction ratio. Furthermore, we evaluate their energy consumption and perceived latency.

The material of this chapter is reported in part in journal paper [J1] and conference papers [C3] and [C5].

- In Chapter 5, we propose clustering algorithms that are characterized by lower complexity than the optimization algorithm studied in the previous chapter. Heuristics are proposed in order to keep the quality of service delivered by edge cloud computing while reducing the process complexity. The chapter contribution is based on the idea of exploiting the low complexity optimization of computation cluster for the single user case. Both algorithms that are proposed rely on the single user selfish cluster optimization.

  A first algorithm consists of scheduling computational tasks at the serving cell where they are received. The serving cell is the cell that receives the computational requests from mobile users. The scheduling policies that can be adopted are various, such as earliest deadline first, and proportional fair to name a few. Computational resources are allocated at the serving cell. Unserved requests are then sent to a centralized entity that re-schedules the tasks sent from all serving cells. Unserved requests are then computed, while respecting the task scheduling, by computational clusters using the single user optimization. Three different implementations of this algorithm are proposed. They differ in the scheduling metrics and the clustering objectives.

  The second algorithm we propose is an iterative algorithm. It is based on operations at the serving cells and a centralized cluster manager, respectively. As a first step, each serving cell selfishly forms the clusters for each of the received requests. The selfish clustering is done by serving cells without considering the presence of other users. As a second step,

clusters report to a centralized entity that controls the feasibility of the resource allocation from all serving cells. Since the same resources can be allocated by several serving cells, the centralized control management validates or corrects the requested clusters according to resources availability. Excess of allocated resources is identified, and feedback is sent to serving cells in order to notify them on remaining requests to be computed, and relative correction values.

We benchmark the proposed algorithms by simulating an indoor mobile edge cloud environment. We evaluate the performance of both algorithms and compare them in terms of requests satisfaction ratio and cluster power consumption.

The material of this chapter is reported in part in patent [P4] and conference papers [C4] and [C6].

- Finally, in Chapter 6, we propose to exploit, in addition to computation capacity, the storage capacity available at the Edge cloud. We introduce the novel concept of *computation caching*. Cache memories at Edge small cells are used for caching users' computations. When a cached computation is requested, it is not computed. The serving small cell retrieves the request from the cache memory. Our proposal is based on exploiting the knowledge of computational requests popularity at each small cell, to choose the requests to be cached. We consider, in this chapter, the same architecture of the small cell cloud clusters. However, we introduce the approach of small cell clustering for cache search. A search cluster is thus defined as the set of small cells whose cache memories are searched for finding the requested computation.

  In the first part of the chapter, we propose a caching algorithm. Caching algorithms define the computations requested to be saved in the cache memories at each small cell. The proposed caching algorithm takes into account, not only the computation request popularity, but also the amount of required computational capacity for its execution, the imposed latency constraints, and the small cell connectivity in the network. The algorithm caching policy aims at increasing the *computation caching* gains in terms of both latency, and power consumption. We compare the performance of the proposed algorithm with the state-of-the-art, and show that even though less computational requests are cached, higher performance gains can be achieved, especially when the storage space at the Edge cloud is limited.

  After the computational requests are cached at the Edge of the network, small cells should be able to locate the cached copies of each request. In the second part of this chapter, we propose a method that allows reducing the set of small cell to be searched for retrieving a computational request. The proposed method exploits the established relationship between requests popularities and their caching probability. This relationship depends on the adopted caching policy. Knowing the popularity of each request in the network small cells, the most *probable* caching location of each request can be identified. Then, according to the small cells connectivity in the network, the cache location that are *reachable* are identified. The proposed method identifies the small cells that have a copy of the requested computation, which can be retrieved while respecting the latency constraints of the application.

  The material of this chapter is reported in part in patent [P5] and conference paper [C7].

Chapter 1

The Evolution of Cloud Enable Mobile Wireless Networks

1.1 Motivation

The famous 'dramatic increase of mobile data traffic' is already happening! Mobile communication is nowadays an essential component in millions of people's lives and daily activities around the globe. The number of mobile subscribers, number of mobile applications, and thus, mobile data traffic are subject to an exponential increase. Indeed, traffic forecasts foresee further data traffic abundance in the coming years [3]. We are currently witnessing a new revolution of the Internet where people and smart objects are connected together in smart environments. The Internet world is converging with cellular mobile communication networks to form a complete services package for mobile users. Data services have been widely incorporated in cellular networks. For instance, 3G and 4G networks are both Internet Protocol (IP) based. Unlike 3G, 4G uses IP even for voice communication data. The interaction and inter-connection between people, devices, sensors, and service providers have opened the door for an eruption of mobile over-the-web applications. New innovative applications are released at a daily basis covering wide areas of communication purposes, going from entertainment and social networking, to industrial and health-care applications. These applications are accessible from mobile devices connected to the Internet through cellular mobile network. The diversity of services that are henceforth available is as large as the diversity of life essentials. Indeed, we are witnessing a rapid creation of an e-life where different essential utilities, favorite interests, and daily life requirements can be controlled, checked or accomplished through a mobile device, throughout a large variety of applications. The merge of IT and communication networks is downright a reality bringing services closer and making them more accessible to mobile users. As a consequence, connectivity is becoming an essential life component rather than a privilege. The enormous amount of web-based applications launched through mobile networks generates a non-negligible amount of additional data traffic, as well as computational requirements in terms of both computation capacity and storage space. Therefore, ubiquitous connection and resources accessibility are essential requirements of cellular networks. Mobile networks should be able to deliver high quality services with high speed connection, assure resources (computing, storage, sensing) availability and reachability by mobile subscribers. On the other hand, mobile users expect to enjoy high end services with a minimal cost especially in devices battery consumption and in latency delay. As a matter of fact, two of the most important marketing keys for mobile devices are autonomy and long battery lifetime. The challenge to overcome is allowing mobile devices to achieve more with the same, nay much lower costs in terms of energy consumption, experience quality degradation, and latency delay. As mobile users will continue to generate increasing amount of data for communication, but also, and more importantly, for computational services, mobile networks face the challenge of delivering high service quality despite the traffic increase. Through the different generations of mobile networks, the architecture has been and is still evolving, and new mechanisms are still being proposed in order to improve service quality and to efficiently integrate computing services. Two essential requirements for overcoming the big data/large computations challenge are the following: (i) a breakthrough in cellular networks functionalities, capacities, and capabilities in order to cope with the ongoing, and upcoming, *tsunami* of data and the ever-increasing number of connected devices. (ii) Find an adequate location, both geographical and in network, for implementing powerful computing solutions. The computational burden in mobile networks is composed of network required signal processing and solution computing for various Radio Resource Management (RRM) mechanisms, and of the users launched computation request through mobile-friendly applications. In the remainder of this chapter, we discuss the future generation of mobile networks, 5G, and show how it can be seen as the next breakthrough for mobile communications to overcome the quickly evolving market. Then, we give a panoramic view of the evolution of network architecture with

a focus on cloud technologies integration within. This overview shows the effort done for mobile networks to incorporate, increase and grant accessibility of computational capacities by means of various solutions.

With the incorporation of cloud services in wireless networks, an important change in traffic patterns takes place: the increase of uplink traffic. While wireless networks have always been designed based on downlink traffic, that represents the largest share of mobile traffic, current networks may not be able to cope with the increase in uplink traffic. Uplink to downlink traffic ratio depends on the application type. However, cloud services promote applications and services that are based on high uplink traffic ratio, such as, computation offloading and online photos and videos storage. In the last part of this chapter, we point out the upcoming traffic changes, the contributing factors, and emerging solutions.

1.2 5G Cellular: The Future of Mobile Networks

Mobile and wireless traffic volume is continuously and rapidly increasing. It is foreseen that the mobile data traffic will grow 1,000 times higher from 2010 to 2020 with a rate that is roughly the double per year [13]. This traffic increase is due to many factors. Mobile devices nowadays are a platform for accessing various types of services requiring high volumes of data. The increase in the number of mobile subscribers, and thus the number of wireless connected devices (e.g. smartphones, tablets) generates additional traffic that approaches the limits of what current cellular networks could deliver. Indeed, the number of mobile-connected devices exceeded the world's population in 2014. Moreover, the forecasts on the number of connected devices by 2019 state that there will be 1.5 mobile devices per capita, i.e. the number of mobile-connected devices will exceed the number of people on earth in 2019 [14]. Furthermore, the scopes of the services that are made available to users through mobile networks are expanding. Some of these services are linked to essential aspects of people's everyday life such as e-banking, e-health, and e-learning, and will continue to be further adapted to mobile environments [15]. Others are on-demand services launched directly by mobile users or through mobile applications. These kinds of services are evolving every day and including more and more complex and data hungry applications such as face recognition, online gaming, augmented reality, instant translation, and video decoding. The expansion of applications in number and diversity leads to an explosive increase of data usage. The interactive side of some of these applications requires a network that supports lower latencies and higher data rates. As the worlds of IT and telecommunication networking are converging, the exploding Internet data traffic has driven towards a new 5th generation of cellular wireless networks. Moreover, various market drivers have pushed towards the conception of this new generation of wireless communication. The Internet of Things (IoT) and its wide range of applications are quickly proliferating in the wireless communication services. IoT is expected to further expand in the future and to impose billions of devices deployment which need to efficiently rely on highly scalable network architecture. In addition of scalability, the network should deliver a high level of reliability since the IoT applications can be linked to highly sensitive domains as tele-medicine for example. Furthermore, with IoT applications such as smart grids and infrastructure monitoring, the latency of the network can be higher than the maximum time delay required by this type of applications. 5G linked applications cover a wide variety of domains such as social networking, automotive, sports, smart cities, industrial monitoring, home security, and health care. Figure 1.1 shows different types of 5G applications and how they differ in latency and bandwidth requirements. 5G networks have been the subject of extensive research in the last years. As the next step in the evolution of mobile communication, 5G capabilities must extend far

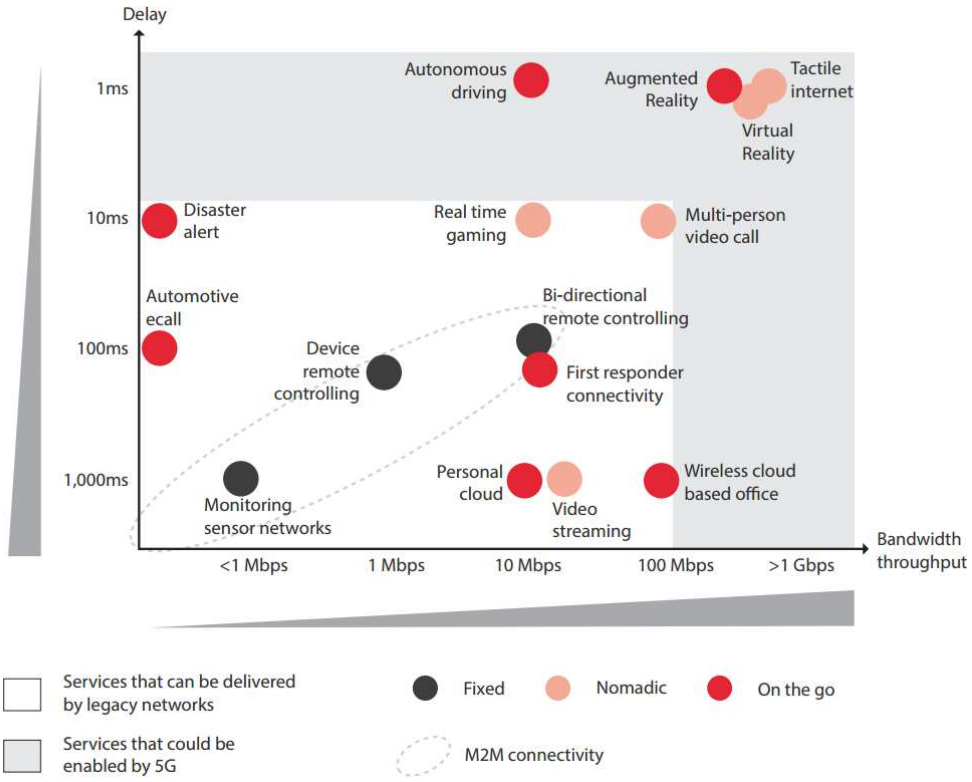


Figure 1.1: Bandwidth and latency requirements for potential 5G applications [1].

beyond previous generations to meet increasing requirements. Despite the fact that 5G is not fully and uniquely defined yet, its technical requirements have been set using an explicit formulation, by one of the first large-scale projects on 5G, the European Project METIS (Mobile and Wireless Communications Enablers for the 2020 Information Society) [16], as follows [17]:

- 1000 times higher mobile data volume per area
- 10 to 100 times higher typical user data rate
- 10 to 100 times higher number of connected devices
- 10 times longer battery life for low power devices
- 5 times reduced end-to-end latency

1.2.1 5G: Design Essentials

The new 5th generation of mobile networks design has to allow the fulfillment of all 5G requirements. Key design principles are needed to guide all requirements and technical solutions. Two main characteristics of 5G networks design are defined by Nokia: flexibility and reliability [18].

- **Flexibility :** As stated earlier, 5G use cases and application cover a very wide area of domains. As seen in Figure 1.1 it can go from use cases that require a data rate as low as

1 Mbps and a latency of 1 second, to others that require data rate in the order of Gbits per seconds and a latency in the order of milliseconds. This gives an example of how the size of the packet to transmit through the network extremely varies depending on the use case in question. Consequently, the quantity of resources that need to be allocated and the urgency of completing the packets transmission are not the same. 5G communication system need to be flexible enough to guarantee a full adaptation to various traffic speeds in both uplink and downlink. This needs to be done without increasing the complexity of the network management nor affecting the mobile users' quality of experience.

- **Reliability :** Reliability is another key design principle for 5G networks that guarantees a quality of experience beyond the best effort and towards reliable communication. Reliability is a very important key principle of 5G networks design since 5G will be the platform of various QoE critical use cases (e.g. e-health, tele-medicine, monitoring). 5G should incorporate new technologies, protocols, designs, and network layers in order to guarantee sufficient reliability for mobile users in all types of applications.

1.2.2 5G Requirements and Enabling Technologies

1.2.2.1 5G Requirements and Capabilities

Even though the 4th generation of cellular networks brought new advances in both design and evolution, the market trend and the expanding connectivity that reached both people's devices and smart objects are imposing new breakthrough in cellular communications. 5G is facing challenging targets that, if met, will allow the network to absorb the ever-increasing data traffic explosion and services high requirements. Many industries, institutions, and research centers have presented a view over the future 5G networks. Nokia [18], Ericsson [19], Huawei [20], GSMA intelligence [1], NTTDOCOMO [21], and others have published white papers in which each company presents its vision of 5G. These 5G requirements, as described by all of these works, converge to the same set of challenges which are the following:

Massive system capacity

The number of mobile devices expected to be connected to the 5G network is in the order of billions. Traffic volumes are expected to be larger by many orders of magnitude that could reach at least 1,000 fold higher capacity demand. The required capacity of the network in order to handle such an extremely high number of connections, including both signaling and data traffic volume, provides a serious challenge. This is considered as the most challenging requirement for 5G networks. Some targets have been set by NTTDOCOMO to achieve a 1000-fold system capacity per km^2 compared to LTE [21]. 5G networks should guarantee a high traffic handling capacity while maintaining mobile users Qos and QoE.

Higher data rates

As the next evolution of cellular networks, 5G should be able to offer, as a minimal requirement, higher data rates compared to its predecessors. However, the focus for previous generations has been on peak data rates instead of individual data rates in all possible real life scenarios. With the proliferation of new services and applications that can be launched by mobile users anytime and anywhere, peak data rates are less significant. Focus should now be accorded to real-life scenarios and the data rate that can be offered to users whenever needed. Various scenarios with different

data rate targets should be studied. According to [19], indoor and dense outdoor environments scenarios should guarantee a data rate for mobile users which is as high as 10 Gbps. For urban and suburban scenarios, Ericsson set a target in the order of 100 Mbps. Furthermore, a minimum data rate of 10 Mbps should be seamlessly and ubiquitously guaranteed at any location even in sparsely-populated areas. NTT-DOCOMO also focused on the target of achieving uniform experience for mobile users. Their target has been set for a 1 Gbps of peak user throughput everywhere.

Massive availability, connectivity, and reliability

5G networks should be able to support all kind of use cases that will be integrated into the cellular network. From cloud services support to IoT devices, all components should *always* find their way into being connected to the network. 5G networks should have massive connectivity to embrace the increasing number of simultaneously connected devices. It should also enable high reliability and availability especially in use cases that handle critical situations or crisis management. In other use cases such as cloud services, the network should be available whenever on-demand resources are required from the users' side. Additional key requirement for 5G networks is robustness. A robust and reliable network is required for guaranteeing data, users, and infrastructure security. Finally, 5G networks will be the essential platform of mission-critical management and monitoring applications such as public safety, water and gas distribution, and home security. This further amplifies the need of a future network with an ultra-high availability and reliability.

Very low latency

5G will gather heterogeneous use cases with requirements that are very different in terms of both required capacity and latency constraints. Some of the applications to be implemented over 5G networks require very low time delay, in the order of milliseconds. Autonomous driving, 3D gaming, and augmented reality are very good examples of this type of applications. Targets have been set to an overall latency in the order of 1 ms, a reduction of $5\times$ to $10\times$ in latency compared to previous generations.

Reduced cost and higher energy Efficiency (EE)

Energy consumption in mobile networks can be seen from two distinct perspectives. A first perspective is related to energy consumption on the network side. One way to reduce the overall network energy consumption is to increase spectral efficiency. However, due to the exploding traffic volumes, attention should also be accorded to the energy consumption per bit (Joules/bit) that has a direct effect on the network energy efficiency. Moreover, network energy performance is a very important component for reducing operational costs [19]. In another perspective, the energy consumption on the wireless devices side should also be considered. Very low energy consumption for wireless devices has always been a well sought requirement. With the integration of IoT that could include various types of sensors, a long battery lifetime is a must. Targets have been set for batteries lifetime of around a decade. Therefore, 5G devices should be able to operate on a very low energy consumption conducted by both adequate hardware design and high energy efficient communication protocols and techniques.

1.2.2.2 5G Enabling Technologies

Based on the key design principles described in 1.2.1 and on the expected capabilities of 5G networks, several driver technologies have been defined. These technologies will help cope with the challenges imposed by the data increase. They will help increase capacity, improve energy efficiency, and reduce spectrum utilization. And in light of the key design features, they will allow better and easier scalability, and help increasing the network reliability.

Massive MIMO

Massive MIMO (Multiple Input Multiple Output) is the evolution of single point-to-point MIMO and Multiuser MIMO (MU-MIMO). In MU-MIMO a set of base stations equipped with more than one antenna (less than 10) serves a set of users, each of which has a single antenna. Massive MIMO is the result of an effort made to expand the MU-MIMO vision into a large scale antenna system where each base station is equipped with approximately 100 or more antennas. Figure 1.2 shows a snapshot of what a massive MIMO system could look like. The concept was proposed by Marzetta

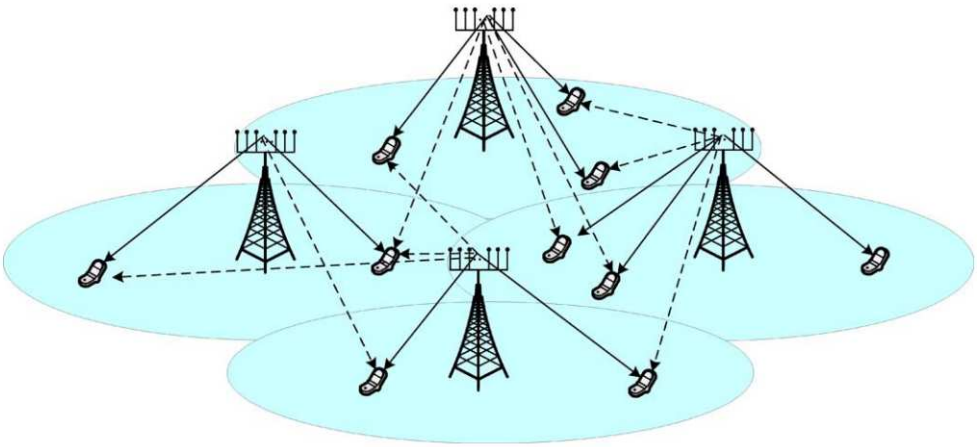


Figure 1.2: Illustration of Massive MU-MIMO systems [2].

[22] [23]. His works showed that as the number of antennas in a MIMO cell grows to infinity, small-scale fading effect is eliminated, and the required energy per bit for transmission is nulled. When a very large number of antennas is available, serving a significantly smaller number of users, then many degrees of freedom are available. These contribute in better shaping the signals to be hardware-friendly, or reducing interference [2]. The extra available antennas allow the system to achieve higher throughput and improve radiated energy efficiency. Indeed, with a large number of antennas, the energy can be concentrated in smaller regions of space, thus energy efficiency is increased by using beam-forming like techniques [24]. Moreover, higher capacity, in the order of 10-fold increase, can be achieved by Massive MIMO systems. This is due to the fact that massive MIMO rely on spatial multiplexing. Interference between terminals in such systems is already lowered thanks to the extreme sharpness of destination- focused energy. The spatial multiplexing is based on channel quality knowledge on the base station side of both uplink and downlink. To

obtain this information relying on reciprocity of the channel, a TDD approach is adopted. In fact, there appears to be a general agreement that the propagation channel is reciprocal a priori. Massive MIMO relies on the law of large numbers and beam-forming which allow to (i) avoid fading clips and thus improve overall transmission latency; (ii) eliminate the effect of frequency domain scheduling and therefore each terminal is granted the whole bandwidth, which simplifies the multiple access layer. Finally, massive MIMO systems can be built with non-expensive and low power components. With all the benefits that could be added by massive MIMO, it does not come challenge-free. One of the most important challenges of the massive MIMO systems is referred to as "pilot contamination". Pilot contamination is caused by the reuse of uplink pilots (used in the TDD channel estimation) from one cell to another. The pilots should be orthogonal for all users. However, the number of existing orthogonal pilots is limited. Therefore pilots are repeated and thus channel estimation for a single user is *contaminated* by a linear combination of the channel of other users using the same pilot.

Device-to-Device (D2D) communication systems

Device-to-Device communication is the direct communication between two end equipments without going through the network infrastructure. In cellular networks, it serves as an additional tier that helps improve network capacity. Network nodes collaborate in relaying information to benefit from spatial diversity advantages. D2D is based on the proximity of end users. Devices can communicate together using either the same spectrum as macro-cells, or unlicensed spectrum. With the recent trends in the wireless market and the introduction of new services that require location and context information, communication between neighbor devices can be useful. When several devices or clusters of devices are acting as relays for each other, they form a massive ad-hoc mesh network. This can serve for offloading traffic into the D2D tier of the network and thus increase per area capacity. As an example of such scenarios, we can think of crowded areas, such as malls or stadiums, where a high number of devices are operating at a close distance. Furthermore, D2D helps reduce end-to-end delay and power consumption. For example, in cell edge scenarios, mobile users require a higher power consumption. However if mobile users communicate with neighbor devices at a proximity acting as relays, power consumption is reduced and end-to-end delay is lowered. D2D communication also plays role in different levels of the network. D2D can have a major role in mobile cloud services, especially in what is linked to mobile cloud computing. Devices at proximity of each other create pools of resources that could be shared among users for a better quality of experience for mobile users. There are four different types of D2D communication as envisioned in [25]:

- **Device relaying with operator controlled link establishment (DR-OC)** where mobile devices act as relays for each other. Link establishment is fully or partially controlled by nodes of the operator's network by communicating with the relaying device. The cell edge scenario is a good use case for using this type of D2D communication.

- **Direct D2D communication with operator controlled link establishment (DC-OC)** where mobile devices do not act as relays, but communicate through direct links. Data transfer and information exchange happen without the assistance of the operator's network, however, it centrally handles links establishment between devices. A centralized links formation policy helps control interference that results from D2D communication using the same spectrum as other core network linked base stations.

- **Device relaying with device controlled link establishment (DR-DC)** Devices act as relays

for each other, but operator's network does not play any role in link establishment. It is for the devices (source and relay) to coordinate relay communication links between each other.

- **Direct D2D communication with device controlled link establishment (DC-DC)** Devices are responsible for creating communication links between source and destination. Devices do not act as relays to operator's base stations; they act as source and destination nodes that directly exchange information through the self-established link. This type of D2D communication is the most exposed to interference and collision problems.

To resume, D2D serves to increase coverage, guarantee connectivity when the link with infrastructure is weak or brakes, increases capacity per area and spectrum utilization, and reduces end-to-end latency and power consumption. Nevertheless, D2D carries some challenges. Security and privacy issues are probably the most constraining in D2D communications. One challenge is to protect user data against any potential attacks. Two devices access modes are then defined. *Closed access* is a mode where mobile devices only communicate with a set of authorized devices that are in their trusted devices list. *Open access* is the communication type where no restrictions are made. All devices can communicate through D2D links with a device configured as open access. Another significant challenge for D2D communication is interference management. When links are established with the help of the operator network nodes, it is easier to control interference since management is centralized. However, in the device controlled link establishment use cases, interference is not centrally managed. Two types of interference can be identified. The first is between the two tiers of the operator network and D2D communications. If both tiers are communicating using the same licensed bands, D2D communicating devices will affect devices communicating with the network base stations. To reduce the impact of inter-tier interference on the existing operator base stations, smart interference management and adequate resource allocations strategies should be designed for the two-tier network. Then, there is the interference between several devices in the D2D tier. Using D2D in the same band by neighbor devices results in possible collisions and interference. This can be addressed through designing smart resource allocation, admission control techniques, and peer discovery protocols.

Ultra-dense heterogeneous networks (Het-Nets)

5G will witness a huge expansion in user base and an in diversity of technologies operating in different bands. From these trends arises the necessity of deploying shorter links to connect mobile users, and the necessity to increase connectivity as well. Network densification is a key mechanism for 5G and for mobile wireless evolution in general. It allows the network to meet the requirements of very high capacity, connectivity and availability. For achieving ultra-dense networks, heterogeneity of network nodes will play an important role [26]. Heterogeneous networks introduce a sort of a dynamic aspect to the cellular network, especially with the introduction of moving networks and ad-hoc networks. With the presence of low transmit power devices that are randomly and densely deployed, the number of network nodes is increasing, and the network itself is getting ultra-dense. This approach will improve spectral efficiency since it reduces the distance between base stations and mobile users [27]. According to a work by Bhusnan *et al.*, which network densification is the main focus area, networks densification is a combination of *spatial densification* and *spectral aggregation* [28]. Spatial densification is the act of increasing the number of base stations deployed in a geographical area. It can also be achieved by increasing the number of antennas at each operating base station node. Several types of base stations co-exist in Het-Nets and can be densely deployed. However, further deployment of macro base stations imposes significant costs of both Operating Expenses (OPEX) and Capital Expenditures (CAPEX). In addition, deploying

more macrocells requires site planning and searching for possible deployment locations. As an alternative, picocells, which are small outdoor operator-deployed base stations, can be deployed more easily and with a much lower cost in both OPEX and CAPEX. Picocells require backhaul access to be directly connected to the operator core network. Relay nodes could be deployed where wired backhaul is not accessible. Relay nodes have the characteristic of appearing as a base station to mobile devices and as mobile devices to base stations. Finally, small cells nodes such as femtocells are the easier to intensively deploy. They are small size low power nodes that can be user-deployed. They require no site planning, and do not impose significant OPEX and CAPEX on cellular operators. Femtocells allow mobile users to have a base station at proximity with direct high speed connection. They also help solve black holes in wireless coverage, and improve signal quality at cell edge and indoor locations. However, sharing the same carrier frequency, between macro and femto cells, results in inter-tier interference. This introduces network design, interference mitigation, and resource allocation challenges.
As for spatial aggregation, it consists in using larger bands of spectrum ranging from 500MHz to the higher bands of $30-300$ Ghz [28]. Aggregating fragments of bandwidth of different frequency bands leads to antenna and transceiver design challenges. Sharing spectrum between several network nodes requires the integration of spectrum-sharing technique to be integrated into licensed carrier networks to assure good QoS and support mobility. Licensed Shared Access (LSA) is a paradigm that helps coordinate spectrum use between spectrum holders and secondary licensed users. LSA spectrum rights holder has the exclusivity of using part of the spectrum when no incumbents are using it. LSA offers the information necessary for LSA licensed users to use the bandwidth when the spectrum rights holder is not using it. It also allows to quickly moving the spectrum whenever spectrum rights holders need to operate. An example of the spectrum sharing and LSA is the TV white spaces concept.
Finally, it is important to note that spatial densification and spectral aggregation need to be supported by a densification of the network backhaul. Backhaul connects base stations to the core network; therefore it should be able to handle all the additional traffic brought by network densification. Otherwise, ultra-dense networks will have limited impact on the overall 5G networks performance.

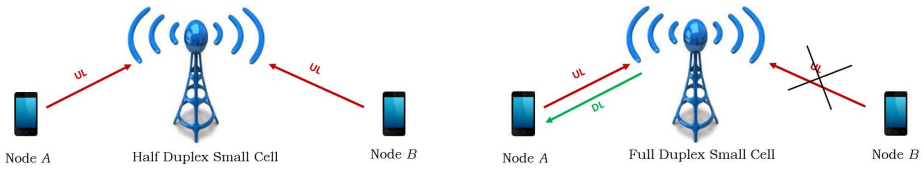
More spectrum and millimetric waves

Mobile cellular systems have almost always been deployed in the 300MHz - 3GHz band. However, the mobile wireless data demand, as well as the number of connected devices are and will continue to grow. The cellular systems spectrum is becoming increasingly crowded. Network densification with reduction of cells sizes is one way for allowing further spectrum reuse. However, this step is not enough since capacity only grows linearly with the number of cells. At the same time, the super and extremely high frequency bands (SHF and EHF) whose combined spectrum goes from 3 to 300 GHz, are underutilized. The signals in this band are referred to as millimeter-Waves (mmW) since their wavelengths are between 1 and 100 mm. Millimeter-waves are characterized by a large bandwidth that results in very high throughput and very small wavelength. Small wavelength has the advantage of allowing the implementation of a large number of very small antennas in a small device area. According to Pi *et al.*, millimeter-wave mobile broadband will offer 100 GHz new spectrum for mobile communication, a 200 times larger spectrum than what is used for the same purpose in the bands below 3 GHz [29]. Including mmW communication in the next cellular networks is an important pillar of 5G. With the increase of used bandwidth, capacity will increase and latency will decrease. This allows a better users' experience in real-time services and data hungry applications. The main challenges for mmW communications are mainly propagation

related issues [30]. Free-space path loss grows with the square of the carrier frequency. Therefore, going from 3 to 30GHz adds 20 dB of signal power loss. As stated by Andrews *et al.*, if antennas aperture are held constant, then free space-loss effect can be compensated. Maintaining the same antenna aperture can be assured by using antenna arrays [30]. In this case, the main challenge would be to co-phase antennas of the arrays so that they can efficiently collect energy. mmW signals are prone to be blocked by various objects in the environment. In a No Line-of-Sight (NLoS) trajectory, the blocking loss is very high an is in the order of $15-40$ dB added to the free-space path loss of around 40 dB [31]. Furthermore, mmW are subject of severe absorption due to rain and air. In conclusion, propagation challenges of mmW communication can be handled by using antenna arrays to collect and steer energy, and also requires narrow and highly directional beams in order to avoid interference problems. New challenges are then imposed in the narrow-beam communication which requires at first link establishment techniques and protocols, and of course adapted transceivers.

Full duplex communications

Full Duplex (FD) communication allows a wireless device to simultaneously transmit and receive data in the same frequency band. Wireless communication have always operated in a half-duplex mode, based on the assumption that wireless nodes cannot transmit while receiving signals due to the generated interference between transmitter and receiver circuits. This kind of signal perturbation is called self-interference. The key of using FD communication is to be able to cancel the effect of self-interference on signals decoding. Recent studies tackled this problem of self-interference cancellation in order to achieve a FD system [32]. FD communication may double the spectral efficiency at the physical layer by not using distinct slots for uplink and downlink anymore. It can also improve the efficiency of contention-based networks by eliminating the hidden node problem. In fact, hidden terminal problem occurs when a node *B* in the network cannot detect the presence of a node *A* that is transmitting data to the same destination at the same time, leading to collision at destination base station. Figure 1.3(a) shows a simple representation of both nodes in half duplex mode. Using full duplex communications, the base station can start sending data back to node *A* while simultaneously receiving data from the same node. Node *B* cannot hear node *A* transmitting, but can hear the base station. Once node *B* detects that the base station is transmitting, it delays its transmission towards the base station. In case the base station does not need to send data back to node *A* it can repeat node *A* signal. This will serve for securing the base station from any hidden terminal collision, and for sending a sort of acknowledgment to node *A*. Figure 1.3(b) shows a simple representation of both nodes in full duplex mode.



(a) Half Duplex scenario with two nodes (b) Full Duplex scenario with two nodes

Figure 1.3: Hidden terminal problem with HD and FD scenarios

Cloud technologies for flexible 5G radio access networks

A major direction adopted by 5G is based on cloud concepts. Exploiting the C-RAN concept (Centralized Radio Access Network), also referred to as cloud-RAN, deployed base stations use shared resources and the centralized cloud. In fact, the C-RAN architecture consists in splitting radio and processing functionalities of a base station. Base stations have always been designed to incorporate both radio protocol stacks as well as base band signal processing. With C-RAN functional split, base stations are reduced to Radio Remote Heads (RRH) that only handle radio modules. The processing functions are then offered to the RRH as a service through a pool of Base Band Unit (BBU). After being separated from analog radio access units, network management and base band processing units are moved to form a virtual cluster where all network functions are pooled. This cluster can be seen as if network functionalities are moved to the cloud and are offered as a service to the RRHs. The BBU pool serves several RRH in a particular area. A cellular network architecture with cloud-based radio access network is shown in Figure 1.4.

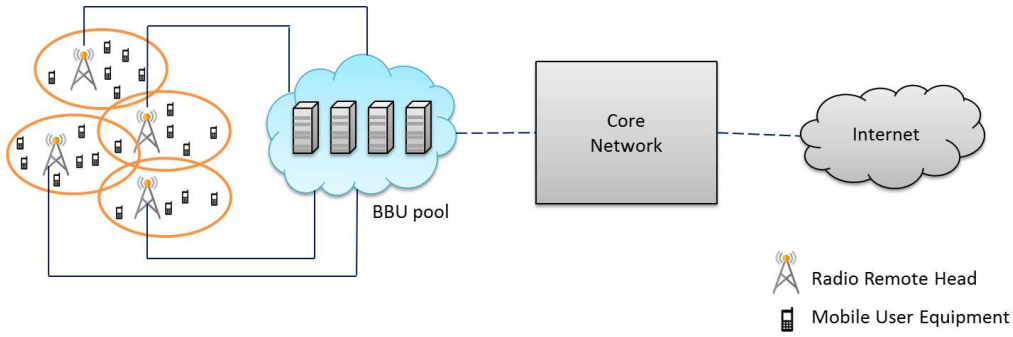


Figure 1.4: System architecture with cloud-based radio access network

C-RAN allows the deployment of more radio remote heads and thus enhances 5G networks scalability, capacity, and extends their coverage. Centralized network management lowers the cost of baseband processing and reduces the power consumption by managing radio resources (load distribution, cooperative processing) from several base stations. Furthermore, by enabling MIMO and Coordinated Multi-Point (CoMP) mechanisms through the C-RAN architecture, energy consumption can be minimized.

In addition to cloud-RAN architecture that offers network functionality as a service, cloud based computing services has been proposed as an effective 5G technology. Indeed, with the proliferation of a wide range of innovative and complex applications and services generating data and computation tsunamis, users' devices are facing a major challenge which is the inability to efficiently perform extensive calculation and process high data volumes. The splitting of hardware and software to enable horizontal services is an existing solution to such problems through cloud computing. As a matter of fact, cloud based solutions in the IT space have revolutionized the IT industry in recent years [33]. Importing the cloud concept to wireless network opens the door for deploying Mobile Cloud Computing (MCC) which consists of offering computing and storage capabilities to mobile users' devices over the cloud. Offering computing and storage resources on demand to mobile users creates a new level of flexibility and elasticity in network services. Mobile cloud computing plays an important role in user experience centric networks, where mobile subscribers expect excellent quality of experience with a minimal cost in terms of services delay,

data usage, and battery consumption. Advocating computation and storage functionalities to the cloud will alleviate mobile devices from executing related resources consuming mechanisms such as performing complex computations and searching large memory spaces for a particular file. In consequence, reachability of the cloud through mobile platforms increases computation and storage capabilities of performance limited devices in battery, processing capacity, and storage space.

In the remainder of this chapter, we focus on the cloud computing technology and its integration in current and future cellular networks.

1.3 Cloud Technologies and Network Architecture: A Joint Evolution

1.3.1 Cloud Computing

1.3.1.1 Definition and Characteristics

As defined by NIST (National Institute of Standards and Technology) [34], cloud computing is a model that enables ubiquitous on-demand network access to a pool of configurable computing resources. The accessible resources are speedily provisioned and provided with low management efforts or service provider interaction. In other terms, cloud computing provides computing resources as a utility and software and applications as a service. Cloud computing is achieved through geographical coalition of powerful servers connected to the internet. This coalition is referred to as a server farm that handles computation in a distributed way. Network users offload computational tasks, applications and services demands to the centralized servers for a cost-effective computation and a higher QoS. Co-located in a single site, large server clusters handle computation through a distributed system, offering users a faster computation. Cloud computing offers its users increase in computational and storage resources capabilities. The cloud computing paradigm is also referred to in literature as *on-demand computing*, *utility computing*, or *pay as you go computing*. The resources offered by the cloud are available over the network and are accessible through network mechanisms by any connected device. This characterizes the cloud with broad network access. Access to cloud resources is granted for users without interaction with their service providers. On-demand self-service is offered to cloud consumers upon request for imminent use or provisioning. The cloud serves multiple users simultaneously through assigning and re-assigning resources. Computing, storage, memory, and other possible resources are pooled in server farms and are not dependent of consumer's location. In other terms, cloud users have no knowledge about the exact location of provided resources source. Resource pooling allows a better management and allocation of resources to cover a wider set of consumers. Therefore, cloud mechanisms for resource allocation, provisioning and release must be rapid enough to scale with varying demands. Elasticity is an important characteristic of cloud computing platforms to always give consumers the impression of having access to unlimited capabilities. Due to the centralized handling of cloud resources, usage tracks can be kept in record and used for improving both users utilization and providers control of cloud services. To resume, cloud computing characteristics can be summed up to on-demand self-service, broad networks access, resource pooling, rapid elasticity, and measured service.

1.3.1.2 Service and Deployment Models

Defined cloud computing service models are: Software as a Service (SaaS), Infrastructure as a Service (IaaS), and Platform as a Service (PaaS).

- Software as a Service is defined as the ability of users to access applications deployed on the cloud by its providers. Consumers reach the application through an interface on their device. However, they have no control on the infrastructure underneath, including storage, operating systems and servers. SaaS is a growing market where applications are delivered in a one-to-many model. Deployed on the cloud, applications are managed centrally and not by users' end devices. Applications updates are not required to be handled by consumers and are directly installed by providers on the cloud servers.

- Platform as a Service gives access to cloud deployed development platform. Consumers have the ability to create and deploy their own applications on the cloud using a provider platform overlaying a development environment (programming languages, libraries, services, and tools). As in the case of SaaS, consumers have no control whatsoever on the underlying cloud infrastructure. As stated in [35], SaaS and PaaS can be seen as analogs where the former is a software delivered over the web while the latter delivers the platform for creating software on the web.

- Infrastructure as a Service offers resources control and provisioning possibility for consumers. Consumers have access to the cloud resources such as processing, operating systems, and storage, where they are able to deploy software and applications and launch computational tasks. Even though consumers do not control the cloud infrastructure, they have control on its computing resources, deployed applications and some networking components. Using IaaS, consumers do not have to invest in expensive hardware, plus, scaling hardware capabilities up and down is much easier and automatic. Requiring more resources could be handled through using a larger part of cloud resources (which imposes higher money costs most of the times) instead of investing in buying and placing new hardware that includes much more costs in terms of deployment and maintenance. On the other side, downscaling resources is achieved by simply releasing provisioned cloud resources instead of keeping on paying for the cost of deployed but unused hardware.

Cloud services, with their three defined models, can be obtained through various types of cloud deployment. First, cloud utilities can be delivered through *Public clouds* to which access is publicly granted for all types and varieties of consumers. This does not impose delivering the same service quality for all consumers. Providers can always propose various cloud plans with various costs depending on the amount of accessible resources. Such clouds can be managed by no other than the providers, or by institutions (business, government, academic) that deploy such clouds on the premises of the cloud providers. Contrary to *Public clouds*, which grant open access to the general public, *Private clouds* grant access to only a closed set of users. *Private clouds* services are exclusively dedicated for a specific set of users defined by the owner/manager of the cloud. This type of clouds is vastly used by business organizations that either own their private clouds or use a private cloud provided by a cloud services provider.
A *Community cloud* as defined in [34] is exclusively used by a community of consumers from

organizations that share the same concerns. Such cloud can be either owned by one of these organizations, a cloud provider, or a combination of both.
Any composition of two access modes of private, public, and community, leads to a *Hybrid cloud* access mode. In Hybrid clouds two or more clouds of two or more deployment modes are bound and can be inter-operatively used for specific portability-enabled applications.

1.3.1.3 Cloud Computing Enablers

Virtualization

Virtualization technique is the main enabler for cloud computing environment. As in the cloud all consumers are sharing the same hardware, a virtualization of resources abstracts them as virtual machines (VMs) along with associated storage and networking connectivity [36]. It is a separation of hardware and software that enables horizontal cloud based solutions. Virtualization creates virtual resources such as operating systems, servers, or storage devices. Resources are available on demand, which introduces a new level of flexibility, scalability and automation in service deployment. Virtualization is possible through a hypervisor layer added above the hardware layer. Several VMs run then on the hypervisor layer, and thus on top of the physical layer running regular operating system. VMs have no access to the hardware but through the hypervisor layer. By virtualization, multiple VMs can run on a single physical machine. For cloud computing environments, virtualization is then a key enabler that allows several consumers to run various tasks on the same hardware, the cloud hardware specifically. They are currently creating a major evolution and transformation in the communication industry by offering efficient solutions that increase the network scalability and flexibility.

High standard servers

Cloud computing is based on computing users tasks, applications, and services using a pool of resources reachable through network access. A main motivation for using cloud services is the lack of capability or the high cost of local computing resources to accomplish requests computations. Advocating computation to the cloud should guarantee, at least, enough computation capacities to handle traffic and computation of a very high number of users. Cloud servers should be high-volume IT hardware in order to support the commercial use of cloud computing. Server components, if standardized, can be rapidly and efficiently changed or updated in a cost-effective manner.

1.3.2 Cloud Technologies in Cellular Networks

Early generations of cellular networks were all about offering voice communication between subscribers. Even though 2G included some very low rate data services through GPRS (General Packet Radio Services) (up to 115 Kbits/sec), it is not before 3G that cellular networks could offer advanced data services to deliver speeds in Megabits per second [37]. In less than two decades, data services demand in cellular networks grew thousands nay millions of times larger. With the new generation of mobile networks being IP-based, data traffic demand has been exponentially increasing. Indeed, Figure 1.5 shows the total traffic from 2010 to 2014 comparing both data and voice traffic. Voice traffic development is almost flat compared to the clear exponential increase in data traffic. This is due to the increasing number of mobile data subscribers since mobile phones have been an effective tool for accessing various internet-based services. Mobile networks are

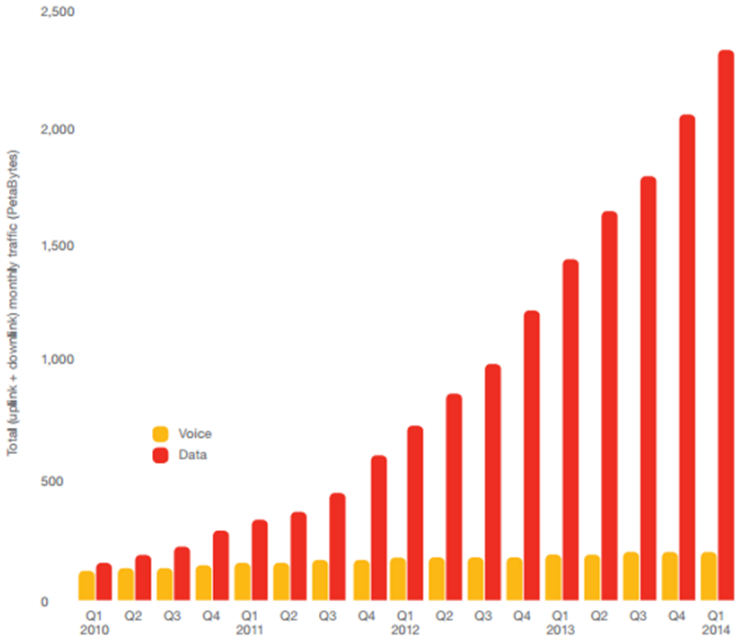


Figure 1.5: Total global monthly data and voice traffic [3].

no longer dedicated to voice calls and text messages; they are henceforth a capable platform for accessing the Internet and for launching applications of unlimited scopes, going from web browsing to video processing. Delivering high quality voice communication is no longer the sole goal of mobile networks. The wide scope of applications, becoming available over cellular networks, demands high computational capacities. First, considering the ever-increasing number of connected mobile devices, more sophisticated mechanisms are needed to compute efficient resources management on the network level. Furthermore, demand for available computing resources is increased when users ask for services and applications requiring computation. As the services expected from mobile networks changed from communication to computing, an evolution of network architecture and serving base stations is a must for involving required computational resources.

1.3.2.1 Classic Base Station Architecture

The traditional architecture of mobile base stations is one where both radio and baseband processing functionalities are integrated within. All base station functionalities are deployed in the same location as the base station itself. Radio (RF) module is placed at a proximity to the antenna linked through coaxial cables. The baseband (BB) processing is located at the same site. Figure 1.6 shows the classic base station architecture.

With the proliferation of 3G networks, new network architecture was proposed which is based on a split of base station main functionalities: radio and baseband processing. This architecture consists in dividing base stations into two separate entities: Radio Remote Heads (RRH) and BaseBand Unit (BBU). RRH is the unit that handles all analog and radio modules and functionalities, along with conversion between analog and digital. BBU is where all the other network functions modules are deployed. A non-exhaustive list of BBU services contains FFT/IFFT operations, modulation/de-modulation, sampling, MIMO management, channel coding and decoding, interference management (e-ICIC), multi-point communication management (CoMP), transport

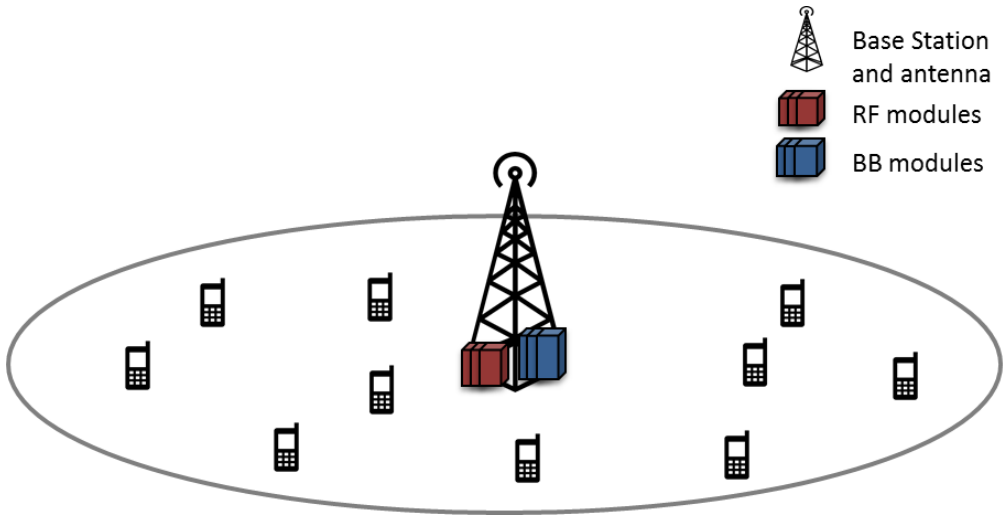


Figure 1.6: Traditional base station model

and MAC layers, and radio resource control. After function splitting, RF modules are placed right next to the antennas. As for the baseband unit, it is relocated to a distance that can go from hundreds of meters to tens of kilometers. RRH and BBU are connected through either optical fiber or microwave connections. A first advantage of such architecture is the ability to link a set of multiple RRHs to one BBU. This will reduce the cost of deploying RRHs. Furthermore, BBUs are placed in a more convenient location, enabling cost savings on site rental for deploying all-in-one traditional base stations. The distributed Radio Access Network (RAN) with RRH is represented in Figure 1.7.

1.3.2.2 Cloud Radio Access Network (C-RAN)

Radio Access Network (RAN) is a very important part of mobile networks. It is where all processing and computation takes place in order to manage network resources and deliver high quality high data rate services for mobile subscribers. In the traditional RAN architecture, each base station handles transmission/reception signals for a certain number of users over a specific geographical area. With the increase of the number of users currently witnessed by wireless networks, this RAN architecture faces severe interference problems and thus degradation of capacity per users. Adapting a solution consisting of deploying more base stations requires more site rental, and thus, imposes additional CAPEX and OPEX costs. Furthermore, the amount of computing necessary for network management is increasing, and thus, more computational capacity is needed. In addition to that, with the Internet proliferation, and IP traffic constant growth, a need for over-dimensioning of processing and forwarding resources in the radio path has emerged [38]. A cost-efficient solution with high quality of delivered services is thus required. Following these requirements, a Centralized Radio Access Network (C-RAN) architecture has been proposed and given high importance [39]. In C-RAN, baseband units are centralized into one entity referred to as BBU pool (Figure 1.8). The centralized pool handles all the processing for different cell sites, and is virtualized. This reduces the number of equipment needed at each base station site. The BBU pool is connected to the mobile core network through backhaul. Fronthaul connects RRHs to BBU

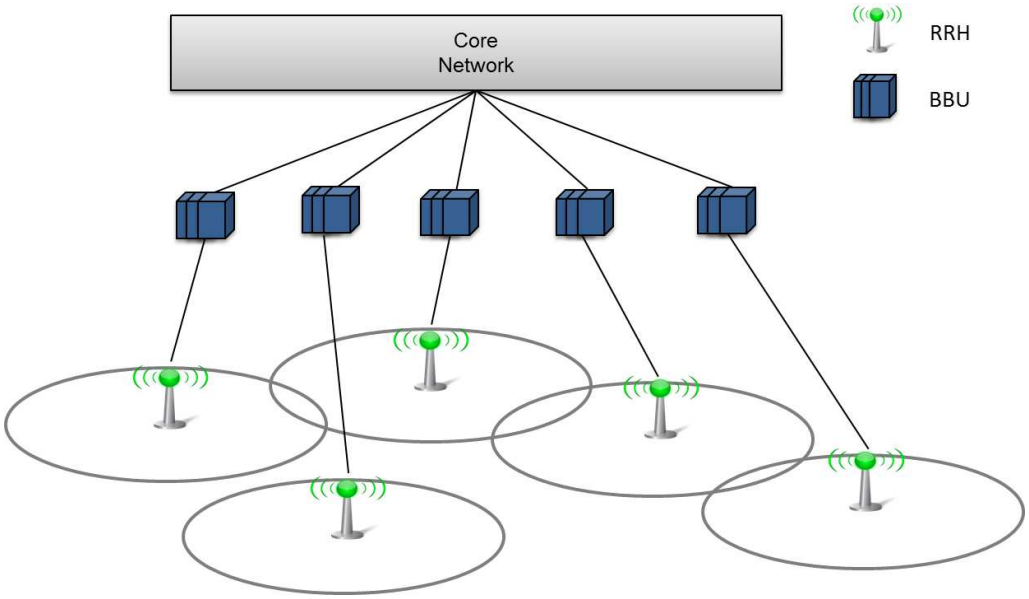


Figure 1.7: RAN distributed architecture

pool. When pooled, BBUs utilization is more efficient and cost-effective. Network flexibility is

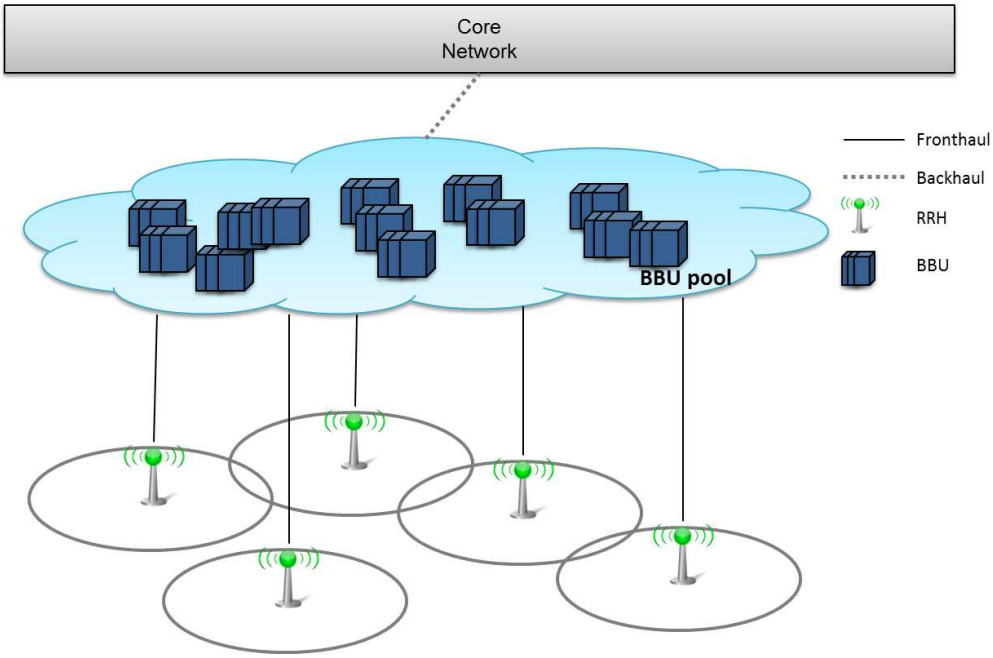


Figure 1.8: C-RAN architecture

increased and power consumption reduced. BBU pool provides a concentration of high processing

capabilities used for faster solutions and resources management computation and for decreasing response time of application servers. Furthermore, BBU pools increase network scalability by allowing a large number and variety of base station (macro-cells, pico-cells, femto-cells) to benefit from C-RAN services. BBU pools can be seen as server farms made available for computing all network required baseband functionalities through the virtualization of the RAN.

1.3.2.2.1 Advantages of C-RAN

Statistical Multiplexing C-RAN transports all baseband processing to a centralized pool of BBUs. The computational capacities needed at each base station are replaced by a centralized capacity at a BBU pool which is supposed to be smaller. Statistical multiplexing is defined as the ratio of the total processing capacity required at the BBU pool to the sum of processing capacities needed, in case of classic RAN, at all base stations covered by the same BBU pool. Statistical gain has been studied and assessed in several works, and potential gain is estimated of around 25% of the computing resources [40] [41] [42]. An important factor contributing in achieving computing resources gain is the adaptability of C-RAN to non-uniform traffic. Base stations were always designed to have high performance at traffic peaks and busy hours. However, daily traffic of mobile users varies throughout the day. Resources are wasted in base stations in off-peak hours and off-peak sectors (location). With BBU pool handling all baseband processing for a large set of base stations, compute resources utilization rate is improved and adapted to the variation of network load.

Scalability Improving coverage and increasing network capacity can be simply achieved by adding more RRHs and splitting existing cells. Since all RRHs are linked to a BBU pool, deploying more RRHs does not require finding a large location site to install a cumbersome base station. RRHs are more easily deployed and accepted by local communities. By increasing the number of operating RRHs, network scalability and flexibility are improved. Additionally, increasing the overall network capacity can be centrally managed at a unique location where BBU pool servers can be expanded, empowered and updated.

Costs savings OPEX costs are reduced with C-RAN architecture since all the equipment is aggregated in a single location. Maintenance interventions and operations costs associated with the large number of BBUs in RAN are saved. Furthermore, due to the adaptability to non-uniform network load, some BBUs in the pool may be switched off without affecting network coverage and performance. This allows saving electricity and cooling costs.

Increase of network capacity BBU pool is associated to a centralized processing of many virtualized base stations. Joint processing between base stations normally requires a non-negligible amount of signaling for sharing traffic data and channel state information (CSI). In C-RAN, these information can be easily shared which permits to implement more efficient interference management and mitigation techniques, such as enhanced inter-cell interference coordination (e-ICIC), and in consequence improve spectral efficiency. All techniques requiring multi-cell cooperation are easily implemented with the C-RAN architecture. A major example is coordinated multi-point (CoMP) which also fights inter-cell interference by a set of cells coordinating for serving a single set of user(s) and thus increase the perceived Signal to Interference plus Noise Ratio (SINR) at the mobile side [43]. CoMP requires tight synchronization and coordination between participating base stations which can be achieved more rapidly, efficiently, and with lower costs in C-RAN.

1.3.2.3 Mobile Cloud Computing: Remote Clouds

New innovative applications are released at a daily basis covering wide areas of communication purposes, going from entertainment and social networking to industrial and health-care applications. Applications that are being released require increasing amount of data processing and computation. This covers a very wide sector of applications that could include, among others, video decoding, image recognition, and online gaming. The mobile devices business industry is endowing devices such as smartphones and tablets with advanced features and services. However, mobile handsets are limited in computational resources, storage capacity, and energy (limited battery lifetime). Therefore, mobile handsets processors, even if adapted to be equipped with computing capabilities, can be easily overloaded: The launching of several applications simultaneously, or a high computation load application, will eventually lead to a lower quality of experience for mobile users. This could result in quick battery discharge, longer response time, or the shutdown of some running applications. This is a problem that we have all experienced, at least once, as mobile users. Offloading computation requests to remote servers has been recognized as an effective solution for guaranteeing good Quality of Experience (QoE) while minimizing mobile handsets energy consumption. On-demand resources, such as storage and computing, have already been implemented through Cloud Computing. For network processing functionalities, on-demand resources are also possible through C-RAN architecture. In wireless mobile networks, offloading computation tasks of mobile users to remote resource providers instead of being computed by the mobile handset itself is referred to as Mobile Cloud Computing (MCC). MCC has been widely discussed in literature. Many comprehensive surveys detail and explain its architectures, taxonomies, motivations and challenges [44] [45] [46] [47].

The term of mobile cloud computing refers then to the ability of running mobile applications and computations by using resource providers other than the mobile device itself. The network architecture through which this is possible is shown in Figure 1.9. Mobile users are connected to the mobile network through the association with a base station of any type (Marco, Pico, Femto). The base station is connected to either a BBU, in the case of RAN with RRH, or to a BBU pool in case of C-RAN (the case of Figure 1.9). BBU is connected to the core of the network through which Internet is accessible. Through the Internet, cloud servers are accessible by mobile users initially communicating with a cellular base station. Mobile users' computational case can be sent over the described architecture to reach cloud powerful servers in order to have access to greater computing resources than available resources on mobiles devices. Examples of applications that motivate the need of mobile cloud computing include: image processing, natural language translation, crowd computing, sensor data applications, multimedia search, and social networking [47] . These examples represent a non-exhaustive list of possible applications that can benefit from the MCC paradigm. MCC is adopted for offloading computational tasks, or retrieving requested information with costs lower than local computation on mobile devices. As cloud computing, mobile cloud computing should also adapt to traffic non-uniformity, and assure a high degree of scalability, flexibility and availability.

1.3.2.3.1 Advantages of Mobile Cloud Computing

Extended battery lifetime Battery lifetime is one of the biggest concerns of mobile users. As mobile phones are becoming multi-service, multi-application platforms generating an ever-increasing amount of traffic, a higher autonomy of such devices is required. Unfortunately, with the growth of mobile phones capabilities, battery industry is not advancing in the same pace for improving battery lifetime [48]. Mobile cloud computing through computation offloading is one of

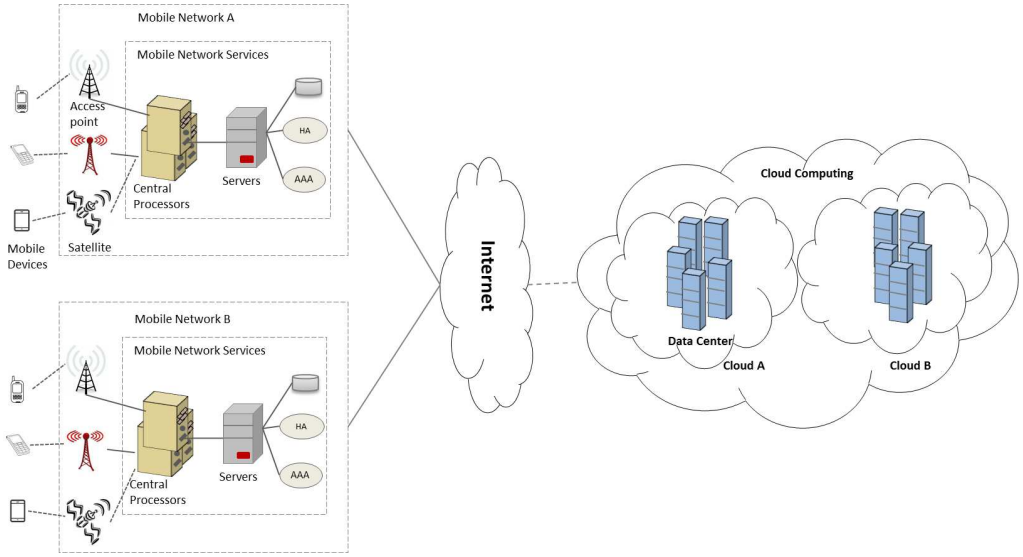


Figure 1.9: Mobile cloud computing architecture [4]

the most effective current solutions for this problem. Furthermore, mobile devices do not consume energy, and in consequence do not lose battery lifetime, for computing large tasks.

Empowering mobile devices A wide variety of applications are often too computation intensive to run on a mobile device. Indeed, mobile devices are resource-limited in terms of energy, computation, processing, and storage space. As discussed previously, MCC can help extending devices battery lifetime. As for computation and processing capabilities, mobile devices executing applications on resourceful and powerful distant cloud servers are given a great increase of computing powers. Through MCC, mobile devices are able to perform, virtually, complex and large computations and process a larger amount of data in shorter delays. Furthermore, being connected to powerful cloud servers allows mobile devices to use available storage space remotely. This allows saving a great deal of local memory space at the mobile device. With a connection to a cloud that is always available, mobile users can access their stored files, photos, documents, and videos stored at the cloud servers instantly.

Improving reliability By delegating applications and data processing to cloud servers, backup and storing data are saved on the cloud servers. These data, along with any personal files stored on the cloud, could be easily retrieved in case of a crash at the mobile device level. Furthermore, applications that are computed on the cloud are accorded necessary computational power by high standard servers that are less exposed and more insusceptible to any system crash or disfunctionality.

Scalability MCC provides mobile users with a powerful tool for performing computation. Furthermore, mobile devices do not have to allocate computation resources and schedule tasks if they are offloaded to the cloud. This gives mobile devices the ability of dynamic on-demand provisioning of resources on a self-service basis. Mobile devices can thus run their applications without prior reservation of resources due to the scalability and high availability of cloud resources.

1.3.2.3.2 Operational Issues

Offloading decisions Mobile cloud computing is based on a main operation of mobile devices: offloading. To use cloud resources, mobile devices offload computational tasks to the cloud through the mobile network. A major operational issue of MCC is to take the right offloading decision. The main questions to ask are: What are the applications to offload? When is offloading beneficial? What should be the offloading decision based on? Answering these questions is not as easy at it seems. A basic approach would be to offload applications that cannot be performed using the mobile device limited resources. But MCC is about more than being an alternative for local computation. Many issues may prevent or push mobile devices to offload computation. Various research studies tackle this paradigm, trying to find the best approach to take the best offloading decision. A very intuitive basic decision rule has been presented in [49] based on the concept of saving energy at the mobile device side. Since the mobile device communicates with its serving base station through wireless connection, the data transfer to the network may impose serious costs depending on the channel conditions, the used transmission power, and the available bandwidth. The work by *Kumar et al.* simply compares the energy consumption of mobile devices in both cases of local computation and offloading in [49]. The energy consumption gain or loss depends on the requested computation size and the communication link quality. Figure 1.10 shows the conclusion reached in Kumar's work, indicating that offloading computation is beneficial (energy wise) when the size of instructions to be computed is large, and the necessary amount of bits to send is small. If the amount of bits to communicate is large comparing to the size of instructions to

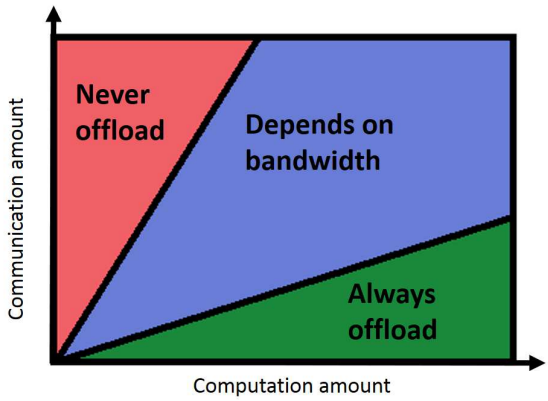


Figure 1.10: Offloading decision based on mobile device energy consumption

compute, offloading will consume more power consumption than local computation. In other than these two extreme cases, the gain of energy consumption depends on the available bandwidth and the channel conditions. This conclusion takes into account that the computation of the requested application is possible on both platforms, local and in the cloud. However, this is not always the case. Mobile devices are resource-limited, and could be sometimes unable to ensure the required resources amount of either computational capacity, memory requirement, or others. In addition, some applications have tight latency constraints that cannot be met by offloading computation to the cloud due to physical distance separating the service need and the computing resources. The offloading decision paradigm is widely studied in literature. A detailed state of the art can be found

in Chapter 3 where we also present our own contribution in computation offloading decision.

Privacy Sending user computation to be performed on remote servers may include sending sensitive and/or private information. An example of this information is the user's GPS location which is necessary to provide location-based services, but remains private and personal information. Being computed on remote servers, application data are henceforth stored at the computing servers and not on users' devices. This introduces privacy and security issues especially with the exchange of sensitive data managed at the cloud. Any loophole in cloud security systems can threaten the privacy of users' data. Many recent attacks and hacking incidents have taken place in cloud environments [49] [47]. These may occur due to either system failure or hacking operations. Therefore, MCC should have strong and resilient privacy policy that guarantees at best users' confidentiality. To fight against such problems, we often notice an agreement between users and cloud services providers on terms and policies including privacy and dissemination of data.

Handover in heterogeneous network access In MCC, mobile users access cloud services through the wireless network. Two types of handover can create issues for service delivery from the cloud. The first is when users toggle between different types of base stations that are linked differently to the cloud. For example, users can be connected to a radio cellular base station at one moment, and switch to a WiFi connection at another. In this case, the cellular base station is no longer the connecting entity between the mobile device and the cloud. Mobile users expect to be delivered computation results and services no matter how often they change connection from one base station to another, or the base station type they are connected to. This variation affects service delivery drops, hence, soft handover schemes are required [50]. The second type of handover than can occur, is between homogeneous types of base stations. This is mainly due to users' mobility and the resulting switching off of base stations. Intelligent mobility management is an important issue in MCC systems. Users should be provided seamless service through a mobility supporting cloud services delivery system.

Network latency Service and applications delay is an important metric in evaluating surrogate computation. Indeed, latency constraints can be very restrictive bottlenecks especially in real-time applications. If on top of that real-time applications require the exchange of a large amount of data, latency is a limiting factor. Looking at MCC architecture in Figure 1.9, we can spot possible latency increasing factors. First, data should be transmitted from mobile devices to the serving base station. The data transmission time between those two parties depends on many factors such as the channel quality, the distance between users and base station, and the transmission protocol in use. Then, data is transferred from base stations through the Internet to remote cloud servers. Unfortunately, the latency of WAN connection for reaching the remote clouds cannot be controlled and might be high. Remote clouds are very far from the mobile devices in both physical location and network transport. As a result, required resources (computing, storage) are far from the need (mobile devices), and thus MCC latency due to transport and computation delays may not meet with the tight delay constraint some applications may impose. Solutions that have been proposed to tackle this problem mainly involve an MCC architecture evolution, and resource management optimization. These solutions will be presented in the following sections of this chapter.

Availability and reliability Another issue resulting from connecting mobile devices the cloud infrastructure through wireless networks is network availability. Wireless transmissions

through wireless networks are less-reliable compared to wired network connections. Services may be found interrupted for random reasons of system failure. Furthermore, the accessibility of MCC services depends on the seamless and ubiquitous coverage of wireless network. Indoor, crowded, cell edge, and out of coverage scenarios are examples of situations where MCC services are either not available or hardly accessible. Furthermore, even though cloud computing is based on powerful servers handling the computations, system outage is however possible. Server outage may occur and result in service failure, and loss of data and connectivity. Not to forget data storage issues that can occur after servers crashes that lead to major losses of users' data or service unavailability. Solutions for availability and reliability problems are mostly based on expanding the cloud infrastructure and empowering it with more powerful servers, in order to enable it to handle peak traffic. As for storage, backup copies of users data is an effective solution, even if CAPEX and OPEX costly.

1.3.2.4 Mobile Cloud Computing: Cloudlets

After the introduction of mobile cloud computing in wireless networks, solutions to operational and technical issues discussed above were investigated. In 2009, Satyanarayanan *et al.* proposed a new concept of MCC using "Cloudlets" [51]. The concept was introduced by the authors as "*A new vision of mobile computing liberates mobile devices from severe resource constraints by enabling resource-intensive applications to leverage cloud computing free of WAN delays, jitter, congestion, and failures.*" The work proposes a new architecture of the wireless network where cloudlets are introduced as a new entity in the network. Cloudlets are defined as resource-rich powerful computers, or cluster of computers, that are deployed as "data centers in a box". They are connected to the Internet and can be used by near mobile devices. The main motivation for cloudlets is overcoming un-controllable WAN delays of mobile cloud computing in remote clouds. Cloudlets are installed at a proximity to mobile users and are accessible through a single hop connection. The cloudlets proximity to users, both physically and in network layers, is an effective solution for overcoming harmful WAN large delays. However, cloudlets are proposed as accessible through local area network connection (LAN) and not through wireless cellular network. Cloudlets are not part of the mobile cellular network, and are not controlled by wireless network providers. Cloudlets are to be managed by end users and can serve up to a few users simultaneously. Figure 1.11 shows where cloudlets are placed in the network. Cloudlets proximity not only allows mobile users to have ubiquitous good service quality, but also to save energy due to the short transmission distance. The introduction of cloudlets brings the cloud closer to mobile devices, i.e. brings resources closer to the need. An additional feature of cloudlets is the fact that in case of system crash or loss of data at the cloudlets level, mobile users are not much affected. Cloudlets connect to remote centralized clouds in order to store data. What is kept at the cloudlets level is a copy of the data. Therefore, cloudlets do not increase the risk of data loss compared to remote clouds. However, the fact that cloudlets are new service providers independent elements to introduce into the wireless network infrastructure has two major drawbacks. Firstly, cloudlets may serve only a few users at a time. Therefore, a cloudlet may not be available even if accessible. In case of unavailability or inaccessibility of a nearby cloudlet, mobile users are then forced either to connect to remote clouds or to compute all tasks locally on their devices. Furthermore, as a newly introduced entity to the network, no wide deployment of cloudlets is available. The case of non-existence of nearby cloudlets risks of being frequent. Secondly, being independent of mobile network, cloudlets do not have access to operator related knowledge. In this case, location aware services, users positioning and mobility management are harder to handle.

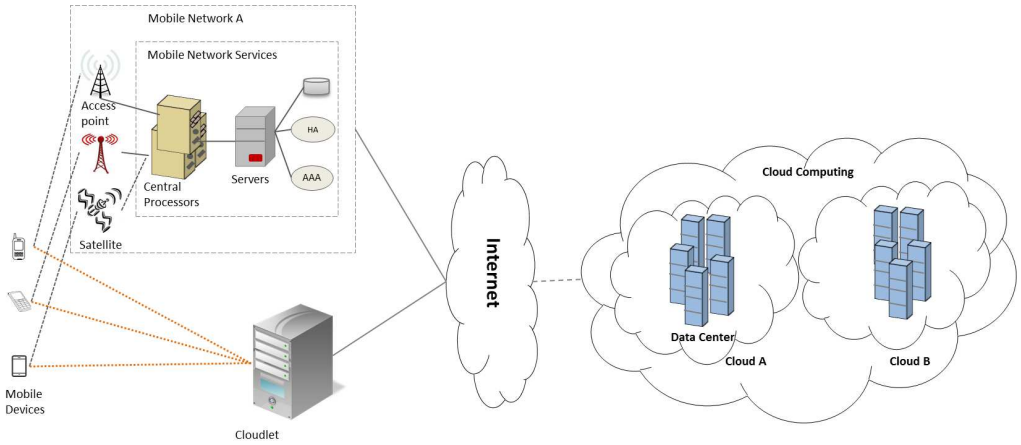


Figure 1.11: Network architecture with cloudlets

1.3.2.5 Mobile Edge Computing

Cloudlets bring the cloud close to mobile users. However, it is, as previously discussed, independent of the mobile operator network. As communication and IT worlds are converging, and with the emergence of over-the-web applications that run on mobile devices and use operator network knowledge (such as location based services), it is more convenient to allow cloud computing elements to be implicitly integrated in the mobile network. Looking at mobile networks, we can notice that capabilities within the RAN extend to a close proximity to mobile subscribers. The RAN edge is characterized by an ultra-low latency, high bandwidth, and direct access to real-time network information. Therefore, deploying cloud services at the RAN edge allows content services and applications to be accelerated.

Moving cloud capabilities to the RAN edge is known as Mobile Edge Cloud (MEC). By moving the cloud to the edge of the network, mobile core utilization is alleviated and latency is reduced for mobile end users. MEC aims at reducing network load by moving computational efforts from the internet to the mobile edge. As discussed in previous Section 1.3.2.1, traditional base stations, which are the devices deployed at the edge of the mobile network, only forward traffic. But they do not actively analyze, nor respond to user requests. Thus, they do not provide computing resources for hosting edge services beyond network connectivity.

Mobile devices-base station links have always been considered as “dumb” links dedicated to only transporting communication data. With the on-going convergence of IT and communication worlds and the advent of software defined infrastructure, network operators can make the mobile devices-base stations link intelligent by overlaying distributed cloud computing solutions onto the RAN. MEC introduces new network elements at the edge, providing computing and storage capabilities at the base stations. MEC can be seen as a cloud server running at the edge of a mobile network and performing specific tasks that could not be achieved with traditional network infrastructure (M2M gateway, control functions) [33]. Mobile edge computing proposes co-locating computing and storage resources at base stations of cellular networks. This requires deploying general purpose processors and storage onto base stations. In cellular networks, outdoor mobile edge is represented by eNodeB or base station (e.g. macrocell) located in close proximity of mobile subscribers. MEC users are typically connected to base stations that loop the traffic through

the MEC server for further processing of the data. Indoor, MEC compact servers are added to serving small cells (basically femto-cells). Indoor scenarios include networks in enterprises, shopping malls, and other commercial buildings. A compact server is added to a small cell that manages traffic to multiple small cells.

In MEC, communication between cloud edge servers and mobile users is done over high-capacity radio link that operate over 3G/4G/or the future 5G network radio access networks. Operating cloud services jointly with the cellular mobile network ensure wide coverage through the wide deployment of operator networks. This further ensures that the vast majority of the customers of a mobile operator can be served.

1.3.2.5.1 MEC Enablers and Characteristics

As for remote mobile cloud computing, virtualization techniques and high standard servers are major key enablers for mobile edge computing. Virtualization, which enables several virtual machines of various users to be deployed on the same hardware, creates readily-available computing capacity for service-oriented software. Resources are therefore available on-demand. However, the unique architecture of MEC requires an additional key enabler: availability and integration of MEC adapted applications. Open environments need to be created to allow the efficient and seamless integration of edge enabled application across multi-vendor MEC platforms. Innovative MEC enabled applications need to be introduced to market in order to push the prosperity of the edge computing concept. When MEC applications emerge with new ideas and are accessible for mobile users and adaptable to all devices platforms, a new ecosystem of MEC applications and services will take its place in the market.

Mobile edge computing architecture is characterized first of all by its proximity to mobile users. Services are hence managed and operated for satisfying mobile users requests directly at the network edge. Data traffic do not have to be forwarded through the internet to remote clouds. In other cases, where computation is offloaded by users to remote clouds, MEC has the ability of forwarding necessary data to the required destination. Furthermore, the availability of resources at the edge of the radio access network eliminates the need of routing data through the core network or through the internet. Edge computing servers are deployed at the very edge of the network, the closest to mobile devices with direct access between both parties. This proximity results in a lower end-to-end delay experienced by mobile users, which is also supported by high bandwidth connection between mobile users and MEC servers. MEC servers can run independently from the rest of the network. Data do not need to travel higher levels in the network if all resources are available at the edge of the network. This aspect of MEC is important for privacy and security issues of users data. Being a part of the mobile network, MEC has access to operator network knowledge as real-time radio network information and location awareness. This will allow the implementation of a wider set of business oriented services and applications that are dependent of location or context information

In outdoor scenarios, MEC servers are deployed in base stations. MEC helps improving mobile users' QoE by reducing latency and improving QoS by providing customized service related to consumers' context. MEC improves infrastructure efficiency with more intelligent and optimized network management and resource allocation. In addition, MEC enables vertical services e.g. M2M, big data management, smart cities. As servers are deployed at base stations where all traffic is routed, it is easier to understand traffic characteristics through probabilistic analysis and thus deal with radio cognition with the help of devices context information. As for indoor, users can enjoy dedicated intelligence that comply with their specific context and needs due to smaller cells and thus better provisioned and dedicated resources. Furthermore, in indoor scenarios, appli-

cations applied to particular locations (M2M, retail, crowds, big data) will thus be widely available through MEC. To resume, MEC provides a highly distributed computing environment that can be used to deploy applications and services, as well as to store and process content in close proximity to mobile users. It creates an ecosystem where new services are developed in and around the base station. The MEC server provides computing resources, storage capacity, connectivity, and access to user traffic and radio and network information [33].

1.3.2.5.2 MEC Applications and Services Ecosystem

The most obvious advantages of edge computing inside cellular networks is given by both reduction of end-to-end delay and context information accessibility. A solution without edge computing would involve a transmission through the core network as well as through Internet links towards the application host and back. In [5] a categorization of several application types which are possible candidates for the deployment at the mobile edge is presented. The promising applications are resumed in Figure 1.12.

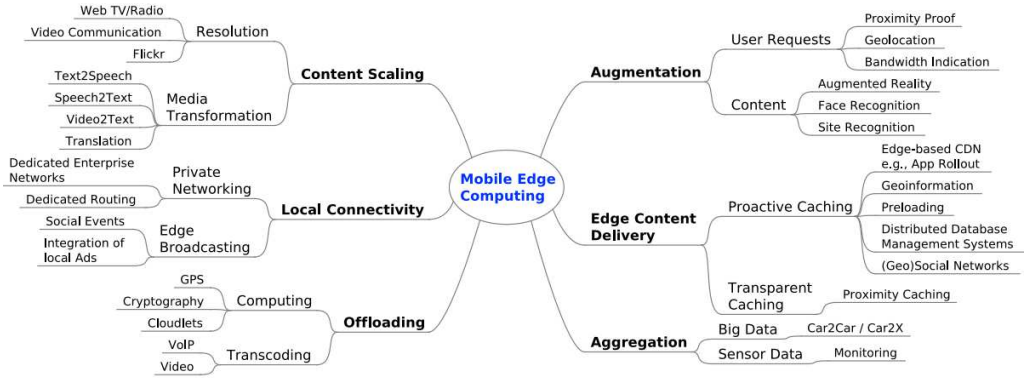


Figure 1.12: Mobile edge applications and use cases [5]

Offloading Computation tasks that are launched at mobile users can be offloaded for remote computation. Offloading decision can depend on several parameters and is due to various reasons. Examples of the most common reasons are the insufficient resources at the mobile devices for intensive computations, the inability of local devices to perform computations with acceptable delay, and the high energy consumption of mobile devices performing computations. Examples of computing intensive applications are transcoding of multimedia traffic, and face recognition. The use of MEC makes offloading feasible in more cases, as today's radio bandwidth is much higher compared to usable Internet bandwidth. Moreover, tasks that would typically be performed on the mobile device due to the size of their input can be performed on the edge [33].

Edge content delivery Located at the network edge and equipped with storage space, MEC servers are able to cache local content information at delivery nodes and serve users directly by retrieving data locally. Caching techniques in general can be classified as reactive and proactive. Caching is transparent if neither the mobile users nor the service provider are aware of the caching MEC server. Proactive caching consists in non-transparently caching content before being

requested. Example of data that could be cached is all location based services data and context related information. MEC servers can either keep their own cache private and not share it with other MEC servers, or share cached information with neighbor MECs. These modes are known as isolated and shared cache, respectively. MEC enables edge content delivery which aims at storing data in a close proximity to the need (requesting users' devices). Edge content delivery leads to a reduction of experienced latency, computational capacity, and energy consumption comparing to centralized database systems.

Data aggregation Several applications that generate huge amount of data and that are region or context related (e.g. Car2Car, GPS route information) generate a lot of similar and region-related event notifications which can be aggregated. Aggregation can also take place in the context of monitoring applications where many devices measure similar data that can be jointly processed at the edge of the network thanks to MEC servers.

1.3.2.5.3 Technical Challenges and Requirements

In order for MEC to be completely and efficiently integrated in the wireless network infrastructure, and contribute in the advancement of such networks, some challenges need to be overcame. The following is a non-exhaustive list of possible technical challenges that can block or delay MEC effective operation in wireless networks.

- MEC servers should be able to integrate the existing network infrastructure without affecting mobile devices and base stations functionalities. MEC servers should comply with all network standards and specifications and their implementation should be transparent to the network architecture and existing interfaces.
- Applications should have the possibility to run on different MEC platforms provided by different vendors. This portability allows a fast transfer of application between MEC servers in case of shared processing, caching and computing. Portability also provides MEC servers with an ability to optimize resources and virtual appliances without location constraints.
- MEC should inherit as well all privacy and security issues of mobile cloud computing.
- MEC servers should be able to deliver high performance services to mobile users. Edge computing is transparent to mobile users who expect high Quality of Experience. A main challenge for MEC is delivering higher performance while minimizing mobile users' energetic and delay costs and the impact of virtualization.
- MEC servers are co-located with network edge equipment. Any failure or crash in MEC servers should not affect the network functionality nor the connectivity of users to the mobile network.
- Legal considerations are to be taken into account in MEC platforms especially in regard of privacy-restricted information diffusion.

In conclusion, MEC allows base stations to increase their functionalities and deliver services adapted to mobile users' requests and contexts. Users are served directly by the edge of the network, which is characterized by high resources proximity and availability. Proximity, context,

agility and speed can be translated into unique value and revenue generation, and can be exploited by operators and application service providers to create a new value chain that will enter in the ecosystem of new MEC-enabled services and applications [33].

1.3.3 Fog Computing

The Internet of Things (IoT) is an emerging wave of connecting things to the network, creating and consuming huge amounts of data. Connected things are usually part of large systems that collect and analyze data for decision making. Computation and storage resources must be available and sufficient for serving the IoT systems. Finding a location for deploying computing and storage resources is not trivial. We have shown in Section 1.3.2 how resources locations have been changing in cellular networks. What is then the best location, or network architecture to server IoT requirements? Deploying resources in IoT endpoints is not practical since they are designed to be very simple and energy efficient. Consequently, outsourcing resources for IoT computations is a must. Unfortunately, due to IoT applications characteristics, cloud computing (we refer here to remote clouds) is not an efficient solution. Resources requested by IoT applications may be generated by tens of millions of devices over a very wide geographical area. Some applications may be characterized by very low latency, high throughput during short time periods, and prompt decision making based on real-time analytics. IoT end devices themselves are often characterized by a low communication power consumption and short range communications, settings that are required due to energy scarcity. Yannuzzi *et al.* [52] give three main reasons for which cloud computing is not adapted for all of the IoT scenarios and applications.

- Iot platforms may require on-demand high throughput and must support mobility, and even rapid mobility patterns. The particular mobility and fast mobility aspects are weakly supported by remote cloud computing. In the case of MCC, devices in mobility connection with cloud servers are subject to frequent variations of network conditions including service degradation. Moreover, reaching the cloud servers can be very time costly through the WAN. In the case of MEC, connection through macro-cells, which are base stations of wide coverage with a very large number of served devices, is not stable and do not always guarantee high data rates. In the case of smaller coverage base stations (pico-cells and femto-cells), higher data rates are available, but the mobility aspect is not supported due to small coverage area. Mobility in IoT can be associated with various examples such as sensors in moving vehicles e.g. cars and trains.

- An IoT platform must be able to deal with systems that require sensing, analysis, control and actuation. Scenarios may vary and objects might be subject to unreliable connectivity to the cloud. Examples of such scenarios are when objects are placed in locations where communication with the cloud is not possible or too weak, as in pipes, gas sectors, aircrafts, etc. In these cases, IoT systems need computing resources, data processing, and storage space for being able to compute a decision, under latency limitations.

- A platform for IoT must be able to manage large amounts of objects that are widely distributed on large geographical areas. This produces data that require different levels of real-time analytics and data aggregation.

For compensating cloud services non-adaptability in IoT scenarios, Cisco has proposed a new vision called *fog computing* to enable the millions of IoT devices operating at network edge [53]. The fog is a new architecture that extends the cloud to be closer to the user. As in the MEC case, the cloud is brought to the edge of the network. However, fog computing is adapted to IoT

systems where any device with computing, storage, and network connectivity can be a fog node, notably routers, switches, cameras, and base stations, to name a few. Treating IoT data close to where it is collected minimizes latency and does not send out sensitive data [54]. Developers can bring their own applications at the edge of the network, where the newly introduced architecture allows them to run at a proximity to where data is collected. In addition to low latency, fog provides location awareness services and improves QoS. It is well positioned for real time big data analysis and support dense data collection points [55]. As resumed in [56], fog computing is characterized by low latency and location awareness, wide-spread geographical distribution, mobility, very large number of nodes, predominant role of wireless access, real-time applications presence and support, and nodes heterogeneity. It consists of (i) fog nodes that are the closest to the IoT devices and equipped with computing and storage resources used for time-sensitive applications; (ii) fog aggregation nodes that are connected to many fog nodes and where data is sent for analysis and action; (iii) the cloud, which is always connected to the system for any big data analysis and applications with large computational demand and large delays. Data sensitivity in aggregation nodes may be in seconds or minutes. Cloud is also used as long term storage entities for data and historical analysis [54].

The benefits of fog computing have been defined by Cisco [54] [57]. First, fog computing provides IoT systems with data privacy whenever data is collected and analyzed within the far edge of the network without being sent to remote data centers. Being co-deployed with network nodes, security of fog nodes uses the same policy of network nodes security controls and procedures. Moving data analysis in IoT to the edge of the network will enable new applications that can be easily and rapidly integrated in the new architecture, and thus fog computing creates greater business agility and innovation. Finally, fog computing helps conserve network bandwidth by lowering the size of data to travel over the network to the cloud, and substitutes remote processing by local data management.

Fog computing satisfies the requirements of many IoT scenarios including smart grid, smart traffic lights, connected vehicles, wireless sensor and actuator networks, decentralized smart building control, and IoT and cyber-physical systems [56] [55].

1.4 Uplink Traffic in Future Mobile Networks: Pulling the Alarm

1.4.1 Motivation

Cellular networks have always been designed, dimensioned, and deployed based on the downlink (DL) mobile users' demand and traffic patterns. The reason for leveraging downlink traffic was the asymmetry — then true — between uplink (UL) and DL traffic. In other terms, the capacity required in the downlink was much higher than the one required in the uplink. Therefore, designing networks with higher data rates to offer in downlink than in uplink was trivial. More precisely, within early 2G based cellular networks, the traffic load for both UL and DL have been roughly the same. This has also been the case for the very early 3G systems. It is not until the 3.5G and 4G systems that downlink traffic load greatly surpassed uplink requirements [58]. In these systems, with the eruption of IP based networks and high speed access to the Internet through cellular networks, traffic is dominated by downlink. The data explosion in downlink and uplink was asymmetrical. While downlink traffic grew exponentially, uplink traffic was also subject to an increase, however, the traffic demands in both directions were not equal. Mobile users downloaded more than they uploaded. The estimated ratio of uploaded to downloaded data is about 1:7 [59]. Thus, current mobile networks are dimensioned based on the amount of data mobile users are

downloading according to downlink traffic models. As the fastest growing segment of the communication industry, wireless communication, and especially cellular systems, have experienced, and are still experiencing, exponential growth over the last decade. Many new applications, services, and technologies have and will integrate the wireless network. The way mobile users see, use, and exploit mobile networks have changed. Mobile networks are nowadays the provider of unlimited number of heterogeneous services that differ in data requirements. As some are mainly downlink based, others have equal requirements of uplink and downlink traffic, or depend on large amount of data upload, like cloud storage for example. Today, the asymmetry between UL and DL is reduced, and sometimes inverted. These changes evoke a set of questions: What is the impact of network evolution on uplink traffic? Have networks started experiencing uplink traffic explosion? Should networks continue to be designed, planned and dimensioned according to downlink traffic only? What has been done to increase uplink network capacities? These questions are of great importance, especially under the fact that very low attention has been given to UL traffic models comparing to DL. Indeed, uplink traffic lacks of tractable models since it depends on users actions and unplanned interventions that are often less easily accessible and predictable. In contrast to downlink traffic that has been given significant attention, attempts to model the uplink have been limited [60]. With an increasing number of connected devices and mobile subscribers, the integration of cloud enabled technologies in wireless networks, the convergence of IoT systems, the development of M2M and MTC platforms, and many other factors, it is important to understand if and how new communication networks will cope with challenging uplink traffic loads. The idea is not about uplink rising over downlink traffic. We do not assume or consider that uplink traffic overtakes the downlink — although this might be the case in specific scenarios. We only present the uplink as a new important player that should be considered when setting network design and dimensions. Even though there are no precise forecasts on uplink, the traffic pattern change is inevitable. A study by NSN in 2013 showed that the overall UL to DL usage ratio reaches approximatively 1:2.4 [61]. In addition, the Ericsson mobility report of 2012 shows that UL to DL ratio reaches 1:1 for bi-directional applications such as P2P TV, email, and P2P sharing [62]. With the availability of high data rate services, new applications are enabled, and mobile devices energy consumption increases. Cloud technologies, sensor networks, device to device communications and social networking are all growing trends that increase uplink traffic and do not rely solely on downlink traffic. All of these trends introduce applications where mobile users create content and launch actions on the network, which changes the classic DL-based traffic pattern adopted in wireless networks. The research community, aware of the upcoming uplink traffic volume change, is already proposing some solutions in the network for improving uplink capacity.

In the remainder of this section, we present the major factors that contribute to the uplink traffic explosion in the current/future mobile networks. Then we discuss some of the efforts that have already started by the wireless community to improve current networks uplink capacity in order to cope with mobile users' increasing uplink demand.

1.4.2 Why Uplink Traffic is Growing

1.4.2.1 Increase in Number of Mobile Subscribers and Devices

The number of mobile subscriptions and mobile devices has been constantly growing since the first deployments of cellular mobile networks. From 6.4 billion mobile subscriptions in 2012 to 7.2 billion in early 2015, the ever-increasing index is to reach 9.2 billion by 2020 according to latest mobility reports [6]. Mobile broadband that was accounted for 2.9 billion out of the 7.1 billion subscriptions will grow its share to occupy 7.7 billion out of the 9.2 billion subscriptions

in 2020, which is around 85% of all subscriptions. As the number of fixed broadband and the number of related devices such as mobile PCs and mobile routers will have very low growth, and the number of total mobile subscribers and subscriptions will increase linearly, the number of mobile subscriptions will increase exponentially. Smartphones, which already are the main mobile equipment (2.6 out of 2.9 billion), are expected to double in number by 2020. Mobile broadband will be accessible to everyone and mobile devices will continue to outnumber the earth population. By 2020, mobile phones will be in possession of 90% of humans over 6 years old. The growth of mobile devices and users showed in numbers gives an idea of how data traffic (in both uplink and downlink) could increase.

1.4.2.2 Evolution of Cellular Networks

Since wireless Internet, wireless generations adopting new technologies for increasing system capacity have been designed and deployed. The increasing users' traffic demand required a network evolution to cope with constant changes. However, for all consecutive technologies and wireless generations, downlink data rate far exceeded uplink. Due to possible technical challenge and asymmetry in traffic demand, mobile networks were always dimensioned to assure higher DL capacity. Table 1.1 shows the difference in up and downstream data rates among technologies. Note that the table shows advertised peak data rates, which are usually higher than nominal achieved rates.

Table 1.1: Cloud architecture evolutions comparison

Technology	**Generation**	**Downstream** (Mbits/sec)	**Upstream** (Mbits/sec)
EDGE	2.5G	1.6	0.5
EVDO (Rev A)	3G	2.45-3.1	0.15 - 1.8
HSPA	3G	0.384-14.4	0.384-5.76
HSPA+	3.5G	21-678	5.8-168
LTE	4G	100-300	50-75
LTE-A	4G	1000	500

Evolution of wireless networks and users' traffic demand are in perpetual evolution and growth, one implying the other. Indeed, wireless network evolves to *"give more"* for mobile users and cope with their increasing traffic. At the same time, when offered more capacity, mobile users would like to *"do more"* with their mobile equipment through the wireless network. Numbers show that the proliferation of new wireless generations offering higher service quality attracts mobile users. Since the introduction of HSPA and then LTE, the number of mobile users continues to grow strongly. In the third quarter of 2012 HSPA and LTE subscriptions increased by 13 and 65 million respectively. As for GSM/EDGE it attracted then 20 million new subscriptions. With LTE proliferation in the market, the numbers in the first quarter of 2015 are as follows: 105 million additions for LTE, 60 million for HSPA, and a decline of 30 million for GSM/EDGE. These numbers and Figure 1.13 show how the market follows the offer of new technologies and increasing service quality. LTE will have, alone, 3.7 billion subscriptions by 2020. In conclusion, the number of mobile users and the evolution of cellular networks are joint in an escalating increase relationship; where the increase of the first requires improvement in cellular networks, which re-attracts more mobile users to subscribe.

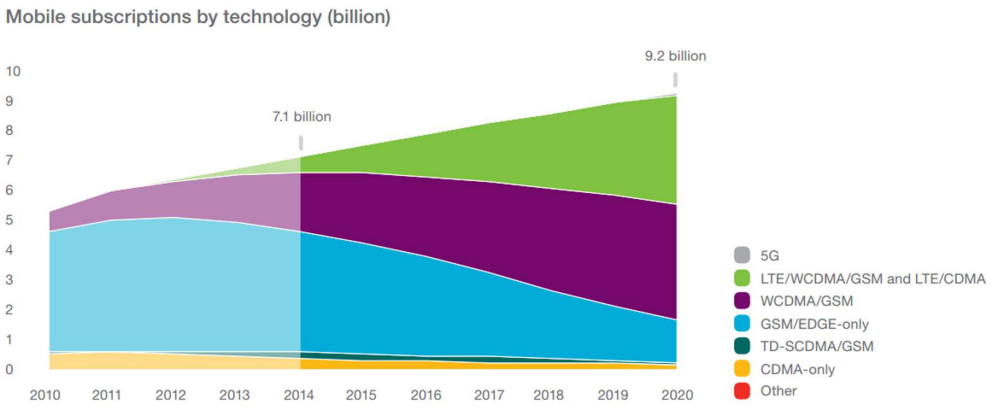


Figure 1.13: Mobile subscriptions by technology [6]

1.4.2.3 Emergence of Cloud Technologies and Dense Heterogeneous Networks

Cloud technologies are progressively but rapidly integrating wireless networks. Cloud radio access network, remote cloud computing, cloudlets, and mobile edge cloud, are all new technologies and architectures in which the cloud concept integrated wireless networks (see Section 1.3). Cloud technologies in mobile networks consist on delegating computing, storage, data processing, and other resources consuming functionalities to a computing entity instead of performing the tasks on the mobile devices. Cloud in wireless networks can take different forms. It can be a centralized remote server pool, a nearby cloudlet, or an edge computing platform. Aside from cloud computing, cloud can be used as a remote storage location. As mobile equipments in general suffer from lack of resources of computing and storage, mobile users are more and more relying on outsourcing required storage and computation capacities. With cloud storage, mobile users can take photos or record videos with their mobile devices and directly upload them for saving on the cloud instead of their devices. In such applications, uplink is as important as downlink and thus should be taken into account in network dimensioning.

Another emerging deployment technology in wireless networks is Heterogeneous networks (HetNets). All mobile users are not served by the same type of base stations. Along with classic large coverage macro-cells, cellular networks are being intensively deployed with pico-cells, relays, and femtocells. One of the main motivations and interests of heterogeneous networks is offloading heavily loaded macro-cells. Users in reach of a femtocell, for example, will communicate with the latter instead of a congested macro-cell. As femtocells are deployed at closer distances from mobile users, communication channels between femtocells and mobile users are very often characterized by a better signal to noise ratio. Due to the lack of tractable models, the impact of such offloading on the uplink performance is not well understood [60].

Uplink traffic modeling has not gained the same attention as downlink. Both directions differ fundamentally in access modes, heterogeneity of transmitters, and resources management. The invasion of wireless networks by cloud enabled heterogeneous network certainly has an effect on traffic patterns especially in uplink, since new offloading opportunities are available to consumers. With the adoption of offloading computation and the concept of virtual machines (VM) and enable applications such as videoconferencing in enterprises and improved network mobility support, upload speeds become critical against users' experience quality and content efficient delivery to the cloud. With the development of cloud technologies, upload speed and capacity will continue to have an important impact.

1.4.2.4 New Applications and Services Ecosystem

Cloud based wireless networks are the next breakout of the wireless communication. Cloud is integrating many functionalities of the wireless networks and increasing their capabilities. Whether a remote cloud or an edge cloud, the cloud unlocks a whole new ecosystem of services and applications. Application developers have now the door open to new types of applications that can be run on the cloud and that were not adapted before to the mobile concept due to heavy resources requirements. Furthermore, cloud and services providers work on increasing their infrastructure ability and performance through improving availability and reliability: An evolving ecosystem that will push forward the cloud based offer and demand, and thus create higher cloud related traffic requirements. Among the applications that are now compatible with the mobile network, we distinguish different types of traffic requirements. Some applications require very high downlink and/or uplink traffic with varying latency constraints. Applications that comply with downlink based networks include streaming basic video and music and web browsing, where upload requirements are relatively low. Streaming relies basically on high downlink traffic, as for web browsing it has in general lower traffic requirements. However, numerous applications do not comply with that model. Many applications require roughly the same amount of upload and download such as web conferencing (cloud-based), video conferencing, tele-medicine, virtual office and connected vehicles safety applications [63]. Others, on the contrary, require more traffic upload than download such as web electronic health records, virus scanning, face recognition, cloud storage, and aggregated data analysis. Hence, the heterogeneity in new services and applications has non-negligible impact on traffic patterns and on the importance of uplink. The diversity of services offered through the Internet requires a management of network capacities in order to avoid both functional and economical harm to wireless communication infrastructure and businesses and their customers.

1.4.2.5 Crowded Networks Scenarios

Mobile networks are designed based on peak network traffic and the ability to serve in peak hours. This has led into excessive energy consumption. Several solutions were proposed for this problem such as base station sleeping. Furthermore heterogeneous networks deployment helps by offloading traffic from congested macro-cells onto smaller base stations. Now that solutions exist, the network should be dimensioned to keep its efficiency in peak data traffic scenarios. Peak traffic does not only concern downlink, uplink traffic is also subject to peak demands. Crowded scenarios are the best example for such situations. We take the example of a football stadium were thousands of people are gathered to watch a game. In such situations, mobile users share their experience through social networks, texting or talking. They post photos and videos during matches. A study by Ericsson [6] about the FIFA 2014 football games showed that social networking and texting were used during the matches and traffic peaked at half time. Ensuring a good user experience in such scenarios is a challenge to operators. Network planning and optimization are necessary. What is important to notice in crowded scenarios is the footprint of uplink traffic. According to the same study, the ratio of uplink in total data traffic was as high as 50% during the final game of the world cup. The normal ratio in the same location is between 12 and 17 %. The increase in uplink traffic is clearly non negligible and should be taken into account during network planning and dimensioning. The study showed that 61% posted or sent pictures via the Internet, and only 25% used the Internet to find and download content related to the world cup. The numbers also showed that more users posted videos (33%) than watched videos (18%) through the Internet. Video uploading data usage is quite high especially that smartphones and tablets camera technol-

ogy is quite advanced and 4K video enabled. Furthermore, in many sports events, uplink traffic surpasses downlink in some time windows. With the accessibility of high data rate services such as 3G and 4G, mobile users will be more active uploading data and using social networking. 4G users, which have higher data rates, are more active than 3G users. This proves what was stated in Section 1.4.2.2 that users with higher services will want to do more. Even though 30% of data traffic was handled by 4G networks, 4G users consumed 70% more data than 3G users.

1.4.2.6 Sensor and MTC Networks

The evolution of sensor networks and Machine Type Communications (MTC) has been more than evident in the last decade. Mobile devices, sensors, and all types of objects already or soon will be equipped with sensors and RF circuits in order to integrate the wireless communication network. The Internet of Things is a well-known application of such networks that is creating a new trend and imposing a breakthrough in network management. In most wireless sensor networks scenarios, data is aggregated from end equipment (sensors) into a gateway that communicates with the network infrastructure. As for machine to machine (M2M) communications that allow devices of the same type to intercommunicate is also requiring an increasing role in the wireless network. Any device to device (D2D) communications can be established through different scenarios, where control link can be managed by end devices or the network, as discussed in Section 1.2.2.2. In D2D communications, especially scenarios where control is done by the network, uplink traffic is at least equal to downlink. Figure 1.14 shows a simplified level architecture of how the M2M system will be connected to wireless networks. The capillary network, which represents the set of communicating machines, aggregates data in a M2M gateway that is connected to the network and may use cellular communication. The convergence of sensor networks and cellular has been

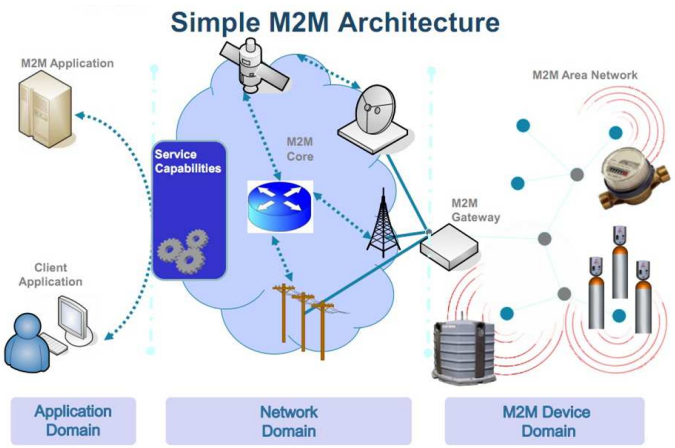


Figure 1.14: High level simplified M2M architecture

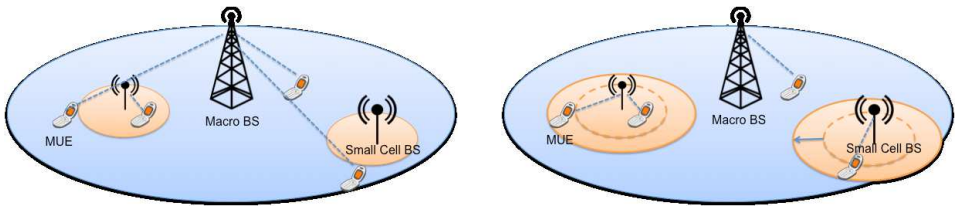
studied in literature. The European FP7 SENSEI project [64] focuses on the integration of WSN (Wireless Sensor Networks) and actuator networks. With the expected increasing deployment of IoT devices and services, the traffic generated by these devices may change current traffic patterns. In fact, regarding the number of connected IoT devices, an estimated 50 billion “things” will be connected to the Internet by 2020 according to Cisco [54]. As for traffic patterns, current traffic models do not take into account traffic generated by smart devices. And since mobile networks are designed according to those patterns, they may not be adapted to such applications that are mostly uplink traffic generating. It is then necessary to understand how smart devices will affect network

traffic and include it in network optimization and dimensioning.

1.4.3 Uplink Improvement Related Work

1.4.3.1 Range Extension in Heterogeneous Networks

Coverage Range Extension (CRE) in heterogeneous networks is a technique that can help increasing the uplink/downlink fairness. In an area covered by both macro-cell and a small-cell, the MUE/ Base Station (BS) association is based on the downlink received signal power only. And since small cells are characterized by a smaller transmit power than macro-cells, and are randomly deployed, they are expected to have large areas with low signal to interference (SIR) conditions [65]. In the uplink, the strength of the signal does not depend on the BS transmit power. It depends on the mobile device transmit power and the received signal power at base stations depends on the channel gain. This results in boundaries mismatch between uplink and downlink handover in heterogeneous networks. And since small cell coverage ranges are smaller than those of macro-cells, we notice unfair distributions of data rates between macro and small cells due to different loadings of connected users. The proposed solution is to balance the load between macro and small base stations by expanding the range of small cells (see Figure 1.15). This is achieved by associating users to base stations based on path loss instead of received signal power. This will be in favor of uplink network performance since minimum path loss association maximizes uplink coverage rate [60]. Nevertheless, range expansion lead to high interference levels in the downlink which imposes using interference coordination techniques.



(a) Base stations coverage limits without CRE (b) Base stations coverage limits with CRE

Figure 1.15: Cell Range Expansion (CRE) impact of base stations footprint

1.4.3.2 Downlink and Uplink Decoupling

From the first generation to 4G, downlink and uplink of cellular networks have been coupled. Indeed, mobile users' equipments have been connected with the same base station in both uplink and downlink directions. As mentioned earlier, the best base station and user equipment couple is not necessarily the same for both directions. While for uplink it is best to connect UEs with the base station with the highest received signal power, for downlink, the best association is the one that minimizes path loss. Adopting a downlink centric association negatively affects load balancing in heterogeneous networks as well as uplink overall performance. Nevertheless, adopting an uplink centric association through cell range expansion creates interference problems for uplink users. As a solution, uplink and downlink association decoupling has been proposed in order to optimize communications in both directions [66] [58]. Association decoupling is expected to increase uplink SNR and reduce transmit power, improve uplink interference conditions, improve uplink data rate, allow distribution of users among macro and small cells, and achieve more efficient resources utilization and uplink rates. This technique indeed proved to achieve up to 200%

improvement in the 5^{th} percentile uplink throughput in a simulation based on a live Vodafone LTE test network deployment in London [66]. Nonetheless, the concept of uplink and downlink decoupling is considered as one of the components of future cellular networks [67] [68]. However, this technique requires changes in system design since it needs mechanisms to allow acknowledgment process between serving base stations for uplink and downlink, strong synchronization, and data connectivity between base stations.

1.4.3.3 Uplink CoMP Techniques

Uplink Coordinated Multi Point (CoMP) is a new technology introduced with the LTE systems, which consists on jointly processing signals that are received at different antennas and/or base stations. It is the uplink analogy of CoMP where a single user is served by more than one base station (see Figure 1.16). In uplink CoMP, users' signals are captured by more than a base station and pro-

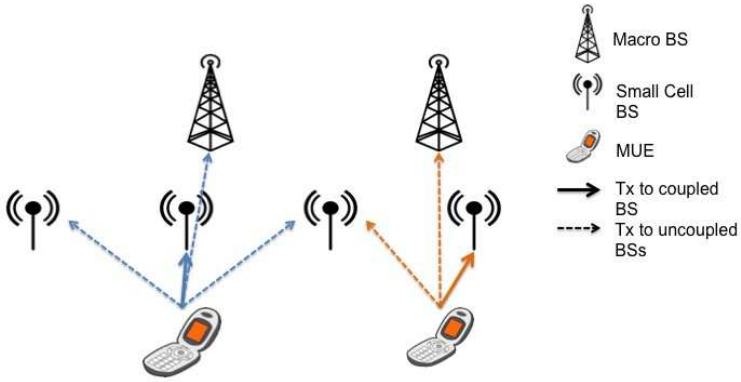


Figure 1.16: Uplink CoMP usecase example

cessed jointly. Uplink CoMP can be deployed through three different scenarios: Intra-cell CoMP, Inter-cell CoMP, and between macro and small cells in heterogeneous networks. Inter-site CoMP is easily deployed since all signals information are inside one cell. Intra-site and heterogeneous CoMP require however low delay high capacity backhaul support between base stations. We note that uplink CoMP is transparent to mobile users in the sense that mobile equipment do not need to be aware of the base stations receiving their signal. Therefore, uplink CoMP does not change the association complexity on the mobile equipment side. By jointly processing received signals at different base stations, uplink CoMP results in uplink improvement. Uplink CoMP achieves uplink gain from both macro diversity reception and from enabling uplink/downlink decoupling in heterogeneous networks. Uplink perceived capacity is improved in high interference and poor coverage conditions. Important gains can be achieved especially in locations where uplink and downlink optimal associations are not the same, i.e. in locations where the most powerful received signals and the minimum channel path loss are not of the same base station. In a full scale field trial in LTE network [69] uplink CoMP proved to achieve 3 Mbps improvement in uplink throughput, and 100% throughput gain if coupled with downlink/uplink decoupling.

1.5 Conclusion

The number of connected devices to the wireless network, both mobile users portable devices (smartphones, tablets), the number wireless connected objects (sensors, machines), and the generated traffic are exploding. The traditional network architecture limits its capabilities, especially in computation, and is rapidly found un-adapted to such changes and increase of requirements and demands. This has imposed a series of changes in the network architecture for satisfying the parallel exponential increase in data traffic and computation requirements. In this chapter, we have presented a description of the future generation of wireless networks, 5G, which requirements are set to meet the changing network demand. We presented the characteristics, enabling technologies, and requirements of the 5G future networks. Then, we focused on the cloud technologies integration in mobile networks. Cloud computing has been first defined and its characteristics, enablers and deployment scenarios have been discussed. Moreover, an overview over the architecture evolutions involving cloud functionalities has been presented. Starting with centralizing the radio access network as a first cloud aspect operation in the mobile network, we discussed the advantages of having network required computational tasks to be performed on a cloud platform. Mobile cloud computing was the first cloud aspect allowing mobile users to compute their own tasks on the cloud. The major MCC platforms, remote clouds, cloudlets, and edge computing have been presented with detailed architectures and analysis. Finally, an overview on the cloud emergence in the world of the Internet of Things detailed edge computing functionalities in a parallel system to cellular networks. Table 1.2 shows a brief comparison of the different cloud systems characteristics.

Finally, we discussed the challenge facing wireless networks in uplink communications. Inscriptions number increase, network development, cloud technologies, cloud computing enabled ecosystem, and convergence of sensor and actuators networks are the main factors that we discussed and showed how they can influence uplink traffic. We then discussed some research works and studies that have been already proposed by the communication society and can help improve uplink network resources management and increase uplink capacity in current and future networks. However, there are still no clear uplink traffic patterns that can validate if the existing efforts are enough for coping with the upcoming challenge.

Table 1.2: Cloud architecture evolutions comparison

	Remote Servers	**Cloudlets**	**MEC**	**Fog**
Latency	Uncontrol-lable (often high)	Low	Low	Low
Availability	High	Moderate	High	High
Reliability	Moderate	High	High	High
Managed by	Cloud services providers	end users	network operators	End users and ISPs
Access to network information	Yes	No	Yes	No
Privacy and security	Low	High	High	High
Mobility support	Moderate	Low	Moderate	High
Proximity to users	Low	Moderate	High	High
Number of hops to be reached	$>> 1$	1	1	1 or more
Computation tasks source	End users	End users	End users	Connected things

Chapter 2

Edge Cloud Cluster Computing: Challenges and Trade-offs

2.1 Introduction

Cloud functionalities are integrating wireless networks, which empowers mobile devices with remotely accessible computational and storage resources. In this thesis, we consider cloud-empowered Het-Nets with added capacities: processing and storage. MUEs have indeed the possibility, in addition to communication, to offload processing and computational requests for an execution on the cloud, through the wireless network. We adopt a MEC architecture, which moves cloud capabilities to the edge of the network, by moving computational resources, and thus computation efforts, from the Internet to the very edge of the network, characterized by its proximity to mobile users. We specifically adopt the architecture proposed in TROPIC [70], in which small cells (SCs) are considered equipped with additional computational and storage capacities. Small cells are used, not only for delivering communication services to MUEs, but also for computing MUEs offloaded computational requests, and storing mobile users' offloaded data. Each MUE is associated to a serving small cell (SSC). SSCs receive offloaded computational requests from connected MUEs, execute the requests, and send back computation results to MUEs. A Small Cell Manager (SCM) entity manages the use, performance, and delivery of cloud services. The cloud-enhanced small cells and SCM form a *small-cell cloud* in which users' computation requests are executed. While SSCs can offer local edge computing services, we propose to extend their capabilities by allowing SCs to cooperate through a small cells cluster (SCC), in which several small cells contribute in the computation of MUEs offloaded computational requests. Even if SSCs capabilities can serve MUEs offloaded requests, a SCC can enhance the local edge cloud capabilities. For example, distributing the computation load one more than one computation entity can speed up the computation. SCCs form a local edge cloud platform at high proximity to mobile users. The introduction of local edge cloud to wireless networks adds extra resources for system operation optimization. In addition to communication resources, computational and storage resources should be optimized.

2.1.1 Contribution

In the first part of this chapter, we detail the adopted small cell cloud architecture. We describe the mobile computation offloading and small cells clustering scenarios. We discuss about the limitations of the cluster-based edge cloud computing, and deduce the optimization degrees of freedom that make the basis of the approaches proposed in this thesis. The set of optimization variables in small cell clustering for computational purposes, are involved in a set of trade-offs. We present an overview over these trade-offs. We start by a preliminary state of the art on trade-offs in heterogeneous networks, before focusing on the cluster edge cloud architecture trade-offs. Finally, we present an in-depth study on the impact of intra-cluster communication backhauling on edge cloud computing.

A part of this chapter is based on the conference paper C2.

2.2 System Model

2.2.1 Small Cells Edge Computing

Using network edge entities, as indoor/outdoor small cells and relays for example, offers some advantages with respect to their counterparts, namely macro base stations or WiFi access points.

The advantages with respect to macro-cell base stations are:

- proximity offered by small cells makes possible energy saving at the mobile handsets, with consequent increase of mobile users equipment battery lifetime.
- the number of users concurrently served by a small cell is much smaller than the number of users served by a macro-cell. This simplifies the setup of computational clusters, and the computation offloading process between mobile users and serving small cells.
- short distance between mobile users and small cells enables the development of proximity-based services, such as home security control, which are not possible in a macro-cells based cloud.

Advantages of using small cells with respect to WiFi nodes are:

- femtocells simplify hand-off as they are fully compliant with the mobile standard
- femtocells provide QoS guarantee and better interference management than WiFi [71].
- WiFi and cellular technologies (3G/4G/5G) are standalone networks, and their integration is considered cumbersome from the operator's point of view. Insufficient authentication, access control techniques, and integration with cellular core network are the concerns of operators in using WiFi as a cloud solution.

2.2.2 Computation offloading

MUEs are limited in computational capacities, storage space, and energy (battery lifetime). Computational requests offloading to the cloud extends MUEs capabilities, and allow mobile users to have access to higher amount of computational and storage resources. MUEs have the possibility of either executing computational tasks locally at the MUEs using the handsets resources, or offloading computational tasks for execution on the cloud. Computation offloading requires sending computational requests to the cloud, via the SSC, in our case. In addition to increasing MUEs computational and storage capabilities, mobile handsets energy saving is an important advantage of computation offloading. However, computation offloading may not always be beneficial from an energy consumption perspective. Energy costs for local computation on MUEs is equal to the tasks execution energy consumption, related to the computation size and the MUE processor characteristics such as the processor speed and its energy consumption per CPU cycle. Mobile energy consumption in case of computation offloading is equal to the communication cost between MUEs and SSCs. This cost depends on the amount of data to send to SSCs, and on the communication channel quality. Moreover, despite the assumption that cloud offer much higher capacity than mobile devices, widely adopted in literature, executing tasks on the cloud may consume more time than local computation. In fact, computation offloading is based on sending the computational requests to the cloud, executing the computation on the cloud, and sending back the computation results to mobile users. These steps impose both communication and computation delays. The gain in computation time, due to higher computational capacities at the cloud side, should be high enough to compensate the additional communication delay.

Computational requests of mobile users are considered to be represented by an instruction block (CPU cycles) to execute within a latency constraint. The computation results should be delivered to mobile users without any violation of these constraints. Hence, an offloading decision strategy is required in order to efficiently compute an offloading decision that guarantees the respect of energy, time, and resources constraints. This decision is subject to various system, tasks, MUE, and cloud characteristics, and to several trade-offs.

2.2.3 Computation Small Cells Cluster

Small-cell Cloud serves MUEs computational requests by executing them using small cell computational resources in the local edge of the network. We propose forming a local edge cloud that extends *small-cell cloud* capabilities by setting up small cells clusters for computational purposes. The idea is to distribute MUEs offloaded computational load on a set of small cell referred to as the Small Cells Cluster (SCC). SCC shifts the paradigm from local edge computing in a small cell to local edge cloud where computational tasks are distributed among several small cells. SCC includes the SSC that receives the computation request from MUE, and a set of neighbor small cells that help execute the offloaded tasks, referred to as Helper Small Cells (HSCs). We make the assumption of possible parallelization of computational requests, in order to be able to find the optimal distribution of computational load on the cluster small cells. SCC is a distributed computing platform that is set up for improving *small-cell cloud* energy efficiency, service latency, and/or power consumption. Various reasons for a SSC setting up a SCC instead of executing the tasks itself are possible. For example, when SSC resources, such as computational capacity or storage space, are lower than the computational request requirements, it cannot execute the tasks, at least without violating requirements constraints. In this case, SSC sets up a SCC for increasing its capabilities by delegating computations to HSCs. Moreover, even if SSC resources are sufficient for executing offloaded tasks, SSC can set up a cluster for enhancing its performance. From a service latency perspective, distributing computational load on several computing entities decreases the service latency through parallelization of computation. Setting up a SCC, is requested by the SSC, and managed by the SCM. An additional functionality of SCM is to manage intra-cluster resource allocation.

Setting up SCC for computing an offloaded computational task is not straightforward. The cluster set up is constrained by the resources availability and demanded requirements. The main conditions and constraints to be respected in a cluster set up are the following:

- **Complete tasks execution**
  Distributed computing of offloaded computational tasks in a small cell cluster, should guarantee the execution of the totality of the task. The distributed computational load should be equal to the total computational size, and all distributed load should be computed.

- **Respect tasks requirements**
  Computational requests consist on executing a number CPU cycles in a fixed time limit, referred to as latency constraint. Latency constraints impose a minimum computational capacity for a task execution *in time*. Furthermore, computational tasks may also have memory and storage space requirements, which should be respected as well. The SCC in which computational tasks are executed should guarantee, at least, the minimum requirements of the computational tasks in terms of computational, storage, and any other requirements. Note that service latency in SCC is defined by both communication (between computing nodes) and computation delays.

- **Small cells resources availability**
  Distributing computational load on several small cells requires allocating computational resources on each of them. Small cells have larger computational capacities than MUEs; however, these resources are limited. We refer to a small cell or computational node as *overloaded* if its available computational capacity is not sufficient for computing the accorded computational load. Load distribution and computational resources allocation should then be jointly orchestrated for preventing small cells computational *overload*.

- **Communication power budget**
  In a SCC, small cells exchange computational data. First, after at the cluster set up, SSC sends computational load and requirements to HSCs, that sends back computational results to SSC after their execution. The data exchange between SCC nodes is referred to as intra-cluster communication, and is often subject to power budget limitations. Power budgets are of high importance especially in the case of wireless intra-cluster communication. Intra-cluster communication resources should be allocated in order to respect imposed communication power budgets.

Respecting the constraints above is crucial for guaranteeing in-time service delivery for mobile users. In this thesis, we consider mobile users' QoE as a cluster performance evaluation criterion. QoE is a subjective measure of mobile users' satisfaction with delivered services. We assume that mobile users are 'satisfied', and achieve desired QoE, when demanded service is delivered without any violation of imposed latency constraints. In general, consumers are willing to wait for service delivery for a reasonable time depending on the application. A small extra waiting time is often acceptable, however, when service delay does not meet consumers' expectations, QoE is judged as insufficient.

In order to achieve high QoE while respecting the imposed constraints and limitations, the cluster set up requires a joint optimization of computation and communication resources. In addition to this joint optimization, efficient load distribution is required. These three optimization sets of variable define the service latency, allocated computational resources, and communication power consumption. We now detail each of the optimization variable sets and the related model and assumptions that we adopt in this thesis.

Intra-cluster communication

We assume that small cells communicate through in-band wireless links that support Line of Sight (LoS) as well as NLoS (No Line of Sight) communication. We adopt wireless intra-cluster communication, even though it can impose larger communication delays (comparing to wired fiber, for example). We choose wireless intra-cluster communication for its scalability and low deployment cost, as no wired connection are required to allow small cells or clusters nodes to communicate. Most importantly, wireless transmission raises the challenge of communication resource allocation for intra-cluster communication. Furthermore, considering wireless communication, several types of nodes (femtocells, pico-cells, relays) could be considered as part of the cluster through plug-and-play or ad-hoc scenarios.
Channel conditions and maximum link capacities between each pair of small cells are known to the SCM. In other words, we consider that the SCM has full Channel State Information (CSI) knowledge. The SCM adapts the transmission rate by tuning the transmission power over each wireless link joining the SSC with a HSC. The transmit power is upper bounded by a maximal power budget, and thus, the rate is also bounded by the maximum Shannon capacity. We consider that the SSC communicates with several HSCs, by adopting orthogonal simultaneous transmissions through orthogonal frequency division multiplexing. This decreases the transmission data rate over each link between small cell and helper small cells, but mitigates the intra-cluster interference problem. Nevertheless, through our simulation channel models, we increase the noise value as an implicit way to take into account a fixed interference level. In all simulations and numerical evaluations presented in this thesis, we consider femto cells edge cloud nodes. However, the formulation of the proposed solutions can be generalized, and applied to any type of wireless communicating node by adapting the simulations parameters.

Computational resources

The SCM also has the functionality of allocating computational capacities at the clusters nodes. Every small cell is characterized by a total computational capacity, and a utilization ratio. The utilization ratio determines the ratio of momentary allocated computational resources. Network small cells report their utilization ratio to the SCM. Sending this information is done either periodically, upon a request of the SCM, or at every variation of the utilization ratio. The SCM allocates computational resources for tasks execution, taking into account resources availability. Resource allocation is subject to a *computing outage* constraint. Small cells should not be overloaded, i.e. allocated computational resources must not be greater than available resources. Otherwise, computational tasks cannot all be executed, and the cell is in computing outage.

Load Balancing

Cluster nodes are connected through limited capacity backhaul links, which differ in capacity and utilization rate. Furthermore, each of the cluster nodes is characterized by a total computational capacity and a utilization ratio. Computational tasks are distributed among the cluster nodes. Computational load in the computing cluster should be efficiently distributed for guaranteeing the execution of the tasks while meeting the mobile users' expected QoE. However, load balancing can follow various strategies. For example, from a latency or computational resources availability aware perspective, the *best* distribution uses all available small cells. Distributing the load on a large number of small cells, results in smaller computational load blocks (CPU cycles) to compute at each small cell. Hence, a lower overall computation latency is perceived. In addition, large clusters require smaller amount of computational resources at each small cells, which increases resource availability for future requests. On the other hand, reducing the number of computing small cells allows to achieve higher system energy efficiency, but can lead to higher computational delay, since the distributed instruction blocks on the cluster nodes are larger.

In order to balance all of these different aspects, load distribution strategy has to take into consideration small cells, computational requests, and communication channels characteristics. A joint algorithm that considers the system radio conditions and computational load of cells must be designed in order to respect all the imposed constraints (available computational capacity, maximum latency tolerance, and maximum communication capacity). In general, splitting an executable task into instructions is not easy. An application can be more easily split into modules (Java modules for example), which provides more coarse granularity than splitting over CPU cycles. Throughout the work of this thesis, and the proposed optimization procedures we assume very high granularity, and we work using task splitting over CPU cycles or instructions. We note that the relationship between the number of CPU cycles and the number of instructions depend on the type of the instructions in question. This assumption was adopted since the aim of this thesis is to find a solution for resource allocation in small cell clouds rather than to distribute computational tasks from an information technology point of view. The applications parallelization is however not a far reality. Many applications can already be distributed with very loose constraints, such as virus scanning for example.

Figure 2.1 represents a small cell cluster architecture showing the MUE that offloads a computational request to its serving small cell. The latter reports the request to the small cell manger (SCM) that already has information about the small cells status in the network, through the continuous status report of small cells to the SCM. Note that SSCs do not send the request in itself but only necessary information: computation size, latency constraints, and application requirements. The SCM computes cluster parameters: it allocates communication and computation resources

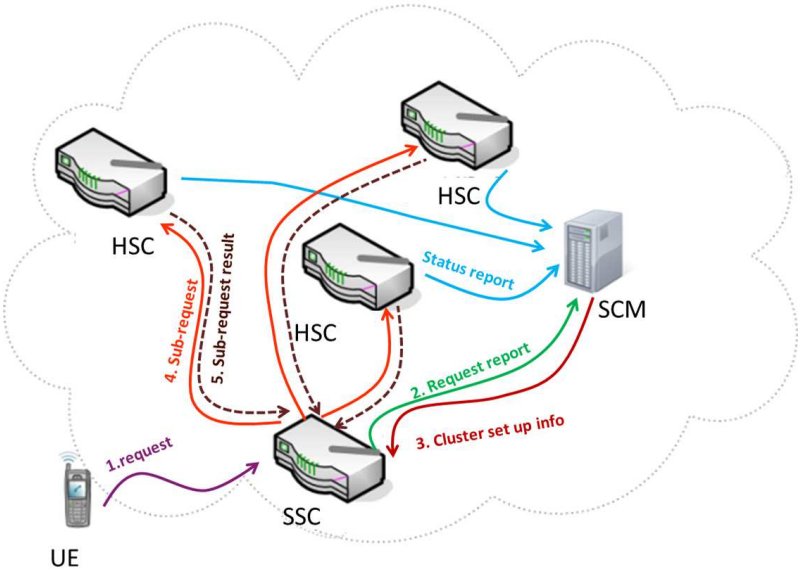


Figure 2.1: Small cell cloud basic architecture and process principles

and distributes the load on cluster nodes. The cluster optimized parameters are then sent back to SSC, which distributes the load on the cluster HSCs. After being computed by HSCs, sub-requests results are sent back to the SSC, that sends the computation result to the mobile user.

2.3 Preliminary on Communication Trade-offs in Heterogeneous Networks

Heterogeneous networks witness the co-existence of at least two-tiers. HetNets are proposed as a solution for increasing wireless network performance by extending the network capacities, but also for improving wireless networks energy efficiency. Improving the communication service quality in HetNets is subject to various trade-offs. in this section, we elaborate some trade-offs that has been studied in literature.

- **Energy efficiency vs system performance**
  One of the main issues of HetNets is the trade-off between energy efficiency and system performance degraded by high interference levels. In a two-tier network where macrocells and femtocells co-exist, communication traffic is offloaded from macro to femtocells. Femtocells, characterized by smaller distance from mobile users, use lower transmit power and deliver higher throughput than macrocells, especially when serving MUEs at macro cell edge. Increasing the density of small cells in the network increases the energy efficiency. With more deployed small cells, MUEs are connected to closer small cells, and thus transmit powers are reduced, and energy consumption is decreased. However, this gain in energy efficiency comes at the cost of higher interference levels with macro users. As small cells deployment can significantly improve energy efficiency, there is a trade-off with macrocell system performance in terms of throughput or spectral efficiency. This trade-off has been studied by Cao *et al.* [72] and Chan *et al.* [73]. Cao shows the trade-off between energy efficiency and macrocell performance degradation with respect to femtocells density in a single macrocell scenario. The study shows that there is an 8% degradation of macrocell

performance, and an energy consumption gain up to 100 times lower, when 80 small cells are deployed within one macrocell. Chen formulates both energy and spectral efficiency for an Additive White Gaussian Noise (AWGN) channel for a point-to-point transmission. The trade-off formula is derived using the Shannon's capacity formula, as follows:

$$\eta_{EE} = \frac{\eta_{SE}}{(2^{\eta_{SE}} - 1)N_0} \tag{2.1}$$

where η_{SE}, η_{EE}, and N_0 represent the spectral efficiency, energy efficiency, and power spectral density of AWGN, respectively. The trade-off curve between EE and SE is monotonically decreasing in this case. However, Chen states that in practical network situations, the trade-off curve will turn into a bell shape after taking into account the effect of resource allocation mechanisms, coding and modulation schemes and transmission channels characteristics.

- **MUE energy efficiency vs system energy efficiency**
  With the dense deployment of small cells, MUE will be associated to closer base stations, resulting in a transmit power reduction, and higher energy efficiency at the mobile side. However, with denser deployment of small cells, some base stations will be activated for serving a very low number of users. Small cells power consumption is mainly due to its activation, and slightly increases with the cell load (see Figure 2.7). Therefore, MUE EE increases for higher small cells density, whereas system EE decreases. This trade-off between MUE and system EE is exploited to propose energy aware MUEs and base stations association schemes [74].

- **Wired and wireless backhauling trade-offs**
  Backhaul is considered as one of the bottlenecks in heterogeneous wireless networks [75]. Backhaul costs and delays affect network performance. We discuss two trade-offs between wired and wireless backhaul for HetNets.

  - **Flexibility vs reliability** Wired and wireless backhaul have each their advantages, which creates trade-offs between the two backhaul types. Wired backhaul has the advantage of higher reliability, data rates, and availability. In addition, throughput is not subject to transmission channel changes or environmental fading. Wired backhaul performance depends on the used technology (optical fiber, xDSL, etc.), the material (fiber, copper), and traffic congestion. However, a major bottleneck of wired backhaul is deployment cost and complexity. Small cells can be deployed in areas or locations where it would be hard to deploy wired backhaul. As for wireless backhaul, various technologies can be used, such as, in band wireless, microwave, or millimeter wave. The choice of wireless technology depends on the deployment scenario and the availability of spectrum. In general, wireless technologies offer higher flexibility in dense deployment scenarios, since no wired connection need to be installed. However, flexibility is at the cost of reliability. Wireless communication is subject to communication channel conditions and variations. Wireless backhaul has lower availability and flexibility than wired network. In addition, higher transmission delays incur due to reception failure, and retransmissions.
  - **Small cells density vs communication latency** As wired backhaul technologies can achieve higher data rates comparing to wireless backhaul, the communication delay witnessed in both cases depends on small cell deployment density. A study by Chen *et al.* evaluates the effect of both wired and wireless backhaul on HetNets performance in

terms of service delay. The authors show that for wireless backhaul, latency decreases with the increase of small cells density. However, the latency gain saturates beyond a certain density, and thus, small cells deployment becomes less cost-effective. As for wired backhaul, it is shown that there is an optimal small cells density, beyond which, wired backhaul performance degrades significantly. Based on the studies by Chen D. *et al.*, we sketch in Figure 2.2 the variation of system performance and deployment cost efficiency with respect to small cells deployment density [76].

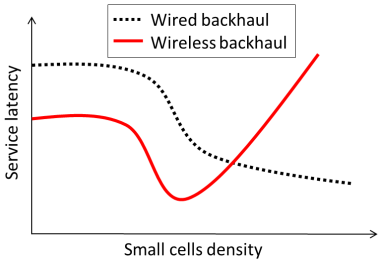


Figure 2.2: Small cells density vs service latency for wired and wireless backhaul

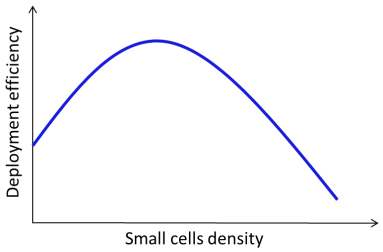


Figure 2.3: Small cells density vs deployment efficiency

- **Energy efficiency vs Deployment efficiency** Deployment efficiency is measured as system throughput per unit of deployment costs including both CAPEX and OPEX. In network design, deployment and energy efficiency are in trade-off. While increasing the number of deployed small cells, energy efficiency increases due to reduction of transmit powers. However, from a deployment efficiency perspective, with less small cells, and larger cells coverage radius, more users are associated per small cell, and thus, deployment costs are reduced per user, or per throughput unit. Increasing the number of small cells in a low density network increases significantly energy efficiency, comparing to increasing the number in already dense deployment. As for deployment efficiency, it increases when adding small cells in low density deployment scenarios, and decreases beyond an optimal small cell density. The optimal small cell density in this case depends on the traffic load and throughput, used for computing deployment efficiency. Based on the studies by Chen Y.*et al.* and Chen D. *et al.*, we sketch the deployment efficiency in function of the small cells density [73, 76].

- **Communication latency vs transmit power** Wireless communication is always subject to a trade-off between transmit power and communication delay, dictated by the Shannon capacity. The relationship between transmit power and delay is monotonically decreasing. However, in practice, taking into account hardware power consumption, and scheduling latencies for example, the relationship between the overall power consumption and the whole service delay does not have the same behavior. The trade-off curve changes its behavior, and, in general, there is no closed formed expression that shows the relationship between power and delay [73]. This phenomenon can be due to, among others, the scheduling latency that increases when transmission delay is larger, circuit power, channel conditions, and traffic arrival rate.

2.4 Advanced MEC trade-offs: Joint Communication and Computation

When empowered with cloud functionalities, HetNets are subject to additional trade-offs. In addition to communication QoS and communication resource allocation, an extra system operation parameter is added to HetNets: computation. Henceforth, MUEs share both communication and computation resources. In previous Section 2.2, we detail the adopted MEC model in this thesis. The required joint communication and computation resource allocation, in addition to computational load distribution in cluster set up, are subject of several trade-offs. In addition to the HetNets classic trade-off axes, new axes are added when computing is considered in HetNets, and some evolve to include more parameters. For example, service latency in classic HetNets is based on the communication delay perceived from MUEs perspective. In a cloud-enabled HetNets, service latency depends, not only on communication delay, but also on computation delay, tasks execution location (SSC or SCC), used computational capacities, load distribution, and intra-cluster communication delay.

2.4.1 Computation Offloading Trade-offs

First, we consider the computation offloading mechanism in which MUEs decide between executing computational tasks locally using handset resources, and offloading tasks to be executed on the cloud.

- **Energy efficiency: local computation vs computation offloading**
  MUE low energy consumption and extended battery lifetime are among the most important performance indicators of mobile handsets. Therefore, MUE energy efficiency is considered as a main parameter in computation offloading decision. However, local computation and computation offloading energy efficiencies vary with respect to system conditions and tasks requirements. This creates a trade-off between energy efficiency and system performance in terms of service latency and computational capacity.

- **MUE energy consumption vs service latency**
  A main trade-off in computation offloading is between energy consumption on the MUE side, and the perceived service latency. Less energy consuming computation strategy does not always have lower service latency. For example, local computation of requests that require the transmission of high volume of data in case of offloading may be less energy consuming than the task offload. However, since local computational capacities on MUEs are lower than the cloud, the computation time is larger. As sending computational data to SSCs imposes communication energy consumption, higher computational capacity is offered by the cloud, and thus the computational service delay can be reduced. We note that this trade-off depends on computational tasks requirements, MUE resources availability, and communication channels quality.

- **MUE energy consumption vs computational capacity**
  While computational capacity offered by the cloud is higher than available computational capacity at the MUEs, computation offloading may not always be beneficial from an energy consumption perspective. Conversely, local computation at the MUE may be less energy consuming, but computational resources may not be enough for respecting resources constraints (latency, memory requirements, or computational capacity). We note that this

trade-off depends on computational tasks requirements, MUE resources availability, and communication channels quality.

- **MUE resources extension vs privacy**
  Having the possibility of offloading computational tasks fom MUEs to the cloud extends the MUEs computational and storage capacities. However, executing computational tasks, even on local cloud, requires sending computational data that may contain private users' information. A trade-off takes place between keeping users' privacy by computing tasks locally, and computational and storage resources availability, which is higher on the cloud. We note that this trade-off depends on the MUE resources availability, the local cloud deployment model (Private, public, community, hybrid), and the SSC deployment model (open access, closed access, hybrid access).

2.4.2 Small Cell Cloud Clustering Trade-offs

In this section, we focus on the trade-offs that are faced in the mobile edge cloud computing cluster set up. Setting up a computational cluster requires identifying the small cells to participate, load distribution among the small cells, computational capacity allocation at each of the small cells, and the communication resources allocation for exchanging computational data between SSC and HSCs. We assume that computational tasks are received by SSCs, and we investigate the trade-offs in the cluster set up between SSC and HSCs, independently from MUEs. The size of the computational SSC is an important parameter in cluster set up. The cluster size is constrained by several parameters such as the aggregated computational capacity and the latency constraints of the computational tasks. Larger clusters have higher aggregated computational capacity, and lower service latency. However, increasing the cluster size reduces the system energy efficiency, and may increase as well intra-cluster power consumption. The imposed constraints on the cluster and the various limitations in terms of power and energy efficiency result in a set of trade-offs that we list below:

- **Transmit power vs computational capacity**
  All participating small cells in the SCC should be able to deliver computational results to the SSC without violating latency constraints. Except for the SSC, where no data exchange is required, the service latency for each HSC can be written as the sum of communication and computational delay.

$$\Delta_{HSC} = \frac{W'_{HSC}}{R_{p_{HSC}}} + \frac{W_{HSC}}{F_{HSC}} \tag{2.2}$$

  The communication delay in Eq. 2.2 depends on the size of data to be exchanged between SSC and HSC, W'_{HSC}, and on the transmission rate defined by the transmit power p_{HSC}. The computational delay depends on the computational load W_{HSC} and the computational capacity allocated at the HSC F_{HSC}. The trade-off between communication and computation delay is clear, where, for respecting latency constraint Δ_{HSC}, if one increases the other should decrease. Figure 2.4 shows the trade-off, for a fixed latency constraint T and computational load, between allocated transmit power and computational capacity. Higher transmit power reduces the communication delay, and thus less computational capacity can be used at the HSC for satisfying latency constraints. In the multi-user case, this trade-off is important for coordinating computational resources allocation and transmission powers for distributing computational resources among several users, while respecting all latency constraints. We note that in case of a fixed offered computational capacity at the HSC, the service latency increases with transmit power decrease. This trade-off shows that the cluster set up depends

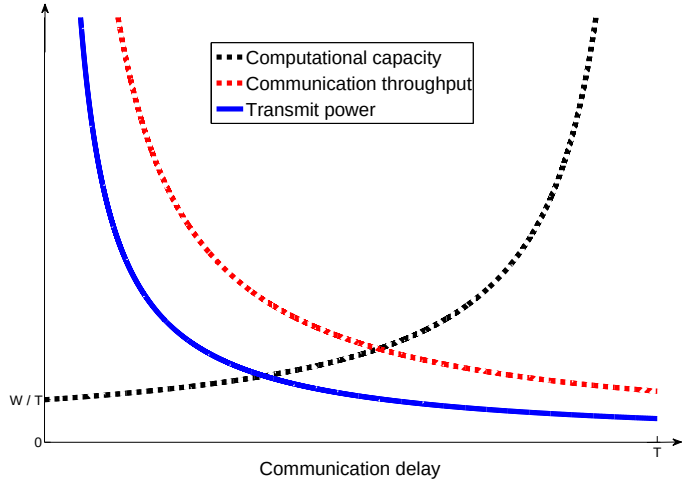


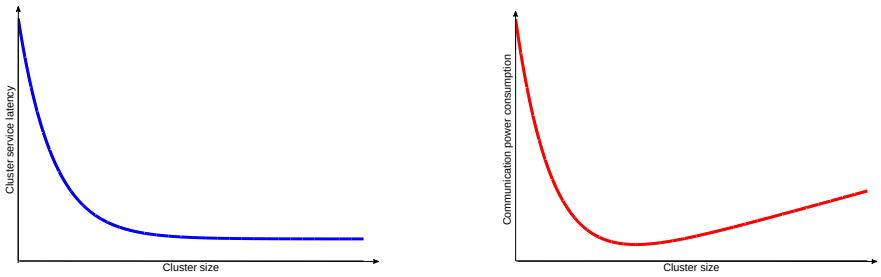
Figure 2.4: Transmit power vs computational capacity trade-off

on a clustering policy. Based on this trade-off we formulate in Chapter 4 two clustering policies based on either reducing the communication transmit power or the cluster service latency.

- **Cluster size vs service latency vs power consumption**
  While a trade-off exists between communication power consumption and service latency at each HSC in a SCC, the whole service latency is not only defined by that trade-off. SCC latency depends on the set of all participating small cells, the computational load distribution, and the communication and computation resources. We distinguish a first trade-off between SCC perceived latency and intra-cluster communication power consumption. Increasing the number of HSC in a SCC, i.e., increasing the cluster size, results in smaller computational loads distributed on participating small cells. For a fixed computational capacity at HSCs, and fixed transmit power, decreasing the load leads to a lower service latency. This can be seen in Eq. 2.2 where it is shown that the perceived latency at each HSC depends on the computational load and the size of computational data to exchange. Smaller service latency at each HSC results in smaller perceived cluster latency, using adapted load distribution.

  With smaller loads to be distributed at each small cell, less computational data is exchanged between SSC and each HSC. Fixing the computational capacity of HSCs, sending less data to HSCs decreases the transmit power between SSC and HSCs. However, an excessive increase in the cluster size, can result in an increase in transmit power consumption. Figure 2.5 represents the variation of latency and intra-cluster communication power consumption with respect to the cluster size. Note that a contribution of this thesis in Chapter 4 is based on this trade-off, where the cluster size vs communication power consumption is exploited in order to decrease power consumption, at the cost of an increasing latency due to cluster size reduction.

- **Cluster size vs system energy efficiency vs aggregated computational capacity**
  Increasing cluster size allows further parallelization of computational tasks, and thus de-



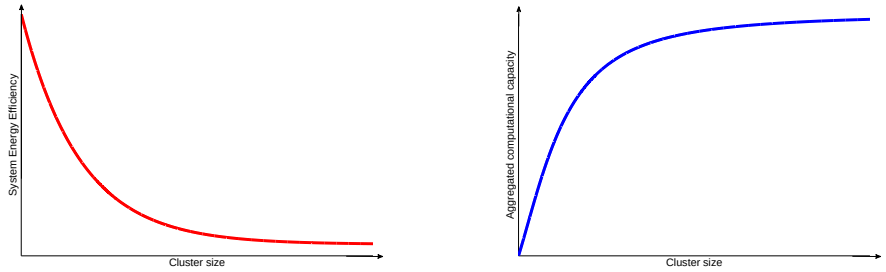
(a) Cluster size vs cluster service latency

(b) Cluster size vs cluster communication power consumption

Figure 2.5: Cluster size vs service latency vs power consumption trade-off

creases the service latency in SCCs. However, energy efficiency from a system perspective decreases with assigning lower computational load to small cells. In fact, small cells power consumption is mostly due to them being activated. The power consumption slightly depends on the communication load. Therefore, assigning higher loads to each small cell improves its energy efficiency. However, cluster set up is constrained by required computational capacity and latency constraints. When aggregated computational capacity at the SCC cannot satisfy the computational tasks requirements, more HSCs should be included in the cluster. Increasing the number of HSCs increases the aggregated computational capacity that could be offered by the cluster. The cluster offered capacity, seen as the sum of perceived capacities at each HSC, increases with the number of HSCs. This is true if we consider a SCC built with HSC of equal computational capacities, which are added to the cluster following an order of ascending distance from SSC. Note that the aggregated capacity saturates when the added HSCs cannot improve the cluster performance due to the high communication delay imposed by large communicating distance. Figure 2.6 shows the variation of System Energy Efficiency (SEE) and aggregated computational capacity of a SCC in function of the cluster size. Note that improving system energy efficiency is the aim of a proposed cluster size reduction strategy based on the cluster latency vs communication power consumption trade-off. We detail the SEE and computational capacity trade-off in the next section.

- **Deployment density vs cluster service latency**
  SCC is based on the joining several small cells for computational purposes. The intra-cluster communication is an important parameter that affects the SCC performance in terms of both service latency and communication power consumption. Ultra-dense small cell deployment is a major key enabler for SCC-based edge cloud computing. Higher small cells density increase the number of neighbor small cells at high proximity to SSC. Proximity of cluster small cells allows faster intra-cluster communication with lower power consumption. Therefore, service latency increases with cloud-enabled small cells density around the SSC. The deployment density relative to the distance between SSC and HSCs is exploited differently according to the cluster set up policy. A latency minimizing strategy will exploit the density of available small cells to increase the cluster size for achieving lower service latency. An energy aware strategy, exploits neighbor small cells density to reduce intra-cluster commu-



(a) Cluster size vs system energy efficiency

(b) Cluster size vs aggregated cluster computational capacity

Figure 2.6: Cluster size vs SEE vs aggregated computational capacity trade-off

nication power consumption while keeping the same cluster size, for example. In Chapter 4, we show the effect of small cells density on cluster set up strategies.

- **Intra-cluster backhaul technology and topology trade-offs**
  Intra-cluster communication has an impact on almost every SCC trade-off. The communication delay and power consumption, are defined according to the adopted communication technology and topology. Intra-cluster backhaul in edge cloud computing platforms has been very rarely studied in literature. In the next section, we propose a detailed study that shows the impact of backhaul on SCC characteristics.

The discussed trade-offs are based on a set of parameters that affect the cluster set up and performance. From the same parameters, we identify more trade-offs that have similar behavior than the discussed trade-offs above. A few of these trade-offs are mentioned in the list below.

- System energy efficiency vs latency
- Deployment density vs required cluster size
- Cluster latency vs Electromagnetic Field (EMF) exposure
- Deployment efficiency vs system energy efficiency

2.4.3 Local Computing vs Computation Offloading

The core of mobile cloud computing is computation offloading, from mobile devices, to be computed by the cloud. Computational requests offloading requires sending data from MUEs to the cloud. In MEC, mobile devices communicate with the cloud through users' serving base stations. From an energy point of view, computation offloading has communication costs, which increase with the size of data transferred to the serving cell. Communication energy consumption also depends on transmission data rate, and transmit power. Let us define the energy efficiency metric, D_{EE} [bits/Joule] as a measure of the amount of data that can be transferred with a given energy budget. Then, the total energy E_o, consumed for offloading a task of N bits, is equal to:

$$E_o = \frac{N}{D_{EE}} \tag{2.3}$$

In case of local computation at the MUE, consumed energy for tasks execution depends on the number of CPU cycles to be executed, and MUE processors computing energy efficiency. Let us define the computation energy efficiency metric C_{EE} [CPU cycles/Joule] as a measure of the amount of CPU cycles for a given energy budget. Then, computation energy E_l, consumed on the MUE for executing tasks of a block of C CPU cycles, is:

$$E_l = \frac{C}{C_{EE}} \tag{2.4}$$

From an energy point of view, the more efficient computing location, between the cloud and MUE, is the one with lower energy consumption. Computing tasks locally at MUE is more energy efficient if $E_l \geq E_o$, and vice versa. Offloading is usually more beneficial when the amount of data to transfer is low, comparing to the amount of computation. Local computation is less energy consuming when large amount of data have to be sent for executing a small amount of computation. In cases that vary between the both previous extremes, the less energy consuming solution depends greatly on energy efficiency parameters of both solutions.

2.4.4 Performance and Energy Savings

Mobile computation offloading is not only seen as an efficient way to save energy, but also as a way to reduce computation delay. Service delivery time is an important performance metric for QoE. When making offloading decision, execution time should be taken into consideration. However, computation offloading decisions that aim at reducing energy, do not necessarily coincide with decisions that reduce execution time. In general, the delay of cloud computing is equal to the sum of the communication and computation latencies. Communication latency is the communication time between MUEs and the cloud (in both UL and DL). Computation latency is the time the cloud takes to execute requested tasks. We define t_o as the whole computation offloading process latency, which is equal to the sum of communication time t_{com} and computation delay t_{comp}.

$$\begin{aligned} t_o &= t_{com} + t_{comp} && (2.5)\\ &= \frac{N}{R} + \frac{C}{F_c} && (2.6)\\ &= \frac{N}{R} + \frac{C}{\lambda F_l} && (2.7) \end{aligned}$$

where N is the number of bits to send, R is the transmission bit rate in bits/sec, and F_c is cloud computational capacity in CPU cycles/sec, which is λ times greater than the local computation capacity F_l.

We define t_l as the local computation delay.

$$t_l = \frac{C}{F_l} \tag{2.8}$$

where F_l is the mobile device computational capacity.

From a delay perspective, offloading reduces execution time when $t_l > t_o$, i.e. when

$$t_l > \frac{N}{R(1-\lambda)} \tag{2.9}$$

Knowing the delay in both local computing and computation offloading use cases, we can formulate the energy consumption of both cases.
The consumed energy for computation offloading and computation on the cloud is formulated as follows:

$$E_o = \frac{N}{R} p_{tr} + \frac{C}{F_c} p_c \tag{2.10}$$

where p_c is the cloud computation power, and p_{tr} the transmission power.

In the case of local computation at the MUE, energy consumption can be written as follows:

$$E_l = \frac{C}{F_l} p_l \tag{2.11}$$

where p_l is the local power for computing.

From an energy perspective, offloading reduces energy consumption when $E_l > E_o$, i.e. when

$$t_l > \frac{N p_{tr}}{R(p_l - \lambda p_c)} \tag{2.12}$$

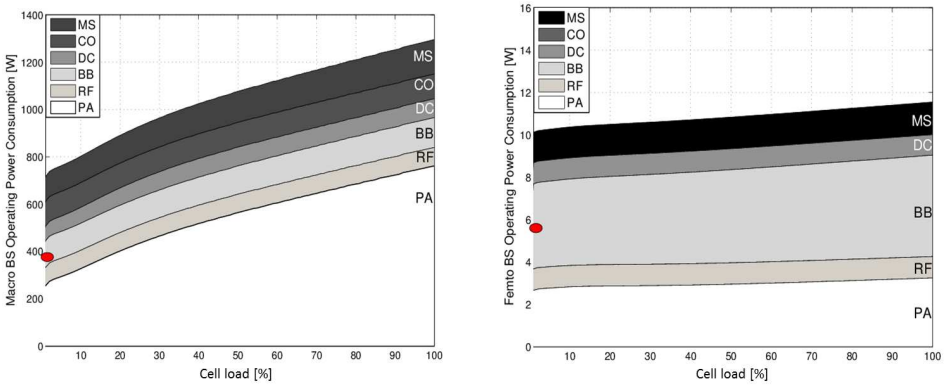
If both conditions in Eq. (2.9) and (2.12) are satisfied, i.e. if $t_l > \max\{\frac{N}{R(1-\lambda)}, \frac{Np_{tr}}{R(p_l-\lambda p_c)}\}$, then computation offloading is beneficial. Not only it reduces energy consumption, but reduces service delivery delay as well. On the contrary, where either of these conditions are satisfied, and thus, $t_l < \min\{\frac{N}{R(1-\lambda)}, \frac{Np_{tr}}{R(p_l-\lambda p_c)}\}$, then local computation at MUE is less time and energy consuming. In the case where t_l respects only one of the two equations, there is a trade-off between energy consumption and execution time. If Eq. (2.9) is valid, and $\frac{N}{R(1-\lambda)} < t_l < \frac{Np_{tr}}{R(p_l-\lambda p_c)}$, then computation offloading execution is faster, but more energy consuming. If Eq. (2.12) is valid, and $\frac{Np_{tr}}{R(p_l-\lambda p_c)} < t_l < \frac{N}{R(1-\lambda)}$, then computation offloading execution is less energy consuming, but it is takes more time to execute the tasks. In the last two cases, the existing trade-off can be biased by varying some parameters, if possible. For example, increasing the transmission power reduces the overall offloading computation delay, at the cost of increasing energy consumption.

We now focus on the adopted SCC cloud scenario. As already showed in Eq. (2.7) the offloading delay depends on both communication and computation components. For SCC, where the cloud is a set of coordinated small cell instead of a single entity executing the tasks, computation delay does not dependent on a single computational capacity as in this equation. The *cloud* computational capacity is offered by more than one small cell and is not fixed a priori. A total perceived computational capacity can be computed taking into account the allocated capacities at each small cell in the cluster, the number of computed CPU cycles, and the total perceived computation delay. The perceived delay depends on both intra-cluster communication for exchanging computational data and results, and computation delay for executing the accorded tasks at every small cell. The formulation of cluster latency depends on intra-cluster communication topology and technology. In addition, load distribution inside the cluster, and allocated capacities at cluster SCs contribute as well in computing the overall cluster perceived latency. Clusters latency and power consumption formulations will be detailed in Section 2.5.

2.4.5 System Energy Efficiency, Cells Density, and EMF Exposure

Mobile networks design has been basically focused on reducing the energy consumption of mobile terminals. Nevertheless,many efforts are put in order to improve energy efficiency of the network

as well. Indeed, a high percentage, around 70%, of the mobile networks energy consumption comes from the base stations [7]. Consequently, the optimization of network base stations energy efficiency is investigated, and several mechanisms and approaches that aim to increase system energy efficiency are proposed. Small cells deployment, and thus the co-existence of macro cells and small cells in heterogeneous networks, helps improve system energy efficiency by offloading macro-cells traffic. Macro-cells power consumption depend on cell load and output transmit power. Hence, reducing traffic of macrocells, lowers the macro network power consumption. On the contrary, power consumption of small cells, particularly femtocells, is independent of the cell load. Figure 2.7 shows the operating power consumption of macro and femto base stations with respect to the cell load [7]. Femto base stations power consumption barely depend on the cell load,



(a) Macro-BS power consumption with respect to load [7]

(b) Femto-BS power consumption with respect to load [7]

Figure 2.7: Power consumption dependency on relative cell load. (PA: Power Amplifier, RF: small signal RF transceiver, BB: Baseband processor, DC: DC-DC converters, CO: Cooling, PS: AC/DC Power Supply, Red mark: BS sleep mode power consumption) [7]

and thus on the output power. From a system energy efficiency point of view, it is best that femtocells operate at full load, when activated. As shown on Figure , the energy consumption of an idle base station is nearly the same as one under full load. In order to save energy, and achieve higher energy efficiency, base stations on-off switching is proposed for switching off under-utilized base stations. Cell activation/deactivation mechanisms can improve the network performance enabling local access points to self-switch off in absence of neighboring end-users [77].

In a small cell cloud cluster scenario, system energy efficiency depend on the load of each computing node in the cluster. Energy efficiency is however considered for both communication and computation. In terms of computation, small cells energy consumption depend on the used computational capacity and the computation load. System energy efficiency is then improved by switching off some cluster nodes that under-utilize their computing capacities. Cells switching-off in a small cell cloud scenario results in reducing the size of the cluster. In a smaller cluster, small cells operate at higher communication and computation load, achieving higher network energy efficiency. With small cells switching-off, the cluster size is reduced, and the cluster can also be geographically sparsified. This means that cluster nodes may end be at a larger distance from the serving cell. HSCs at a larger distance from SSC use higher transmit power, in order to

achieve desired SINR, or compensate for the latency loss brought by increasing the distance. The sparsification is less harmful in cases of ultra-dense deployment of small cells. The high density of neighbor small cells increases the probability of limiting the geographical expansion of the small cell cluster.

Increasing system energy efficiency is however subject to some trade-offs or limiting factors. The main limiting factor of small cell clusters sparsification, and size reduction, is latency. Reducing the size of the cluster, results in according higher loads to the participating nodes. Higher loads not only impose longer computation time at each small cell, but also larger data size to be transmitted to each node, which imposes higher communication delay, and/or higher transmission power.

From a system energy efficiency point view, the best case scenario is to compute tasks at SSCs. In this case, no intra-cluster communication is required, and thus, no communication energy costs are imposed. However, this solution is not applicable in case SSC available computational capacity is less than required. The cluster size is constrained by latency constraints and thus computational capacity requirements. A representative graph in Figure 2.8 shows that there's a minimum cluster size imposed in order to be able to serve the computational request, i.e. to have sufficient computational capacity for not violating latency constraints. As shown on Figure 2.8, the aggregated computational capacity offered by the cluster is smaller than the sum of cluster nodes capacities. This is because the aggregated capacity takes into account the intra-cluster communication latency cost, and thus, load distribution. The perceived computational capacity is then equal to the ratio of the total request size to the total cluster computation latency.

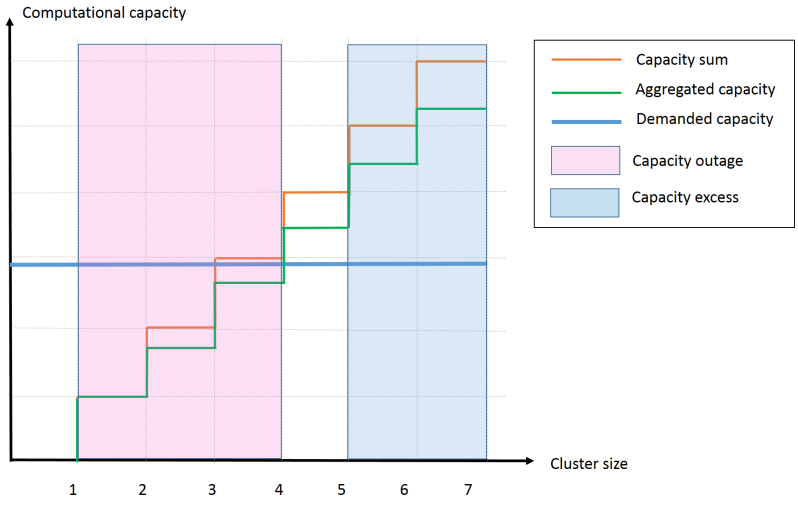


Figure 2.8: Example of aggregated computational capacity with respect to the cluster size

Another issue gaining a lot of attention with the proliferation of mobile cloud computing paradigm, is Radio Frequency Electromagnetic Fields Exposure (RF-EMF). Wireless devices omnipresence raises health concerns due to possible effects of electromagnetic radiations on the human body, and especially on the brain, due to its proximity with the hand-held radio devices [8]. When radio frequency transmission is used for extended duration, implications may take place, especially due to the heating effect. Mobile cloud computing increases uplink traffic, and consequently, EMF exposure. Moreover, in small cell cloud, intra-cluster communication creates a

source of EMF exposure, since small cells are normally deployed at a proximity to mobile users. While this enables user devices to transmit at lower power, it also moves the transmitters closer to the users, which increases the EMF effect. EMF exposure is measured via a Specific Absorption Rate (SAR) expressed in W/Kg. EMF exposure depends mainly on distance between the communicating device and the human brain, and on the uplink transmit power. Reducing the EMF effect can be achieved through using the minimum amount of transmission power, especially when good channel conditions are available. However, when the communication signal is subject to severe losses, caused by either large distance between communicating devices, or severe fading caused by obstructions, communicating devices are forced to increase the transmission power. Figure 2.9 is a heat map that shows the impact of transmit power and distance from the device to human brain, on EMF. The map considers only the effect from the mobile users, and not the surrounding base stations that despite the fact that they are farther from the users, they use much higher transmit power. In a small cell cloud scenario, EMF exposure is in a trade-off with system en-

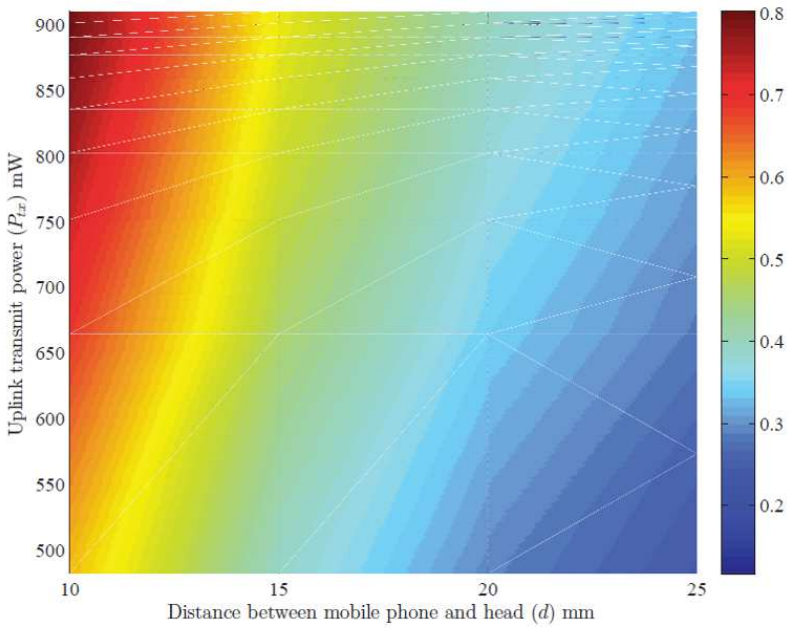


Figure 2.9: Specific absorption rate in respect with distance and transmit power [8]

ergy efficiency. Indeed, as deduced earlier, increasing the system energy efficiency requires using the cluster nodes at full load in both communication and computation. As cluster node loads are larger, and the latency constraints are the same, then the communication time needs to be smaller. This leads to increasing the transmit power, and thus, the EMF exposure. Small cells density plays also a role in this trade-off. Cell switching-off limits the number of active small cells, and if the deployment density is low, then the distance between nodes risks being larger. Therefore, higher transmit power should be used in order to maintain desired communication efficiency.

2.4.6 Small Cell Cluster Cloud

All of the discussed trade-offs above apply to the MEC scenario adopted in this thesis. In a small cell cloud, the main steps are the following: (i) computation offloading decision (ii) communica-

tion resource allocation for sending computational request to the SSC (iii) load distribution among cluster nodes (iv) communication resource allocation for intra-cluster communication (v) computational capacity allocation at each small cell in the cluster.

In each of these steps, an optimization of the decision/solution is required. In the first step of offloading decision, it is necessary to incorporate all parameters that affect the most essential optimization criteria, which is users' QoE. Energy consumption, available computational capacity, service latency, memory requirements, are examples of aspects that should be considered for deciding between computation offloading and local computation at mobile devices. The offloading decision process must be of low enough complexity, especially if it is to be implemented on mobile devices.

SCC latency depends on both data transmit time, and computation delay. Therefore, transmit powers and computational capacities should be optimally allocated, in order to equilibrate the communication and computation latencies for respecting latency constraints. The transmission power is subject to a trade-off with EMF, device energy efficiency and perceived latency.

Computation load distribution, computational capacity, and intra-cluster communication resource allocation are the main steps that create the small cell cloud cluster. The three steps are related since the cluster characteristics in terms of latency, computational efficiency, and power consumption are affected by all the steps. Therefore, a joint allocation of computation and communication resources is required for guaranteeing the respect of latency constraints. Building the small cell cluster is subject to numerous trade-offs, basically between latency and power consumption. Cluster size expansion and sparsification is a way for manipulating these trade-offs. SCC dimensioning creates, however, a trade-off between energy efficiency of system and mobile devices. Intra-cluster communication technology and topology also have an effect on the load distribution and resource allocation. Backhaul technology and topology have an impact on the perceived cluster latencies, power consumption, and resources utilization. For example, in a small cell cluster where nodes communicate through high capacity fibers, communication latency is small, nay negligible, and thus, the focus will then be on load distribution and computational capacity allocation. On the contrary, OTA (Over The Air) intra-cluster communication requires optimization of communication resource allocation in order to guarantee in time service delivery.

2.5 Extending the Impact of Backhaul Network on Small Cell Cloud Computing

In the considered SCC (small cell cloud), the cooperation of small cells, through cluster formation, merges cooperation for communication with computation offloading. The cloud network boosts the computational capacity of mobile terminals, and its proximity to users' equipment reduces the end-to-end latency. To enhance the computation and storage capacities, small cells are backhauled together and exchange data. Accordingly, when an application is offloaded to the SCC, one or more cells could contribute in the data processing and computing. The cluster management process, i.e., choosing the cluster size and the set of cooperative small cells, depends on multiple constraints such as, latency and power consumption. Another important parameter is the backhaul type through which the nodes are connected. Nevertheless, there is a lack of studies that investigate these relationships. With the dependency on several parameters, getting the optimal number of base stations to include in a cluster would request more analysis.

2.5.1 System Model

To model and evaluate cluster latency and power consumption, we consider a cellular deployment model of hexagonal compartments of radius of 5m. Each cell is assumed to be equipped with a deployed small cell (see Figure 2.10). We consider an LTE system with K mobile users and N femtocells. Users are served by a femtocell base station within a distance d. We consider a connection channel of bandwidth B between UE (User Equipment) and the serving base station.

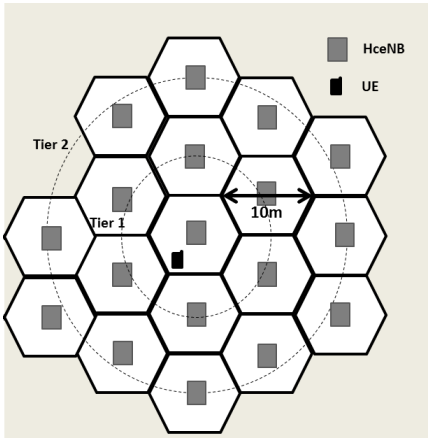


Figure 2.10: Cellular deployment of small cells.

Instantaneous bit rate is maximized based on adaptive modulation and coding scheme (AMC) [78]. A parameter Γ in the channel model indicates the SNR margin to guarantee the minimum error rate. We adopt a Rayleigh channel model with path loss exponent β, noise power N_0, and fading channel coefficient h_k. We assume perfect estimation of the channel coefficients h_k and the channel fading is assumed constant for a whole transmission period. The maximum information rate that can be achieved through this channel is calculated using the following equation:

$$R = Blog(1 + aP_{Tx}) \tag{2.13}$$

where $a = \frac{|h_k|^2}{\Gamma d^{\beta} N_0}$, B is the channel bandwidth, and P_{Tx} is the transmission power. We consider that user k asks for the computation of W CPU cycles to the femtocell it is connected. We assume that the number of bits to be transmitted through uplink and downlink communications is proportional to W: $N_{UL} = W\beta_{UL}$ for uplink and $N_{DL} = W\beta_{DL}$ [79], where the constants β_{UL} and β_{DL} account respectively for the overhead due to the uplink and downlink communications and for the ratio between output and input bits associated to the execution of CPU cycles at small cells. The uplink and downlink transmission length are expressed respectively as: $\Delta_{UL} = \frac{N_{UL}}{R_{UL}}$ and $\Delta_{DL} = \frac{N_{DL}}{R_{DL}}$, where R_{UL} and R_{DL} are the instant maximum rate that can be achieved in uplink and downlink transmissions, evaluated with Eq. (3.1).

We denote the latency constraint of the application as L_{app}, and the cluster overall latency as $\Delta_{cluster}$. For the application latency to be respected, we should have:

$$L_{app} \geq \Delta_{UL} + \Delta_{cluster} + \Delta_{DL} \tag{2.14}$$

Finally, the power consumption of the overall process can be formulated as:

$$P = P_{Tx}^{UL} + P_{com} + P_{comp} + P_{Tx}^{DL} \tag{2.15}$$

where P_{Tx}^{UL} and P_{Tx}^{DL} are, respectively, the radiated power of uplink and downlink transmissions. P_{com} and P_{comp} represent the power consumed in the cluster for communication and computation

respectively. In the rest of this section we model $\Delta_{cluster}$ and P_{com} and their relations with the cluster characteristics such as, size, backhaul topology, and backhaul technology.

2.5.2 Latency Models

To guarantee an acceptable quality of experience (QoE), applications latency constraints must be respected. We identify three major latency components: the transmission duration from MUE to the SSC, the overall cluster latency due to the data sharing amongst cluster nodes and load computations, and data transmission from the SSC back to UE. Data transmission between UE and the serving cell depends on the channel quality and on the amount of data to be transmitted. Particularly, data processing latency depends on the number of HSCs, the amount of computing tasks assigned to each one of them, and their allocated computational capacity. Data transport inside the cluster depends on the cluster size, the backhaul topology used to interconnect the base stations, and the backhaul technology.

The number N of femtocells in the cluster should be set in order to satisfy the following constraint:

$$N = \{n \in \mathbb{N} / \Delta_{cluster}(n) \leq L_{app} - \frac{N_{UL}}{R_{UL}} - \frac{N_{DL}}{R_{DL}}\} \tag{2.16}$$

We now model the cluster latency when using the ring, binary tree, and full mesh topologies (see Figure 2.11). As for transmission technologies we consider fiber backhaul, microwave backhaul, and over the air (OTA) LTE wireless backhaul. As already discussed, the cluster latency is due to the communication latency between cluster nodes and the computation latency at each one of them.

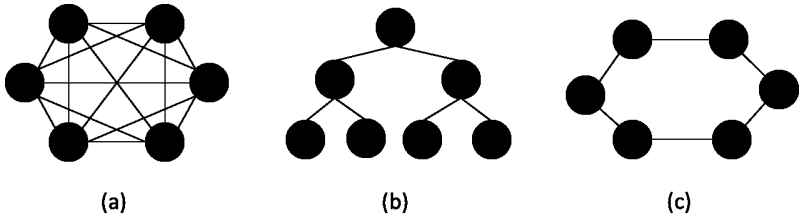


Figure 2.11: Wireless backhaul topologies: (a) full mesh topology, (b) tree topology, and (c) ring topology.

2.5.2.1 Full Mesh Topology

Hence, in the full mesh case, the cluster latency can be written as:

$$\Delta_{cluster} = \max_{n=1}^{N}(W_n f_n^{-1} + \delta_{Tx,bh}(n) + \delta^r_{Tx,bh}(n)) \tag{2.17}$$

where W_n is the computation task assigned to each base station, f_n its computational capacity, $\delta_{Tx,bh}$ the one way communication latency through the backhaul between the serving small cell and helper small cells, and $\delta^r_{Tx,bh}$ the communication latency for the reverse way.

2.5.2.2 Wireless LTE Backhaul

Considering LTE wireless backhaul, $\delta_{Tx,bh}(n) = \frac{N_{in}^n}{R_{in}^n}$ and $\delta_{Tx,bh}^r = \frac{N_{out}^n}{R_{out}^n}$, where $N_{in}^n = W_n \beta_{UL}$ is the number of bits sent to base station n, and $N_{out}^n = W_n \beta_{DL}$ is the number of bits that base station n should send back. R_{in}^n and R_{out}^n represent the rates of data transmission achieved in downlink and uplink through the channel between base station n and the base station connected to the UE.

$$\Delta_{cluster} = \max_{n=1}^{N}(W_n f_n^{-1} + \frac{N_{in}^n}{R_{in}^n} + \frac{N_{out}^n}{R_{out}^n}) \tag{2.18}$$

Fiber and microwave backhaul

For fiber and microwave backhaul, the latency of a transmission is assumed to be load independent because of the high throughput than can be achieved using these technologies. A categorization of non-ideal backhaul latency based on operator inputs can be found in [11]. Therefore, in both cases the total cluster latency can be formulated as the following:

$$\Delta_{cluster} = \max_{n=1}^{N}(W_n f_n^{-1} + 2\delta_{Tx,bh}) \tag{2.19}$$

2.5.2.3 Tree Topology

In the tree topology case, for the data to reach base station n at level l_n, it should be transmitted through l_n base stations (The serving small cell is considered of level $l = 0$). Therefore, the total cluster latency can be formulated as:

$$\Delta_{cluster} = \max_{n=1}^{N}(W_n f_n^{-1} + \sum_{l=1}^{l_n} \delta_{Tx,bh}(n) + \sum_{l=1}^{l_n} \delta_{Tx,bh}^r(n)) \tag{2.20}$$

Wireless LTE backhaul

For LTE wireless backhaul, $\delta_{Tx,bh}$ and $\delta_{Tx,bh}^r$ depend respectively on the number of bits to be sent in uplink and downlink, and on the channel capacity at the transmission time.

$$\Delta_{cluster} = \max_{n=1}^{N}(W_n f_n^{-1} + \sum_{l=1}^{l_n} \frac{N_{in}^n}{R_{in}^l} + \sum_{l=1}^{l_n} \frac{N_{out}^n}{R_{out}^l}) \tag{2.21}$$

where R_{in}^l and R_{out}^l are the transmission rates in downlink and uplink at the base station backhauling the traffic at level l.

Fiber and microwave backhaul

For fiber and microwave backhaul cases, as the transmission latency is constant, ($\delta_{Tx,bh}(n) = \delta_{Tx,bh}^r(n) = \delta_{Tx,bh}$), the total latency can be represented as:

$$\Delta_{cluster} = \max_{n=1}^{N}(W_n f_n^{-1} + 2l_n\delta_{Tx,bh}) \tag{2.22}$$

2.5.2.4 Ring Topology

In the ring topology case, the total cluster latency can be formulated as:

$$\Delta_{cluster} = \max_{n=1}^{N}(W_n f_n^{-1} + \sum_{h=1}^{h_n} \delta_{Tx,bh}(n) + \sum_{h=1}^{h_n} \delta_{Tx,bh}^{r}(n)) \tag{2.23}$$

where h_n is the number of hops needed for HSC of index n to reach the SSC. Similar to the tree topology case, LTE wireless backhaul latency depends on the data load and channel conditions, and for fiber and microwave backhaul backhaul transmission time is constant.

Wireless LTE backhaul

So the equation for LTE wireless can be written as:

$$\Delta_{cluster} = \max_{n=1}^{N}(W_n f_n^{-1} + \sum_{h=1}^{h_n} \frac{N_{in}^{n}}{R_{in}^{h}} + \sum_{h=1}^{h_n} \frac{N_{out}^{n}}{R_{out}^{h}}) \tag{2.24}$$

Fiber and microwave backhaul

And for fiber and microwave backhaul:

$$\Delta_{cluster} = \max_{n=1}^{N}(W_n f_n^{-1} + 2h_n \delta_{Tx,bh}) \tag{2.25}$$

Given the latency formulas for different backhaul topologies, we can distribute the computational load across the distributed cloud in order to minimize latency. In all previous cases, for any set of computational rates f_n and channel states, the optimization problem can be cast as:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \max_{n=1}^{N}(W_n f_n^{-1} + L_n) \\ \text{subject to:} \quad & \sum_{n=1}^{N} W_n = W \\ & W_n \geq 0 \end{aligned} \tag{2.26}$$

Where L_n is the delay associated to communications across the computing nodes.

2.5.3 Power Consumption Models

Another important issue in the formation of the small cell cloud cluster, is the power consumption. Cloud computing leads to an increase in network traffic, and thus, in power consumption and EMF exposure. Power consumption in transport and switching can be a significant percentage of total power consumption in cloud computing [80]. This section presents power consumption models for data transport inside the SSC cluster. The cluster power consumption depends on the number of base stations within, the backhaul technology, and topology. All traffic from the base stations is assumed to be backhauled through the SSC, playing the role of a hub node. If more than one backhaul link originates at any node, the base station is assumed to be equipped with a switch. The equations in this section are based on the study done in [12]. The equations below assume that the backhaul topologies are height balanced (formed with the lowest possible level depth).

2.5.3.1 Wireless LTE Backhaul

For LTE wireless backhaul transmissions between base stations, the total power consumption is expressed as [9]:

$$P = \sum_{n=1}^{N} \sum_{j=1}^{N_{ant}^{n}} (P_0 + \Delta p P_{Tx}^{n,j}) \tag{2.27}$$

In this equation, N_{ant}^{n} is the number of active antennas at SSC n, P_0 the base station power consumption at zero load, Δp is the slope of the load-dependent power consumption, and $P_{Tx}^{n,j}$ the transmission power used to transmit at base station n through antenna j.

Considering constant transmission power P_{Tx} for all base stations, the total power consumption for the different types of backhaul topology is modeled differently for every topology.

Full mesh topology:

The base station connected to UE will transmit to all the $N-1$ base stations in the cluster, that will transmit back once computing tasks are accomplished. Then, the total number of transmissions in this case is $2(N-1)$, and the power consumption can be formulated as:

$$P = 2(N-1)(P_0 + \Delta p P_{Tx}) \tag{2.28}$$

Tree topology:

The number of base stations that will transmit through two antennas to two different base stations is $\lfloor \frac{N-1}{2} \rfloor$, and thus, the number of base stations that will transmit through only one antenna is $(N-1) - 2\lfloor \frac{N-1}{2} \rfloor$, which is equal to 1 if N is even and 0 if odd. All $(N-1)$ base stations transmit back when computing tasks are accomplished. Therefore, the total power consumption is expressed as:

$$P = \begin{cases} (2\lfloor \frac{N-1}{2} \rfloor + N)(P_0 + \Delta p P_{Tx}), & \text{if } N \text{ is odd} \\ (2\lfloor \frac{N-1}{2} \rfloor + N - 1)(P_0 + \Delta p P_{Tx}), & \text{if } N \text{ is even} \end{cases} \tag{2.29}$$

Ring topology:

Only the SSC will transmit through two antennas to two different base stations. In addition, $N-1$ base stations will transmit back after accomplishing computing tasks. The total number of transmissions in this case is $2(N-1)$, and the total cluster power consumption is:

$$P = 2(N-1)(P_0 + \Delta p P_{Tx}) \tag{2.30}$$

2.5.3.2 Fiber Backhaul

For fiber backhaul, the communication power consumption in the small cell cluster is formulated as:

$$P = N_{UL} P_{UL} + N_{DL} P_{DL} + \sum_{n=1}^{N} N_s^n P_s \tag{2.31}$$

with $N_s^n = \begin{cases} 0 & \text{if } N_{ant}^{n} = 1; \\ \lceil \frac{N_{ant}^{n}}{max_{dl}} \rceil & \text{otherwise} \end{cases}$

where N_s^n is the number of switches needed at base station n and max_{dl} is the maximum number

of interfaces available at one switch. For the different types of backhaul topology considered here, the total power consumption can be modeled as follows:

Full mesh topology:

$$P = (N-1)(P_{DL}+P_{UL}) + \lceil \frac{N}{max_{dl}} \rceil P_s \tag{2.32}$$

Tree topology:

$$P = (N-1)(P_{DL}+P_{UL}) + \lfloor \frac{N-1}{2} \rfloor \lceil \frac{2}{max_{dl}} \rceil P_s \tag{2.33}$$

2.5.3.3 Ring topology

$$P = (N-1)(P_{DL}+P_{UL}) + \lceil \frac{2}{max_{dl}} \rceil P_s \tag{2.34}$$

2.5.3.4 Microwave Backhaul

For microwave backhaul, the communication power consumption in the small cell cluster is formulated as [12]:

$$P = \sum_{n=1}^{N} \sum_{j=1}^{N_{ant}^n} (P_{agg}^{n,j}(C_{n,j}) + P_{ss}^n) \tag{2.35}$$

$$P_{agg}^{n,j}(C_{n,j}) = \begin{cases} P_{low-c}, & \text{if } C_{n,j} \leq Th_{low-c} \\ P_{high-c}, & \text{otherwise} \end{cases}$$

$$P_{ss}^n = \begin{cases} 0, & \text{if } N_{ant}^n = 1 \\ P_s \lceil \frac{C_{n,j}}{C_{switch}^{MAX}} \rceil, & \text{otherwise} \end{cases}$$

Where $P_{agg}^{n,j}$ is the power consumption for transmitting and receiving the aggregate backhaul traffic through base station n via antenna j. This power consumption is modulated as a two steps function that depends on whether the backhauled capacity traffic through the same antenna ($C_{n,j}$) is low or high. The capacity traffic is considered as high if it exceeds a defined threshold (Th_{low-c}), and considered as low otherwise. P_{ss}^n is the function that accounts the necessary switch power consumption that depends on the backhauled capacity and the maximum capacity of a switch (C_{switch}^{MAX}).

2.6 Numerical Evaluation

We evaluate the backhaul cluster latency and communication power consumption, using the models proposed in Sections 2.5.2 and 2.5.3. Our evaluations compare latency and power consumption for the different considered backhaul technologies and topologies, with respect to the cluster size. We adopt the same system model described in Section 2.5.1, considering a single user ($K = 1$) at the cell edge of its serving base station. All N small cells are assumed to have the same computational rate $f = 2.10^7$ CPU cycles/sec. We assume that computational load is equally distributed

on the cluster small cells. Tables 2.1,2.2, and 2.3 resume the parameters used for LTE wireless, fiber, and microwave backhaul respectively.

B [MHz]	β	N_0	T_c[s]	$P_{Tx}[W]$	P_0[W]	Δ_p
20	5	10^{-3}	1	1	6.8	4

Table 2.1: LTE wireless backhaul parameters [9] [10].

$\delta_{Tx,bh}$[ms]	P_{DL}[W]	P_{UL}[W]	P_s[W]	max_{dl}
5	2	1	300	12

Table 2.2: Fiber backhaul parameters [11] [12].

$\delta_{Tx,bh}$ [ms]	P_{low-c} [W]	P_{high-c} [W]	Th_{low-c} [Mbps]	P_{ss} [W]	C_{switch}^{MAX} [Gbps]
15	37	37	500	53	36

Table 2.3: Microwave backhaul parameters [11] [12].

As assumed in section 2.5.3, backhaul topologies are assumed to be balanced. For the whole mesh topology, we assume that a cluster of N is formed with the SSC and the $N-1$ closest HSCs. We consider three different sizes of application CPU cycles W corresponding to 1MB, 50MB, and 100MB traffic, representing low, medium, and high traffic load scenarios.

2.6.1 Cluster Latency

As already shown in Eq. (2.18), the cluster latency for the LTE wireless backhaul depends on the traffic load. In both tree and ring topologies, the cluster latency for wireless LTE backhaul will be greater than the full mesh case. In fact, not every small cell is reachable via a direct link, and thus, cluster nodes will have to backhaul traffic for farther base stations which increases the overall cluster latency. We show simulation results on the effect of traffic load on cluster latency only for a full mesh topology (Figure 2.12). As the assumption in our simulations is to always form the cluster with the closest SSC to the UE, the cluster latency will be subject to a brutal increase with the increase of cluster radius (distance between the SSC connected to the user and the farthest HSC in the cluster).We notice that for low load scenarios, including more nodes in the cluster will not have an effect on the latency since the computation time with only one small cell is already low. However, for higher load scenarios, including more HSCs in the femtocell cluster will decrease the overall latency since the tasks are distributed on different computing entities.

In the cases of fiber and microwave backhaul, as can be seen in Eq. (2.19), (2.22), and (2.25), the cluster communication latency does not depend on the traffic load. The load will only have an effect on the computation latency in the cluster through $W_n f_n^{-1}$.

Figure 2.13 shows cluster latency for medium traffic load for different backhaul topologies and technologies. As can be seen, wireless LTE backhaul is the most time costly for lower cluster sizes. Full mesh is the topology less time consuming for both fiber and microwave backhaul, followed

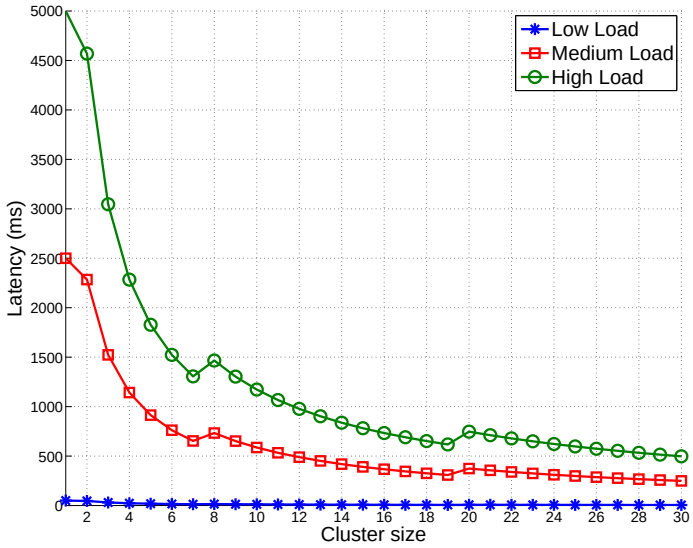


Figure 2.12: Wireless LTE full mesh backhaul cluster latency for different traffic loads.

by tree and then ring topologies. As the tree and ring topologies are assumed height balanced, we can see a step when a level is added to the topology (every 2 small cells for ring topology, every 2^n for tree topology). We notice that we always gain in latency for a cluster of size N over a cluster of size $N-1$ when the addition of the N's base station does not increase the cluster radius. However, adding too much base stations can result in more time consuming as can be seen for the fiber ring backhaul. In fact, when the number of base stations increases, the task computation delays at each node decrease. When computing delay becomes less than the transmission time between two nodes, the addition of new base stations to the cluster will increase the total latency.

Figure 2.14 compares between the adopted assumption of equal load distribution and the optimal load distribution among cluster base stations for full mesh wireless LTE and tree fiber backhaul. Same kind of results goes for other technologies and topologies. We notice that the optimal load distribution is optimal for the fiber tree backhaul, and can outperform equal load distribution in the case of wireless LTE backhaul. This is due to the fact that in the latter case the transmission latency is highly affected by the distance, whereas it is not the case of fiber backhaul. The Major difference of performance is noticed when the cluster radius changes. This graph shows that optimal load distribution in a cluster with wireless intra-cluster communication improves the cluster performance and is thus necessary.

2.6.2 Cluster Communication Power Consumption

The cluster communication power consumption for the wireless LTE backhaul depends on the transmission power P_{Tx}. If this transmission power is kept constant, as in our simulations, traffic transport power consumption will be a linear function of the number of HSCs in the cluster, however, it will consume more time as seen in 2.6.1.

For the microwave backhaul, the communication power consumption depends on the traffic load through Eq. (2.35). For this reason, a full mesh topology in this case would be the most interesting. Indeed, a previous study on microwave backhaul power consumption in [12] shows that the ring topology is the most costly in terms of power, followed by the tree topology. Figure

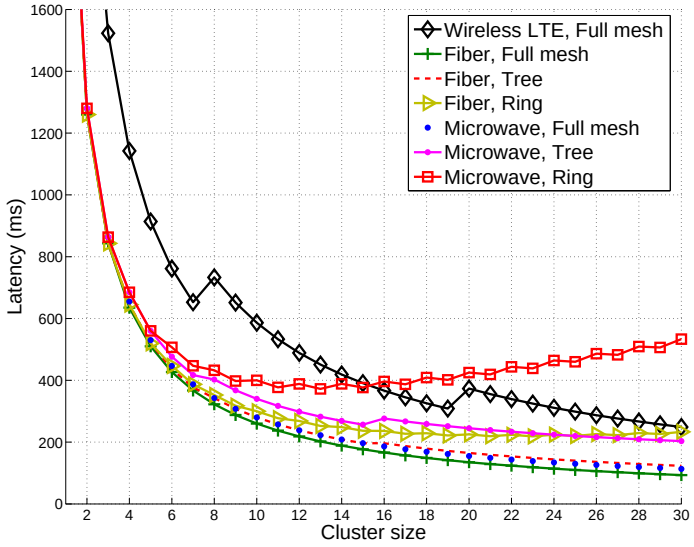


Figure 2.13: Cluster backhaul latency for different backhaul technologies and topologies for medium traffic load.

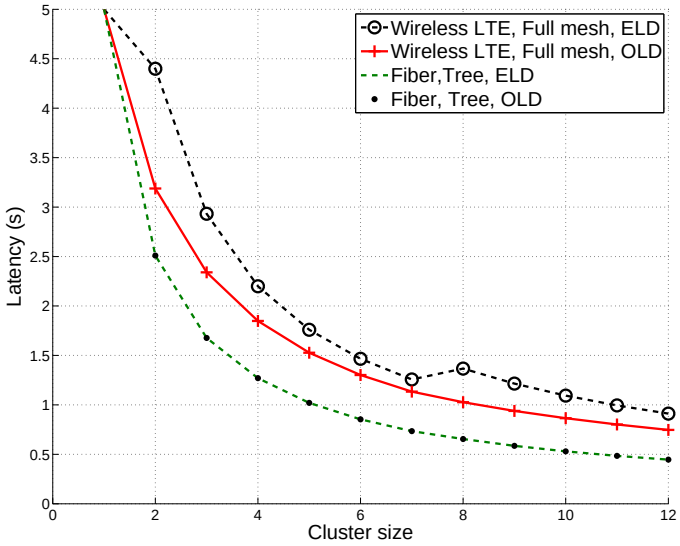


Figure 2.14: Equal load distribution (ELD) and optimal load distribution (OLD) comparison.

2.15 shows the variation of power with respect to the number of base stations. In the case of low traffic load the nodes are always operating in the low consumption regime (P_{low-c}) and the total consumption is a linear function of the number of HSCs in the cluster. In the case of high and medium traffic loads, we notice that the power consumption is reduced when the cluster size exceeds the values of 9 and 19 small cells. This is due to the fact that the traffic backhauled through each base station decreases with the increase of the cluster size. And at a certain point, the traffic backhauled through each base station gets lower than $Th_{low,c}$, and thus, the base stations switch from operating at a higher power consumption P_{high-c} to a lower power consumption P_{low-c}.

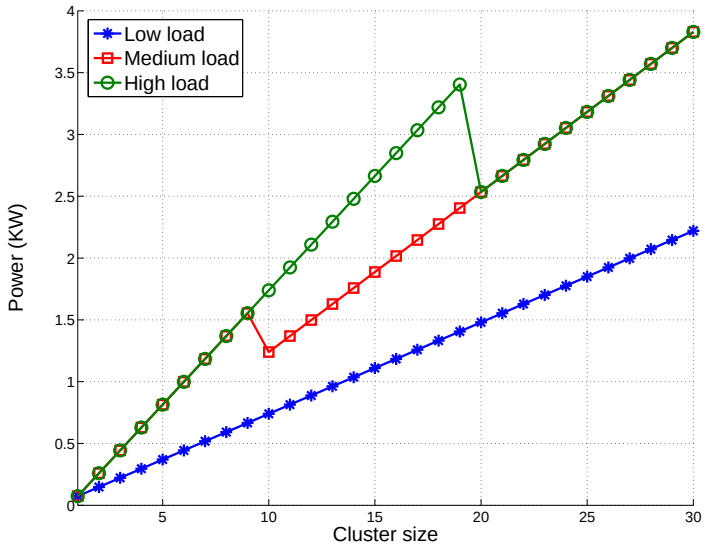


Figure 2.15: Microwave backhaul traffic power consumption for different traffic loads.

In the case of fiber backhaul, the communication power consumption in the cluster does not depend on the traffic load. As shown in Eq. (2.32), (2.33), and (2.34) it depends on the number of small cells in the cluster. Figure 2.16 shows the power consumption for the three topologies for fiber backhaul, and for the full mesh topology for microwave backhaul. It can be seen that fiber backhaul consumes less power than microwave backhaul in a full mesh topology. Fiber backhaul power consumption for a full mesh topology increases by a step each max_{dl} base stations, because an extra switch is needed. Ring topology is the less consuming since it requires the least number of switches which consumes the major part of the total power consumption. For the tree topology, at each addition of two base stations an additional switch is needed. For this reason, it is the most power consuming.

A comparison of different backhaul technologies characteristics are summarized in table 2.4.

2.7 Conclusion

Mobile cloud computing can be implemented through various and different architectures. In this chapter, we presented in details the adopted mobile cloud computing architecture. It is based on a mobile edge computing scenario case where the cloud functionalities are offered by the small cells base stations serving the mobile users. The small cells are endowed with computational

Criterion	Wireless LTE	Fiber	Microwave
Load dependent latency	Yes	No	No
Load dependent power consumption	Yes	No	Yes
Topology with lowest latency	Full mesh	Full mesh	Full mesh
Topology with lowest power consumption	Full mesh	Ring	Full mesh
Latency classification	⋆	⋆⋆⋆	⋆⋆
Power consumption classification	⋆	⋆⋆⋆	⋆⋆

Table 2.4: Comparison of backhaul technologies.(⋆⋆⋆ -the best, ⋆ -the worst)

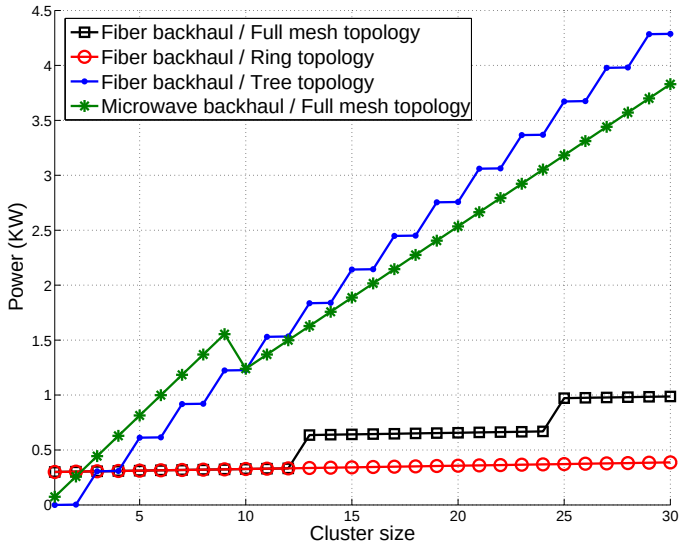


Figure 2.16: Cluster backhaul power consumption for different backhaul topologies and technologies.

and storage capacities. A cluster of small cell can be formed for distributing the execution of offloaded requests. We refer to this paradigm as the small cell cloud cluster. We also described the use case scenario we consider by specifying the assumptions we fix for our work. We consider wireless intra-cluster communication, and therefore we presented the trade-offs that are faced in a mobile edge computing with clustering possibility in a wireless scenario. We then focused on the trade-offs that can be specifically encountered in the considered architecture and scenarios. We discussed energy efficiency from both devices ad system perspectives. We showed the relationships that joins latency, cluster size, power consumption, and EMF exposure. The presented trade-offs overview shows clearly how edge cloud computing shifts the paradigm of HetNets operation optimization to include additional computing-related parameters. Edge cloud computing, and notably, computation clustering, adds a new level of trade-offs that control both network service performance and users perceived service quality. We presented as well the impact of the backhaul technologies and topologies on the small cell cluster characteristics. The set of trade-offs

are a main basis for our contributions in this thesis.

Chapter 3

Multi-parameter Computation Offloading Decision

3.1 Introduction

3.1.1 Motivation

The services offered by mobile devices have shifted from communication only to computing, storage, sensing, and communicating. This is a new revolution of Internet where people and smart objects live connected in smart environments. Today, mobile handsets are the platform used for launching and running abundant and various services and applications [47]. Indeed, new mobile applications are integrating the market, producing tsunami of data traffic and imposing high resources requirements. Handset battery lifetime is henceforth reduced due to communication, but also computational tasks that are run over the equipment hardware. Offloading computational tasks from mobile handsets to the '*cloud*' is seen as an effective solution to limit computing energy consumption at the handsets side. Mobile cloud computing offers the potential of increasing mobile devices computational and storage capabilities, and extending their battery lifetime. Nevertheless, computation offloading comes at the expense of generating extra communication load, the offloading traffic (uplink, intra-cloud, and downlink traffic). Uplink communication costs vary with the distance between the handset and the cloud gateway and the wireless channel conditions. On another hand, computation tasks are generally constrained by time limits and memory requirements. The first proposed mobile computation offloading decision algorithms are mainly based on the energy trade-off between local computation at the mobile device and computation offloading to the serving cloud. However, mobile users' Quality of Experience (QoE) must be taken into account in the offloading decision process. The energy minimizing decision does not necessarily deliver the best QoE. On one side, all applications requirements should be respected (latency constraints, memory requirements, tasks offloadability). On the other side, offloading decision should allow to minimize energy consumption on the mobile side without losing experience quality. Hence, energy consumption is not the only parameter that affects the offloading decision. Integrating all parameters that affect offloading decision is fundamental to guarantee a good QoE while minimizing the offloading process cost. In this chapter, we look into computation offloading decision from the mobile devices to the cloud with a single hop wireless communication.

3.1.2 Related Work

Several studies related to offloading decisions for mobile cloud computing frameworks have been proposed. Nevertheless, the majority of existing work proposes offloading decision processes and algorithms based on the energy trade-off without considering all parameters that affect the offloading decision [49, 81–86].

Maui [81], *CloneCloud* [83], and *ThinkAir* [82] are frameworks that enable mobile computation offloading to the cloud. Both *Maui* and *CloneCloud* propose to solve the problem of optimizing applications partitioning between local execution and offloading, with quite similar optimization targets: maximize energy savings for *Maui*, and minimize execution time or energy consumption for *CloneCloud*. Both framework architectures include solver and profiler entities, but they differ in implementation. In *Maui*, the profiler monitors program and network characteristics continuously at runtime, no persistent result is stored across multiple runs, and the solver is periodically run at runtime. Whereas *CloneCloud* creates device clones operating on the cloud. It profiles and solves, before the partitioned application begins, by assuming different running conditions, and profiling results are used to generate partition configuration files. Nevertheless, building profiles at runtime is energy consuming for the mobile handset and offline profiling does not easily cover all possible running conditions.

Chen *et at.* propose an offloading decision process based on communication, compilation and communication energies comparison [84]. With the goal of conserving energy on the mobile client, the decision process is launched when a method is invoked. It consists in comparing different energy consumption values corresponding to different of task interpreting strategies (remotely, locally, through an interpreter), and choosing the alternative with the lowest energy cost. Energy cost takes into account estimated energies for bytecode interpretation through local computation or remote execution, and parameters representing communication power, the size of data to be sent and received in case of remote computation, and the complexity of local execution. Communication power, data size, and computation complexity are predicted as weighted average of current and past values. Various execution strategies are considered in Chen's *et al.* work. A decision is made over both the computation granularity (bytecode form or with code compilation) and the computation location (remote or local). The decision aims at reducing energy consumption at the mobile side, but it does not take into account latency constraints of each method or task. In fact, a local execution of a method can be more energy saving in some cases but it could require a period of time larger than the latency constraint of that task. Focusing only on energy consumption does not guarantee mobile users' QoE. In addition, no memory requirements are taken into account.

Another study by Kumar *et al.* shows that the energy saved by computation offloading depends on the wireless bandwidth and the amount of computation to be performed [49]. The authors also discuss some offloading challenges such as security, reliability, and real time data. This study concludes that not all applications are energy efficient when offloaded if additional energy overhead for security and reliability is considered. The energy saving is represented with respect to the system bandwidth and the amount of computation.

Related work algorithms incorporate only the energy consumption in the mobile offloading decision algorithms. However, many other parameters and conditions could be introduced in the algorithm in order to solidify decisions and adapt them to the system conditions. If an offloading decision only depends on energy saving, the chosen computation strategy (local or remote) could violate, among others, latency and memory constraints. Indeed, mobile handsets limitations, other than energy consumption, have an influence the offloading decision. In addition to the limited battery lifetime, mobile handsets have limited computational capacity and memory space. Furthermore, the offloading decision depends strongly on the application to be offloaded. Mobile applications are characterized by latency constraints, computation complexity, and memory requirements that should all be met. Therefore, an optimization trade-off approach that is based on a single parameter does not guarantee meeting the users' expected. Additionally, not all tasks can be offloaded. Some computational tasks require the use of specific information available on the mobile handset, and/or the use of mobile devices hardware.

Other related work in literature proposes mobile offloading decision algorithms that take into consideration more than one parameter [10, 87–90]. Barbarossa *et al.* propose a joint offloading decision and resource allocation solution [10], . The offloading decision process takes into account both latency constraints of the tasks, and computation queues state. Computation queues represent a buffer on the mobile side where all computations that are not yet executed are stored. First, the set of users that are able to transmit data in a defined block of time lower than the imposed latency limits is identified. This set is reduced in order to satisfy the global condition of the sum of the allocated computation capacities for all task is smaller than the total surrogate computational capacity. Reducing the set consists in removing mobile users, i.e., forbidding some users from offloading computations in the considered time block. Users with larger queues are prioritized to remain in the set in order to minimize the cases of queues instability at the mobile side. Users that are removed from the set execute their tasks at the mobile handset. For users admitted to transmit, transmit power and allocated computational capacity at the cloud side are jointly optimized. Joint

optimization helps taking advantage of all the server capacity in order to meet latency constraint while minimizing as much as possible the transmission power. The joint optimization is formulated as a convex optimization problem. The optimization problem solutions takes full advantage of the server computation capacities in order to meet the latency constraints while minimizing the transmission power which varies according to the channel state. This approach could also lead in case of bad channels conditions to an increase in the transmission power.
Gu *et al.* propose a fuzzy control approach that consists on initiating an offloading decision process whenever the mobile handset memory status is critical [88]. The available memory space is represented with three linguistic fuzzy states: *low*, *moderate* and *high*. When memory availability is low, an offloading decision process is triggered. The offloading decision optimization is based on a multi-criteria decision function which takes into account bandwidth, delay and memory. The initiation of the offloading decision process in this approach does not include other important parameters such as mobile battery level, latency constraints, and available computational capacity.
Gao *et al.* propose a strategy where time and energy are evaluated for both local and offloading computations [89]. A computational task is allowed to be offloaded when offloading is less time and energy consuming than local computing. After taking an offloading decision for a task, a tasks clustering algorithm is called. The clustering algorithm aims at taking similar decisions for the tasks that communicate with each other, in order to reduce communication energy consumption. Even though both energy and time are considered as decision parameters, memory requirements and availability is ignored.
Kovachev *et al.* [90] consider energy, memory, and execution time in a multi-criteria utility function. The multi-criteria function is introduced to avoid the complexity of solving an optimization problem that jointly optimizes all the criteria. However, multi-criteria utility functions require a fine tuning of the weights associated to each parameter in this function. The multi-parameter optimization complexity is twofold. First, the problem itself is complex to solve, and second the decision needs to be refreshed whenever system conditions change.

3.1.3 Contribution

In this chapter, we tackle the problem of multi-parameters optimization complexity, which is a major bottleneck of classical multi-parameters optimization techniques. We propose a novel Sequential Multi-Parameters Offloading Decision algorithm under the name of SM-POD . We adopt a sequential approach where decisions are sequentially made according to a decision tree. The proposed approach introduces a multitude of parameters in the decision process while keeping it simple to implement. We propose indeed to approach the multi-parameters optimization with a multi-fold task classification. We define successive and nested classifications of tasks at the mobile handset. Calvanese Strinati *et al.* propose a single parameter classification to improve downlink packet scheduling [91], . Following the same intuition, we introduce a multi-parameters classification. In the proposed approach, we adapt the offloading decision to the current state of the system that is defined through the series of classifications. We classify computational tasks into *virtual* buffers each of which is associated with an offloading decision. Requested tasks that are offloadable are classified and buffered depending on the criticality of their latency constraints. Time critical tasks are to be executed immediately, either locally or at the server. According to the application requirements and mobile available resources (computational capacity, memory space, battery life), the offloading decision is made. In order to keep high user QoE, time critical tasks are prioritized for offloading, regardless the offloading costs. Less time critical offloadable tasks are allowed to be offloaded if the offloading communication cost low. The proposed approach is a solution for incorporating several parameters in the computation offloading decision at the mobile

side without imposing complex optimization solving. Note that the offloading decision concerns the single hop communication between MUEs and their serving cell. The proposed algorithm does not propose a resource allocation algorithm at the mobile edge cloud. The novelty of this work is based on a patent [P2] and a conference paper [C1].

3.2 System Model

We consider a system with K users served by either a macro base station or a small cell base station (femtocell), within a distance d. We consider uplink connection, between MUE (mobile user equipment) and the serving base station, with a bandwidth B. Instantaneous uplink bit rate is maximized based on adaptive modulation and coding scheme (AMC) [78]. We adopt a non-ergodic Rayleigh channel model with path loss exponent β, noise power N_0, and fading channel coefficient h_k. We assume perfect estimation of the channel coefficients h_k. The channel is assumed constant for a whole transmission period, with a coherence time T_c. A parameter Γ in the channel model indicates the Signal to Noise Ratio (SNR) margin to guarantee the minimum error rate. We consider a constant transmission power P_{Tx} through the defined channel. The maximum information rate that can be achieved through this channel is calculated using the following equation:

$$R_{Tx} = B \times log(1 + aP_{Tx}) \text{ where } a = \frac{|h_k|^2}{\Gamma d^{\beta} N_0} \tag{3.1}$$

Indeed, R_{Tx} determines the maximum number of tasks that can be transmitted through the wireless link in one time slot.

Each MUE is characterized by a set of parameters summarized in Table 3.1.

Parameter	Description
F	CPU computational capacity [CPU cycles/sec]
Tot_E	Total energy capacity of the mobile handset [Wh]
EPI	Energy consumption per cycle [J/CPU cycle]
M_{av}	Current amount of available memory [MB]
B_{lev}	Available battery level percentage

Table 3.1: Mobile Handset Characteristics.

Applications are launched by the user at the mobile handset. The applications arrival is modeled as a Poisson process with a rate λ, where λ represents the number of launched applications in a time window T_w. We consider that an application call generates a burst of tasks to be computed. Each generated task is a set of instructions requiring W CPU cycles that has to be executed with a required memory m, and a maximum latency Δ. A parameter ρ indicates if the task is offloadable ($\rho = 1$) or not ($\rho = 0$). The percentage of tasks that cannot be offloaded is defined by a parameter α_{no}. In case the task is offloadable, a parameter W' indicates the number of bits to send to the small cell. Table 3.2 resumes applications related parameters.

We compute energy consumption relative to each task. For the tasks that are computed locally, the mobile handset energy consumption is evaluated as the product of the number of executed CPU cycles and the energy consumption per cycle:

$$E_{local} \text{ [J]} = W \times EPC \text{ [J/CPU cycle]} \tag{3.2}$$

For the tasks that are offloaded, the energy consumption of the mobile handset is evaluated based on the mobile user power consumption model proposed by Jensen *et al.* [92]. In the adopted model, the consumed energy for transmission is evaluated as:

$$E_{offloading} = P_{Tx,C} + P_{Tx,BB} + P_{Tx,RF} + P_{con} \tag{3.3}$$

$\mathbf{P_{Tx,C}}$: Power consumption of active transmission chain.
$\mathbf{P_{Tx,BB}}$: Power consumption of the baseband (BB) components. It depends on the uplink data rate R_{Tx} as the following equation:

$$P_{Tx,BB}[\text{mW}] = 34.5 + 0.87 R_{Tx}[\text{Mbits/s}] \tag{3.4}$$

$\mathbf{P_{Tx,RF}}$: Power consumption of radio RF components. It depends on the transmission power S_{Tx} as the following equation:

$$P_{Tx,RF}[\text{mW}] = -943 + 117 S_{Tx}[\text{dBm}] \tag{3.5}$$

$\mathbf{P_{con}}$: Average power consumption in connected mode. It is equal to 1.35W, according to [92].

Parameter	Description
λ	Applications arrival rate
ρ	Application offloadability $\in \{0,1\}$
α_{no}	Ratio of non-offloadable tasks
ω	Task number of CPU cycles
m	Task memory requirement
Δ	Application maximum tolerated latency
W	Number of bits to be sent in case of offloading

Table 3.2: Applications and tasks Characteristics.

3.3 Problem Statement

Consider a set $\mathcal{K} = \{1, \cdots, K\}$ of K users served by a set $\mathcal{N} = \{1, \cdots, N\}$ of small cells. Mobile users $k \in \mathcal{K}$ have computational tasks to accomplish. Each task is characterized by a maximum tolerated latency and memory requirements. MUEs offload their computation through sending the computation requests to their serving cell. In order for MUE to make offloading decisions for each of the tasks, many parameters are taken into account. The first application characteristic that is considered is the tasks the ability of the task to be offloaded, or its *offloadability*. Not all computational tasks are offloadable. Many tasks require launching embedded sensors and hardware on the mobile phone. Tasks that require using the microphone, or using the hart beat sensor embedded in some handsets are examples of tasks that cannot be offloaded to the cloud. Furthermore, the possibility of computing tasks locally is subject to the availability of sufficient resources of computation, memory, and battery life. Offloading decision algorithms in literature are mainly based on energy consumption comparison between local computation and offloading. Energy saving is indeed an important aspect at mobile handsets; however, minimizing the energy consumption should

not affect the perceived QoE by mobile end users. Therefore, applications latency constraint is an important parameter to consider in the offloading decision process. In addition, offloading tasks requires wireless transmission of data to the cloud gateway. Wireless channel quality affects both transmission latency and handsets power consumption, and thus, should be included in the offloading decision. To the best of our knowledge, any offloading decision algorithm that considers this variety of parameters and keeping low complexity have been proposed. Keeping low complexity of computation offloading decision algorithms is important in case the algorithm is executed at the mobile handsets. Engaging the handsets in complex computations for making an offloading decision is both time and energy penalizing. Hence, to find a quick, effective, and low energy consuming offloading decision, we propose an algorithm based on successive and nested classifications—SM-POD. The SM-POD algorithm can be easily implemented on handsets, has low complexity, and helps increasing handsets battery lifetime. The algorithm exploits the delays imposed by each application to find the *right moment* to offload or compute tasks without violating any constraint, and keeping low the mobile handset energy consumption.

3.4 Proposed Offloading Decision Algorithm: SM-POD

We propose to perform a series of classifications that joins a multitude of parameters, without including them in a complex optimization problem. Each of the tasks classifications is based on a characteristic of either the mobile handset, the computational task, or the communication channel. At the end of the successive classifications, tasks will be classed in various *virtual buffers*. By *virtual buffer* we refer to a set of computational tasks that share the same offloading decision. The novelty of this work is twofold. First, the offloading decision process complexity is reduced while a dependency on a variety of parameters is considered. SM-DOP trades complex optimization problem solving with a series of successive and nested tasks classifications. Second, mobile energy consumption is reduced by offloading tasks depending on time criticality, handset available resources, and channel conditions.

At each time slot, the algorithm of mobile application offloading decision is run on the set of tasks generated by the launched applications. The proposed SM-POD algorithm is summarized as follows:

Step 1: Offloadability classification

First step of the proposed offloading decision algorithm is to divide tasks that are offloadable and ones that are not. To this end, in a first classification, computational tasks are classified into two distinct sets. The classification is based on the tasks characteristic ρ that specifies if the task is offloadable or not (see Figure 3.1). The first set, ***"Off"***, includes all tasks that have the possibility to be offloaded ($\rho = 1$). The second, ***"NOff"***, includes all tasks that cannot be offloaded by characteristics definition ($\rho = 0$).

Step 2: Urgency classification

Then, the algorithm classifies tasks in both sets *Off* and *NOff* as *urgent* and *not urgent* tasks as shown in Figure 3.1.

- ***Off* set:** An offloadable task is labeled as *urgent* when the remaining latency is less than a predefined percentage Δ_{th} of the original latency constraint Δ. Computational latency

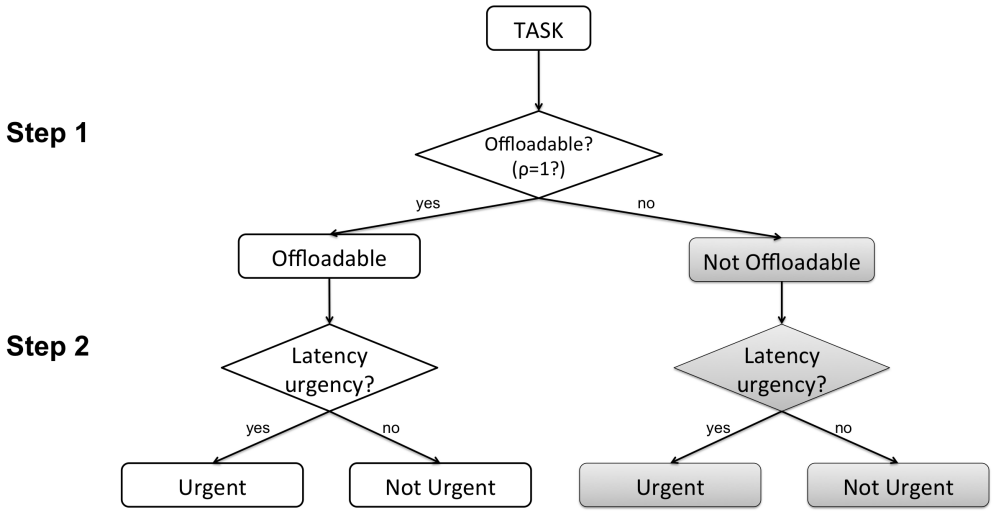


Figure 3.1: First and Second classification steps of the proposed algorithm.

constraints are seen as Time To Live (TTL) limits beyond which the service delivery fails. Computational tasks are set as *urgent* when a TTL margin is reached, in other terms when Δ_{th}% percent of the initial TTL Δ have passed. The adopted margin is either set as a constant, or updated according to the current system state. Communication channel conditions and the number of bits to transmitted, as well as statistics over the computation offloading success can be used for adapting the TTL margin. We consider a constant ν that takes into account the current system state parameters. Then computations are classified as *urgent* if:

$$\Delta_{th}.\Delta + \nu \geq \Delta_R \tag{3.6}$$

where Δ_R is the time remaining to reach the maximum latency constraint (current TTL). This classification divides the *Off* set into two parts: the *urgent* offloadable tasks set referred to as ***"OffUrg"*** and the *non-urgent* offloadable set referred to as ***"OffNUrg"***.

- ***NOff* set:** Tasks in the *NOff* set cannot benefit from computation offloading, they must be computed locally at the MUE. Even for local computation, we classify tasks are urgent or not. To make such a classification, we account on the mobile handset computational capacity. We check if all *NOff* tasks can be computed at once while meeting each task latency constraints. The check is done for equal computational capacity distribution among all tasks. Total available computation capacity at the MUE is defined by the parameter (MH_{cap}). Each task is then given a computational capacity of $F = \frac{MH_{cap}}{|NOff|}$ where $|NOff|$ is the $NOff$ set cardinal number. Tasks that cannot be executed with the allocated computational capacity are identified through the following classification criterion:

$$\Delta < \frac{W}{F} + \varepsilon \tag{3.7}$$

where W is the task computational load. Tasks that verify this condition are set as *locally urgent* and are added to the not offloadable urgent set referred to as *NOffUrg*. The *NOffUrg* tasks must be computed locally and therefore local resources are allocated. The remaining tasks, that do not verify the above condition, can be differed and are added to the set referred to as *NOffNUrg* grouping not offloadable not urgent tasks.

Step 2 allows then to allocate resources for tasks that are the most demanding in terms of computational capacity first. This step can be seen as a scheduling process that prioritizes *urgent* tasks.

Step 3: Resources availability check

The third SM-POD step concerns the offloadable urgent tasks represented by the *OffUrg* set. Basically, each of these tasks has the possibility to be offloaded ($\rho = 1$) and is not prevented from being computed locally ($\rho \neq 0$). However, local computation of some of these tasks may not be possible due to a lack in available resources at the mobile handset. In this case, the tasks have to be offloaded. This classification aims to find the tasks that have to be offloaded, i.e., it finds tasks for which resources demands are higher than what the mobile handset can offer. This decision is based on a series of tests taking into account computational capacity, memory requirements, and mobile battery consumption.

A task is classified as "*should* be offloaded" if any of the following conditions is satisfied:

- Mobile is battery level is critical (lower than a predefined threshold $BLev_{off}$).
- The task consumes more than a predefined percentage of the available battery level BAT_{off}.
- Task memory requirements surpass the allowed percentage of available memory at the mobile handset MEM_{off}.
- The task requires a computational capacity greater than a predefined percentage of the total locally available capacity CAP_{off}.

The offloading thresholds $BLev_{off}$, BAT_{off}, MEM_{off}, and CAP_{off} are parameters that can be defined by the user through its mobile equipment operating system.

The set of tasks verifying one of these conditions is referred to as ***"SOffUrg"*** in reference to offloadable urgent tasks that should be offloaded. The remaining tasks form a set that we refer to as ***"COffUrg"*** in reference to offloadable urgent tasks that could be either offloaded or computed locally. This classification is represented in Figure 3.2.

Step 4: Energy consumption comparison

Each of the tasks in *COffUrg* is checked for whether it can be computed locally applying the same conditions in section Step 3. If the task is allowed to be computed locally, then two options are available: offloading or local computation. In order to take a final decision between both options, the mobile handset energy consumption is investigated. E_{local}, the energy spent in case of local computation of this task using Equation (3.2) and the energy $E_{offloading}$ spent at the mobile handset in case of offloading using Equation (3.3) are compared. If $E_{local} < E_{offloading}$, the task is computed locally and transferred to the *NOffUrg* set, otherwise, the task is offloaded. In case of a local computation decision, memory and capacity resources are allocated to the task in question. Tasks that are decided to be offloaded are added to the set *SOffUrg* which represents the set of tasks whose offloading is necessary. This classification is also shown in Figure 3.2.

Step 5: Resources aware decision for non urgent tasks

SOffUrg is the set of tasks to be offloaded. The number of bits that should be sent radio link from MUE to the cloud is known.
The transmission power is S_{Tx} and the transmission rate is R_{Tx}. In the case where data rate does

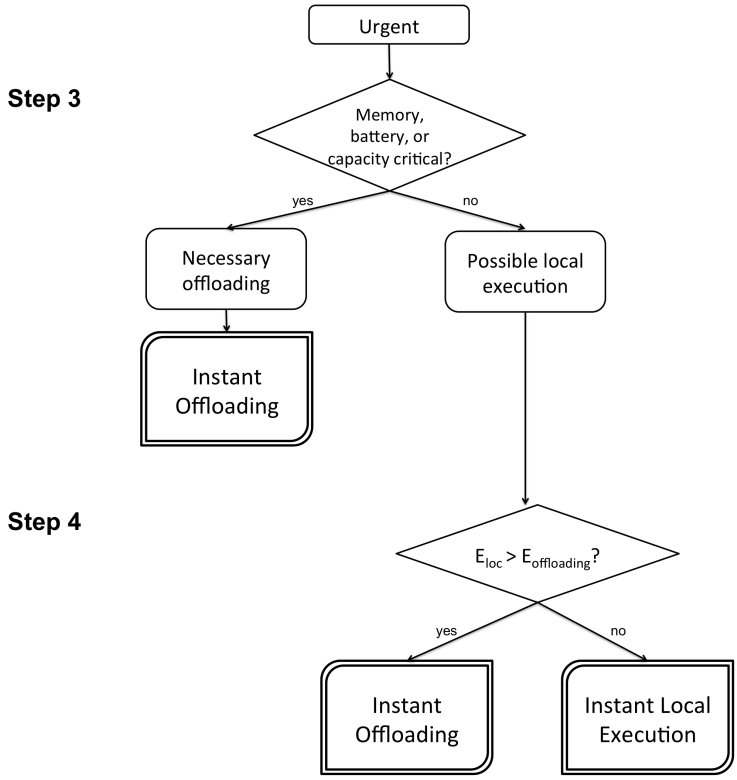


Figure 3.2: Third and fourth classification steps of the proposed algorithm.

not allow the transmission of all of the tasks offloading traffic, tasks with lowest latencies are prioritized. The mobile will offload as much tasks as possible, and the remaining tasks will be differed to a posterior time slot.
In the case where the number of bits to be sent is less than what the channel capacity offers, we refer to the transmission channel conditions. The channel coefficient is compared to a statistical average channel coefficient calculated and updated over time. If the current channel realization is above this average, it is considered that the channel is in a relatively *opportunistic* state. The novelty in this case is to include current channel conditions in the decision process. Opportunistic channel conditions allow data transmission at a lower cost. Indeed, better the channel conditions allow having greater aggregated throughput, and thus higher energy efficiency. This is seen as an opportunity to offload non urgent offloadable tasks from the set *OffNUrg*. Priority is given to tasks in *OffNUrg* that have lower latencies.

Offloading non-urgent offloadable tasks can be seen not only as an opportunistic utilization of the radio link, but also as a tool to alleviate the system in the future so it would not face *urgent* tasks too often.

The set of tasks that are going to be computed locally is identified (*NOffUrg*). Therefore, the remaining mobile resources can be computed. Following the same conditions as in 3.4 to the set *NOffNUrg*, we assign more tasks to be computed locally with the remaining available resources. Tasks that could not be computed are deferred, i.e. no decision is anticipated for these tasks, they will be re-classified in the next time slot. The *non-urgency* of these tasks permits deferring the decision to a future time slot, since the algorithm guarantees that whenever the task become *urgent*, it will be associated to an execution decision. This step is illustrated in Figure 3.3.

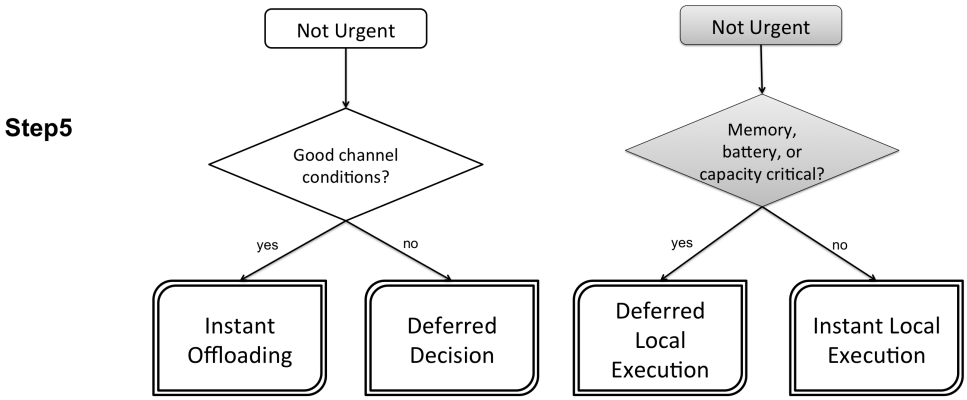


Figure 3.3: Last steps of the proposed algorithm.

The proposed SM-POD algorithm can be seen as a smart buffering process. The algorithm distributes the tasks at the mobile side into four different virtual buffers. Each of these buffers is associated with an offloading decision. Table 3.3 summarizes the virtual tasks buffers and the associated offloading decision. Figure 3.4 shows the whole algorithm steps and indicates the corresponding buffers to each of the decisions.

Buffer	**Offloading Decision**
$NOffUrg$	Instant local computation
$SOffUrg$	Instant task offloading
DLE	Deferred local task execution
DD	Deferred offloading decision

Table 3.3: Computation offloading virtual buffers labels and decisions

3.5 Numerical Evaluation

In this section we investigate the offloading efficiency achieved with the proposed SM-POD algorithm. Our evaluation highlights cost reduction in terms of handset battery life, memory, computational capacity, and tasks latency violation. We adopt the same parameters as the system model described in Section 3.2, considering a single user served by a femtocell base station, within a distance $d = 5$m from the serving station. The considered uplink bandwidth is of $B = 20$MHz (which is among the standard LTE uplink bandwidths), the path loss coefficient $\beta = 5$ that complies with a multi-level building scenario [93], and the noise power as $N_0 = 10^{-3}$.We consider a transmission power of $P_{Tx} = 0.2$W. The simulations are averaged over approximately 2.10^5 channel instances per hour.

The mobile handset EPI is estimated between 17nJ and 19nJ which is in line with some Intel processors EPI (Pentium, Pentium Pro, Dual Core) [94], its total energy capacity between 4Wh and 8Wh, its available memory of $M_{av} = 5$MB.

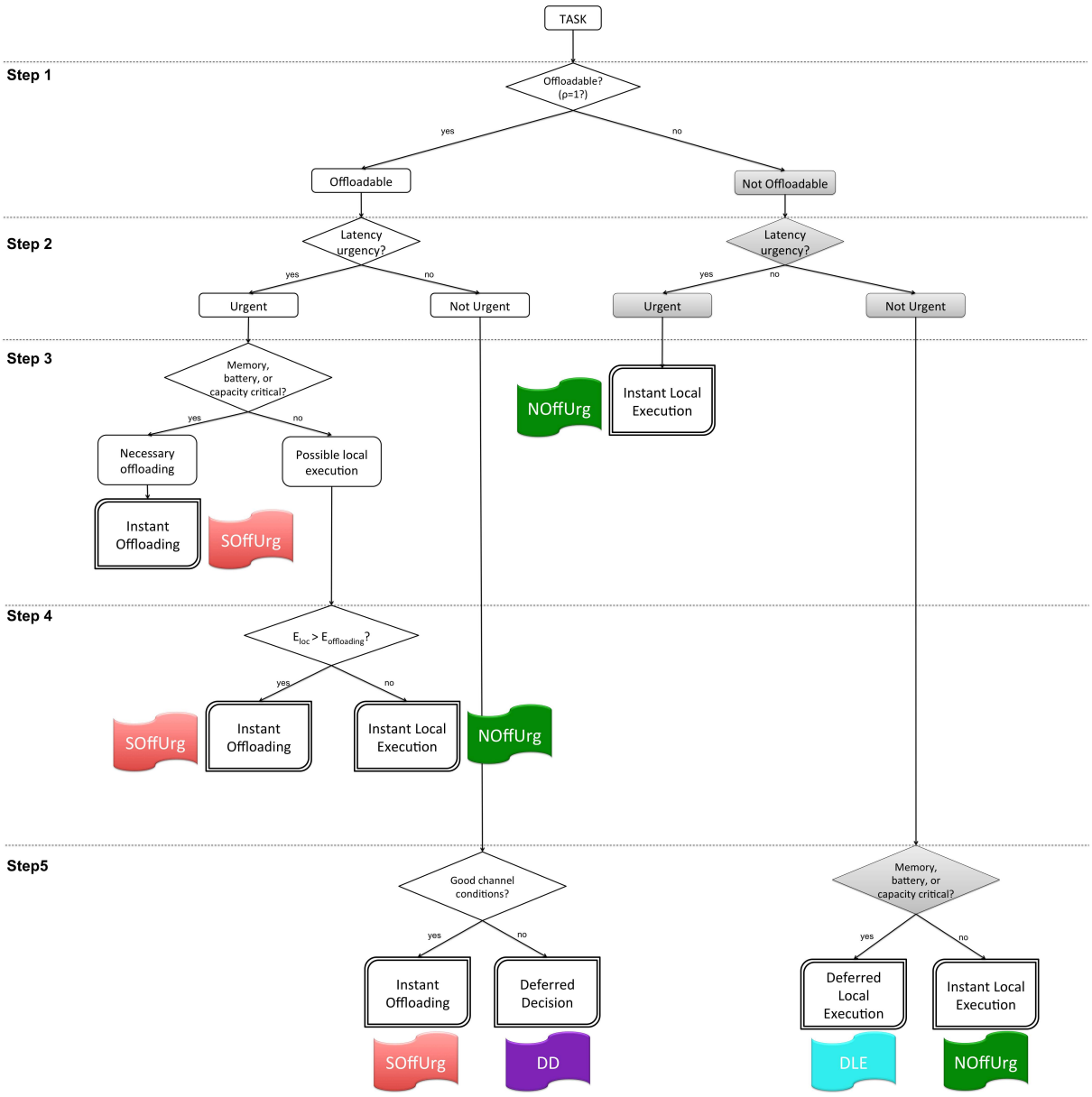


Figure 3.4: Successive classifications of the offloading decision proposed algorithm.

We consider application calls that generate a burst of tasks (10 tasks). Each task is characterized by a required memory 1KB $\leq m \leq$ 1MB, a number of bits to be offloaded 1KB $\leq N \leq$ 20KB, a latency constraint 90ms $\leq L \leq$ 300ms and a number of CPU cycles to be executed that varies with each task and different scenario assumptions.

The application arrival modeled as a Poisson distribution with a rate $\lambda = 2$ for a time window of $T_w = 10$ms. The percentage of tasks that cannot be offloaded is $\alpha_{no} = 40\%$. The constraints defined in section 3.4 are set as: $BLev_{off} = 20\%$, $BAT_{off} = 30\%$, $MEM_{off} = 70\%$ and $CAP_{off} = 50\%$.

According to the work by Kumar *et al.*, the offloading decision must be to never offload when large amount of communication is needed with relatively small amount of computation, and must be to always offload when large amount of computations is needed with relatively small amount of communication [49]. The offloading decision is based on energy consumption. To benchmark the proposed algorithm, we run simulations for different scenarios that represents scenarios for which we are in neither of the cases above. In such scenarios, Kumar *et al.* state that the decision depends on the available bandwidth (Figure 1.10). We use this offloading decision criterion to benchmark the proposed SM-POD algorithm. The adopted scenarios vary in channel conditions and amounts of computation per task. Each scenario is defined by a combination of two parameters defining the channel conditions and the average computation size of the requested tasks. The set of parameters are defined as follows:

- $\{\alpha_{min}, \alpha_{av}, \alpha_{max}\}$ representing respectively a *low*, *random* and *good* channel coefficient average. For *low* channel coefficient we consider the lowest 20% of a random Rayleigh channel coefficients generation. For *random* channel we adopt a random generation of Rayleigh channel. For *good* channel coefficient average, we consider the highest 20% of the random coefficients.

- $\{TC_{min}, TC_{mix}, TC_{max}\}$ representing respectively a *small*, *mixed* and *large* amounts of computation for each task.

We show numerical results for the three following representative scenarios:
Scenario 1 - *Max-Max*: good channel conditions (α_{max}) and large amounts of computation per task (TC_{max}). For good channel conditions, we select the best 20% of the generated channel instances.
Scenario 2 - *Min-Min*: bad channel conditions (α_{min}) and small amounts of computation per task (TC_{min}). For bad channel conditions, we select the worst 20% of the generated channel instances.
Scenario 3 - *Mix-Mix*: random channel conditions (α_{mix}) and mixed amounts of computation per task (TC_{mix}). For random channel conditions, no selection over the generated channel instances is made.

In order to evaluate the algorithm performance, we compare to the following reference algorithms:
No Offloading Tasks are never offloaded.
Total Offloading Offloadable tasks ($\rho = 1$) are always offloaded.
Energy Reference Offloading Decision (EOD) Task offloading is based on the offloading energy trade-off between local computation energy cost and offloading cost.

$$E_{local} \lesseqgtr E_{offloading} \tag{3.8a}$$
$$E_{UE,loc} \lesseqgtr E_{UE,off} + E_{femto} \tag{3.8b}$$

where $E_{UE,loc}$ is the energy spent at the mobile handset for locally computing the requested task. $E_{UE,off}$ is the energy spent at the mobile handset for sending the necessary information to the small cell where the task will be offloaded. $E_{femto} = W.f$ is the energy spent at the femtocell

for computing the requested tasks. W is the number of cycles to be executed at the femtocell, and f is the computation capacity accorded to the execution of the cycles. This entity depends on various components such as system implementation, femtocell clusterisation, CAC (Call Admission Control) at base stations, etc. Two approaches are possible: The first one compares the energy of the whole system in order to decide on offloading. In this case, $E_{femto} \neq 0$. Another possible approach, which is adopted in this work, is the user centric approach. In this case, the mobile handset searches only for reducing its own energy consumption, and the decision does take into account the whole system energy efficiency. It consists on comparing the energy spent only by the mobile handset for the cases of both offloading and local computation. In this case, $E_{femto} = 0$ and the trade-off equation will be reduced to:

$$E_{UE,loc} \lessgtr E_{UE,off} \tag{3.9}$$

On figures 3.5, 3.6, and 3.7, blue curves with *plus* marks represent battery level in the case where the proposed SM-POD algorithm is applied. Black curves with *point* marks represent the case of energy reference offloading decision algorithm (EOD). Green *circle* marked and red *diamond* marked lines represent, respectively, the cases of total offloading and no offloading.

Figure 3.5 shows the mobile handset battery discharge for **Max-Max** . In this case, offloading is beneficial, supported by good channel conditions. Data transmission, in case of offloading, is done through high capacity links. The graph shows that the solution that computes all tasks locally costs the most in terms of handset energy. Reference offloading and total offloading share the same results because this is an extreme case where offloading is less battery consuming than local computing in this scenario and thus the algorithm based on the energy trade-off will always decide to offload the requested tasks. Figure 3.5 also shows that SM-POD algorithm outperforms all other algorithms in terms of handset battery lifetime. In fact, taking advantage of good channel conditions to deal with non-urgent tasks prevents the system from having a large amount of urgent tasks to deal with in the future time slots. Therefore, by opportunistic transmission of some tasks on better channel conditions, data is sent using lower transmission power, and thus energy is saved and battery lifetime is prolonged. Using SM-DOP, battery lifetime is 2.3 times longer compared to the total offloading case, and approximately 1.5 times longer compared to full local computing. Table 3.4 resumes the battery lifetime, CPU memory overflow, and CPU capacity outage results. It shows that the proposed algorithm prevents the system from having CPU memory or capacity outage while respecting latency constraints. For other algorithms, local CPU resources have experienced outage in at least 4.3% of the times, while achieving lower battery lifetime. Including CPU memory and computational capacity in the decision process prevents the outage use case from taking place.

For **Min-Min** scenario, offloading tasks is not beneficial. Considering the extreme case of bad channel conditions, offloading data is both time and energy consuming. Figure 3.6 shows the battery discharge in such conditions. Results show that the algorithm that does not allow offloading outperforms the total offloading algorithm. In this case, the EOD algorithm that is based on the energy trade-off gives results that are close to the no offloading algorithm decisions, which are less energy consuming. SM-POD is more energy consuming than the reference offloading and the no offloading algorithms. This is due to the fact that the proposed algorithm decide to offload offloadable tasks that are assigned as *urgent* regardless of the channel conditions. This affects the energy consumption of the mobile handset, but on the other hand will guarantee a good user QoE. As it is shown in Table 3.4, latency constraints are violated only for the case of total offloading up to 3%,due to bad channel conditions. The proposed algorithm trades the outage situations that occur in both the No offloading and EOD use cases by increasing energy consumption.

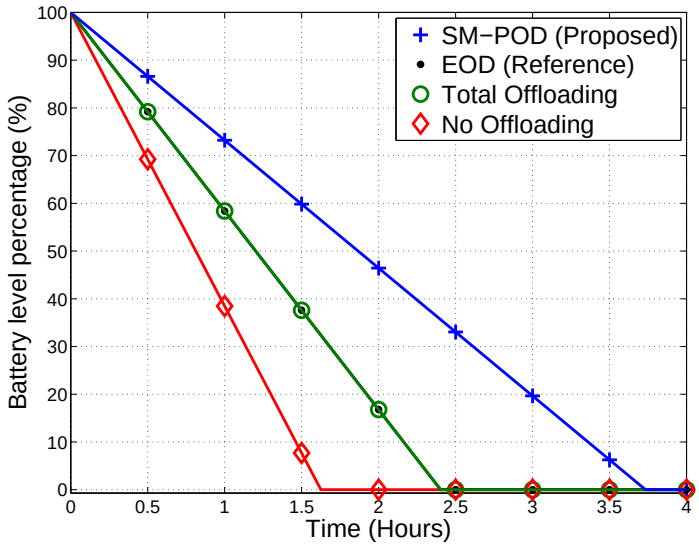


Figure 3.5: Max-Max Scenario: Mobile battery discharge due to tasks computation/offloading for all considered algorithms

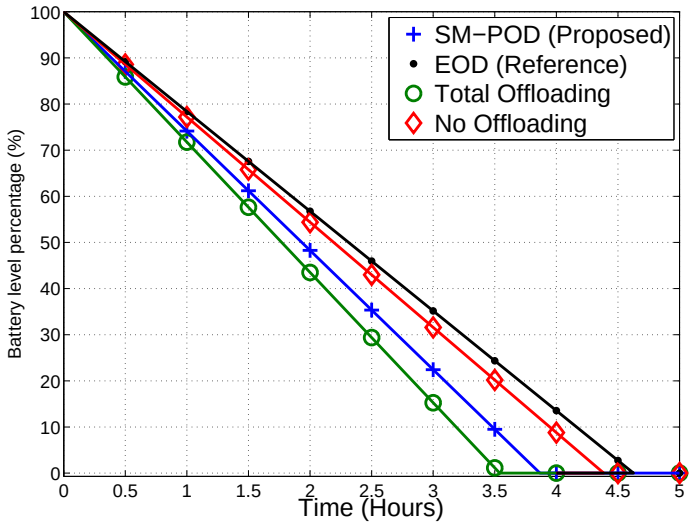


Figure 3.6: Min-Min scenario: Mobile battery discharge due to tasks computation/offloading for all considered algorithms

Mix-Mix scenario results are shown on Figure 3.7. The figure shows that even in this random case scenario, the proposed algorithm outperforms the reference EOD algorithm. The latter is clearly seen as less energy consuming than the no offloading and the total offloading algorithms. The ability of both algorithms to adapt the decision to the current situation results in better performance and prolonged battery life. However, the flexibility of SM-POD and the fact that it encompass a multitude of parameters, allows it to achieve higher end performance. It adapts the offloading decision to current system parameters considering at the same time the application requirements, the handset available resources and the radio channel quality. The proposed algorithm prolonged the battery life 1.45 times in these random conditions scenario compared to worst case scenario (No offloading)

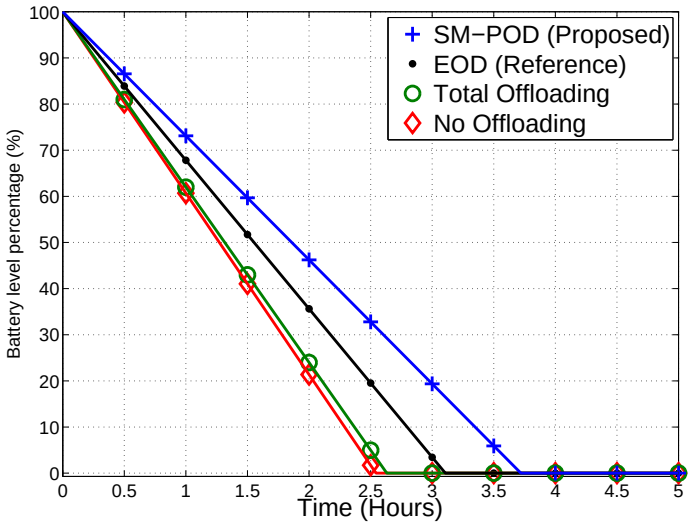


Figure 3.7: Mix-Mix scenario: Mobile battery discharge due to tasks computation/offloading for all considered algorithms

In addition of energy savings at the mobile handset, SM-POD prevents the mobile from suffering of CPU memory overflow and computational capacity outage and allows the user to always have good QoE by respecting each task latency constraint. Those benefits are also validated by simulations (see Table 3.4).

The proposed algorithm, allows, thanks to its flexible structure, the integration of several parameters in the offloading decision. By simple classifications and comparison steps, it guarantees a user good quality of experience, even if sometimes this comes at the cost of increased energy consumption and thus shorter battery lifetime. Nevertheless, in the random scenario case, it proved that it could adapt to the changing situations and exploit different latency constraints and varying channel conditions to save energy consumption and extend battery lifetime.

3.6 Conclusion

With the proliferation of mobile-enabled applications, and the computations they require, mobile computation offloading has a grown as an effective solution for enabling handsets to do more. Mobile devices have access to greater computational resources, and larger storage space, if they

Scenario	Algorithm	Battery Lifetime [mn]	CPU memory overflow (%)	CPU capacity outage (%)	Latency Violation (%)
Max-Max scenario (1)	SM-POD Proposed algorithm	**223.9**	**0**	**0**	**0**
	No offloading	97.86	2.61	5.02	0
	EOD Reference algorithm	144.77	0.17	4.31	0
	Total offloading	144.77	0.17	4.31	0
Min-Min scenario (2)	SM-POD Proposed algorithm	**235.64**	**0**	**0**	**0**
	No offloading	263.19	1.29	0	0
	EOD Reference algorithm	277.8	0.17	4.31	0
	Total offloading	231.02	0	0	0
Mix-Mix scenario (3)	SM-POD Proposed algorithm	**223.96**	**0**	**0**	**0**
	No offloading	153.09	0.3	0	0
	EOD Reference algorithm	187.09	0.02	0	0
	Total offloading	158.87	< 0.01	0	3

Table 3.4: Simulations results for scenarios 1, 2, and 3

offload computational tasks to the cloud. However, in this chapter, we show how computation offloading is not always beneficial for mobile users. In mobile cloud computing, offloading required sending computational data to the cloud. The energy consumption of the data transmission varies with the size of the data to be sent, and the transmission channel conditions. A computation offloading decision strategy that incorporates all the aspects that affects the decision, is needed. Basically based on a simple energy comparison, proposed decision algorithms do not consider all the parameters that could affect the offloading decision. Such observations have led to the design of a novel multi-parameter offloading decision algorithm, characterized by a series of successive and nested low complexity classification operations to be executed at the mobile side. In this chapter, we propose an approach that exploits multi-fold task classification to deal with multi-parameters optimization. No complex optimizations or multi-criteria utility function based linear programs optimization are needed. The proposed algorithm classifies the computational tasks in *virtual* buffers, each of which is associated with an offloading decision. Classifications depend on several parameters including tasks offloadability, time criticality, handsets resources availability, energy consumption, and radio channel conditions. The classifications lead to one of the following offloading decisions: instant offloading, instant local computation, deferred local execution, and deferred offloading decision. Furthermore, opportunistic computing and offloading are integrated in the process. Identifying the tasks that should be offloaded, and when it should be done, is a step forward that permits achieving higher computation offloading gains.

The proposed algorithm is seen as a smart buffering process where distinct virtual buffers join tasks that are associated with the same offloading decision. The buffers output orders depend on the system conditions and are subject to quality of experience based or opportunity scheduling.

Simulation results proved that the proposed algorithm is able to achieve extended battery lifetime while preventing any CPU memory overflows and capacity outage, and while keeping users' quality of experience by always respecting the imposed latency constraints.

We note that the proposed algorithm constitutes a base for algorithms variants that can be designed by varying the classification hierarchy. As future work, it would be interesting to investigate the impact of the classification order on the computation offloading gain.

Chapter 4

Small Cells Clustering for MEC: From Single-user to Multi-user

4.1 Introduction

4.1.1 Motivation

Edge cloud computing is a combination of cloud computing and mobile Internet paradigms. As in cloud computing, computations are offloaded and processed on remote servers. Future 5G ubiquitous mobile Internet allows mobile users to be always connected and have access to centralized resources pool that can handle offloaded computation. Mobile cloud computing has evolved during the past years and has adopted various architectures definitions. A first architecture definition of MCC is where mobile devices reach the remote cloud server directly through the Internet. The major bottleneck of this architecture is the cloud access latency. Mobile devices are connected to the cloud through a Wide Area Network (WAN) with uncontrollable access delay. Furthermore, energy consumption for accessing the cloud through the radio network can be significant. This also causes a major drawback for using MCC. Edge cloud consists in moving the mobile computation process to the edge of the logical extremes of a network. A novel edge cloud architecture was proposed in the European Project TROPIC. It consists of joining the emerging paradigm of mobile cloud computing with the ever-evolving trend of heterogeneous networks creating thus a local cloud in close proximity to MUEs [70]. Small cells (SCs) are small sized low-power base stations, some of which powered by mobile subscribers (femtocells). Even if small cells are endowed storage space and computational capacities, they cannot be compared to remote cloud servers. For this reason, small cell cloud proposes to enable cells federation in what we call a 'computation cluster'. MUEs send computational requests to their serving small cell (SSC) and get the computation results from that cell as well. The MUEs communicate with the Edge cloud only through a single hop communication with the serving cell. Serving small cells have the ability of distributing the computation load among neighbor small cells. The set of SCs in which the SSC and helper small cells (HSCs) participate in the computation is thus called SCs computation cluster. In this case, the second hop (and more if necessary) is between the SSC and HSCs. The SC cluster acts as a local cloud and delivers cloud services to the mobile user, always through its SSC. In fact, mobile users will not have to establish various communication links with several SCs. It is the SSC that sets the strategy for handling users' requests. The whole process is transparent to mobile users whose only interest is perceiving the desired QoE, here considered as respecting the services delay constraints.

Joining a set of small cells in one computation cluster is possible through virtualization, parallelization and Virtual Machines (VMs) deployment. The small cell cloud architecture assumes the presence of a virtual entity named Small Cell Manager (SCM). One of the SCM responsibilities is to handle VM deployment and manage small cell cloud resources. The computation load is distributed among the cluster cells. The distribution depends on both computational resources availability at each small cell, and communication channel quality between SSCs and HSCs.

The main motivation of this chapter is to extend the potential of MEC from one-hop offloading between MUEs and the SSCs, to multi-hop coordinated offloading, in which we exploit small cell clustering. The SSCs handle MUEs offloaded computational tasks and distribute computational load within the cluster. As computation requests are often subject to latency constraints, time limits should be respected for offering high quality of service to mobile users. At the same time, the offloading and computation process should be applied in an energy efficient way in order to reduce the system power consumption. The small cell clusters should then be set up such that service is efficiently delivered to all mobile users.

4.1.2 Related Work

Several works in literature investigate the problem of cloud resources management. Cloud resources management in MCC can be classified according to several criteria.

Many offloading strategies and methodologies focused on application partitioning for offloading decision purposes. The decision for each partition is either to be computed locally (at the mobile device that launched the application) or to be offloaded to local cloudlet or distant cloud. This kind of strategies can be classified according to the adopted partition model and the offloading objective. Mobile applications are modeled, partitioned, and attributed to a computing host. Application partitioning has taken different forms. Graph-based models are used to show the computation components context and relationships [95–97]. Using this model, computation components are identified as to be computed locally or offloaded. Other works used linear programming to cast the partitioning problem through linear equations [10, 79, 98–100]. Add to that, many heuristics have been proposed in order to deal with high complexity situations and to include a larger set of decision parameters [101–103]. Heuristics help dealing with scalability problems and decision computation delay.

Other works focus on partitioning for computation on the cloud. In [104], Verma *et al.* presented an algorithm for VM placement in virtualized systems. The goal was to design a cost aware dynamic VM placement controller. Two types of costs were considered: power costs (activation and computation) and migration costs. The problem was identified as a bin-packing problem which is NP-hard. The proposed solutions were three algorithms based on a FFD (First-Fit Decreasing) policy. VM are ordered according to a specific metric, and each is accorded with the first server that can accommodate it. Three different implementations are considered. In the first the goal is to minimize the power consumed by all servers. Therefore, VM are sorted by size. This strategy does not lead to global allocation optimality. Solutions are only locally optimal. An additional strategy is proposed in order to minimize VM migration cost. This strategy aims at minimizing the number of VM migrations. A third strategy joining both power and migration cost is proposed. It is based on comparing two VM placements and identifies VM migrations that allow the passage from one placement to another. These migrations are sorted by power per unit migration cost, and the most energy efficient are selected only if the power cost decrease is higher that the migration cost. In [105] a similar approach based on a BFD (Best-Fid Decreasing) strategy is adopted. A heuristic based on allocating each VM to the server that can guarantee the least power consumption is adopted. VM are sorted by decreasing CPU utilization. These works are part of the family of proposed heuristics that are approaching the problem as a bin-packing problem with differently sized bins [104–108]. Costs that are considered are all related to power. Time cost has not been considered in these works, despite the fact that computation latency is a of great importance especially for the current emergence of real-time applications and augmented reality.

Another point of view, is formalizing the problem as a Markov Decision Process (MDP) [109–111]. An MDP approach is proposed in [111] where the goal is to optimize long term system performance. The work tackles specifically the small cell cloud platform and assumes the presence of a SCM that manages cloud resources through VM placement. The MDP optimization problem has the objective of minimizing resources offloading, network delay, and VM migration costs. The factors that are taken into consideration in the process are each SC load, the network delay for sending and receiving data, and the migration cost. The costs are expressed in time. A final conclusion lead to the fact that allocating computational resources, i.e. deploying virtual machines, at the SSC of each mobile user is the best choice since it is has the lowest cost in terms of network delay. This conclusion is only true because a single VM destination is assumed per user, or in

other words, each user is assigned a single VM. If a user request could be assigned more than one VM, the conclusion is not always true since deploying additional virtual machine may accelerate the process through computation parallelization, and thus cells clustering. In addition, only time costs were taken into account in this work.

MAPCloud, a hybrid tiered cloud architecture, has been proposed by Rahimi *et al.* for computation load distribution [102]. In this work both local and public clouds are considered in a 2-tier cloud architecture. It is a service oriented framework where users call for services at mobile devices or on the cloud. Each service is characterized by a utility metric that depends on the user's and service location, service price, delay, and power consumption. Maximizing the utility over the set of possible services allocation solutions is a NP-hard problem. An annealing based heuristic under the name of CRAM (Cloud Resource Allocations for Mobile applications) is proposed. In CRAM, an iterative approach over increasing distance is used to progressively include more services in the search process. For each iteration, a set of services is randomly chosen according to four different metrics that are normalized price, power, delay, and QoS. A good possible solution is found through simulated annealing. In [103], a very similar approach is adopted, this time considering users mobility. Instead of a location based work-flow as in [102], a location-time work-flow is proposed. For introducing the mobility aspect, a center of interest location is computed for each user. The iterative annealing algorithm is then applied over increasing distance from the center of interest location of each user instead of its static position as in [102].

Multi-user edge cloud computing clustering has not been extensively investigated yet. Of the few existing work that investigate the multi-users case is of Yang *et al.* [99]. This work assumes that the cloud does not have unlimited computing resources. In case of multiple simultaneous users' requests, cloud resources should be jointly allocated in order to guarantee good QoE for all users. The problem is formulated as a Multiple Integer Linear Programming (MILP). The objective function is to minimize the average application delay for all the users. Scheduling is considered in order not to allow servers to compute more than one module at the same time. Users requests are assumed to be formed of several components, each of which can be computed at a different location: either locally on the mobile device, or on a cloud server. Two heuristics based on greedy algorithms are proposed in order to approach the optimal solution. A first algorithm computes the optimal resource allocation for each user alone, as if it was a single user case. This leads to overloaded allocation at servers. Allocation is adjusted by slightly increasing the average application delay. This is assured by searching for the module which adjusting leads to lowest increase in overall average latency. Another alternative proposal is based on sorting requests modules by non-increasing ready time. The same order is used to allocating resources for each module at the server that minimizes the extra delay. This work presents an interesting formulation of the problem. However, an important aspect is not taken in consideration, which is power consumption. Furthermore, connection between VMs is considered as infinite.

In [112] the partition problem is studied for the multi-users case where mobile devices can share communication bandwidth. The goal is to maximize system throughput in data stream applications. Mobile users also share cloud computational resources. The optimization problem is formulated and addressed in terms of a genetic algorithm.

4.1.3 Contribution

Most of the works that tackled computation offloading were based on offloading decision for energy saving at the mobile handsets. In Chapter 3 we proposed our own offloading decision algorithm. The algorithm is studied for a single user - single cloud scenario where a mobile user is connected to the cloud through its serving base station. However, it can be easily extended

to a multi-user case - single cloud case since it is decentralized and implemented at the handsets side. In this chapter, we tackle a cloud clustering set up paradigm. We study the cluster set up optimization by choosing which small cells contributes in the computational clusters, as well as the resource allocation and load distribution. In this chapter, we adopt the local small cell cloud paradigm. We assume that small cell base stations communicate and exchange data through wireless links. Active mobile users are connected and associated with one SC. Users have the possibility of offloading computation tasks to serving SCs. We focus on resources management inside the small cells cluster. Resources management consists of load distribution, radio resource allocation, power allocation, and computational capacities assignment. Proposed solutions are based on a joint resource allocation of computational and communication resources, in addition to joint computational load distribution. The approaches we propose are centralized, and can be computed by a small cell cluster managing unit. We note that the centralized control functionality can be implemented at any of the network small cells as well. In order to form the cluster, we assume that the SSC distributes the computational load on a set of HSCs. According to the adopted communication technologies and topologies, the cluster can be formed by multi-hop communications between SSCs and HSCs. However, in wireless intra-cluster communication, routing protocols should then be considered in order to optimize the computational data delivery. In our work, we consider only the special case of two-hop MEC computing. The first being between MUEs and the SSCs, and the second, between SSC and HSCs in the computation cluster.

Our contribution in this chapter is x-fold.
First, for the single user multi-cloud use case, we propose three different novel approaches of small cells clustering via optimization of computational load distribution between small cells. The first approach is the optimization of the load distribution with a goal of minimizing the cluster latency. The second consists in optimizing the cluster overall power consumption, and the third consists in minimizing the power consumption from a small cell centric point of view under the condition of respecting imposed latency constraints. In addition, we propose a clustering strategy that exploits the trade-off between the perceived latency and the cluster size. This trade-off, discussed in Section 2.4.5, increase system energy efficiency by using less small cells operating at higher load. The energy gain is at the cost of an increased perceived latency. We propose a cluster *sparsification* approach that reduces the size of the cluster without violating the latency constraints.

Then, we tackle the case of multi-user multi-cloud use case where computation clusters should be formed for all requests of all users. We propose a multi-user clusters optimization that allocates jointly communication and computation resources. We focus on respecting imposed latency constraints and the minimization of the clusters communication power consumption. A first novel aspect of the solution we propose, is the cluster scalability according to the computation requests requirements. In fact, small cell cloud has always been considered in previous works as an established set with known characteristics. Our proposed solution allows the cluster to have adaptive size, load distribution, and intra-cluster communication and computation resource allocation. Computation clusters should be built so that all users are satisfied, i.e. have their computation request handled without violating the imposed latency constraints. Hence, the second novel approach, which is to jointly form clusters such that all active users' requests simultaneously in order to better distribute computation and communication resources for a better users' QoE. We formulate the clustering problem for multiple users as an optimization problem. We distribute the computation load of all requests among the active small cells in the network. And we jointly allocate transmission powers for each of the small cells, and the computational capacity allocated for each user. The objective of the optimization problem is to minimize the clusters power consumption while respecting the imposed latency constraints of each user request.

The novelty of this work is based on a journal paper [J1] and conference papers [C3] and [C5].

4.2 System Model

We consider a multi-user indoor scenario where a set $\mathcal{N}$ of N small cells are deployed. A set $\mathcal{K}$of K mobile users are served each by a small cell denoted by S_k. The set of SSCs is denoted by $\mathcal{S}$. The set of devices associated to the small cell s is denoted by $\mathcal{K}_s$. Every device k in $\mathcal{K}$ sends a computational requests (W_k, Δ_k) to its SSC S_k. W_k and Δ_k denote for the number of CPU cycles to execute and the maximum latency imposed by the application, respectively. Note that the relationship between the number of instructions and the number of CPU cycles depends on the instructions type. Computational requests are characterized by the number of input and output bits, which are the bits to be sent to the computing small cell and back to the user. Each small cell n in $\mathcal{N}$ is characterized by a computational capacity of F_n CPU cycles per second. Each small cell can serve multiple devices simultaneously by according a part of its computational capacity, say f_{kn}, to each user k. We consider that the computation requests are already sent to the SSC. Each of the SSC forms a computation cluster for each of the requests it received. The computation load of each request W_k is distributed among the small cells (SSC and HSCs) of the computation cluster. Each small cell n is accorded W_{kn} of user's k request. We assume high granularity, and we split computational load over CPU cycles. The SSC sends the necessary input bits to the cluster small cells. The number of input bits is equal to $\theta_{UL}W_{kn}$. The cluster small cells processes the bits and sends back the output bit to the SSC S_k. The number of output bits to be sent is equal to $\theta_{DL}W_{kn}$. We consider point-to-point wireless backhaul connection between serving and helper small cells. The transmission power used to send input and output bits between the SSC s and a helper small cell n is p_{sn}. The information rate that can be achieved through the wireless channel link between small cells, taking into account packet retransmission is:

$$R_{sn} = B_{sn} log(1 + \frac{\sigma_c |h_{sn}|^2 p_{sn}}{(1-PER)\Gamma d^{\beta} N_0}) \tag{4.1}$$

where σ_c is the shadow fading coefficient of the adopted Rayleigh channel model. The channel fading is assumed constant for a whole transmission period. We assume perfect estimation of the coefficients h_{sn} of the channel between small cells $s \in \mathcal{S}$ and $n \in \mathcal{N}$. PER is the target packet error rate, Γ indicates the SNR margin to guarantee a minimum bit error rate BER ($\Gamma(BER) = -\frac{2log(5BER)}{3}$), d represents the distance between s and n, β indicates the path loss exponent which depends, in an indoor environment, on the number of walls separating the two communicating SCs [113], and N_0 is the noise power. Equation 4.2 details the path loss model, where d is the distance between the transmitter and the receiver, $d_{2D,indoor}$ is the two dimensional separation between the transmitter and the receiver, n is the number of penetrated floors, q is the number of walls that separate the transmitter and the receiver small cells, and L_{iw} is the penetration loss of walls.

$$\begin{aligned} PL(dB) =& 38.46 + 20 log_{10} d + 0.3 d_{2D,indoor} \\ &+ 18.3 n^{(n+1)/(n+2)-0.46} + q.L_{iw} \end{aligned} \tag{4.2}$$

We adopt the small cells deployment model for urban scenarios proposed in the 3GPP framework [113]. This model represents a single floor building with 10m x 10m apartments in a 5 x 5 grid. Each apartment is assumed to be equipped with a deployed small cell. Parameter ρ indicates the deployment ratio of SCs, which is the percentage of apartments in which the deployed base station is active. In the adopted system model we only consider a single floor, therefore, $n = 0$ and $d = d_{2D,indoor}$ in Equation 4.2.

$$P^* = \begin{cases} P_0 + \Delta_p P_{Tx} + P_{comp}, & 0 < P_{Tx} < P_{max}; \\ P_{sleep}, & P_{Tx} = 0. \end{cases} \tag{4.3}$$

where $P_{comp} = EPC \times W$ represents the computation power consumption which is proportional to the number of CPU cycles W executed by the SC characterized by an energy consumption per cycle equal to EPC.

4.3 Single-user Multi-cloud Use Case

4.3.1 Problem Statement

We consider at first a single user case scenario, where a mobile user terminal offloads a computation request to its SSC (see Figure 4.1). We do not tackle the problem of user association with a small cell. Users are already connected to one small cell to which they can send their computation requests. Furthermore, we do not deal with the offloading decision process at the mobile handset side. We assume that mobile handsets have already an offloading decision process that takes into account offloading related parameters. Note that by making this assumption we do not link the cluster set up with the computation offloading at the mobile side. It is a realistic scenario since MUEs only expect the service to be delivered with no additional delay, regardless if it is computed on the MUE, SSC, or in a SSC. However, incorporating the offloading decision and the SSC is possible by sending all requests to the SCM. The SCM computes than the offloading decisions and reports back to the MUEs. In this case, SCM can jointly optimize the set of offloaded computations and their computing clusters. In this section, we consider that the SSC (SSC) receives

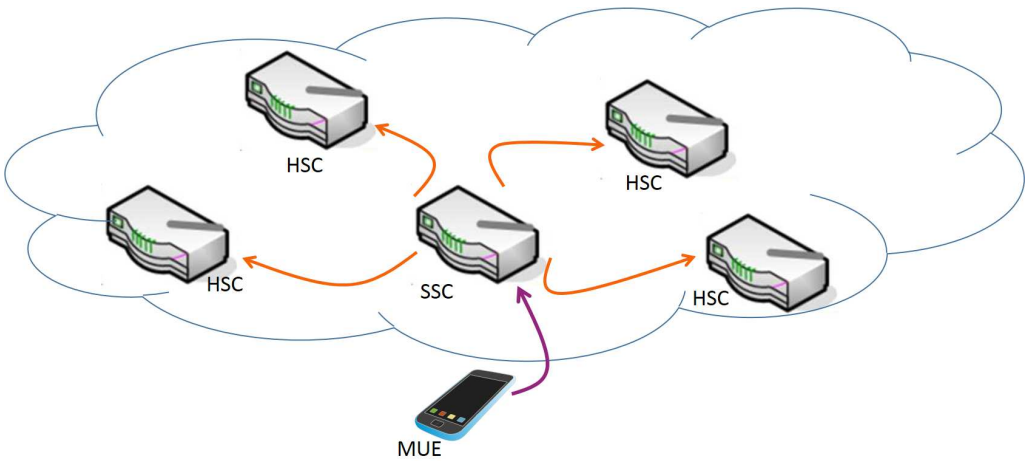


Figure 4.1: Single-user small cell cloud scenario.

a computation request from one of its connected mobile devices. The computation request is defined as a set of CPU cycles/instructions to be executed/computed under some latency constraint defined by the application. The goal of the SSC is to compute the user request without violating the imposed latency constraint. Depending on the system state and its available resources, SSC may decide to either compute users' requests locally (i.e. using its own computational resources), or build a cluster of small cells in order to distribute computation on the small cell cluster. In case of computation distribution on a cluster of small cells, load distribution should be optimized

along with computation resources at each of the cluster small cells. Furthermore, communication resources for sending computation data from SSC to HSCs should also be adequately allocated in order to guarantee tolerable data delivery time and communication power consumption.

We consider a set of N small cells each endowed with a total computational capacity of F_n [CPU cycles/sec]. Serving small cell s receives a computation request for running W CPU cycles. The maximum processing time allowed is set by the application parameter Δ_{app}. Therefore, Δ_{app} is the latency constraint to respect for the computation delay in order to serve the mobile user's request. The SSC s has the choice of either computing the request with its own local resources, or establishing a small cells computation clustering.

For the computation request to be satisfied, defined by (W, Δ_{app}), by using only local resources on SSC s, the following condition should hold:

$$\frac{W}{F_s} \leq \Delta_{app} \tag{4.4}$$

The left term of this equation represents the minimum computation time that can be achieved at the SSC. In this case, the totality of computational capacity F_s at the SSC should be allocated for the computation of the request. If equation (4.4) does not hold, then the SSC forms a small cells computation cluster in order to distribute the computation load. Each of the HSCs in the computation cluster is accorded a fraction of the computation load. However, to guarantee service delivery to the mobile user, resources should be adequately optimized. The SSC has the following tasks: (i) Choose which small cell to include in the computation cluster and distribute computational load on the chosen HSCs. (ii) Allocate computational resources at each HSC. (iii) Manage communication resources for sending and retrieving necessary data to and from SCC to HSCs. We consider that small cells communicate through point-to-point wireless backhaul connection (5G, 4G, 3G, WiFi, WiGig, etc.). For formulations and simulations, we adopt a wireless communication over Rayleigh non ergodic channels. Nevertheless, SSC may optimize a clustering process to compute the request in question even if the condition in equation (4.4) is verified. This decision depends on the strategy adopted for computing each user request. In the following of this section, several strategies for small cells clustering are proposed. The variety of proposed strategies covers different type of applications and user requirements.

For notations simplification, and since this section tackles the single user case, the index k of all notations is omitted since it always refers to the single user we consider.

4.3.2 Latency Minimization

In this section, the goal is to compute the mobile user request while minimizing the service latency. In general, the total overall service latency is measured from the moment the request is received by the SSC until all components are computed and received at the SSC. The total latency expression can be written as follows:

$$\Delta = \max_{n \in \{1,\ldots,N'\}} (\Delta_{comm}(n) + \Delta_{comp}(n)) \tag{4.5}$$

where N' is the number of HSCs that can be part of the computation cluster. Δ_{comm} is the time needed for sending necessary data to and from HSCs. It is then composed of two components: Δ_{UL} and Δ_{DL}. Δ_{UL} is the transmission time from SSC to HSC, and Δ_{DL} is the transmission time from HSC to SSC. In the case of the SSC (i.e. when $n = s$) there is no communication delay and $\Delta_{comm} = 0$. Δ_{DL} and Δ_{UL} depend respectively on the number of bits N^n_{DL} and N^n_{UL} to be sent

and received at HSCs. They are related to the load distribution and to the computation load W_n allocated at each HSC n through the following equations:

$$\begin{aligned} N_{DL}^n &= W_n\theta_{DL} \\ N_{UL}^n &= W_n\theta_{UL} \end{aligned}$$

where θ_{DL} and θ_{UL} are constants that account respectively for the overhead due to the uplink and downlink communications and for the ratio between output and input bits associated to the execution of W_n CPU cycles at HSC n. We note that parameters θ_{DL} and θ_{UL} vary according to the application type. Indeed, different classes of applications give rise to different sets of values for the pair (N,W). Not all applications are equally amenable for computation offloading. The classes of computation more suitable for offloading are the ones where, for a given computational load W, the number of bits N to be exchanged to enable the transfer of the program execution is low [49].

The total transmission time can then be written as:

$$\begin{aligned} \Delta_{comm}^n &= \Delta_{DL} + \Delta_{UL} \\ &= \frac{W_n\theta_{DL}}{(1-PER)B_{s,n}log(1+a_{s,n}p_{s,n})} + \frac{W_n\theta_{UL}}{(1-PER)B_{s,n}log(1+a_{s,n}p_{s,n})} \\ &= \frac{W_n\theta'}{(1-PER)B_{s,n}log(1+a_{s,n}p_{s,n})} \end{aligned} \tag{4.6}$$

with $\theta' = \frac{\theta_{UL}+\theta_{DL}}{1-PER}$, $B_{s,n}$ is the bandwidth used for transmitting data between SSC s and HSC n; $p_{s,n}$ is the power spent for transmitting this data which in this case takes the maximal value; $a_{s,n} = \frac{\sigma_c|h_{s,n}|^2}{\Gamma(BER)d^\beta N_0}$, where h is the channel coefficient, σ_c the shadow fading coefficient, $\Gamma(BER)$ the SNR margin for meeting a target BER, d the distance between SSC s and HSC n, β the path loss coefficient, and N_0 the noise power. $\frac{1}{1-PER}$ the average number of retransmissions assuming independent errors on each packet for a packet error rate PER. The packet error rate is determined by the bit error rate BER and the transmission packet size ps determined by the used modulation and coding scheme for each transmission:

$$\text{PER} = 1-(1-BER)^{ps}.$$

We consider, for simplicity, that both SSC and HSC transmit with the same optimized power $p_{s,n}$. Otherwise, the transmission delay in downlink from HSC to SSC cannot be estimated unless it is fixed a priori.

Δ_{comp} is the time required to compute the load accorded to the small cells. This term depends on the load distribution in the cluster, and on the computational capacity allocated at each small cell of the cluster. With f_n the allocated computational capacity, and W_n the computation load at HSC n, $\Delta_{comp}(n)$ is defined as follows:

$$\Delta_{comp}(n) = \frac{W_n}{f_n}. \tag{4.7}$$

The first strategy consists in minimizing the cluster latency that is the time that is consumed for load distribution, computation at the cluster nodes, and the computation results reporting to the SSC. This kind of strategies could be requested by the user in order to increase his QoE. This strategy does not impose power consumption constraints nor cluster size limitations. For

these reasons, the system is forced to include all of the active and reachable small cells in the computation cluster. Since we consider point-to-point communication between small cells, the overall cluster latency is the maximum latency as defined in (4.5). This latency depends on the computational load through the computation time at the small cell, and on the channel quality through both communication latency in uplink and downlink. For this strategy, we consider that the SSC communicates with all other HSCs in the cluster with the same transmission power $p_{s,n} = Pmax$. All transmission links are fully used in order to maximize effective throughput and decrease the total experienced latency.

The optimization problem is formulated as follows:

$$\begin{aligned} \min_{\{W_n\}_{n=1}^{N'}} \quad & \max_{n=\{1,\dots,N'\}} \quad A_n W_n \\ \text{s.t.} \quad & W_n \geq 0, \quad n = 1,\dots,N', \\ & \sum_{n=1}^{N'} W_n = W \end{aligned} \tag{$\mathcal{PB}_1$}$$

where we define $A_n \triangleq \dfrac{1}{f_n} + \dfrac{\theta'}{\log(1+a_{s,n}P_{max})}$. The conditions in $\mathcal{PB}_1$ guarantee the totality of the computation block is distributed among the cluster small cells. The solution of this optimization problem leads to a load distribution among all active base stations in a way that unifies the experienced latency at each small cell. This is intuitive: if two small cells do not have the same experienced latencies, then we can always adjust the load distribution in order to decrease the higher latency and increase the lower one in order to have a smaller maximal value.

Problem $\mathcal{PB}_1$ is a non-smooth problem. However to find its optimal solution, by introducing a slack (real positive) variable $t = \max\limits_{n=\{1,\dots,N'\}} A_n W_n$ we can solve the following equivalent problem:

$$\begin{aligned} \min_{t,\{W_n\}_{n=1}^{N'}} \quad & t \\ \text{s.t.} \quad & W_n \geq 0, \quad n = 1,\dots,N', \\ & \sum_{n=1}^{N'} W_n = W \\ & A_n W_n \leq t, \quad n = 1,\dots,N'. \end{aligned} \tag{$\overline{\mathcal{PB}_1}$}$$

Theorem 1. *The convex problem $\overline{\mathcal{PB}_1}$ is equivalent to $\mathcal{PB}_1$ and its optimal solution is given by*

$$W_m = W\left(\sum_{n=1}^{N'} \frac{A_m}{A_n}\right)^{-1} \quad \textit{and} \quad W_n = \frac{A_m}{A_n}\cdot W_m,\ \forall n \neq m. \tag{4.8}$$

Proof. First observe that problem $\overline{\mathcal{PB}_1}$ is convex, then any local optimal point is a global optimal solution satisfying the KKT conditions (note that Slater's condition holds true). Given the Lagrangian function defined as

$$\begin{aligned} \mathcal{L}(t,\mathbf{W}) \triangleq t &- \sum_{n=1}^{N'} \lambda_n W_n - \nu\left(\sum_{n=1}^{N'} W_n - W\right) \\ &+ \sum_{n=1}^{N'} \mu_n (A_n W_n - t) \end{aligned} \tag{4.9}$$

where $\mathbf{W} = [W_1, \ldots, W_{N'}]$ and λ_n, ν, μ_n are the Lagrangian multipliers, the KKT conditions can be written as

$$\begin{aligned}
&\text{(a):} \quad \frac{\partial \mathcal{L}}{\partial t} = 1 - \sum_{n=1}^{N'} \mu_n = 0, \\
&\text{(b):} \quad \frac{\partial \mathcal{L}}{\partial W_n} = -\lambda_n - \nu + \mu_n A_n = 0, \quad \forall n, \\
&\text{(c):} \quad \nu \in \mathbb{R}, \quad \sum_{n=1}^{N'} W_n - W = 0, \\
&\text{(d):} \quad 0 \leq \lambda_n \perp W_n \geq 0, \quad \forall n, \\
&\text{(e):} \quad 0 \leq \mu_n \perp (t - A_n W_n) \geq 0, \quad \forall n.
\end{aligned} \tag{KKT$_{\overline{\mathcal{PB}_1}}$}$$

To find the optimal solution of this system, observe that at least one multiplier λ_n has to be null, since $\lambda_n > 0$ for all n leads from (d) to $W_n = 0$ and this contradicts condition $\sum_{n=1}^{N'} W_n = W$. Then let us suppose that there exists at least a positive multiplier $\lambda_n > 0$. From (d) we get $W_n = 0$ and from (b) $-\nu + \mu_n A_n > 0$. Hence using (e) it results $t > 0$ since $t = 0$ implies $W_m = 0$, $\forall m$ and this contradicts condition (c). Therefore it follows $t > 0$ and from e) $\mu_n = 0$ so that to meet condition $-\nu + \mu_n A_n > 0$ it results $-\nu > 0$, $\forall n$. On the other hand to satisfy (c) it exists at least a value of $W_m > 0$ for which we yield from d) $\lambda_m = 0$ and from (b) $-\nu + \mu_m A_m = 0$. This is an absurdum since $-\nu > 0$. Then it results $\lambda_n = 0$ and $\nu = \mu_n A_n$, $\forall n$. Let us now focus on the multiplier μ_n. Note that if there exits a value $\mu_n = 0$ this leads to $\nu = 0 = \mu_m$ for each m and then condition in (a) never holds. It follows that $\mu_n > 0$ for all n and from (e) $t = A_n W_n = A_m W_m$ for each $n \neq m$ or $W_n = \frac{A_m W_m}{A_n}$. Hence from (c) we get $\sum_{n=1}^{N'} W_n = \sum_{n=1}^{N'} \frac{A_m W_m}{A_n} = W$ so that the optimal solution of $\overline{\mathcal{PB}_1}$ is

$$W_m = W \left(\sum_{n=1}^{N'} \frac{A_m}{A_n} \right)^{-1} \quad \text{and} \quad W_n = \frac{A_m}{A_n} \cdot W_m, \ \forall n \neq m. \tag{4.10}$$

□

Note that this strategy may in some cases result in assigning very small computation loads to some HSCs that either experience a very bad communication channel quality with the SSC, or are very far from the SSC and subject to severe path loss, or both. In these situations, HSCs in bad conditions will consume the major part of the time for receiving and transmitting data. Even if energy consumption is not in the goal of this strategy, this kind of situations pushes the energy-latency trade-off to its extent. It consumes a lot of energy for a very small amount of computation, and thus system energy efficiency can be improved. This problem could be solved by adding a pre-selection step that limits the number of participating HSCs. This could be done by setting a threshold on channel quality, distance, or both. However, if the sole goal is to guarantee a QoE and to server the users' requests regardless of the cost, $\mathcal{PB}_1$ is able to deliver the optimal solution. For improving local cloud clusters energy efficiency, we propose a clustering strategy that reduces the cluster size while keeping the QoE guaranteed.

4.3.3 Cluster Sparsification

Forcing a small cell exclusion rule at the beginning of the clustering process may result in service delivery failure. If we exclude too many small cells, the computational and communication resources of the considered HSCs may not be enough for computing a request. And since the top objective is to serve the user's request, increasing the cost but guaranteeing the service is a better solution than reducing the cost and failing at delivering the service. In this section, we base on the solution of the previous latency minimizing strategy. The objective is to remove nodes that are accorded very small computational tasks in order to reduce the cluster size, and eventually its energy consumption. The main goal is to keep the service guarantee while trying to exclude costly HSCs. This may be seen as an exploitation of the latency/power consumption and cluster size/latency trade-offs. We propose to reduce the size of the SSC, by distributing higher loads on less small cells. Some small cells will then have no computational load, while others will have more. To minimize the size of the cluster, a cost is imposed for each used HSC. Minimizing the size of the cluster is equivalent to minimizing the number of small cells that are accorded computational load. The optimal cost function to use is the l_0 norm, which associates a zero cost to every non used HSC and a unit cost for used HSCs. Minimizing l_0 norm costs is minimizing the size of cluster as much as possible. In this case, the optimization problem can be cast as follows:

$$
\begin{aligned}
\min_{\mathbf{W}} \quad & ||\mathbf{W}||_0 \\
\text{s.t.} \quad & \frac{W_n}{f_n} + \frac{W_n \theta'}{log(1+a_{s,n}P_{max})} \leq \Delta_{app}, \quad n = 1, \cdots, N' \\
& W_n \geq 0, \qquad\qquad\qquad\qquad\quad n = 1, \cdots, N' \\
& \sum_{n=1}^{N'} W_n = W
\end{aligned}
\qquad (\mathcal{PB}_{l_0})
$$

where $\mathbf{W} = [W_1, \cdots, W_{N'}]$. Unfortunately, implementing the minimization of a l_0 norm is not an easy task due to the discontinuous nature of its objective function. Therefore, we replace the l_0 norm by a cost function with similar properties. For having the same behavior of the l_0 norm, the cost function should be null at zero and positive otherwise. We propose the following function:

$$F_2(x) = \frac{\alpha|x|}{1+\alpha|x|} \tag{4.11}$$

with α a parameter that sets the sharpness of the function. Figure 4.2 shows the variation of $F_2(x)$ as a function of α.

If x represents the percentage of computation load accorded to each HSC, F_2 penalizes more HSCs with higher computational load. As seen in Figure 4.2, the higher α is, the more $F_2(x)$ approaches the form of the l_0 norm. However, in some cases, the solution could lead to advocating, if possible, many HSCs with low load for a lower cost. This is caused by the continuity of the F_2 and depends on its sharpness. If the solver is sensitive enough to capt the weak gradient that exists even for high values of α, then the solution risks to be driven toward low load HSCs. Even though this case is less probable to happen with small scale scenarios, it is more likely to occur with a large scale system.

For this reason, we propose a second cost function that is null at zero, continuous, and that associates lower cost for high loaded HSCs. The function is designed in a way that drives the solver for assigning higher loads at each considered HSCs. It is designed to inverse the gradient variation with the load increase comparing to F_2. The cost function we propose is the following:

$$F_3(x) = 1 + (10\gamma|x| - 1)e^{-\gamma|x|} - \frac{|x|}{2} \tag{4.12}$$

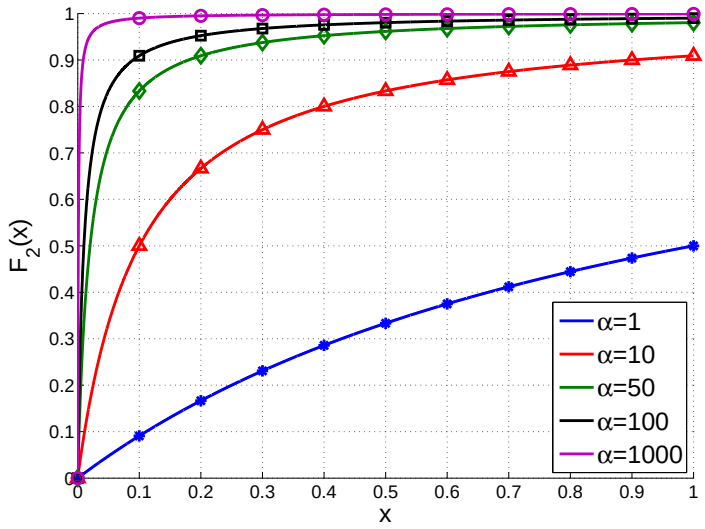


Figure 4.2: Cost function $F_2(x)$ variation with α

where γ also sets the sharpness of the function as can be seen in Figure 4.3.

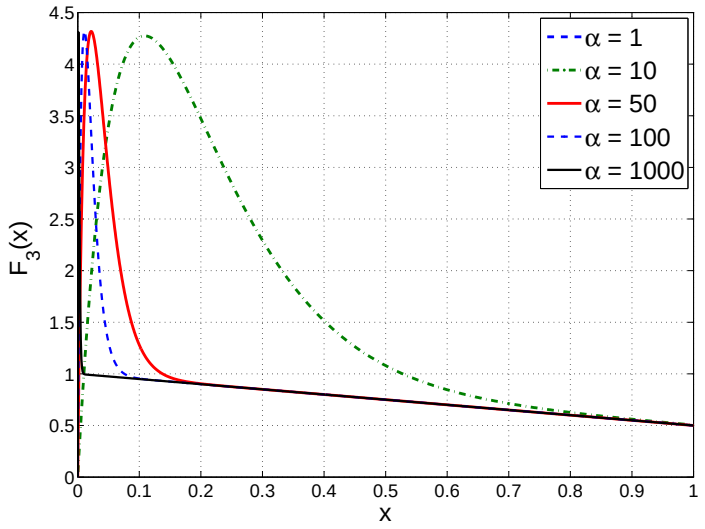


Figure 4.3: Cost function $F_3(x)$ variation with γ

As it is shown in Figure 4.3, the cost function sharpness increases for high values of γ. This enormously decreases the chances that the solution derives low load HSCs. In addition, it is clear that lower costs are given to higher load HSCs. This further helps the solution to tend toward high loaded HSCs, and thus, to smaller cluster size. F_3 does not have the property of the l_0 norm of associating a value of 1 to the non-zero elements. However, as explained above, this function could help achieve the required solution while speeding up the optimization process. Indeed, the slope in F_3 cost function will accelerate the convergence to a solution. Both F_2 and F_3 can thus be used as cost functions for reducing the computation cluster size. They have similar behavior at

small scale scenarios, with F_3 giving a higher cost for higher loaded HSCs. For both cases of F_2 and F_3 as cost functions, the optimization problem is cast as follows:

$$\begin{aligned} \min_{\{W_n\}_{n=1}^{N'}} \quad & \sum_{n=1}^{N'} F_i(W_n), && i=2,3 \\ \text{s.t.} \quad & \frac{W_n}{f_n} + \frac{W_n\theta'}{log(1+a_{s,n}P_{max})} \leq \Delta_{app}, && \forall n=1,\cdots,N' \\ & W_n \geq 0, && \forall n=1,\cdots,N' \\ & \sum_{n=1}^{N'} W_n = W. \end{aligned} \qquad (\mathcal{PB}_i)$$

The first condition in $\mathcal{PB}_i$ guarantees the respect of the application latency constraint by each of the HSCs, and the SSC. The second and third conditions guarantee that the whole task components will be computed. The solution of this problem, regardless of the cost function, tends to include HSC that are close to the SSC to the computation cluster. The closer the HSC, the higher, in general, is the channel achievable rate. Therefore, closer HSCs have lower latencies that farther HSCs and can be accorded larger computation tasks.

4.3.4 Minimization of Cluster Power Consumption

Both proposed strategies in 4.3.2 and 4.3.3 deal with the clustering optimization problem from a latency minimization and cluster size reduction point of views. These strategies do not account for the cluster power consumption. Power consumption is an important issue in local MEC and especially in small cell cloud since the local cloud servers are typical cellular network base stations. Both first strategies aimed at achieving latency gain and good experience quality. Another approach to the problem, is based on the fact that the latency constraint given by each application (Δ_{app}) should be respected, but not necessarily anticipated. Indeed, if a computation is executed and delivered to the user before Δ_{app} seconds, this does not necessarily increase MUEs perceived QoE. What is necessary, is service delivery within — at most — Δ_{app}. Users' experience quality won't be affected if this constraint is respected. Even if no latency gain over Δ_{app} is achieved, users will still be able to receive the required service in time. The main idea is to exploit the latency-power consumption trade-off in local MEC in order to reduce the small cells cluster power consumption while keeping a good QoE. The following strategies stress on power consumption minimization, constrained by latency limitations.

The computation power consumption can be formulated as the product of the number of processed CPU cycles and the Energy Per CPU cycle (EPC) of the small cell processor,

$$P_{comp} = \frac{W.EPC}{\Delta_{comp}} = f_n.EPC. \qquad (4.13)$$

In the following, we focus on communication power consumption considering that the sum of computing power consumption is fixed for each task independently of the load distribution. Communication power consumption can be optimized according to the channel quality, the computational capacity offered by each HSC and the application latency constraint. If no limitations were imposed by the offered computational capacities and the latency constraints, the problem would be similar to water-filling [114]. However, with the additional constraints, the problem is a joint optimization of the transmission power $p_{s,n}$ and of the percentage of computation W_n accorded to each HSC. Since the optimization problem aims at minimizing the communication power consumption, the optimal solution would be to compute the request at the SSC. Indeed, the

SSC allocates all of its computational capacity to compute a maximum load within the latency constraints, with any communication cost ($p_{s,s} = 0$). If computational resources at the SSC are sufficient for computing the whole request without violating the latency constraint, then no optimization problem solving is needed and the load is accorded to the SSC. This is true only for the case where $f_s\Delta_{app} \geq W$. Otherwise, in the case where $f_s\Delta_{app} < W$, the SSC will compute as much as its resources allow in a Δ_{app} time. The SSC load will be equal to $W_s = f_s\Delta_{app}$. Then, for the remaining computational load, the optimization problem can be cast as follows:

$$
\begin{aligned}
\min_{\mathbf{p},\mathbf{W}} \quad & \sum_{n=1,n\neq s}^{N'} p_{s,n} \\
\text{s.t.} \quad & \text{(a)} \quad W_n \geq 0, && \forall n = 1,\cdots,N'; n \neq s \\
& \text{(b)} \quad \sum_{n=1,n\neq s}^{N'} W_n = W - W_s, && (\mathcal{PB}_4) \\
& \text{(c)} \quad \frac{W_n}{f_n} + \frac{W_n\theta'}{log(1+a_{s,n}P_{s,n})} \leq \Delta_{app}, && n = 1,\cdots,N'; n \neq s \\
& \text{(d)} \quad 0 \leq p_{s,n} \leq P_{max}, && n = 1,\cdots,N'; n \neq s.
\end{aligned}
$$

Problem $\mathcal{PB}_4$ is non-convex, due to the non-convexity of the delay constraint (c). In the following we case $\mathcal{PB}_4$ into a convex equivalent problem. The delay constraint in (c) is equivalent, for $p_{s,n}W_n > 0$ and under the feasibility condition $\Delta_{app} f_n > W_n$, to:

$$
-log(1+a_{s,n}p_{s,n}) + \frac{f_nW_n\theta'}{\Delta_{app}f_n - W_n} \leq 0 \tag{4.14}
$$

Note that the delay constraint in (4.14) is convex, as can be easily proven by showing that its Hessian is a semi-definite positive matrix. Therefore, the problem $\mathcal{PB}_4$ can be reformulated as:

$$
\begin{aligned}
\min_{\mathbf{p},\mathbf{W}} \quad & \sum_{n=1,n\neq s}^{N'} p_{s,n} \\
\text{s.t.} \quad & W_n \geq 0, \quad n = 1,\ldots,N'; n \neq s, \\
& \sum_{n=1,n\neq s}^{N'} W_n = W - W_s && (\overline{\mathcal{PB}_4}) \\
& 0 \leq p_{s,n} \leq P_{max}, \quad n = 1,\ldots,N'; n \neq s \\
& -log(1+a_{s,n}p_{s,n}) + \frac{f_nW_n\theta'}{\Delta_{app}f_n - W_n} \leq 0, \quad n = 1,\ldots,N'; n \neq s, \\
& W_n - \Delta_{app}f_n < 0, \quad n = 1,\ldots,N'; n \neq s.
\end{aligned}
$$

Problem $\overline{\mathcal{PB}_4}$ has the following properties:

Theorem 2. *Given problem $\mathcal{PB}_4$ and $\overline{\mathcal{PB}_4}$, the following hold:*
(i) Necessary condition for $\mathcal{PB}_4$ to be feasible is:

$$
W_s + \sum_{n=1,n\neq s}^{N'} \frac{\Delta_{app}}{\frac{1}{f_n} + \frac{\theta'}{log(1+a_{s,n}P_{max})}} \geq W; \tag{4.15}
$$

(ii) $\overline{\mathcal{PB}_4}$ is a convex problem, then any local optimal solution is a global optimal minimum;
(iii) $\mathcal{PB}_4$ and $\overline{\mathcal{PB}_4}$ are equivalent

Proof. To prove point (i) of Th. 2 observe that Equation (4.15) is a global condition that can be easily derived from (b), (c) and (d) in $\mathcal{PB}_4$. For proving point (ii), observe that problem $\overline{\mathcal{PB}_4}$ is convex since the objective function as well as all the constraints are convex. Then any stationary point is a *global* optimal solution of the problem. As for point (iii), since for $\overline{\mathcal{PB}_4}$ the Slater's constraint is verified, any optimal solution satisfies the KKT conditions of $\overline{\mathcal{PB}_4}$. The Lagrangian function associated to $\overline{\mathcal{PB}_4}$ is:

$$\begin{aligned}\mathcal{L}(\mathbf{p},\mathbf{W}) \triangleq & \sum_{n=1,n\neq s}^{N'} p_{s,n} - \sum_{n=1,n\neq s}^{N'} \beta_n W_n + \lambda\left(\sum_{n=1,n\neq s}^{N'} W_n - (W - W_s)\right) \\ & + \sum_{n=1,n\neq s}^{N'} \mu_n p_{s,n} + \sum_{n=1,n\neq s}^{N'} \alpha_n \left(p_{s,n} - P_{max}\right) \\ & + \sum_{n=1,n\neq s}^{N'} \tau_n \left(-log(1+a_{s,n}p_{s,n}) + \frac{W_n f_n \theta'}{\Delta_{app} f_n - W_n}\right) \\ & + \eta_n (W_n - \Delta_{app} f_n)\end{aligned} \tag{4.16}$$

where β_n, λ, μ_n, α_n, τ_n, and η_n are the Lagrangian multipliers. The KKT conditions can be written as:

$$\begin{aligned}
&\text{(a):} \quad \frac{\partial \mathcal{L}}{\partial p_{s,n}} = 1 + \alpha_n - \mu_n - \tau_n \frac{a_{s,n}}{1 + a_{s,n} p_{s,n}} = 0, \quad \forall n \neq s, \\
&\text{(b):} \quad \frac{\partial \mathcal{L}}{\partial W_n} = \lambda - \beta_n + \tau_n \frac{f_n^2 \Delta_{app} \theta'}{(\Delta_{app} f_n - W_n)^2} + \eta_n = 0, \quad \forall n \neq s, \\
&\text{(c):} \quad \lambda \in \mathbb{R}, \quad \sum_{n=1,n\neq s}^{N'} W_n - (W - W_s) = 0, \\
&\text{(d):} \quad 0 \leq \beta_n \perp W_n \geq 0, \quad \forall n \neq s, \\
&\text{(e):} \quad 0 \leq \alpha_n \perp (P_{max} - p_{s,n}) \geq 0, \quad \forall n \neq s, \\
&\text{(f):} \quad 0 \leq \mu_n \perp p_{s,n} \geq 0, \quad \forall n \neq s, \\
&\text{(g):} \quad 0 \leq \tau_n \perp \left(-log(1 + a_{s,n} p_{s,n}) - \frac{W_n f_n \theta'}{\Delta_{app} f_n - W_n}\right) \geq 0, \quad \forall n \neq s, \\
&\text{(h):} \quad 0 \leq \eta_n \perp (\Delta_{app} f_n - W_n) > 0, \quad \forall n \neq s.
\end{aligned} \tag{KKT$_{\overline{\mathcal{PB}_4}}$}$$

where $a \perp b$ stands for $\langle a, b \rangle = 0$. From the complementary condition (h) we get $\eta_n = 0, \forall n \neq s$. We study these two cases separately: i) $\tau_n > 0$ and ii) $\tau_n = 0$. Under assumption i), it follows from (g) that the delay constraint is always active. Therefore, if $p_{s,n} > 0$ then $W_n > 0$ and we have a one to one relationship established between the transmission power and the computational load. From (d) we have that $\beta_n = 0$, then from (b) it results that $\lambda < 0$. Finally, assume that $\tau_n > 0$ and $p_{s,n} = 0$, then the solution $p_{s,n} = 0, W_n = 0$ is achievable according to the one to one established relationship between $p_{s,n}$ and W_n. We now consider the case ii= where $\tau_n = 0$. Under this assumption (b) reduces to $\lambda = \beta_n$ which contradicts the fact that $\lambda < 0$ and this leads to an absurdum.

□

The optimal strategy tends to assign the high computation loads to small cells with larger computational capacities and better communication channels.

4.3.5 Minimization of Small Cell Selfish Power Consumption

The previous optimization targets the minimization of the overall communication power consumption. This may lead to that some small cells hold greatly higher energy costs than others. This happens, for example, when a certain HSC has very high computational capacity. This small cell will then be allocated very high load which leads to increasing its communication power consumption compared to other HSCs. In this problem, we address selfish minimization of the communication power consumption, i.e., each small cell in the cluster tends to reduce its own energy consumption. The power minimization should always take into account the application latency constraints. For the same reasons as in $\mathcal{PB}_4$, the SSC s will be assigned a load equal to $fs\Delta_{app}$. The optimization problem can be set as follows:

$$
\begin{array}{lll}
\min\limits_{\mathbf{p},\mathbf{W}} & \max\limits_{n=\{1,\dots,N'\}} p_{s,n} & \\
\text{s.t.} & \dfrac{W_n}{f_n} + \dfrac{W_n\theta'}{log(1+a_{s,n}p_{s,n})} \leq \Delta_{app}, & \forall n = \{1,\cdots,N'\}; n \neq s, \\
& W_n \geq 0, & \forall n = \{1,\cdots,N'\}, \\
& \sum\limits_{n=1}^{N'} W_n = W, & \\
& 0 \leq p_{s,n} \leq P_{max}, & \forall n = \{1,\cdots,N'\}, \\
& p_{s,s} = 0, & \\
& W_s = min\{f_s\Delta_{app}, W\}. &
\end{array}
\qquad (\mathcal{PB}_5)
$$

The solution will tend to accord to all small cells an equal power consumption. The same reasoning of $\mathcal{PB}_1$ applies. If any small cell has a greater power consumption than the others, the load distribution can be modified, if possible, to decrease the maximal power consumption value. This policy will most likely increase the overall cluster power consumption comparing to $\mathcal{PB}_4$.

4.3.6 Numerical Evaluation

We presented four different strategies for small cells clustering in the concept of local mobile computing through computation offloading to SSCs. These strategies differ in their optimization objective, and therefore form different clusters. As proved in [111], in a cluster of size 1, the less costly solution for small clustering is to compute the tasks at the SSC. For the latency minimizing strategies, this is clearly not the case when we can include more than one computation server. For the power minimizing strategies, it is the most beneficial to compute the totality of the tasks at the SSC. By doing so, there is no communication cost. But this is not always possible due to the limited computational capacities at small cells.

In this section, we compare the solution of $\mathcal{PB}_1$ and $\mathcal{PB}_4$ in a 3GPP 5×5 apartments grid scenario. ρ is the ratio of apartments where active small cells are deployed. For simplicity we consider a static user connected to the small cell in the center of the grid (see Figure 4.4). Direct neighbors HSCs (separated of SSC by a maximum of 2 walls) are referred to as Near HSCs. χ_{Near} determines the percentage of Near HSCs among active small cells.

Figure 4.5 shows how much load can be allocated to the SSC with both latency and power minimizing strategies. The computational load ration of SSC, near HSCs, and far HSCs ($\frac{\sum W_n}{W}$) are reported with respect to the ratio of near HSCs with respect to the number of active HSCs in the grid (Ξ_{near}). Simulation parameters are listed in table 4.1

$\mathcal{PB}_1$ solution tends to give larger computation tasks to far HSCs than $\mathcal{PB}_4$, despite the fact that they are subject to an average weak transmission channels due to larger distance from SSC. This

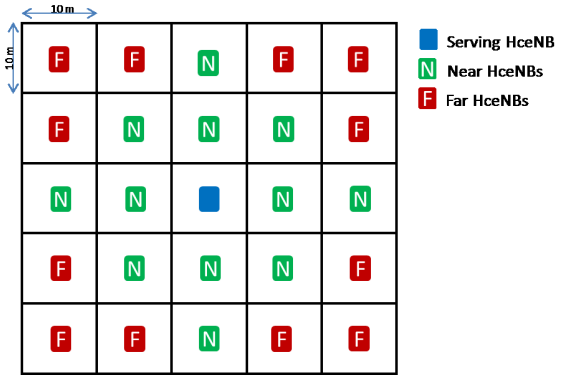


Figure 4.4: 3GPP apartment grid with small cells emplacement

Table 4.1: Simulation parameters values

Parameter	**value**	**Parameter**	**value**
Runs	3000	ρ	0.25
B	20MHz	σ_c	10
N_0	-118.4 [dB/Hz]	BER	10^{-6}
W	$[10^8; 2.10^8]$	Δ_{app}	[5;8]
f_n	$[10^6; 2.10^6]$	θ_{UL}	1
θ_{DL}	0.2	P_{max}	1 [W]
P_0	10.1 [W]	Δ_p	15

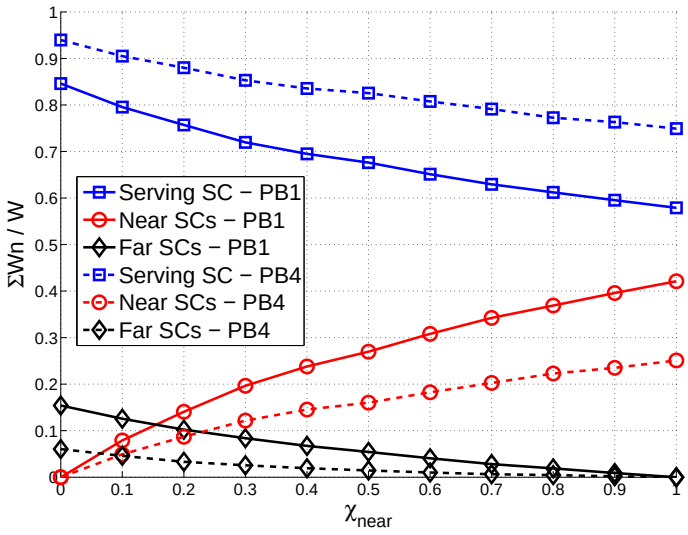


Figure 4.5: Comparison of load distribution to each HSC set and SSC for latency and power consumption minimization

strategy allows the cluster to reduce its latency. $\mathcal{PB}_1$ takes advantage of all active HSCs, especially

Near HSCs to which it allows a greater computation load in order to achieve a lower overall latency since they normally have better channel conditions and are subject to lower path loss. The solution of $\mathcal{PB}_4$, which objective is to minimize the cluster power consumption, assigns more computation load to the SSC when possible, because it has no communication power consumption. This comes at the cost of increasing the cluster latency since computational load is not distributed in a way that guarantees faster cluster computations. In $\mathcal{PB}_1$ load is distributed such that all small cells in the cluster have the same overall latency of receiving, computing, and sending back the results to the SSC. Whereas in $\mathcal{PB}_4$, the load is not distributed according to perceived latency, and thus some small cell with higher loads, notably SSC, will consume more time for tasks execution that others, and therefore, increase the overall cluster perceived latency. As it is shown in Figure 4.6, $\mathcal{PB}_1$ has the largest latency gain, as expected. It shows also that the optimizations that target the

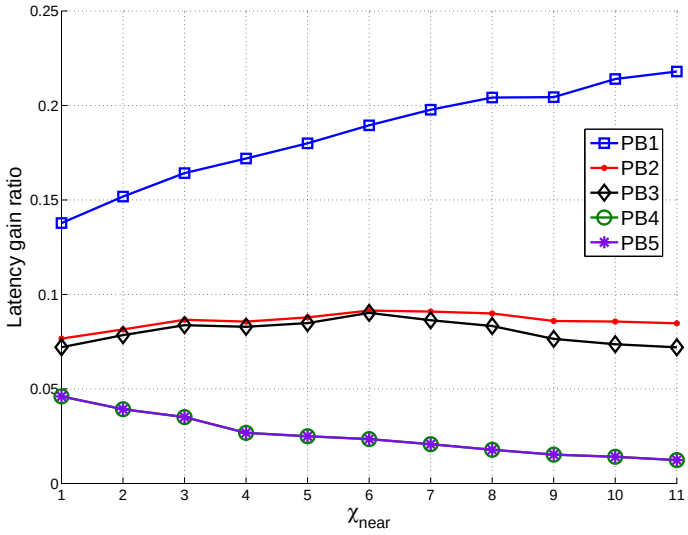


Figure 4.6: Latency gain of different strategies comparing to Δ_{app}

minimization of the power consumption, i.e. $\mathcal{PB}_4$and $\mathcal{PB}_5$ do not achieve almost any latency gain. In fact, these strategies take advantage of all the available time delay in order to further reduce power consumption. Transmitting with lower transmit power increases the bit duration, and thus the communication time. They push the latency-power consumption trade-off to its limits defined by assuring QoE. Figure 4.6 also shows that when we tend to sparsify the solution in order to eliminate HSCs with very low computation tasks ($\mathcal{PB}_i$), we can lose up to 15% in terms of latency. However, this gain in latency is traded with power consumption as can be seen in Figure 5.7. As it is shown in the graph, $\mathcal{PB}_i$ achieves the higher power consumption gain comparing to the consumption of $\mathcal{PB}_1$. This gain is between 50% and 60% for all χ_{near} value, i.e. the gain in power consumption is considerable for all HSCs distribution in the apartments grid. We notice that for the case where the power minimization is HSC centric, the gain decreases of approximately 10% for scenarios where far HSCs are dominant, and it decreases less when the majority of HSCs are in the Near HSCs set. As for $\mathcal{PB}_3$ where the solution of latency minimization is sparsified, it is shown that we can increase the power consumption gain from 0% in the case of $\mathcal{PB}_1$ up to 33% for deployments where far HSCs dominate. In fact, when far HSCs are numerous, the chances of being in a situation where far HSCs are accorded very low computation load increase. Therefore, more gain can be achieved in such scenarios. Since in the case of $\mathcal{PB}_3$ we assumed

that transmission power is constant and equal to P_{max}, the gain in power consumption comes only from the reduction of the cluster size. For $\mathcal{PB}_i$ and $\mathcal{PB}_4$, transmission power consumption can be controlled. For this reason, the power consumption gain in these cases is a result of both transmission power adaptation and cluster size reduction. Figure 4.8 shows the ratio of used HSCs

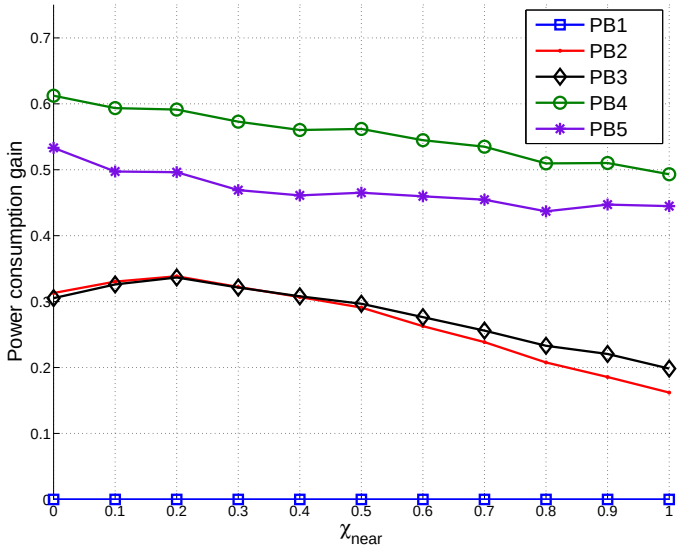


Figure 4.7: Power consumption gain for different strategies comparing to the maximum power consumption

to form the cluster among the total number of active small cells. This plot shows that all strategies that gained in power consumption own a part of it to the cluster size reduction. $\mathcal{PB}_i$ is the strategy that uses the smallest cluster size. Moreover, it tends to use more HSCs whenever they are in the Near HSCs set. This can be seen from the gain decrease when the χ_{near} increases. We notice that ($\mathcal{PB}_4$), which has higher power consumption gain than $\mathcal{PB}_3$, achieves almost the same gain in cluster size. Additional power gain is then due to power control and load distribution. The difference of power consumption gain between $\mathcal{PB}_i$ and $\mathcal{PB}_4$ is also due to the load distribution since the first minimizes the overall power consumption and the second does an HSC centric power consumption. Power consumptions are compared in Figure 4.9 that shows the power consumption distribution over the HSCs set. It is clear that the HSC centric approach allocates more power to far HSCs to be able to lower the power allocation for Near HSCs. This will balance the power consumption over the cluster HSCs. But as seen in Figure 5.7, this comes at the expense of higher overall cluster power consumption.

4.4 Multi-user Multi-cloud Use Case

In this section, we extend the cluster set up optimization problem in a local MEC, to the muli-user case (See Figure 4.10). Each of the SSC will form clusters for its own requests. SSC will then share HSCs and their computational capacities. This requires a joint and simultaneous set up of the computations clusters. Almost all previous work considers that computational capacities at the cloud are always sufficient for computing users' requests. Such an assumption is not always true,

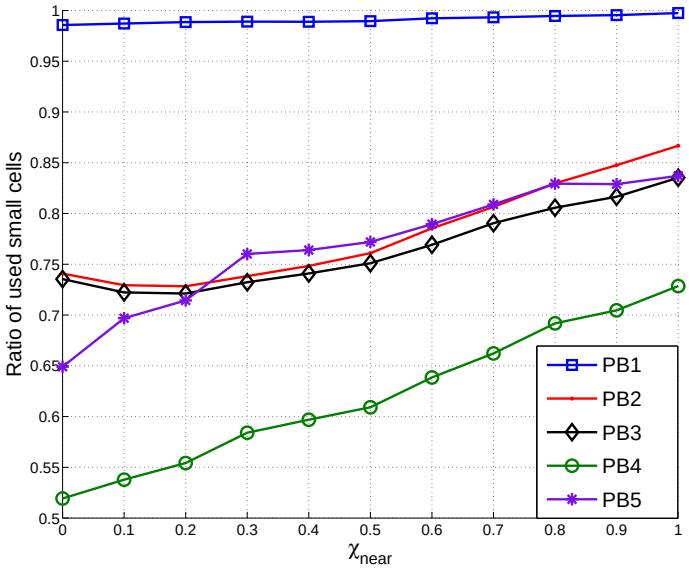


Figure 4.8: Ratio of used HSCs for different strategies

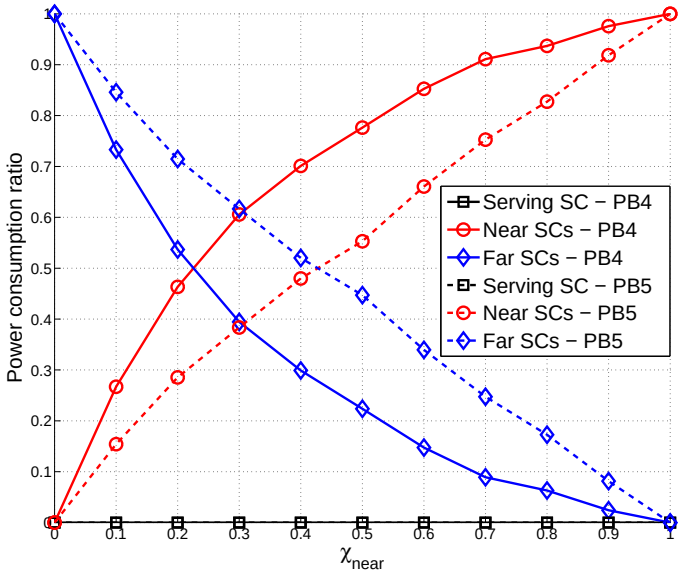


Figure 4.9: Power consumption distribution for power saving strategies

especially in local small cell cloud. SSCs cells are small base stations with limited power and computational capacities. In the case where there are a lot of users, SSCs may receive concurrent requests at the same time. The load distribution and computational resource allocation should be jointly assigned for all users in order to guarantee QoE for all users. The offloading decision would be far more interesting if done with a global view on all active users in the system. If we consider that each user allocates its own resources with a selfish behavior where each SSC forms its own cluster without taking into account the presence of other users' requests, then, not all users

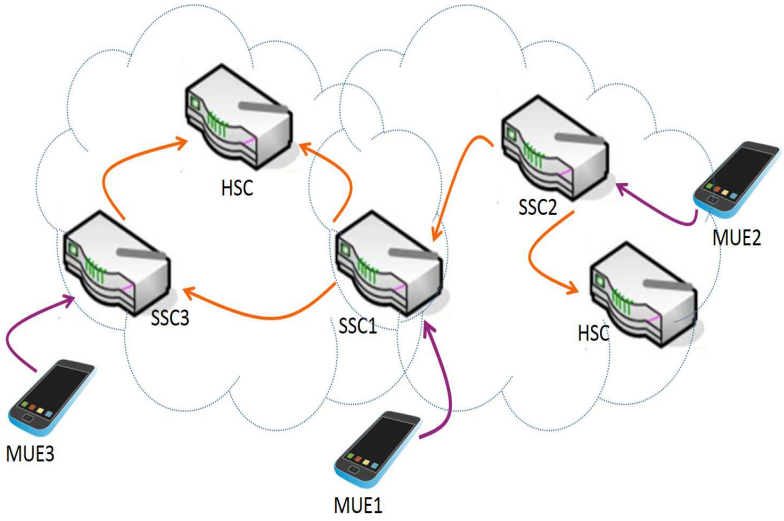


Figure 4.10: Multi-user small cell cloud scenario.

will have enough resources to compute their own tasks. Even if the system resources are sufficient for computing all requested tasks, non-coordinated resource allocation by various SSC can cause system overload. Instead of distributing the load in order to satisfy all requests, SSC may ask for the same resources at the same HSCs, which results in failure in requests computation due to lack of resources.

We study the multi-user computation partitioning and cloud resource allocation problem under application latency constraints. In our work, we consider an idealistic approach where transmissions between small cells are orthogonal. In this thesis, we do not tackle interference management techniques in small cell computation clustering. The idea is to jointly allocate communication and computation resources in the novel proposed edge cloud architecture under considered system characteristics. The goal here is to set a main insight about what can be done in cluster-based distributed edge cloud platforms. The proposed solutions and concepts can be upgraded to be interference-aware, and to adapt cluster set up to radio interference map. Interference aware small cell clustering in edge cloud computing is indeed an interesting step that can be considered as an advanced in-depth future investigation of small cell clustering solutions. For example, cluster set up can target interference limitation goals. Interference in general can be tackled through various mechanisms such as orthogonalization (frequency, time, or space duplexing), diversity increase through repetition coding and MIMO systems for example, coordinated multi-points transmissions, colored graph techniques, and interference margins.

For simplifying notations, we refer to the set of active small cells as $\mathcal{N}$.

We adopt the system model described in 4.2, and we denote by $\mathbf{p} \triangleq (p_{sn}^{k})\forall n,s,k$, $\mathbf{f} \triangleq (f_{kn})\forall k,n$, $\mathbf{w} \triangleq (w_{kn})\forall k,n$ respectively, the transmit powers, computational rates and computational loads associated to each mobile user.

4.4.1 Multi-user Clustering Optimization

We consider a multi-user MEC scenario, where SSC set up computational clusters with the objective of reducing intra-cluster communication power consumption. Our proposed solutions is an extension of $\mathcal{PB}_4$ solution of Section 4.3.4 to the multi-user case. In the case of multiple users, we propose to formulate the problem with the objective of minimizing the sum of transmission

powers inside the cluster. Under the constraints of resources availability at each small cell, the transmission power budget limitation, and the respect of latency constraints, the problem is set as follows:

$$
\begin{aligned}
\min_{\mathbf{p},\mathbf{f},\mathbf{W}} \quad & \sum_{s\in\mathcal{S}}\sum_{k\in\mathcal{K}_s}\sum_{n\in N, n\neq s} p_{sn}^k \\
\text{s.t.} \quad & \text{(a)} \quad w_{kn}\geq 0,\, f_{kn}\geq 0, \quad \forall k\in\mathcal{K}, n\in\mathcal{N} \\
& \text{(b)} \quad \sum_{n\in\mathcal{N}} w_{kn} = W_k, \quad \forall k\in\mathcal{K} \\
& \text{(c)} \quad 0\leq p_{sn}^k\leq P_{max}, \quad \forall s\in\mathcal{S}, k\in\mathcal{K}_s, n\in\mathcal{N}, n\neq s \\
& \text{(d)} \quad \sum_{k\in\mathcal{K}} f_{kn}\leq F_n, \quad \forall n\in\mathcal{N} \\
& \text{(e)} \quad \Delta_{sn}^k(p_{sn}^k, f_{kn}, w_{kn})\leq \Delta_k, \quad \forall n,s\in\mathcal{S}, k\in\mathcal{K}_s
\end{aligned}
\tag{$\mathcal{P}$}
$$

where we define the delay function

$$
\Delta_{sn}^k(p_{sn}^k, w_{kn}) \triangleq \begin{cases} \frac{w_{kn}}{f_{kn}} + \frac{w_{kn}\theta}{R_{sn}^k(p_{sn}^k)} & \text{if } p_{sn}^k.f_{kn}.w_{kn}>0, \forall n\neq s \\ 0 & \text{if } p_{sn}^k.f_{kn}.w_{kn}=0, \forall n\neq s \\ \frac{w_{kn}}{f_{kn}} & \text{if } f_{kn}>0, n=s \\ 0 & \text{if } w_{kn}.f_{kn}=0, n=s \end{cases} \tag{4.17}
$$

with $\theta=\theta_{UL}+\theta_{DL}$ and $R_{sn}=B_{sn}log(1+\frac{\sigma_c|h_{sn}|^2P_{sn}}{(1-PER)\Gamma d^\beta N_0})$. Note that the delay defined as in (4.17) means that: i) for all the non SSCs, the delay function is forced to be strictly positive only if the transmit power, the computational rate and load assigned to the mobile user are non-zero; ii) in case of computation at the SSC, i.e. for $n=s$, the transmit power p_{sn}^k is null then the delay constraint is reduced to a computation time constraint. We note that in the multi-user case, we assume orthogonal transmission and thus we do not consider interference. We are aware that the model stays simple, however, the objective in this thesis was to overview edge cloud architecture to explore what solutions can be found for cluster-based edge cloud computing. We consider the proposed solutions, with the adopted model, as a starting point for devising further in-depth investigations of edge cloud computing implementation. Unfortunately problem $\mathcal{P}$ is non-convex, due to the non-convexity of the delay constraints (e). Nevertheless, in the following we cast $\mathcal{P}$ into a convex equivalent problem. To this end, observe that the delay constraint can be equivalently rewritten for $p_{sn}^k.f_{kn}.w_{kn}>0, n\neq s$ and under the feasibility condition $\Delta_k f_{kn}>w_{kn}$, as

$$
g_{sn}^k(p_{sn}^k, f_{kn}, w_{kn}) \triangleq -B_{sn}\log_2(1+a_{sn}^k p_{sn}^k) + \frac{w_{kn}f_{kn}\theta}{\Delta_k f_{kn}-w_{kn}} \leq 0 \tag{4.18}
$$

where $a_{sn}^k \triangleq \frac{\sigma_c|h_{sn}^k|^2}{(1-PER)\Gamma d^\beta N_0}$. Note that the delay constraint in (4.18) is convex as can be easily verified by proving that the Hessian of g_{sn}^k is a semi-definite positive matrix. The delay condition (e), which imposes a non-convex constraint to our optimization problem, can be reduced for $f_{kn}>0, n=s$, to the linear convex constraint $w_{ks}\leq\Delta_k f_{ks}$. Hence, problem $\mathcal{P}$ can be reformulated as:

$$
\begin{aligned}
\min_{\mathbf{p},\mathbf{f},\mathbf{W}} \quad & \sum_{s\in\mathcal{S}}\sum_{k\in\mathcal{K}_s}\sum_{n\in N,n\neq s} p_{sn}^k \\
\text{s.t.} \quad & \text{(a)} \quad w_{kn}\geq 0,\, f_{kn}\geq 0, \quad \forall k\in\mathcal{K}, n\in\mathcal{N} \\
& \text{(b)} \quad \sum_{n\in\mathcal{N}} w_{kn} = W_k, \quad \forall k\in\mathcal{K} \\
& \text{(c)} \quad 0\leq p_{sn}^k\leq P_{max}, \quad \forall s\in\mathcal{S}, k\in\mathcal{K}_s, n\in\mathcal{N}, n\neq s \qquad (\mathcal{P}_c) \\
& \text{(d)} \quad \sum_{k\in\mathcal{K}} f_{kn}\leq F_n, \quad \forall n\in\mathcal{N} \\
& \text{(e)} \quad g_{sn}^k(p_{sn}^k, f_{kn}, w_{kn})\leq 0, \quad \forall s\in\mathcal{S}, k\in\mathcal{K}_s, n\in\mathcal{N}, n\neq s \\
& \text{(f)} \quad w_{kn}-\Delta_k f_{kn}<0, \quad \forall s\in\mathcal{S}, k\in\mathcal{K}_s, n\in\mathcal{N}, n\neq s \\
& \text{(g)} \quad w_{ks}-\Delta_k f_{ks}<0, \quad \forall s\in\mathcal{S}, k\in\mathcal{K}_s
\end{aligned}
$$

Problem $\mathcal{P}_c$ enjoys some desirable properties as stated in the following theorem.

Theorem 3. *Given problem $\mathcal{P}$ and $\mathcal{P}_c$, the following hold:*
(i) Necessary conditions for $\mathcal{P}$ to be feasible are:

$$
\frac{W_k}{\sum\limits_{n\in\mathcal{N}} F_n}\leq\Delta_k,\; W_k\leq\sum_{n=1}^{N}\frac{\Delta_k}{\frac{1}{F_n}+\frac{\theta}{R_{sn}^k(P_{max})}}; \tag{4.19}
$$

(ii) $\mathcal{P}_c$ is a convex problem then any local optimal solution is a global optimal minimum;
(iii) $\mathcal{P}$ and $\mathcal{P}_c$ are equivalent.

Proof. To prove point (i) of Th. 1 observe that from the constraint (b) it exists $\forall k$ at least a server n for which $w_{kn}>0$. Then from (e) in $\mathcal{P}$, we can write $w_{kn}<\Delta_k f_{kn}$ which leads to the first condition in (4.19). This implies that the maximum delay imposed by the application Δ_k cannot be less than the minimum execution time which can be achieved by a single equivalent server with computational capacity equal to that of the overall network, i.e. $\sum\limits_{n\in\mathcal{N}} F_n$. The second condition in (4.19) is a global condition which can be easily derived from (e) in $\mathcal{P}$. To prove point (ii) in Th. 1 it is sufficient to observe that problem $\mathcal{P}_c$ is convex since the objective function and all the constraints are convex. Then any stationary point is a *global* optimal solution of the problem. It remains to prove point (iii). since for $\mathcal{P}_c$ the Slater's constraint qualification holds true, any optimal solution satisfies the KKT conditions of $\mathcal{P}_c$. The Lagrangian function associated to $\mathcal{P}_c$ is:

$$
\begin{aligned}
\mathcal{L}(\mathbf{p},\mathbf{f},\mathbf{w}) \triangleq & \sum_{s\in\mathcal{S}}\sum_{k\in\mathcal{K}_s}\sum_{n\in\mathcal{N},n\neq s} p_{sn}^k+\sum_{k=1}^{K}\lambda_k\Big(\sum_{n=1}^{N} w_{kn}-w_k\Big) \\
& -\sum_{n=1}^{N}\sum_{k=1}^{K}\beta_{sn}^k w_{kn}+\sum_{s\in\mathcal{S}}\sum_{k\in\mathcal{K}_s}\sum_{n\in\mathcal{N},n\neq s}\Big[\alpha_{sn}^k(p_{sn}^k-P_{max}) \\
& -\mu_{sn}^k p_{sn}^k+\tau_{sn}^k\Big(-R_{sn}^k(p_{sn}^k)+\frac{w_{kn}f_{kn}}{\Delta_k f_{kn}-w_{kn}}\Big) \\
& +\eta_{sn}^k(w_{kn}-\Delta_k f_{kn})\Big]+\sum_{n\in\mathcal{N}}\gamma_n\Big(\sum_{k=1}^{K} f_{kn}-F_n\Big) \\
& -\sum_{n\in\mathcal{N}}\sum_{k=1}^{K}\rho_{kn}f_{kn}+\sum_{s\in\mathcal{S}}\sum_{k\in\mathcal{K}_s}\kappa_{ks}(w_{ks}-\Delta_k f_{ks})
\end{aligned}
$$

where the non negative variables $\alpha^k_{sn}, \mu^k_{sn}, \tau^k_{sn}, \eta^k_{sn}, \beta_{kn}, \rho_{kn}, \gamma_n, \kappa_{ks}$ and $\lambda_k \in \mathbb{R}$ are the Lagrangian multipliers. The KKT conditions are:

$$
\begin{aligned}
&\text{(a')}: \quad \frac{\partial \mathcal{L}}{\partial p^k_{sn}} = 1 + \alpha^k_{sn} - \mu^k_{sn} - \tau^k_{sn} \frac{B_{sn} a^k_{sn}}{1 + a^k_{sn} p^k_{sn}} = 0, \quad \forall s \in \mathcal{S}, k \in \mathcal{K}_s, n \in \mathcal{N}, n \neq s,\\
&\text{(b')}: \quad \frac{\partial \mathcal{L}}{\partial f_{kn}} = \gamma_n - \tau^k_{sn} \frac{w^2_{kn}\theta}{(\Delta_k f_{kn} - w_{kn})^2} - \rho_{kn} - \eta^k_{sn}\Delta_k = 0, \quad \forall s \in \mathcal{S}, k \in \mathcal{K}_s, n \in \mathcal{N}, n \neq s,\\
&\text{(c')}: \quad \frac{\partial \mathcal{L}}{\partial w_{kn}} = \lambda_k - \beta_{kn} + \tau^k_{sn} \frac{f^2_{kn}\Delta_k\theta}{(\Delta_k f_{kn} - w_{kn})^2} + \eta^k_{sn} = 0, \quad \forall s \in \mathcal{S}, k \in \mathcal{K}_s, n \in \mathcal{N}, n \neq s,\\
&\text{(d')}: \quad \tfrac{\partial \mathcal{L}}{\partial f_{ks}} = \gamma_s - \kappa_{ks}\Delta_k - \rho_{ks} = 0, \quad \forall s \in \mathcal{S}, k \in \mathcal{K}_s,\\
&\text{(e')}: \quad \tfrac{\partial \mathcal{L}}{\partial w_{ks}} = \lambda_k - \beta_{ks} + \kappa_{ks} = 0, \forall s \in \mathcal{S}, k \in \mathcal{K}_s, \quad \forall s \in \mathcal{S}, k \in \mathcal{K}_s,\\
&\text{(f')}: \quad \lambda_k \in \mathbb{R}, \forall k, \quad \textstyle\sum_n w_{kn} - w_k = 0,\\
&\text{(g')}: \quad 0 \leq \beta_{kn} \perp w_{kn} \geq 0, \quad \forall k, n,\\
&\text{(h')}: \quad 0 \leq \alpha^k_{sn} \perp (P_{max} - p^k_{sn}) \geq 0, \quad \forall k, n \neq s,\\
&\text{(i')}: \quad 0 \leq \mu^k_{sn} \perp p^k_{sn} \geq 0, \quad \forall k, n \neq s\\
&\text{(l')}: \quad 0 \leq \gamma_n \perp (F_n - \textstyle\sum_{k=1}^K f_{kn}) \geq 0, \quad \forall n,\\
&\text{(m')}: \quad 0 \leq \rho_{kn} \perp f_{kn} \geq 0, \quad \forall k, n,\\
&\text{(n')}: \quad 0 \leq \tau^k_{sn} \perp (R^k_{sn}(p^k_{sn}) - \frac{w_{kn} f_{kn}\theta}{\Delta_k f_{kn} - w_{kn}}) \geq 0, \quad \forall k, n \neq s,\\
&\text{(o')}: \quad 0 \leq \eta^k_{sn} \perp (\Delta_k f_{kn} - w_{kn}) > 0, \quad \forall k, n \neq s,\\
&\text{(p')}: \quad 0 \leq \kappa_{ks} \perp (\Delta_k f_{ks} - w_{ks}) \geq 0, \quad \forall k, s
\end{aligned}
\tag{KKT$_{\overline{\mathcal{P}\mathcal{B}_4}}$}
$$

where $a \perp b$ stands for $\langle a, b\rangle = 0$. Observe that from the complementary condition (o') we get $\eta^k_{sn} = 0, \forall k, n \neq s$. Let us first consider $n \neq s$ by studying separately the two cases i) $\tau^k_{sn} > 0$ and ii) $\tau^k_{sn} = 0$. Under assumption i), it follows from the complementarity condition (n') that the delay constraint is always active. Hence if $p^k_{sn} > 0$ then $w_{kn} f_{kn} > 0$ and conditions (g') and (m') lead to $\beta_{kn} = \rho_{kn} = 0$. Then from (b') we get $\gamma_n > 0, \forall k, n$ so that the computational rate constraint holds with equality and from(c') it results $\lambda_k < 0, \forall k, n$. Finally, assume $\tau^k_{sn} > 0$ and $p^k_{sn} = 0$. then $w_{kn} > 0$, $f_{kn} > 0$ is not an admissible solution since it contradicts the constraint qualification (n') being the delay constraint always active. On the other hand, $w_{kn} > 0$, $f_{kn} > 0$ lean to an absurdum since from (c') one gets $\lambda_k = 0$ while it must always be $\lambda_k < 0$. Under assumption i), it remains to check if the solution $p^k_{sn} = 0, w_{kn} = 0$ is achievable. In this point conditions (b') and (c') reduce respectively to $\gamma_n = \rho_{kn}$ and $\lambda_k - \beta_{kn} + \tau^k_{sn} = 0$. The condition $\gamma_n = \rho_{kn}$ implies $f_{kn} = 0$ since γ_n is always positive. It is important to remark that albeit the feasible set of $\mathcal{P}_c$ does not include the all zeros solution, the admissible solution $p^k_{sn} = w_{kn} = 0$ leads to $f_{kn} = 0$. This permits to reach along gradient directions for which the KKT conditions are not violated, the null delay point enclosed through (4.17) in $\mathcal{P}$.

Let us consider now the case ii), i.e. $n \neq s, \tau^k_{sn} = 0$. From (c') one gets $\lambda_k = \beta_{kn}$ and this contradicts the fact that $\lambda_k < 0$.

The only case left to study is $n = s$. Under this assumption, observe that $\kappa_{ks} = 0$ is not admitted since from (e') λ_k could not be strictly negative. Then let us consider $\kappa_{ks} > 0$. From the complementary condition (p') $w_{ks} = \Delta_k f_{ks}$ and w_{ks}, f_{ks} can assume non-negative values while meeting the

KKT conditions. This implies that according to (4.17), the delay can assume the zero value for $w_{ks} = f_{ks} = 0$.

It is important to remark that $\mathcal{P}$ is a hard to be handled problem due to the discontinuous non-convex delay constraints. Nevertheless as stated in Th. 1, we can find out its optimal solution by solving the equivalent convex problem $\mathcal{P}_c$. □

We note that Eq 4.19 could represent a form of admission control, in the sense that only when that condition is satisfied for all users, then all users' requests are accommodated. When Eq. 4.19 is not met, the solution would require some admission control strategy that decides which users' requests should be accommodated and which ones should be discarded. The admission control process optimization is not addressed in this work, but the theory developed here can be the starting point for devising an admission control strategy that identifies a proper priority of users' requests and selects which users to serve and when. For performance evaluation of the proposed theory, we consider that, in case not all requests could be accommodated, SCM discards request that requires the highest computational capacity (in cycles/sec).

4.4.2 Numerical Evaluation

For evaluating the performance of the joint clustering optimization, we compare it to the case where all requests are handled by the SSC ('No Clustering'), the case where a static clustering rule of equal load distribution between active neighbor small cells is imposed ('Static Clustering'), and the case where clusters are formed for each user successively ('Successive Clusters Optimization'). Comparing to the 'No Clustering' case shows the gain that is introduced by allowing computations execution on small cell clusters instead of a single cell cluster, the SSC. 'Static Clustering' represents the case where a fixed cluster is associated to every SSC. Our solution brings a dynamic approach to the cluster setup, where the number of small cells and their choice is QoE aware and depends on the current system load, computations characteristics. The 'Successive Clusters Optimization' allows each SSC to dimension its clusters for computing its own task. However, the clusters set up for different SSCs is not done jointly. Every SSC sets up its cluster with remaining computational resources. Comparing successive clustering to our proposed solution shows the gain of jointly optimizing multiple users' clusters for a better resource allocation resulting in higher QoE. The simulations parameters are resumed in Table 4.2. First

Table 4.2: Simulation parameters values for the multi-user case

Parameter	**value**	**Parameter**	**value**
ρ_a	0.5	ρ_s	0.32
B	20MHz	σ_c	10
N_0	-118.4 [dB/Hz]	BER	10^{-6}
W	$[2.10^6; 10.10^6]$	Δ_{app}	[0.5;3.5]
F_n	$[10.10^6; 15.10^6]$	P_{max}	1 [W]
θ_{DL}	0.2	θ_{UL}	1

of all, we show how the small cells are chosen to participate in the computational clusters. The solution of problem $\mathcal{P}_c$ jointly forms computation clusters for all users. In a single user case, the optimal power minimizing strategy would be to allocate as much computational load as possible to SSC as seen in Section 4.3.2. Furthermore, power minimizing solutions for a single user case

tend to allocate more computational capacities to near HSCs in order to reduce the transmission cost. Figure 4.11 shows how the clusters are formed in the multi-user case where computational capacities are shared. With an increasing number of users per SSC, each user is allowed to use

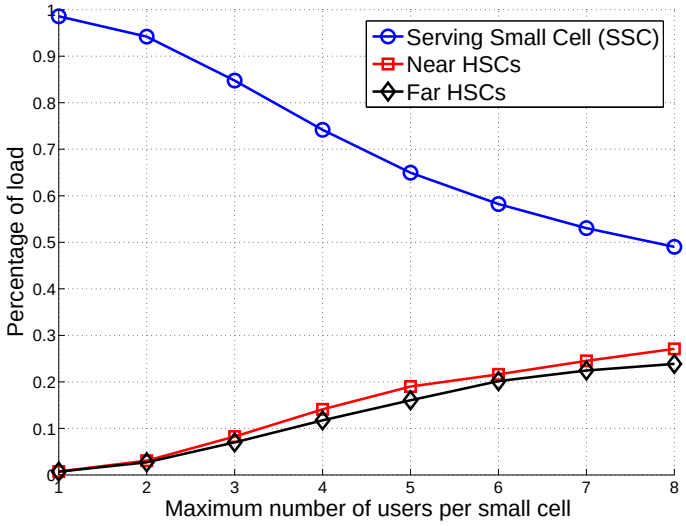


Figure 4.11: Cluster load distribution according small cells distance to SSC

less of the SSC computational capacity and offload more computation to HSCs. Furthermore, we observe that far HSCs are used in the clusters nearly as much as near HSCs. This strategy allocates more computational load to far HSCs when possible in order to increase the system performance and achieve higher satisfaction ratio. Figure 4.12 shows the percentage of satisfied users. A user is satisfied if its computation request result is delivered without violating the imposed latency constraint. In order to evaluate this percentage, we try to solve the optimization problem with the total number of active users in the network. In case of failure of reaching a cluster solution, requests with highest computation loads are dropped one by one until all considered users are satisfied. The satisfied users ratio is evaluated for the considered clustering strategies with respect to the number of served users per SSC.

On Figure 4.12 we show how the joint clustering strategy for all users greatly outperforms all other strategies. The fact of taking into account all the active devices in the system allows better distribution and allocation of both computation and communication resources, and thus, higher QoE.

On Figure 4.13 we observe the average power consumption in the computation clusters. With the *no clustering* case, no data transmissions take place so there's no computation offloading, no communication power consumption but extremely low QoE. Relating with the extremely low satisfaction ratio offered by this strategy (figure 4.12), this strategy is obviously not a suitable choice as soon as the number of devices per small cells increases. In the case of *static clustering*, the power consumption in the computation clusters is higher than in the proposed joint optimization. Despite the fact that in static clustering the SSC communicates with its close neighbors small cells that are subject to better channel quality, we observe high power consumption. With a fixed number of computing small cells in the cluster, the aggregated computational capacity that can be offered is limited. Therefore, the SSC may end up increasing its power consumption for a faster transmission of input data in order to assure the tasks computation without any latency violation.

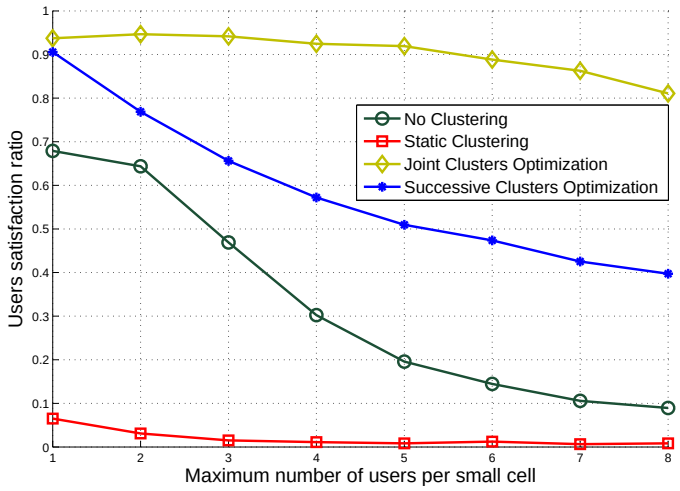


Figure 4.12: Users satisfaction ratio in dependence on number of users per small cell

This leads to higher power consumption for lower users' satisfaction ratio. This figure also shows that the *joint clusters optimization* consumes more transmission power than *successive clustering* with the goal of increasing the satisfaction ratio through more adapted resource allocation. Our proposed solution achieves much higher performance while managing to keep a lower power consumption than the static clustering strategy.

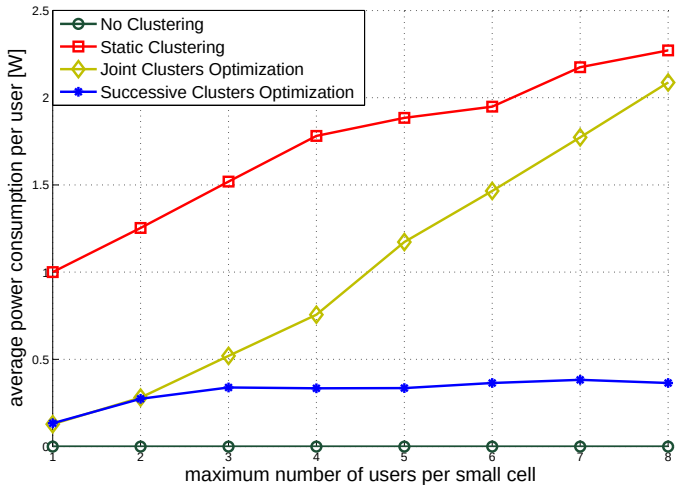


Figure 4.13: Average power consumption per user in dependence on the number of users per small cell

Even though the objective function of the joint clustering optimization problem targets power consumption minimization, we notice that it achieves some gain in the cluster latency. This is mostly due to the cases where local computation resources at the SSC are enough for computing its users' requests. In this case, local computational capacity is accorded to the users' power consumption free, and a latency gain can be achieved. It is clear that *joint clusters optimization*,

compared to *successive clustering*, exploits the latency gain and trades it with higher satisfaction ratio.

Different latency trade-offs can be exploited, where we trade latency for higher gains on other optimization dimensions, such as, cluster cells energy efficiency, cluster size, reduced EMF exposure, and power consumption. Such trade-offs are discussed in Chapter 2. For example, we could reduce the cluster size by means of sparsification and exclude some small cells in order to put them in an idle state for lower equipment power consumption. The computation load of some small cells could be redistributed to others at the cost of increasing the experienced latency.

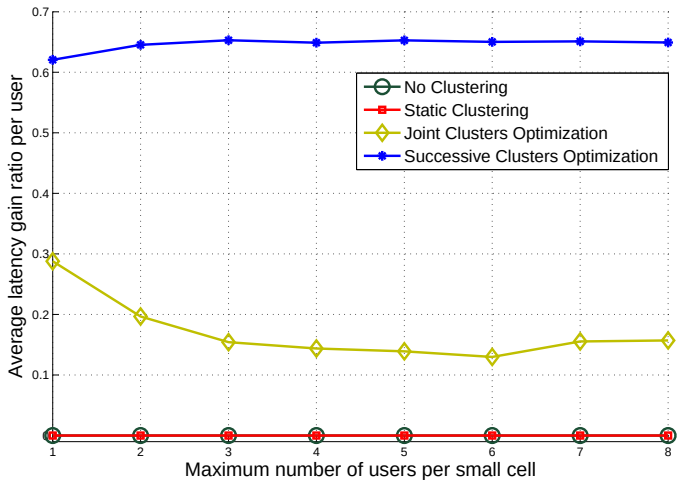


Figure 4.14: Average user latency gain in dependence on the number of users per small cell

The proposed joint clustering optimization might not give the ultimate optimal solution for the multi-user computation clusters establishment problem. This is due to the fact that the small cells are in a computation idle state, i.e. not performing any computation, during the time they are receiving input data. The data transmission time depends itself on the cluster characteristics and parameters, and especially on load distribution and transmission powers. Computing idle time is seen as a loss of computational resources that could eventually be used for serving local devices at each small cell. In addition, when clusters are formed and computational loads distributed, it is possible that some computations execution finish before the others and thus some small cells will have free computational capacities. These capacities are not re-used or re-integrated in the computation process, and can thus be seen as wasted resources. Our proposal is based on a single *one shot* optimization that does not update the load distribution when computational resources are freed. The computational resources that are gradually liberated are however used for the following clusters set up. A possible way to overcome this problem would be to launch the optimization problem with a different starting point of the system state that allows better exploitation of computational resources. However, and despite its non-optimality, the proposed method achieves relevant gains comparing to static strategies where each computation cluster is predefined.

4.5 Conclusion

In this chapter, we consider scenarios where edge cloud small cells are empowered with computational and storage capacities. Small cells have then the ability to act as a local cloud through

small cells cooperation through clustering for computational purposes. The pools of resources thereby created are to be shared among the mobile subscribers connected to the set of small cells. We tackled the issue of load distribution and resource allocation inside the clusters in the small cells local cloud. We started by studying the single-user case where only one computational task is requested at a SSC. The optimal cluster depends on the computational task requirements, the network capabilities and the desired cluster characteristics. First of all, we propose a solution for setting up a computational cluster that can offer the fastest service time, i.e. cluster latency is minimized. Then, we proposed two additional cluster set-up strategies that aim at reducing the cluster power consumption. In a further step, we introduce a novel concept of cluster sparsification. We propose a strategy that aims at reducing computation cluster size, without violating computational requirements especially in terms of time limitations. The proposed solutions in the single user case are based on efficient load distribution between participating small cells in order to guarantee the service delivery to the mobile users. In the multi-user case, the load distribution is more complex. The formulated joint resource allocation and load distribution optimization problem is not convex. Our solution is based on the *convexification* of the formulated problem. We wrote a convex equivalent problem and we proved that it can converge to the solution if it exists. Our proposed solution is based on the trade-off of leveraging overall users satisfaction on user centric QoE. Indeed, if each of the mobile subscribers act selfishly by setting-up its own optimal cluster using one of the proposed single-user solutions, pooled resources will be fought over by several clusters which will dramatically lower the overall users satisfaction ratio. The proposed solution was shown to outperform existing policies by achieving a higher satisfaction ratio. However, the computation of such a solution requires a powerful server with high computational capacities since the complexity scales with both the number of mobile users and small cells. We assumed wireless communication between small cells with known channel characteristics and known characteristics of every small cell. In practice, a signaling system would be necessary for the central unit that computes the solution can collect such information. Furthermore, the proposed solution is a one shot optimization. However, computations do not have neither same time limitations nor are all completed simultaneously. Thus, some computational resources are freed as soon as some computations are done. The proposed solution does not include a strategy for re-using these capacities.

We note that the presented work can be applied to current third generation cellular wireless mobile networks and its future evolutions (5G, 4G, 3GPP LTE, 3GPP LTE A, WiFi, LiFi, WiGiG, WiMAX, etc.) with the modification of the adopted transmission and deployment models. This is because the proposed methods do not request changes in the current standards to be implemented.

Chapter 5

Small Cells Clustering Approaches for MEC

5.1 Introduction

5.1.1 Motivation

In this chapter, we address the same context as Chapter 4. however, we aim at designing low complexity clustering algorithms, that do not contain complex optimization problems solving.

We tackle the computing cluster set up paradigm for mobile edge cloud. In MEC, MUEs have the possibility to offload their computation to be executed at the local edge cloud. In the architecture we adopt, the cloud functionalities are deployed at network edge small cells in close proximity to mobile users. MUEs send offloading computation requests to their SSC. The SSC sets up a computing cluster for executing the received requests. Computation load is distributed among the small cells participating in the cluster. In order to respect the latency constraints imposed by the offloaded tasks, the cluster should consider jointly the allocation of both computational and communication resources, and load distribution within the cluster. In this chapter, we propose heuristic small cell clustering algorithm that also considers joint communication and computation resource allocation.

The clustering optimization solution proposed in Chapter 4, for both single user and multi-user use cases, are based on joint optimization of cloud cluster resources formulated as an optimization problem. The proposed solution proved to be efficient and guarantee high users' satisfaction ratio, i.e. guarantee the service of a high number of users without violating the latency constraints. For the single-user case, the optimization problem is simple to solve. In some cases, as in the case of cluster latency minimization, a closed form solution can be found. However, in the multi-user case, the optimization problem is more complex, and non-convex. Even though we proposed a solution to compute a clustering solution using an equivalent convex optimization problem, the number of parameters to optimize increases with the number of users and the number of small cells. In small scale scenarios, this solution can be implemented and guarantee better QoE. On the contrary, in medium or large scale scenarios, the number of parameters to optimize becomes very large. Solving the optimization problem in this case can be time, and computational capacity consuming, which may have a negative impact on finding a feasible solution. In cases where the solution computing time is not negligible, the delay is omitted from the computational time tolerance, and thus reducing aggregated resources. The number of variables to optimize is equal to $3 \times N \times K$ where N and K are the number of considered small cells and users respectively. Therefore, the implementation of such a solution requires a powerful small cell manager that can derive the solution without affecting the feasibility of the derived solution. In Chapter 4, the challenge was to find a clustering solution that jointly optimizes computational and communication resources, and distributes the load, while guaranteeing a high QoE. In this chapter, the challenge is different. The motivation of this chapter is to relax the cluster set up complexity for avoiding the implication of small cells in complex clustering set up calculations. Hence, alternative algorithms with low complexity are needed for solving the cluster set up problem.

5.1.2 Related Work

The topic of small cells clustering for computing purposes in the MEC is relatively new and very few solutions have been proposed in the literature. In addition to the related work reported in Section 4.1.2, we report the following related work. Three resource allocation algorithms, *'Path'*, *'Comp'*, and *'ACA'* for SCC clustering are proposed as part of the solution that TROPIC propose [115]. The first algorithm, *'Path'*, is based on transmission quality between UEs (Users' Equipments) and SCs. The SCs with the best 'path' quality are selected for participating in the

computation process. The *'Comp'* algorithm is based on the computational power available at each SC. It is the SCs with the highest computational power at the estimated data delivery time that participates in the computation process. In addition, a combination of both algorithms is proposed under the name of *'ACA'* algorithm. By estimating the computational load to be executed (according to the application type), an overall delay is computed taking into account both path quality and available computational power. SCs with lowest overall delay are selected for participating in the computation process. Tasks are handled in all of these algorithms in a FIFO (First In First Out) manner. Indeed, for simplicity, no scheduling policy is considered. Treating tasks as FIFO may not always be the best scheduling solution especially in the case where tasks characteristics vary in latency constraints and computation load. Furthermore, the proposed algorithms are applied in a scenario where the number of participating SC is predefined. On the contrary, with our proposed approaches, the SCC size is adapted to the need and dimensioned to satisfy the computational requests.

Niyato *et al.* propose a game theoretic modeling of cooperation in mobile cloud computing [116]. However, the addressed cooperation is between different service providers data centers, which are separated from the radio access points. The idea is to allow multiple providers of mobile cloud services to cooperate and create a resource pool and serve a higher number of users, and support a higher number of applications instances. This work addresses the computation part at the computing data centers coalitions as well as the communication part between MUEs and radio access points. Wireless access points communicate with computing servers through wired links. It proposes an admission control method enabling the users to access the resource pool owned by the coalition (cluster) of mobile service providers. In order distribute communication and computation resources of base stations and data centers among mobile users, an admission control mechanism is proposed. An optimization problem is formulated to obtain the maximum number of supported users by base station and data center. The goal of this optimization is to maximize the revenue of data center providers. The coalition of data centers is assumed to be chosen a priori, and any coalition algorithm is proposed or presented. However, a distributed algorithm that allows the data centers and base station to increase the amount of resources used in the cluster is proposed. Base stations and data center have the option of increasing their participation in the cluster in order to maximize their revenue, but once they are part of a coalition, they are forced to participate in the computation cluster since it is considered that this participation will generate revenue for the service providers. However, in the context of small cell cloud where the data centers are users' deployed small cells, more strict privacy policies, security concerns, or cost reduction strategies may be encountered. Therefore, forcing participation in a coalition is not always possible, and it depends on adopted small cells deployment models (open access, closed access, hybrid access) [117].

Garg *et al.* tackle computational capacity and resource allocation on a cloud data center [107]. Even though the authors do not consider the same context of small cell cloud computing, their work shows different policies to allocate resources on a computing entity that we also use in the proposed joint clustering and resource allocation algorithm. It considers an admission control and scheduling policy that allow virtual machines to run on servers while minimizing the penalty resulting from the violation of Service Level Agreement (SLA). SLA is modeled as a time limit for computing requests. The admission control policy consists in estimating the amount of available resources at each server based on an Artificial Neural Network (ANN) forecasting model that predicts future demands of computational tasks. Requests are admitted based on the resources estimation. As for the scheduling policy, two different resource allocation strategies are considered. The first resource allocation strategy consists in giving applications the total amount of computational resources they require. The second strategy consists in allocating computational resources

to applications within the limits of availability at each server. This means that applications are given the maximum between required and available resources at the server. If available resources are lower than required by the application, computation delay is larger, and a SLA violation may occur. Then, for minimizing the SLA violation penalty, resources are redistributed inside each server according to the need of the running applications.

5.1.3 Contribution

In this Chapter, we approach the problem of small cell clustering for computation purposes with two different algorithms. We propose two novel solutions, which both aim at keeping advantages of dynamic clustering and adapting cluster size according to current users' demands. We focus on managing resources for the set of small cells forming the local small cell cloud. Furthermore, we do not consider the cloud as an established entity — as opposed to the state-of-the-art. Instead, we dynamically set up the computation clusters by choosing which small cells to include and how to distribute the load among them.

Our first proposition is based on a sequential algorithm that can be split into two phases: local computation resource allocation and small cell cluster establishment. The first phase consists in scheduling computational tasks and the second in setting up the clusters for computation. Three different implementations of the algorithm are proposed. They differ in scheduling metrics and cluster optimization objectives. Details are given in Section 5.3.

Our second proposition is based on an iterative clustering algorithm. The process is divided into two phases: A first phase in which serving small cells computes their clusters *selfishly*, and a second in which the small cell manager validates or corrects the clusters of each SSC. Details are given in Section 5.4.

As computational resources are shared by the set of serving small cells, their allocation should be coordinated between small cells. Allowing each small cell to set up its own cluster without taking into account the presence of other computational requests will lead to a problem of resources management. Some small cells will be overloaded, while others will have unallocated available resources. Load balancing is required in order to minimize the number of dropped requests. In both proposed algorithms, we rely on the low complexity of the single-user multi-cloud cluster optimization. The designed algorithms introduce cluster set up coordination between various SSCs in order to efficiently distribute computational load and resources among computational requests.

The novelty of this chapter is based on a patent [P4] and two conference papers [C4] and [C6].

5.2 System Model

In this Chapter, we adopt the same system model of Chapter 4. We consider a multi-user scenario where the set of users $\mathcal{K}$ are served by a set of small cells $\mathcal{N}$. Computation offloading requests are defined by the pair (W_k, Δ_k) that represent the number of CPU cycles to execute and latency constraints. Note that the relationship between the number of instructions and the number of CPU cycles depends on the instructions type. We assume high granularity, and we split computational load over CPU cycles. SSCs communicate with HSCs through direct point-to-point wireless. The full system model details can be found in Section 4.2.

As for simulations, we consider also a femtocell deployment in a grid of 25 10m $\times$ 10m apartments, which is known as the 3GPP grid urban deployment model [113]. Simulation parameters can be found in Table 4.2 of Section 4.4.2.

5.3 Small Cell Cloud Clustering: A Scheduling Approach

5.3.1 General Algorithm

In this approach, we propose a novel five steps clusters establishment procedure. We propose to split the resource allocation process into two major phases: local computational resource allocation and small cell cluster establishment. The general scheme of the algorithm is described through the following steps resumed in Figure 5.1.

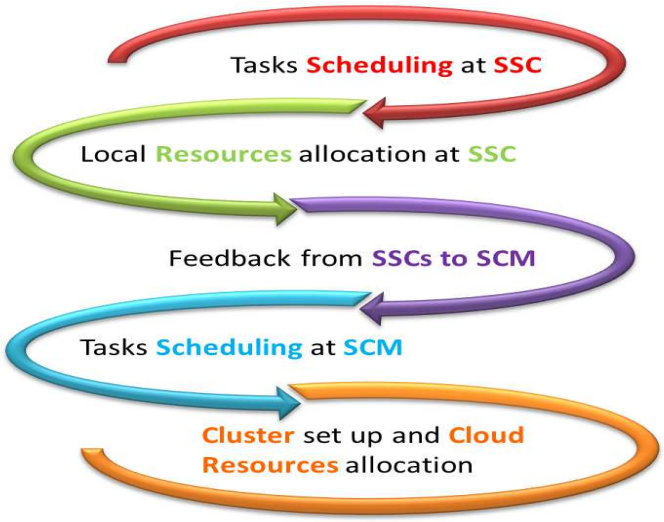


Figure 5.1: Scheduling aware clustering proposed algorithm scheme

Step 1: ***Local computational resource allocation***

First, local computational resources are allocated at serving small cells. Local resources at the SSCs are allocated for users that are served by this cell represented by the set of users $\{k \,|\, \mathcal{S}_k = s\}$.

- **Step 1.a)- Scheduling at SSC** Each serving SC accords priorities for tasks received from its connected users, according to a specified metric such as latency constraint, computation load, minimum required computational capacity, minimum required energy efficiency, or arrival time. The priority assignment defines the scheduling rule for local resource allocation. Some examples are the following: (i) sorting according to latency constraint corresponds to an Earliest Deadline First scheduling (EDF) [118]; (ii) sorting according to the order off arrival corresponds to a First In First Out (FIFO) scheduling; (iii) sorting according to each user service rate corresponds to a Proportional Fairness (PF) scheduling [119]. This step gives different priorities to users' requests depending on the adopted sorting metric or the scheduling policy.

- **Step 1.b)- resource allocation at SSC** Serving small cell computational resources are allocated to users' requests following the ordered list established in step 1.a. The resource allocation policy may have different objectives: increasing latency gain, or increasing resources availability time. For example, users can be accorded the maximum amount of re-

sources and thus computation are done in a minimal duration. Or, conversely, lower amount of resource allocation can be allocated over a larger time window.

Step 2: ***Small cell cluster establishment***

As local computational resources are limited, only a limited number of requests can be served with SSC local resources. Whenever the local SSC has insufficient resources to serve the user, unserved remaining requests are handled using a small cell cluster. To this end, a computing cluster is set up by the SSC to comply with their specific requirements.

- **Step 2.a)- Requests update** Information on requests to be offloaded and remaining actualized computational capacities remaining at each serving small cell is reported to the small cell manager (SCM). Note that this step does not introduce additional overhead comparing to traditional centralized solutions. In fact, all centralized algorithms and solutions suppose the presence of a small cell cloud control entity that receives and stores system parameters and requests requirements.

- **Step 2.b)- Scheduling at SCM** Unserved requests of all SCs are classified at the SCM according to a specific metric or a scheduling rule. Note that the scheduling policy can be the same or not as in step 1.a. As the first scheduling on mobile handsets depends on the policy of the device in computing tasks, in this step it depends on the small cell clustering policy and the desired clusters characteristics.

- **Step 2.c)- Clusters set up** Computation cluster are built for each of the unserved requests following the scheduling order established in step 2.b. Clusters are built for each request independently of the other requests presence. Possible single-user cluster policies that can be adopted in this step are proposed in Section 4.3. Setting up computation clusters for one request at a time reduces the complexity of the clustering process. For guaranteeing a lower outage probability, i.e. a lower number of unserved requests, the focus is then on choosing the best scheduling policy and clustering policy. When clusters are set up successively for the scheduled requests at SCM, some small cells computational capacities will be fully allocated. These small cells are excluded from the following cluster set ups. Optimization search space is indeed reduced. Moreover, small cells that have computational resources released will be added to the cluster optimization search space. Computational resources are then allocated to as long as necessary and reused when released.

5.3.2 Algorithm Implementations

These general algorithm steps constitute a customizable sequence for small cell clustering for a multi-user scenario. Several versions can be built using this algorithm by varying the scheduling metrics and resource allocation and clusters set up policies. We consider three different implementations of the algorithm EDF-PC, EDF-LAT, CS-LAT where the notations refer to the local scheduling rule and the clustering optimization objective.

(i) EDF-PC
EDF-PC is based on an Earliest Deadline First scheduling on SSC, and a power consumption minimizing clustering strategy. **[1.a]** The requests are sorted in ascending order of latency constraints. This choice of sorting metric imposes an EDF (Earliest Deadline First) scheduling. Priority is given to tasks with tightest latency constraints. FIFO, used for testing

algorithms in [115], is not an adequate scheduling policy for the specific case of our adopted testing scenario because FIFO scheduling does not take into account QoE parameters. Our objective is to associate the scheduling policy with a clustering policy for achieving higher QoE.**[1.b]** Local resources are allocated following a policy that blocks the minimal required computational capacity for each request. This capacity can be expressed as the ratio between the computation load and the latency constraint ($\frac{W_k}{\Delta_k}$). We refer to this policy as *horizontal allocation* (see Figure 5.2(a)

[2.b] Unserved requests are sorted in ascending order of available latency.

[2.c] Clusters are formed for users with the objective of minimizing overall clusters communication power consumptions. The problem formulation used for this clustering strategy is cast for user k served by SC s $\mathcal{PB}_4$ of Section 4.3.4 of Chapter 4.

(ii) EDF-LAT

EDF-LAT is based on an Earliest Deadline First scheduling on SSC, and a latency minimizing clustering strategy. This implementation has the same steps of **EDF-PC** except for step 2.c where clusters are formed with the objective of minimizing overall cluster latency. The latency minimizing clustering problem is cast for user k served by SC s as described in Section 4.3.2 of Chapter 4.

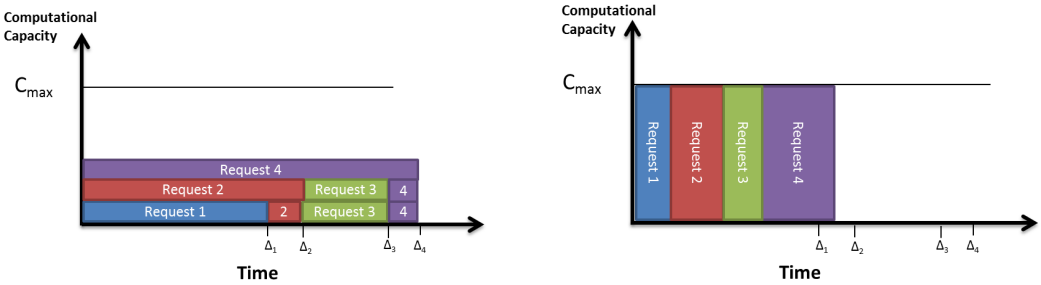
(iii) CS-LAT

CS-LAT is based on a *largest computation first* scheduling on SSC, and a latency minimizing clustering strategy. **[1.a]**: Requests are sorted in decreasing order of requested computations size.

[1.b] Local resources are allocated following a policy that blocks the maximal computational capacity available for each request. This will block local resources for smaller time periods and achieving higher latency gains. We refer to this policy as *vertical allocation* (see Figure 5.2(b)

[2.a] Unserved requests are sorted in ascending order of latency tolerance.

[2.c] Clusters are formed for users with the objective of minimizing overall clusters latencies using the optimization problem $\mathcal{PB}_1$ of Section 4.3.2 of Chapter 4.



(a) Horizontal Allocation

(b) Vertical Allocation

Figure 5.2: Resource blocks allocations of several requests r of delay constraint Δ_r with horizontal and vertical allocations.

5.4 Small Cell Cloud Clustering: An Iterative Approach

Our second proposal takes the form of an iterative algorithm that is based on a novel strategy for radio access points based on two management layers: decentralized and centralized. This strategy allows to reduce cluster management time, through its decentralized layer, and maintain mobile users' QoE through the centralized layer by guaranteeing the respect of latency constraints. We assume homogeneous data generation of computational data at set of mobile devices $\mathcal{K}$, i.e. all generated requests have the same range of latency constraints. The decentralized part of the algorithm requires a form of signaling between the network SCs. It consists on *selfish* clustering at each serving small cell, where each small cell forms its own optimal computation cluster. Serving small cells interrogates neighbor small cells about their available computational capacity. According to link qualities and available computational rates, the serving small cells builds its own computational cluster. As for the centralized part, it requires the presence of a cluster management unit that has knowledge of the SCs characteristics in terms of both available computational capacities and channel link qualities between SCs. SCs available resources are reported by SCs to the SCM at a regular basis. As for channel link qualities, either statistical or instantaneous knowledge can both be considered. Each serving small cell sends its clusters load distribution to the central management unit, which acknowledges the global load distribution. If any SC has been accorded more computational load that it can offer, the management unit re-distributes the excess of allocated load among serving small cells. Management unit reports back the remaining computation load to serving small cells. This process is repeated until the entire computation load is distributed, or until the system has reached the maximum computation capacity that can be offered. A maximum delay or power consumption cost can also be used as stopping criteria for the algorithm.

The novelty of this proposal is indeed threefold. First, we propose to exploit optimization solutions for the single user case, and we propose to perform a first guess of resource allocation per user neglecting that there are other concurrent requests from other users. We will refer to this case as *selfish* for which optimization is performed without considering the presence of other users' requests. Second, we propose to introduce a supervisor which verify the feasibility of the union of *selfish* users' requests. Third, we propose to inform/notify user equipment on excess of demands which are computed with the proposed *selfish* approach. Users are informed if they are or not in offloading request excess. Therefore, the supervisor checks the combination of available resources and their cost in terms of connectivity. More details will be given in the algorithm description that follows hereafter.

The proposed iterative algorithm is represented on Figure 5.3, it has several steps described as follows:

A . Decentralized management layer steps

- **Step 1: Update of available computing capacity of neighbor small cells**
  Each serving SC $s \in \mathcal{S}$ with a computation task demand, requests neighbor SCs $n \in \mathcal{N}, n \neq s$, using special form of signaling, to report available computational capacity.
- **Step 2: One user guess *selfish* optimal allocation**
  Each serving SC s computes its optimal computational cluster for each request of user $k \in \mathcal{K}_s$, independently of other parallel allocations requested by different SSCs. The optimal cluster is computed by using the optimization problem $\mathcal{PB}_4$ whose objective is to minimize the communication power consumption in the cluster. This problem has been studied in 4.4 of Chapter 4.
- **Step 3: Cluster reporting to central management unit**

Each serving SC s reports its cluster load distribution and resource allocation variables W_{kn}, p_{kn}, and f_{kn} to the central management unit. Failure of building a computation cluster, if occurred, is also reported.

B . Centralized management layer steps

- **Step 4: Central Unit Check on Feasibility of distributed allocation requests**
  In this steps there are two possibilities:

  (a) No failure of clustering has been reported

  (b) Failure of clustering has been reported

  It could happen that SSCs are not able to set up their clusters. This could be a reason of lack of computing resources, errors in resources availability information, or low connectivity with neighbor small cells. Therefore depending if we are in the case (a) or case (b) we have different sub-steps in the proposed algorithm. Hereafter, we refer to steps with '.a' when referring to case (a) and '.b' when referring to case (b).

Case (a): No failure of clustering has been reported
In this case, all serving small cells have found at least one *selfish* solution (allocation that does not consider potential resources allocated by other serving small cells forming their own clusters).

- **Step 4.a: Compute W_n: total aggregated allocated load at each SC**

$$W_n = \sum_{k=1}^{K} w_{nk} \tag{5.1}$$

- **Step 5.a: Update available computing capacity (F'_n) and load excess (X_n) at each SC**
  Update available computational capacity at each SC n as:

$$F'_n = (F_n - \sum_{k=1}^{K} f_{kn})^+ \tag{5.2}$$

  and compute excess of allocated computation X at each SC $n \in \mathcal{N}$

$$X_n = (W_n - Th_n)^+ \tag{5.3}$$

  where Th_n is the maximum computation load that can be allocated at each SC, $Th_n \leq F_n\Delta$.

- **Step 6.a: Redistribute available resources excess**
  In this step of the algorithm, the proposed optimization distributes the excess of load on users that has the best opportunity of computing the remaining CPU cycles with available resources. Indeed we propose to redistribute the excess of computational load allocation at each SC $b \in \mathcal{N}/X_b > 0$. This excess of load is distributed among mobile devices that include n in their computing cluster. The redistribution follows the following rules:

$$
\begin{array}{lll}
\min\limits_{\mathbf{w}'} & \sum\limits_{n\in\mathcal{N}}\sum\limits_{k\in\mathcal{K}} w'_{nk} bpt_{nk} & \\
\text{s.t.} & 0 \leq w'_{nk} \leq w_{bk} & \forall n,k \\
& w'_{nk} = 0 & \forall k / w_{bk} = 0 \\
& w'_{nk} = 0 & \forall n / F'_n = 0 \\
& \sum\limits_{n\in\mathcal{N}}\sum\limits_{k\in\mathcal{K}} w'_{nk} = X_b &
\end{array}
\qquad (\mathcal{P}')
$$

where bpt_{nk} is the minimum bit processing time (transmission and computation) that SC n can offer for user's k request as:

$$bpt_{nk} = \frac{1}{F'_n} + \frac{1}{R_{max}(P_{max})}$$

where R_{max} is the maximum bit rate that can be achieved. This optimization distributes the excess of load on users k that has the best opportunity to execute remaining CPU cycles. The first condition guarantees that redistributed load does not exceed the allocated load. The second condition limits requests that can be re-allocated to the set of requests that were served by n'. The third condition forbids re-allocation to SCs that have no available computational capacity. The last condition guarantees distribution of all excess of computation load.

- **Step 7.a: Centralized controller report W' and F' to SCs**
  Report $\mathbf{W}'$ and $\mathbf{F}'$ to SCs.
- **Iteration.a**
  Repeat from Step 2 with computation load request $\mathbf{W}'$ and system computational capacity $\mathbf{F}'$.

Case (b): Failure of clustering has been reported
In this case at least one serving small cell has not found in its *selfish* resource allocation guess enough resources to serve offloading remands within the required QoE.

- **Step 4.b: Drop requests in excess**
  Drop non-served computation request with the highest required bit processing time defined by $\frac{W_k}{\Delta}$.
- **Step 5.b: Update computing capacity at each SCs (F')**
  After the drop of a request, some allocated computing resources are then freed. Therefore we update the available computational capacity in the system as $F'_n \leftarrow F'_n + f_{nk}, \forall n$, where k is the index of the eliminated request.
- **Step 6.b: Centralized controller report W' and F' to SCs**
  Report W' and F' to SCs that serve the dropped user k. The task drop decision is then sent to user k.
- **Iteration.b**
  Go back to Step 2 with updated computation load request W' and system computational capacity F'.

The algorithm is run until one or more of the considered stopping criteria occur:

(i) all requests are either successfully computed or eliminated

(ii) a fixed maximum number of iterations is reached

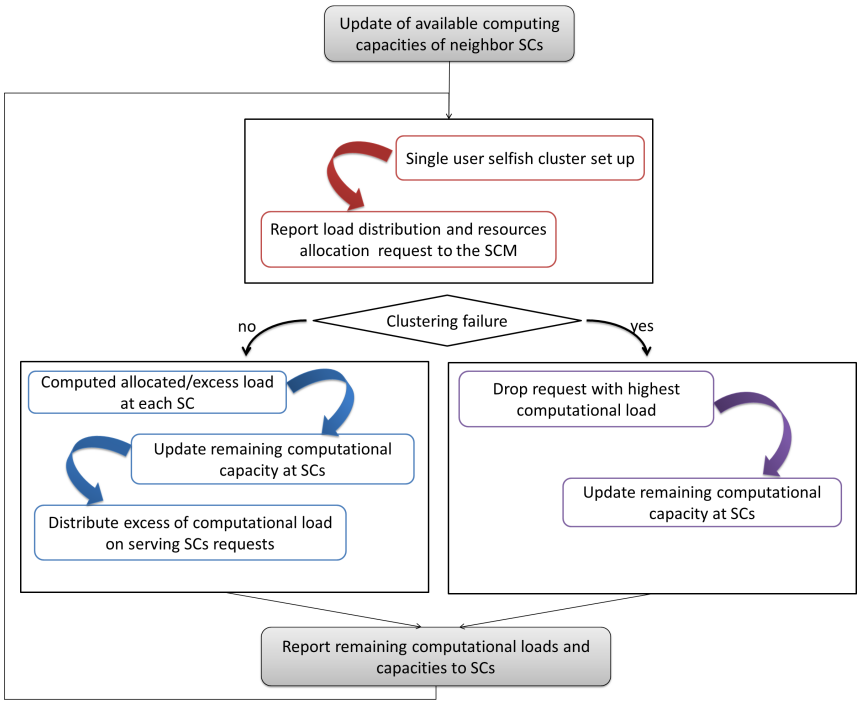


Figure 5.3: Proposed iterative clustering algorithm scheme

(iii) maximum algorithm delay tolerance is reached

(iv) maximum algorithm power budget is reached

The proposed management method joins benefits of both centralized and decentralized management. The central management unit does not solve heavy problems since clusters are computed individually at SSCs for a single user request. This reduces management time loss, since single-user clustering solutions are rapid to compute. At the same time, SCs report to one central managing unit and not to each other; this reduces the feedback signaling overhead. The presence of the central managing unit can be seen as a control check that readjusts clusters composition whenever decentralization fails, which increases QoE.

5.5 Numerical Evaluation

In this section, we evaluate the proposed strategies of small cell clustering in computation clusters. We provide numerical evaluation considering the case of femtocell deployment for urban scenarios model of the 3GPP framework [113]. This scenario is represented by a single floor building of a 25 apartment grid. In each of these apartments a femtocell is deployed. Parameter ρ_a determines the ratio of active femtocells in the grid. Parameter ρ_s determines the ratio of active femtocells that are connected to mobile users (serving femtocells). We adopt the same system model described in Section 5.2 with parameters values listed in Table 5.1.

In the following, we benchmark our proposed strategies algorithms with four different clustering strategies that we define below: **No Clustering**: all requests are handled by the serving SC, there is no computation offloading. There is no cluster formation. The comparison with this strategy shows the benefits of small cells clustering for mobile edge computing.

Table 5.1: Simulation parameters values

Parameter	value	Parameter	value
ρ_a	0.5	ρ_s	0.32
B	20MHz	σ_c	10
N_0	-118.4 [dB/Hz]	BER	10^{-6}
W	$[2.10^6; 10.10^6]$	Δ_{app}	[0.5;3.5]
F_n	$[10.10^6; 15.10^6]$	P_{max}	1 [W]
θ_{DL}	0.2	θ_{UL}	1

Static Clustering: a static clustering with a fixed set of small cells contributing in the clusters. Equal load distribution is applied among the cluster cells. Each small cells has in its computation cluster neighbor small cells only (separated by 1 wall). This imposes then the size of the cluster, which is not dynamically adapted to the computation demands.
One shot optimization: dynamic clustering with one shot optimization. This strategy is the one we proposed in Section 4.4. It consists in computing all computation clusters for all computation requests simultaneously. Computation requests parameters (number of instructions, number of CPU cycles, delay constraints, requesting user, etc.) are sent by SSCs to the managing unit, SCM, that has the role of jointly allocating resources for all requests computation.
Successive clustering: dynamic clustering where each serving SC computes its own computation clusters after gathering necessary information from neighbor SCs. Interrogated SCs wait for the serving SC feedback in order to update system state information before engaging in a new computation cluster. Clusters are computed successively at serving SCs starting with requests with highest computational capacity demand.

We start by comparing the performance of the three implementations of the first approach described in Section 5.3 and resumed in Table 5.2.

Table 5.2: Algorithm implementation metrics and policies

Step	Step description	EDF-PC	EDF-LAT	CS-LAT
1.a	Local scheduling at SSC	Latency (ascending)	Latency (ascending)	Computation size (ascending)
Step 1.b	Local resource allocation policy	Horizontal	Horizontal	Vertical
2.b	Tasks scheduling at SCM	Latency (ascending)	Latency (ascending)	Latency (ascending)
2.c	Clustering policy	Power consumption minimizing	Latency minimizing	Latency minimizing

Figure 5.4 shows the users' satisfaction ratio for considered algorithms (number of users served without violating latency constraints) with respect to the maximal number of users simultaneously offloading computation requests to the same serving small cell.

With increasing number of users and incoming requests, pooled resources are to be shared by SSCs to serve a larger number of offloaded computational requests. Thus, due to the lack

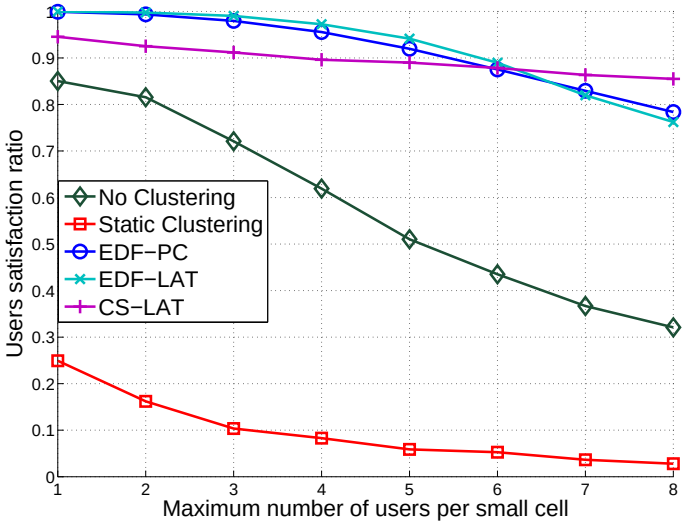


Figure 5.4: Users satisfaction ratio in dependence on number of users per small cell

of computational resources, satisfying all users becomes harder and, in some cases, impossible. In the case where each small cell computes the requests of its own users (*No clustering*), the users satisfaction ratio drops decreases almost linearly with the increasing number of users as seen in Figure 5.4. It is important to note on the figure, the low performance achieved by the static clustering strategy where computation load is equally distributed on neighbor small cells. This shows that clustering can be a inadequate solution if not adaptively orchestrated. All of the three variants of the proposed method show important gain in satisfaction ratio even for a high number of users per small cell. We can see that EDF-PC and EDF-LAT can maintain over 95% of satisfaction ratio for up to 4 users per femtocell. Classical usage of femtocells is about an average of 4 mobile users [120]. We extend our numerical evaluation for up to 8 users considering that a larger number of femtocells connected users can be foreseen in the future. Since both of these algorithms schedule the users based on latency urgency, they manage to serve a larger portion of users compared to the CS-LAT, which aims at achieving larger latency gain. Nevertheless, CS-LAT manages to keep a satisfaction ratio of at least 90% for less than 4 users per femtocell.

In Figure 5.5 we also compare performance of the iterative proposed algorithm proposed in Section 5.4 with the EDF-PC implementation and other considered clustering strategies.

The centralized *one shot optimization* outperforms all other strategies. Since the central managing unit is fully aware of the system state, and since it jointly computes clusters for all requests, it yields to better performance in terms of satisfaction ratio. Nevertheless, this comes at the cost of higher complexity since larger optimization problems need to be solved. Furthermore, in this strategy, if all requests cannot be satisfied with a *one shot optimization*, the central management unit drops a request and runs again the optimization until a solution is found. In this particular case, the time complexity is almost multiplied by the number of solved optimization. The proposed algorithm achieves lower satisfaction ratio than the centralized *one shot optimization*, but it clearly outperforms the *successive clustering* and *no clustering* strategies. It approaches the performance of centralized strategy especially at low number of users per small cell with less than 10% performance degradation for less than 3 users per small cell and less than 20% for up to 6 users per small cell. The gain comparing to *successive clustering* and *no clustering* becomes

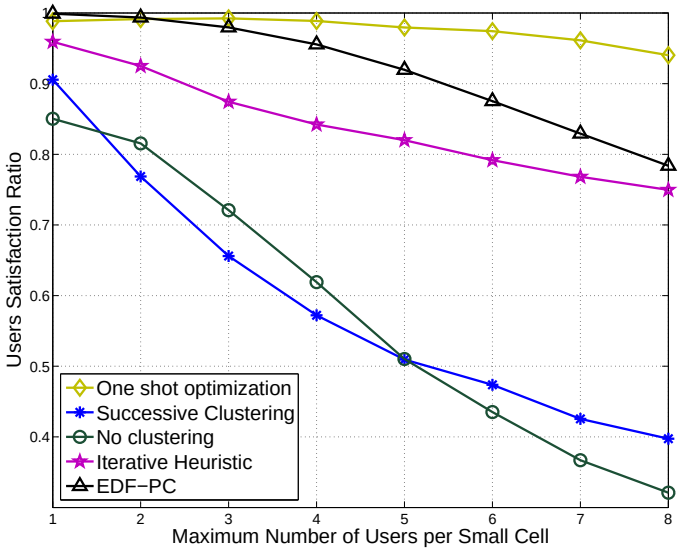


Figure 5.5: Users satisfaction ratio in dependence on the number of users per small cell

of more importance with higher number of users per small cell. It is shown that even with high number of users per small cell, the proposed strategy can guarantee a minimum of 75% of satisfaction ratio. On Figure 5.5, it is important to notice that *successive clustering* can have even lower performance than computation at the SSC (*no clustering*). *Successive clustering*, will allocate the SSC computational capacities to If we compare *successive clustering* and no clustering, we notice that *no clustering* has, in some cases, a better performance. This shows that clustering is only a good solution when we use a good strategy that is adapted to the system conditions.

In Figure 5.6 we show the average latency gain per user versus the number of users per small cell. The latency gain for each user is defined as the ratio between the experienced latency for computing the request and the imposed latency constraint. We designed the iterative algorithm to minimize cluster power consumption ($\mathcal{PB}_4$). This means that the cluster will trade the delay allowed by the computation requests in order to reduce energy consumption. Therefore, in this case, no latency gains are observed.As shown in Figure 5.6, the CS-LAT algorithm achieves the highest average latency gain. This due to both local and cluster resource allocation. In fact, in this algorithm, the local serving femtocell allocates its full computational capacity to compute the requests, which results in completing the task with the lowest possible latency. Furthermore, clusters are formed for unserved requests with the objective of minimizing the overall cluster latency. On the other hand, the EDF-LAT latency gain is a result of latency minimizing clustering strategy. As a matter of fact, serving femtocell local resource allocation policy exploits all the available time for each computation request by allocating the minimal required computational capacity. The more users are connected to the femtocells, the more requests are received by the SSC, and the fewer requests are able to be computed locally on the serving femtocell. Therefore, higher latency gain can be achieved with this algorithm with the increasing number of users since more requests are handled to latency minimizing clusters, as can be seen on Figure 5.6.

The high increase in latency gain achieved by algorithm with a latency minimizing clustering policy comes at the cost of increasing the communication power consumption in the cluster. This is due to the basic wireless communication trade-off between power consumption and latency (see

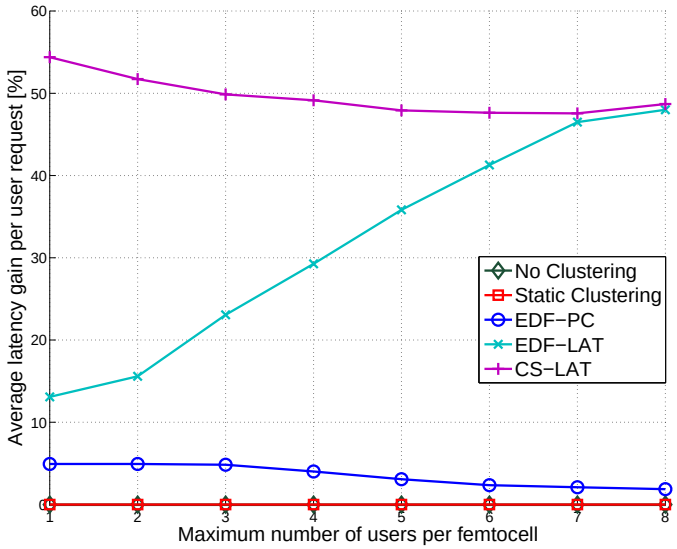


Figure 5.6: Average latency gain in dependence on number of users per small cell

Section 2.4.5). Figure 5.7 shows the average power consumption per user request for the compared algorithms. For the case of *no clustering*, no data is sent between small cells and therefore there is no communication power cost. An interesting result is the very low communication power consumption for the EDF-PC algorithm even for high number of users per femtocell. This shows the convenience in the choice of both *step 1* and *step 2* metrics and rules. This algorithm can in fact achieve high energy efficiency while keeping a very high quality of service. It is a very good solution to implement whenever latency minimization is not an issue. In fact, when traffic is elastic, optimization could focus on respecting the elasticity limits instead of minimizing latency. Even though EDF-LAT and CS-LAT implement both latency minimizing clusters, which imply high power consumption, it is clearly seen that CS-LAT is less power consuming. In fact, since CS-LAT schedules requests in *step 1* based on their computation size instead of adopting and EDF rule, it can serve more users' request locally communication cost free. This comes at the cost of lower users satisfaction ratio as can be seen in Figure 5.4 since users with high requirements of computational capacity (computation size and latency ratio) can be found dropped using such strategy. We also notice a difference in the average power consumption between the *one shot optimization* solution and EDF-PC. This difference is potentially due to several factors: (i) the difference in satisfaction ratio shows that the *one shot optimization* serves more users, and thus exploits the QoE-power consumption trade-off to its limits by increasing the consumed power for achieving higher QoE. (ii) EDF-PC schedules computational tasks on the SSC based on latency constraints. Requests with the earliest deadlines are computed locally on the SSC. Computed on the SSC, there is no intra-cluster wireless exchange of data and thus no transmission power consumption. The minimization of offloaded tasks increases the availability of computing resources at nearby small cells, i.e. at a small cost. These resources can be exploited by users that are less urgent, and thus can further reduce their power consumption by opting a lower transmit power. (iii) as computational resources are allocated locally and on the edge cloud following and EDF rule, then freed resources on both sides can be re-used for the computing tasks with higher delay tolerance. It is especially the re-use of local computational capacities at the SSC and then at nearby

HSCs with high throughput connection links that achieves this gain of power consumption.

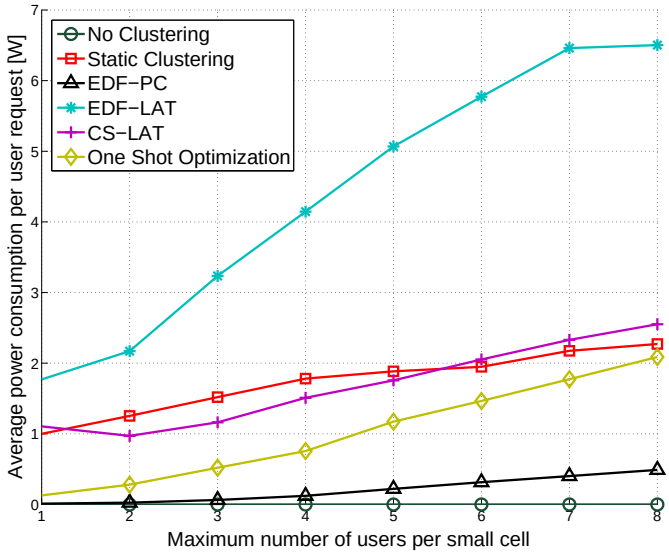


Figure 5.7: Average power consumption per user in dependence on number of users per small cell

An important aspect to evaluate in the iterative algorithm is the number of iterations that are necessary to compute all clusters solutions. Each iteration imposes signaling between SSCs and the SCM. However, the signaling amount at each iteration decreases comparing to previous iterations, since SSC that have their clusters established do not need to report back to the SCM. The cumulated signaling overhead increases with the number of iterations, and therefore the whole system resources and power consumption. Note that the signaling data to be exchanged between SSC and SCM can be detailed as follows:

(i) SSCs send their computed clusters parameters that consists of the participant small cells, and, for each cell, the computational capacity allocated, and the number of CPU cycles to execute. In case of clustering failure, a negative acknowledgment (NACK) is sent.

(ii) SCM response to SSC is (i) in case no modifications on the cluster are required, an acknowledgment (ACK); (ii) in case of necessary cluster adjustment, small cells that have been overloaded, and the number of CPU cycles (instructions) that should be re-allocated to another small cell; (iii) in case of a dropped request, a NACK.

Since the SSC population that participate in the cluster set up algorithm decreases at every iteration, the signaling delay varies linearly with the number of iterations — in the worst case scenario.

A study on the number of iterations required for establishing a clustering solution is represented in Figure 5.8.

Figure 5.8 shows the average, 25th, and 75th percentiles of number of iterations needed with respect to the number of users per small cell. On each box, the central line is the median. The edges of the boxes are the 25th and 75th percentiles. The most extreme data points are delimited by the whiskers. Red cross marks represent outliers. Blue, red and green curves represent respectively the maximum, median, and minimum number of iterations for different number of users per small cell. It is shown that the median number of iterations does not exceed 4 iterations for less than 4 mobile users per small cell. For a higher number of users per small cell, the number of iterations

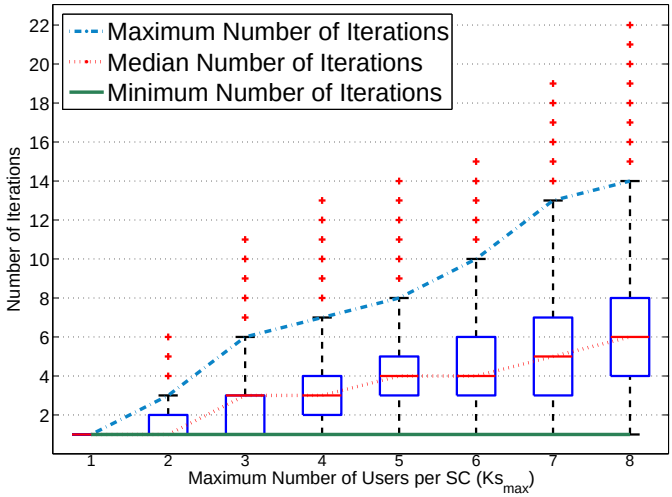


Figure 5.8: Average Number of Iterations Needed for Cluster Establishment

is mostly limited between 4 and 8 iterations. The variance of the number of iterations increases with the number of users per SC. Figure 5.9 shows the cumulative distribution function of the

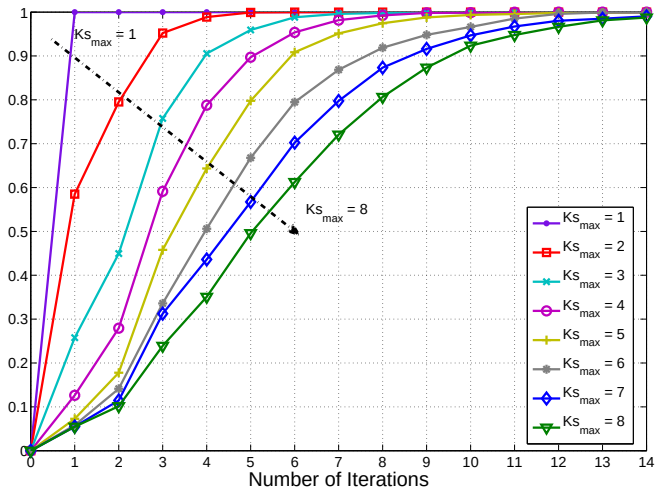


Figure 5.9: CDF of the Number of Iterations Needed for Cluster Establishment

necessary number of iterations for a varying maximum number of users per small cell. It shows again that a solution is faster derived for a small number of users. It is important to notice that even for high number of users per small cell, a large population reaches a clustering solution with a small number of iterations. For example, for the most extreme case of 8 users per small cell, 50% of users have a clustering solution in 5 iterations and less. At each iteration, a set of users is served. This means that at each iteration less optimization problems need to be computed than the previous one. Therefore, the complexity, which is already low because of distributed cluster optimization, further decreases for each iteration.

5.6 Conclusion

In this chapter, we proposed two heuristic algorithms as a solution for the small cell clustering problem. The benefit is low complexity and the low implementation cost it can offer. As most of the joint computation and communication resource allocation for mobile cloud computing are formulated as NP-hard problems, they are usually solved either by an approximation of the real problem, by loosening a constraint, or by proposing heuristic algorithms. In a previous chapter we opted for the optimization solution after loosening the scheduling and resources re-use constraint. In this chapter, we took resources re-use into consideration, as well as tasks scheduling. We proposed two heuristic with several implementation.

A first approach based on resources scheduling at the SSC and resource allocation of the SSC as a first step. In a second step, unserved request are sent to a small cell managing unit which schedules the tasks and then sets up computing clusters for each of them. Three implementations that differ in scheduling and clustering metrics and policies are then derived from the main algorithm. We compared the EDF-PC, EDF-LAT, and CS-LAT implementations in terms of achieved QoE, power consumption, and latency gain.

In the second part of this chapter, we proposed an iterative clustering and resource allocation algorithm for distributed mobile cloud computing environment. The algorithm consists on distributed roles between SSCs and the SCM. SSCs form their own clusters that are then managed and controlled by the SCM. In other words, the algorithm proposes to:

(i) Set up the clusters of small cells serving computation offloading requests from multiple users, having potentially each request accommodated to an ad hoc established cloud. This exploits combined centralized and distributed approach.

(ii) Perform Distributed *Selfish* Guess: The distributed part of the cluster set up algorithm considers all users' requests as *selfish* players that propose to a centralized unit their preferred allocation (and associated serving cluster) based on *selfish* interests (own minimization of delay, energy, etc). Note that while single *selfish* user clustering set up procedure is optimal, scalable, practical, and is of a low complexity, for the multi-user case it can drive to notable losses of system performance.

(iii) Centralized Feasibility Check: The centralized unit supervises which requests can be granted as requested and which must be modified. This is needed to check if the set of *selfish* requests can be accommodated by the system.

(iv) Centralized Correction Evaluation: The central unit evaluates which serving small cell presents excess of requests based on union of *selfish* requests. The central unit classifies requests as 'granted as demanded' and 'to be corrected'. All 'granted as demanded' are allocated, available computing capacities of small cells updated. The centralized units informs SSCs on remaining request to be re-computed and relative correction values.

(v) SSCs of unserved requests, set up new cluster for computing these requests and sends the cluster guess details to SCM (iteration).

A performance evaluation of the iterative algorithm has been presented. An analysis of the number of iterations required to converge towards a clustering solution for all users has also been discussed. The algorithm achieves guarantees a QoE for at least 75% of mobile users, in the case of 8 users per small cell.

We note that the presented work can be applied to current third generation cellular wireless mobile networks and its future evolutions (5G, 4G, 3GPP LTE, 3GPP LTE A, WiFi, LiFi, WiGiG, WiMAX, etc.) with the modification of the adopted transmission and deployment models. This is because the proposed methods do not request changes in the current standards to be implemented.

Chapter 6

Computation Caching in Cluster-based Cloud Computing

6.1 Introduction

6.1.1 Motivation

In previous chapters we studied and proposed resource allocation solution for small cell clusters-based Mobile Edge Cloud. The majority of the proposed solutions aim at minimizing the SCC power consumption, while maintaining QoE, and delivering requested services to users, without violating any of the imposed latency constraints. To this end, communication and computation are jointly allocated, and load distribution is adequately balanced. In addition to resource allocation and scheduling, resource provisioning can bring major benefits into the MEC computing context. In Chapters 4 and 5 we propose *reactive* solutions for resources management. In this kind of solutions, communication and computation resources are allocated according to the current requested load — a *demand and compute* operation mode. In this chapter, we introduce the concept of caching to the computation cluster. We propose to allow storing computations in cache memories of small cells. Conversely to what caching has traditionally referred to, we propose caching tasks computation results instead of caching communication data files. Hence, we exploit caching for computing storage and not data storage. If the cached task is requested again, it is then retrieved from cache and not computed. This approach shifts the MEC to a *search and download* operation mode.
We propose the computation caching in small cell cloud according to the following motivations:

- In the context of edge computing, small cells are endowed with computational and storage capacities. These resources can be exploited for both users and system usage. Computational resources are exploited for executing computations requested by mobile users. The idea is to also exploit available storage space for storing mobile users related computational data.

- Mobile data traffic has been recognized as predictable, correlated, and can be identified to patterns [121] [122]. Furthermore, several tools for data prediction and traffic statistical pattern learning exist. Partial traffic information knowledge can help improve caching efficiency by choosing the right files to cache. Caching prevents the system from being overloaded and saturated in computational capacity. Redundant computation of the same tasks is reduced, and more computational capacity is available.

- Caching computations at small cells reduces the computation costs in terms of power consumption, or energy efficiency. Small cells power consumption while computing is higher than in idle mode. If fewer computations are executed, then less computational power is consumed.

- Delay is a very critical aspect of mobile users QoE. Caching computation results will prevent their computation when they are requested. This can reduce experienced latency, since data is only sent from the SC where it is cached to the SSC, then to the user.

- Caching some computation results prevents their repetitive computation. It also prevents sending necessary computational data to the computing entity. Data sent from SSC to HSCs is, generally, larger than computational results sent back from HSCs to SSC. Thus, caching not only minimizes computation power consumption, but can reduce communication energy consumption and EMF radiation and exposure.

- With the increase of uplink traffic and the consequences it has on wireless networks (see Section 1.4), computation caching can help reducing uplink traffic. Computation caching

limits uplink requests by reducing uplink sending of data for computational offloading. In case of computation caching, requests can consist in sending only a label for example.

In this chapter, we address the challenge of further enhancing the performance of cluster-enabled MEC, by reducing perceived latency and power consumption. To this end, we propose to exploit cache memories at small cells for caching popular computation offloading requests.

6.1.2 Related Work

Distributed cache systems

Cache servers were initially introduced to reduce the processing load of web servers. Distributed caching is presented as a solution for the problem of overloading centralized cache servers and decreasing their performance. Normally, every cache server serves users connected to it. To further improve the system performance, cache servers can exchange files in order to achiever higher hit ratio over the whole system. Two important characteristics of distributed caching are (i) caching of the same files at different cache servers; (ii) adopting a light request search process. For the Internet Cache Protocol (ICP) [123] all cache servers are searched for finding the requests, which is not suitable for large scale networks. Kataoka *et al.* propose a centralized cache server controller, that can provide contents cached in any of the caching servers [124]. The server controller keeps a list of cached *url*s and their cached server ID. Cache control server is responsible for sending an instruction to the cache server that has a cached copy of the requested file. This instruction consists of a request to send the file to the user who requested it. An extension of splitting the load, i.e., the content list, on several (three) cache control servers is discussed. In this case, a cache cooperation router is needed to forward requests to the right cache control server.
Zhang *et al.* propose a distributed cache systems for real-time cloud services [125] . Cache services are organized in a Peer-to-Peer (P2P) style, and use a Distributed Hash Tables (DHT). The distributed storage process relies on creating three replicas of the requested data files in the cache system. When a request is received, and is not entirely found at the distributed cache system, the hash value of the file is calculated, and two nodes from DHT are chosen to store replicas of the file. File distributed system in wired network do not tackle the problem of connection between cache servers, nor do they discuss the costs in terms of delay and energy consumption.

Data caching in cellular networks

In this section, we discuss some related work to data caching in mobile network. Data caching is linked to data storage at cache memories of network entities, such as base stations. Anandharaj and Anitha propose a cache management for mobile hosts in the context of cloud computing [126]. Base stations are assumed to be connected to database servers. Files are stored at the mobile clients according to a weight metric that is based on the available bandwidth, the CPU speed, access latency and cache hit ratio. A cache discovery algorithm is presented, in which, if a requested file is not found in local cache, a broadcast request is sent to active clients. The client with the most updated version of the file is selected to transmit the file to the requesting client. A replacement algorithm is as well developed. It replaces files with lowest *Relevant Value*, which is a metric based on the access probability, the number of hops between requesting client and responding client, and the file size. This caching process proved to outperform the Least Recently Used (LRU) algorithm [127] by achieving higher download traffic and lower delays. Bastug *et al.* propose a caching system for cellular networks where users' files are stored at small cells base stations [128]. The authors propose an algorithm for choosing the files to be cached, based on the

files popularities (popularity matrix). The algorithm stores the most popular files, i.e. files with highest popularities, at each base station until cache memory is full. If files are not found they are downloaded using small cells backhaul. Performance of this algorithm is compared to random caching, and it proved to achieve better performance and lower backhaul load.

In [129], Bastug *et al.* follow the same concept, with modifications in the popularity matrix generation distribution . In the first proposal ([128]), the popularity matrix is randomly generated and assumed fully known. Whereas in the second ([129]), the authors introduce a training and placement strategy for the matrix, to be more realistic. Furthermore, they introduce a social-aware caching through the modeling of social content dissemination between users. Request probability of content is affected by the number of previous content requests within the same community. The higher the number of the same content request in a same community, the more D2D file sharing is leveraged, instead of downloading files from the network. Gu *et al.* proposed in [130] an MDP decision process for cache replacement in base stations cache memory. The strategy is based on Q-learning and replaces cached data taking into consideration both service popularities and the transmission cost between base stations.

The presented work tackles the paradigm of data caching in cellular networks. Our proposal is to exploit cache memories in edge network entities in order to store also computation data.

Computation caching

Computation caching means storing computation related information, which can be used when needed by a computation. The main issues of computation caching are what to cache, how to link request to cached information. Linking the computational request to the search action can be done either using DHTs as in the data caching case, or using other techniques similar to Information Centric Networking (ICN), for example. Computation caching has been proposed in the context of Named Function Networking (NFN) in [131]. In this proposal, the network is able to cache not only data but also computation results. The proposal is based on the network recognition of the function by a naming definition. ICN offers names for functions, enabling users to say what result they need by writing expressions that refer to data and function names. The network substrate would then be in charge of finding out how these results can be obtained, either by computing them, or by looking them up in case it was already computed by others. Three scenarios are described. First, the cached result is entirely available and thus downloaded. Second, data is partly processed, downloaded, and then remaining code is downloaded and executed. Third, program is not cached, then it is fully computed and the result is downloaded.

Another paradigm of computation caching is to decide what to cache. Of the existing solutions, Waterland *et al.* propose to cache the act of computation, so that it is applicable later in the same or different contexts [132]. Caches are used to avoid re-computation by storing the results of computations. The proposal is based on caching the *act* of computation instead of the results because computing can be dynamic. The function is cached along with actual executions results and predicted possible executions as well. Furthermore, they consider an approach that uses semantic information about the relationship between the function and the desired computation value. This way only the value can be matched and not the function. If the function is found in cache, the computation is speed up and forwarded to the final state (results). The idea is exploitable in a network of computers or servers, that collaborate by sharing model parameters and cache entries without middleware problems. If the act is stored, it can be applied repeatedly in new contexts. Another option would be to store computational results, or entire (or parts) of the computational codes, as proposed in [131]. In our work, we adopt a solution that caches computational results,

but that treats the problem from both perspectives of computational caching at the level of computed instructions, and how to retrieve the information. None of the discussed work proposes to form information retrieval ad hoc clusters, even in distributed caching systems. We exploit the distributed caching in local edge cloud in order to change the set up of the cluster to take into account computational caching.

6.2 Contribution

In this chapter we propose a novel paradigm to further reduce both power consumption, and time delay, of mobile computation offloading to the edge cloud. As we suppose that small cells are equipped with computational and storage capabilities, we propose to use their storage space for computation caching. Computation caching consists in caching popular computations, in order to prevent re-executing the same instruction blocks. More concretely, if the same computation is asked for several times, the small cells do not need to compute it at every request. If the computation is cached, the results are retrieved from the edge cloud. The benefits of this paradigm are two-fold. First, it allows users to have faster response to computational requests service. Second, it prevents small cells from consuming energy, time, and computational capacity for computing the same tasks repeatedly. The two main contributions of this chapter are the proposal of a computation caching algorithm in the context of edge cloud computing with small cells clusters, and a cluster search reduction (sparsification) method including computational caching empowerment. In the first contribution, we propose a caching algorithm for the small cell edge cloud paradigm. The algorithm is based on computing a caching metric that is a function, not only of the computation popularities, but also the computation size, and latency constraints. The proposed algorithm is described and compared to state-of-the-art algorithms.

When a SSC searches for a cached computation, it may search on its own cache memory, but also on HSCs caches. However, cache searching algorithms impose time complexity. In addition to our first proposal of clustering small cell for serving computational requests, we propose, in this chapter, to form clusters for cache searching. We shift the cluster set up paradigm from a joint communication and communication aware clustering, and add a caching aware clustering. Searching a cache memory where it is not likely to find the desired files increases the costs of the cache search operation. Hence, our second contribution consists in reducing the set of small cells caches that are searched for finding the requested computations. We refer to the set of searched small cells as the search cluster. We therefore propose to identify the set of eligible small cell candidates, for which the probability of finding the requested files meets a specified target. Obviously, reducing the search cluster by focusing on a selection of small cells reduces the cache search process costs.

6.3 System Model and Notations

We consider a scenario of K mobile users, each connected to a SSC from the N active small cells in the network. Each small cell n is characterized by a computational capacity of F_n [CPU cycles/sec]. The set of small cells communicate through a backhaul connection. Parameter ρ indicates the connectivity ratio between small cells. A connectivity matrix $\mathbf{X}$ shows available

connections between small cells as follows:

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & \dots & x_{1,N} \\ x_{2,1} & \dots & x_{2,N} \\ \vdots & \vdots & \vdots \\ x_{N,1} & \dots & x_{N,N} \end{bmatrix} \in \{0,1\}^{N \times N} \tag{6.1}$$

where $x_{n,n'} = 1$ if small cells n and n' can communicate through a direct backhaul link.

In our model, we adopt the same concept of probability matrix proposed by Bastug *et al.* [128]. We therefore consider that a matrix **P** gives the probability of a computation c to be offloaded to SC n. A set of a maximum of C popular computations is considered. The computation probability matrix is then defined as follows:

$$\mathbf{P} = \begin{bmatrix} p_{1,1} & \dots & p_{1,C} \\ p_{2,1} & \dots & p_{2,C} \\ \vdots & \vdots & \vdots \\ p_{N,1} & \dots & p_{N,C} \end{bmatrix} \in [0,1]^{N \times C} \tag{6.2}$$

where $p_{n,c}$ is the probability of computation c to be requested at small cell n. Note that the sum of each row of **P** is equal to 1. Matrix **P** serves as an input for a caching algorithm that decides which computation to cache at which small cells. The algorithm returns is a caching matrix **a** such that:

$$\mathbf{D} = \begin{bmatrix} D_{1,1} & \dots & D_{1,C} \\ D_{2,1} & \dots & D_{2,C} \\ \vdots & \vdots & \vdots \\ D_{N,1} & \dots & D_{N,C} \end{bmatrix} \in \{0,1\}^{N \times C} \tag{6.3}$$

where $D_{n,c} = 1$ if computation c is cached at small cell n, and $D_{n,c} = 0$ if not. Each of the C computations is characterized by an offloading request (W_c, Δ_c), which consists of the computation of W_c instructions in a maximum time of Δ_c seconds. We assume that the number of bits N_{UL} and N_{DL} to be transmitted to each small cell in uplink and downlink, respectively, is related to the number of CPU cycles W as follows:

$$\begin{aligned} N_{UL} &= W.\theta_{UL} \\ N_{DL} &= W.\theta_{DL} \end{aligned} \tag{6.4}$$

where θ_{UL} and θ_{DL} are constants that account, respectively, for the overhead due to the uplink and downlink communications. The formulation in Equation 6.4 is valid for both UE to SSC and SSC to HSC communications.

The intra-cluster communication assumptions and models are the same as described in Section 4.2.

6.4 Computation Caching for Edge Computing

We adopt the concept of MEC described in Section 2.1. It consists on deploying cloud services at the edge of the network, in our case, in small cells. Mobile handsets offload computation tasks by communicating only with their serving small cell (SSC). SSCs take the role of computing the

requested tasks without violating the latency constraints imposed by each task. SSCs form computing clusters with a set of its neighbor HSCs in order to compute received tasks. Computational load is distributed among the cluster nodes. The process of small cells clustering is transparent to mobile users; however the perceived cluster computing delay affects the users' QoE. Clustering steps, that affect the perceived cluster computing latency and energy consumption, can be resumed as follows: (i) uplink computation request transmission from MUE to SSC (ii) computational load distribution on the cluster SSC and HSCs (iii) computation execution at each small cell of the cluster (iv) computational results transmission back to the SSC (v) downlink computation response transmission from SSC to MUE. All of these components depend directly on the quantity of data bits to be transmitted or processed. Even with high data transmission energy efficiency in Joules/bit, the consumed energy will always scale with the number of bits. The same applies on computational power consumption which is linked to each processor EPC (Energy Per CPU cycle). Thus, with higher number of instructions to compute, energy consumption increases.

In this chapter, we propose a new paradigm that allows further reducing the computation offloading process costs. We merge computation with caching in a new paradigm that allows small cells to store popular computations. If a computational request is already cached, the SSC will only have to retrieve it from the cache location. This will reduce the quantity of transmitted data since only computation results are to be exchanged. Computing cluster set-up is thus not required, and consequently, no computations are to be executed. Computation caching will also allow sparing computational resources, and therefore using these capabilities to satisfy more users. An observation that further supports such a proposal, is the fact that human behavior is highly predictable and correlated. Human behavior prediction may not be able (yet) to predict *exact* actions at exact locations and moments, but statistical patterns may be observed [128]. An example of statistical patterns is the requests popularity distribution. Popularity is a matrix that associates each file with a popularity value that translates the probability of it being requested. We also consider in this work the presence of a known computation popularity matrix that we use for caching popular requests. Whenever a computation request is sent to a SSC, it searches for a cached copy. If found, the computation details are fetched and downloaded. Otherwise, a computing cluster is set up by the SSC and the computations are run inside the cluster.

The general scenario is the following:

- **Step 1 - Offloading request:**
  Serving small cell (SSC) receives an offloading request (W_k, Δ_k) from mobile user k.

- **Step 2 - Local search:**
  SSC searches for the computation in its own local cache memory. If found, results are sent to UE.

- **Step 3 - Search & Download:**
  If computation is not found on local cache, SSC sends a search request for reachable helper small cells in the network. If found, the file is downloaded from the *best* HSC. After that, results are forwarded to the UE.

- **Step 4 - Cluster computation:**
  In case a cached version of the computation is not found, SSC forms a computation cluster and distributes the load on participating small cells.

The total costs components of the cluster computing process can be written as follows:

$$C_{tot} = C_{TX}^{UL} + C_{clustering,Tx} + C_{comp} + C_{Tx}^{DL} \tag{6.5}$$

where C_{tot} is the total cost of the process,C_{TX}^{UL} and C_{Tx}^{DL} are the communication cost between UE and SSC for uplink and downlink respectively, $C_{clustering,Tx}$ is the cost of intra-cluster communication for sending and receiving computational load and results, and finally C_{comp} is the computing cost at the SSC and HSCs. We note that $C_{clustering,Tx}$ and C_{comp} are the sum of costs relative to all small cells participating in the computation cluster:

$$\begin{aligned} C_{clustering,Tx} &= \sum_{n \in cluster; n \neq SSC} (C_{SSC,n}^{UL} + C_{SSC,n}^{DL}) \\ C_{comp} &= \sum_{n \in cluster} C_{comp}^{n} \end{aligned} \tag{6.6}$$

Costs in equations above can represent both energy and latency costs. For transmission costs, both, energy and time, scale with the number of bits to be sent. We note that UL, DL, and cluster communication depend on both transmission and overhead costs, that both increase with data load. The same applies for computing cost, where the computation time is computed according to the processor speed, and the computing energy consumption according to the processor energy consumption per cycle. Indeed, computation time at each small cell n is computed as $\Delta_{comp}^{n} = \frac{W_n}{f_n}$ and energy cost as $\mathcal{E}_{comp}^{n} = EPC(n).W_n$

The costs details allow us to assume that computation caching can severely reduce offloading costs, and that by greatly reducing the size of that data to be transmitted and processed.

6.5 Proposed Caching Algorithm

The scenario described in the previous section assumes the knowledge of matrix **D** (see Section 6.3). In this section, we propose a novel caching algorithm that we name *ClusterCaching*, which takes into account several parameters of the requested computations. In [128], the authors propose a caching algorithm for data under the name of *PropCaching* for caching files at femto base stations, with the objective of maximizing the satisfaction ratio defined by finding the files at the base station cache. The algorithm is based on caching the most popular files at each base stations.

We propose a caching policy metric and a caching algorithm adopted for the context of computational caching. We re-engineer the same approach of data files popularity for exploiting cache clustering for computational data. We propose a caching algorithm in which the common point with the state of the art [128] is the exploitation of the concept of popularity matrix. Nevertheless, our work is based on computational instructions results caching and not user data files. Furthermore, we design a multi-parameter caching metric λ that does not only takes into account the tasks popularity as in state of the art. We introduce additional parameters in the caching metric, which will adopt the caching policy to dynamic ad-hoc clusters set up for computation fetching. The aim of the proposed caching algorithm is to adapt the caching policy to achieving higher energy gain (lower mean energy consumption per computational request) when instructions results are fetched in cache memory using ad-hoc clusters that include caching resources. In our work, and in the context of mobile computation offloading, we define the users satisfaction ratio as the percentage of mobile users that have a response of their offloaded requests, without any violation of the latency constraint Δ_k. We propose a novel caching algorithm whose objective is to increase users satisfaction while reducing the cost of the local edge cloud computing process. The proposed algorithm takes into account, not only computation popularities, but also other characteristics, such as, data

size, latency constraint, and the mean of data rates with which cached copies can be retrieved. In fact, our algorithm does not target a high hit ratio in cache memories. Instead, we find the right computations to cache at the right place (small cell) that will allow us to reduce the edge computing and offloading costs. Our policy will provision computing and communication resources by caching the most resources demanding computations. Following this policy, more resources will be free to be used by small cells for executing other tasks and computations requested by mobile users.

We define the parameter λ as:

$$\lambda = \frac{p_{n,c} W_c |\mathcal{N}(n)|}{L_c \sum\limits_{m \in \mathcal{N}(n)} R_{n,m}} \tag{6.7}$$

where $\mathcal{N}(n)$ is the set of neighbor small cells connected to n, defined as the set of SCs $\{m | C(n,m) = 1\}$, and $|\mathcal{N}(n)|$ its cardinal number. As noticed in its definition, λ increases with the computation popularity $p_{n,c}$ and with the required computational capacity, defined by $\frac{W_c}{L_c}$. Caching computations according to λ gives priority to popular computations requiring high computational capacities. By giving priority to higher capacity requiring computations, higher amounts of computational capacities can be provisioned. Furthermore, λ is inversely proportional to the intra-cluster communication mean rate with which the cell is connected to its neighbors ($\sum\limits_{m \in \mathcal{N}(n)} R_{n,m}/|\mathcal{N}(n)|$).

The caching matrix **D** is then obtained by applying Algorithm 1. This algorithm computes parameter λ for each pair (small cell, computation request), and then stores computations with the highest λ, until small cells cache memory, of storage capacity denoted as Z_n, are full. We denote as Y_c the amount of data to be stored on the cache memory for a computation request c.

Algorithm 1 ClusterCaching algorithm

for $n = 1, \cdots, N$ **do**
    compute $\boldsymbol{\lambda}_n$ according to (6.7)
end for
$\mathbf{a}_{N \times C} \leftarrow \mathbf{0}_{N \times C}$
$\mathbf{z}_{N \times C} \leftarrow \mathbf{0}_{N \times C}$
for $n = 1, \cdots, N$ **do**
    $[\hat{\boldsymbol{\lambda}}, \mathbf{s}] \leftarrow \text{sort}(\boldsymbol{\lambda}_n)$
    for $c = 1, \cdots, C$ **do**
        $\hat{c} \leftarrow s_c$
        if $\hat{z}_n + Y_{\hat{c}} \leq Z_n$ **then**
            $\hat{z}_n \leftarrow \hat{z}_n + Y_{\hat{c}}$
            $a_{n,c} \leftarrow 1$
        else
            break
        end if
    end for
end for

6.6 Caching Algorithm Evaluation

In this section, we first evaluate, and then benchmark, the computation caching paradigm along with the proposed *ClusterCaching* algorithm. We consider a scenario where in a time window of length T, a set of Q users computation requests are randomly generated. Users communicate with their SSC through a wireless link of a delivery rate fixed to $R_{u_{k,n}}$. The generated requests are chosen from a set of selected C popular requests. The Q requests are also randomly distributed among the N active small cells in the network. The number of active small cells is determined by the small cell activation ratio α. Each of the small cells has a computational capacity F_n randomly generated, and a computing power consumption of $P_{comp} = 10\,\text{W}$. Small cells communicate through wireless links of a capacity of $Rf_{n,n'}$. The simulation parameters values are listed in Table 6.1.

Table 6.1: Simulation parameters values

Parameter	Value	Parameter	Value
N	25	α	0.8
ρ	1	T	1024
Q	$60 \times T$	R_u	5 Mb/time slot
R_f	5 Mb/time slot	C	128
W_c	$[2,10]$ Mb	Δ_c	$[0.3,3.5]$ s
Y_c	W_c	F_n	$[10,15]$MIPS
θ_{UL}	1	θ_{DL}	1

As for the storage of small cells, we consider different values of cache memory capacities. We define S_C as the total storage space available at each SC. $\sum_{c=1}^{C} Y_c$ is the required storage space at one small cell to store all C requests. We define μ as the storage ratio at each small cell defined by:

$$\mu = \frac{S_c}{\sum_{c=1}^{C} Y_c} \tag{6.8}$$

However, considering that small cells have access to each others cache through clustering, lower memory space than S_c may be needed for each SC to have access to cached copy of each computation. This comes at the cost of backhauling communication. Consequently, the amount of required storage space depends on the connectivity between small cells. Indeed, when small cells are inter-connected through high capacity and ubiquitous availability backhaul, each node can store or cache a smaller set of tasks, knowing that non-cached tasks can be easily retrieved from neighbor small cells through intra-cluster communication. Nevertheless, when cluster connectivity is lows, small cells have lower chances of retrieving cached computation from a neighbor small cell due to the low number of available and reachable neighbor small cells. A cluster with high connectivity ratio lead to a better aggregation and exploitation of neighbor small cell cache memory, and thus, lower storage space is required for accessing all cached tasks.

Figure 6.1 shows the ratio of cached and accessible files in function of both the small cells connectivity ratio and the cache memory size. We notice indeed that we can achieve the same ratio of cached and accessible requests, in a well-connected network, by using lower cache memory space. In Figure 6.2, we compare both *PropCaching* and *ClusterCaching* algorithms in terms of cached requests. We show the probability of a request being found in the SSC cache in function of

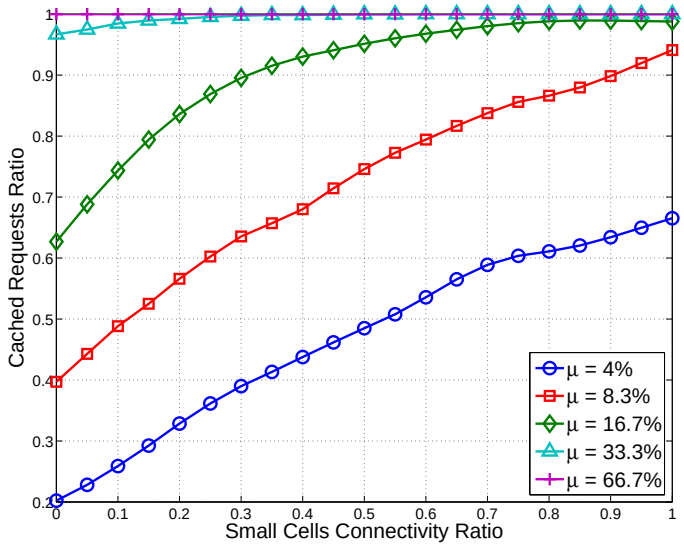


Figure 6.1: Cached requests ratio for different connectivity levels and memory space

its popularity at the same SSC. This plot shows how *PropCaching* clearly gives caching priority to most popular requests and *ClusterCaching* does not. For example, as indicated by the double arrow shown on the graph, a file with a popularity of $p = 0.05$ will be found on a cache memory with a probability of 80% in the case of *PropCaching*, and only 26% for *ClusterCaching*. The

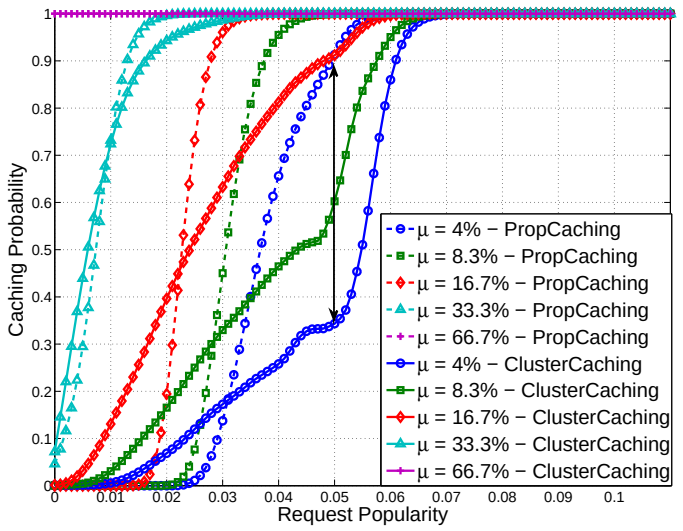


Figure 6.2: Cached requests probability in function of popularity

objective of *ClusterCaching* is to minimize the computation cluster energy in the context of local cloud computing. Choosing the right requests to cache is imperative in order to achieve higher

energy efficiency. To this end, we show on Figure 6.3 the energy consumption per request with

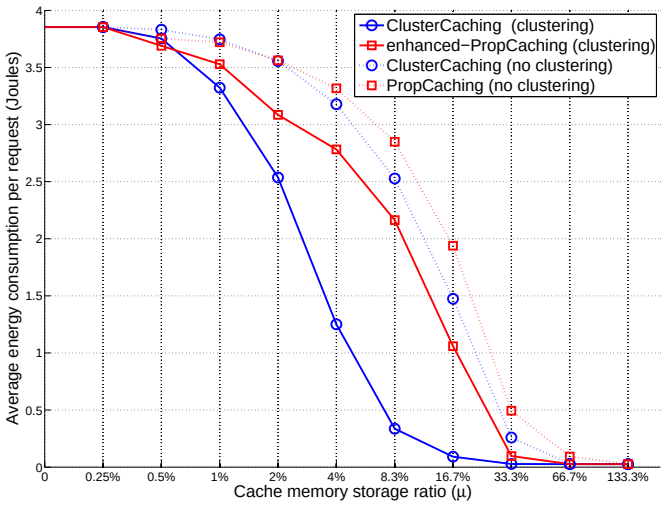


Figure 6.3: Energy consumption of different caching algorithms

respect to cache memory storage space. We evaluate the energy consumption in cases of no cache clustering, i.e. SSC only searches for the computation results locally on its own cache, and in case of unit size cluster, i.e. if not found on local cache, SSC downloads the computation results from a single small cell cache source. Solid and dotted lines on Figure 6.3 represent cases with and without cache download clustering, respectively. We compare the energy consumption of our proposed algorithm to the state of the art *PropCaching* algorithm. We also proposed an *enhanced-PropCaching* where we include clustering possibility for the *PropCaching* algorithm. Square marked red lines represent state of the art and reference algorithm; while circle marked blue lines represent the proposed algorithm. Two important observations from 6.3 are important. First, comparing solid and dotted lines, we see the energy consumption gain brought by small cells cache clustering. Indeed, having the possibility to download computational results, even if not from the SSC, leads to savings in energy consumption when the download cost is lower than tasks computing costs. As shown on Figure 6.3, the energy for computing the set of requests costs 5 times more the energy comparing to the case of computation caching, in the considered scenario and parameters. Second, it is shown that *ClusterCaching* achieves higher energy savings than *PropCaching* algorithm comparing to the case where there is no caching. Note that the no caching case is represented by a zero cache memory size on all small cells. We notice that *ClusterCaching* is more adapted to the cache clustering scenario, since it exploits, in addition to computations popularity, the cost of computation and download for choosing *which* tasks to cache. By caching results of computations that imposes high execution or download costs, even if less *popular*, the proposed algorithm reduces the *search and download* cost. Figure 6.4 shows the percentage of energy savings comparing to the state of the art (*PropCaching* with no clustering). On this figure, we distinguish three regions with different behavior.

1. Very low cache storage space: In this case, where cache space is very low comparing to the considered computations population (0.25-0.5%), we notice that both algorithms have almost the same behavior, with a slightly better performance for *enhanced-PropCaching*. This can be explained by the fact that *PropCaching* stores the most popular tasks, while *Cluster-*

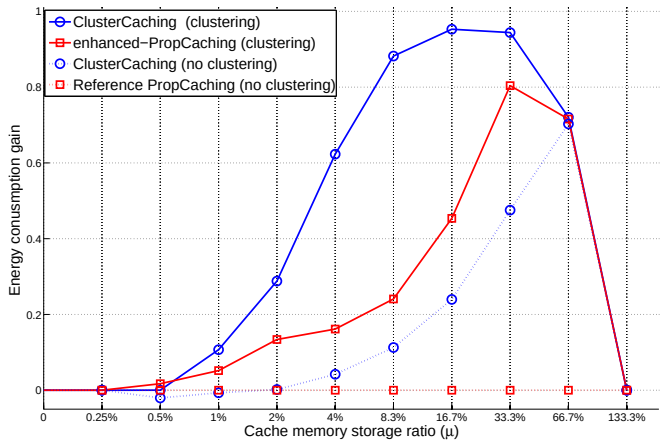


Figure 6.4: Energy consumption gain of different caching algorithms

Caching caches the computations based on a multi-parameter metric that depends, among others, on the computation size. Therefore, with very low cache memories, *clusterCaching* is able to store less computational tasks, that can be less popular, than *PropCaching*. Hence, *PropCaching* achieves a higher cache hit ratio and lower average energy consumption per request.

2. Moderate cache storage space: It is in this case that we notice higher gains for both the cache clustering and the proposed *ClusterCaching* algorithm. When cache memory space is not very low, and not large enough to contain all the computations considered population, *ClusterCaching* outperforms the reference algorithm for up to a gain of 70%. Cluster caching brings for *PropCaching* algorithms a gain that goes up to 80%. Finally, combining the effects of both *ClusterCaching* and cache clustering, a gain of 98% can be achieved. This clearly shows that the proposed algorithm is adapted to the cache clustering mechanism. Even if *PropCaching* can achieve higher cache hit ratio, *ClusterCaching* achieves higher energy gain with lower cache hit ratio, because it stores popular computations that are the most energy consuming.

3. Cache storage excess: In this case, cache storage space on each small cell is high enough to contain a cached copy of all the considered computation population. In this case, computational results are always found at the SSC cache. The caching algorithm has no impact on energy consumption in this case.

It is important to note that energy consumption in the considered scenarios depend on several parameter, most importantly the communication channels condition and the computing processors power consumption that can vary by tens of watts. The results shown in this chapter are for a specific set of parameters. While energy consumption values can vary and the energy gain percentage depend on the correlation between popularity and tasks size, the analysis of these results remains general, and the relationships between cache memory space, computation size, and popularities hold up even in the case of different system parameters.

6.7 Search Cluster Sparsification

6.7.1 Motivation

Computation caching is a way of reducing the computation cluster perceived costs in terms of latency and power consumption, and of increasing resources utilization efficiency.

A first problem identified in the concept of computation caching, is the complexity of the cache search process. In medium and large scale networks, the search algorithm can be severely penalizing in terms of time complexity. Small cells processors could be overloaded with search jobs, which can reduce the availability, and the efficiency of computational capacities. SSC has interest in reducing its decision set, and hence, reducing the size of the cache clusters. Nevertheless, the cache cluster size reduction must not eliminate candidates of having the requested cached file. Indeed, the larger the cluster, the more small cells caches it has, and the higher the probability of finding the required files since the search space is bigger. Search cluster sparsification should be designed in order to reduce the number of searched small cells while guaranteeing a minimum cache hit probability. The reduction of the search cluster set should not prevent the SSC from finding and retrieving the required files and cached computations. The excluded small cells from the search cluster should not hold the required files in their cache memories. Distributing the list of cached files on every small cells into the network results in extensive overhead, especially that the cache update mechanism can be dynamic and very frequent. In addition, the main motivation for introducing computation caching is to enable further latency and energy consumption of small cell clusters. These objectives have to be maintained in order to keep the gain imported by computational caching. Occupying small cells resources in cache search jobs may result in low resources utilization efficiency.

There is a trade-off between reducing the search cluster and guaranteeing the required files retrieval in the discussed computation caching scenario. To overcome this existing trade-off, we propose a method that reduces the cluster size, while guaranteeing a target cache hit probability.

6.7.2 Contribution

We already introduced an additional resource to the clustering process which is the cache storage space. The idea we propose now, is to exploit the popularity matrix for interconnecting small cells in cloud connected cache system.

Exploiting the popularity matrix, we can reduce the search or decision space without negatively affecting the search results, i.e. we can perform *safe* search cluster sparsification.

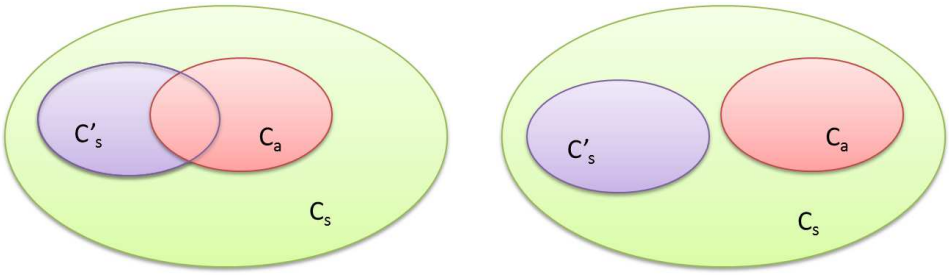
We note as $\mathcal{C}_s$ the decision space that contains all SCs with cache storage space. If each of these small cells ($n \in \mathcal{C}_s$) stored M computations in its cache memory, then the search algorithm will have to go over NM files. The complexity of the search algorithm (linear search) is of $\theta(NM)$. However, the searched file may not be cached in all of the N small cells in $\mathcal{C}_s$. We define $\mathcal{C}'_s$ as the set of N' small cells where the computation is cached ($\mathcal{C}'_s \subseteq \mathcal{C}_s$). The small cells follow a policy to cache computations defined by the caching algorithm. This policy depends on the popularity of each request at each small cell (Examples are in Section 6.5). Establishing a relationship between popularity and caching will help identify possible caching location of the requested computation. If we exploit the knowledge of both the popularity matrix and the caching policy, we can reduce the cache cluster size without removing possible small cells candidates. The decision space will thus be reduced from $\mathcal{C}_s$ to $\mathcal{C}_a \subseteq \mathcal{C}_s$. The sparsified cluster $\mathcal{C}_a$ should include at least one cached copy of the required files. The required and necessary condition for $\mathcal{C}_a$ to be a *safe* sparsification

of the search space is:

$$\mathcal{C}_a \cap \mathcal{C}'_s \neq \emptyset, \text{ where } \emptyset \text{ represents an empty set} \tag{6.9}$$

Figure 6.5 shows a graphical explanation of the decision space size reduction. $\mathcal{C}_a$ is the image of $\mathcal{C}_s$ by popularity matrix aware transformation. If the searches in $\mathcal{C}_s$ and $\mathcal{C}_a$ have the same results, then the decision space reduction includes no loss in the search process. We propose a process that defines this popularity aware transformation that can guarantee a minimum intersection with $\mathcal{C}'_s$. The implementation of this idea is possible thanks to (i) the introduction of computation caching; (ii) the existing relationship between popularity and caching location.

The ultimate case for optimal size reduction is the knowledge of exact locations (small cells) where the request is cached, i.e. the N' small cells in $\mathcal{C}'_s$ can be perfectly identified and $\mathcal{C}_a \cap \mathcal{C}'_s = \mathcal{C}'_s$.



(a) Valid sparsification ($\mathcal{C}_a \cap \mathcal{C}'_s \neq \emptyset$) (b) Non-valid sparsification ($\mathcal{C}_a \cap \mathcal{C}'_s = \emptyset$)

Figure 6.5: Search space reduction use cases

We propose a method for reducing the search cluster set based on the knowledge, at each small cell, of the files popularity and the caching policy. We refer to the list of computations cached on a small cell as the *cache inventory*. If all the network small cells are aware of the cache inventory of all other small cells, the requested computation cached copy can be localized. However, the cache inventory at each small cell changes at a high pace, up to the frequency of computations requests. Indeed, at each computation request, the caching policy may result in replacing older requests with the newly requested. Diffusing the cache memory inventory to small cells in the network results in excessive overhead. Furthermore, even if a small cell can collect the cache inventory of others, it will still have to search through the inventories. We propose to exploit the *in cache* probability, defined in Section 6.5, in order to efficiently reduce the size of the search clusters. We define an *in cache* probability matrix that associates to each popularity, a probability of being cached at the small cell in question. This matrix is diffused to the network small cells. The *in cache* probability matrix, does not require frequent update as the cache inventory. The former depends on the caching policy of the small cells, whereas the latter depends on the computational traffic. *in cache* probability drastic update occurs rarely, for example, when a small cell changes its caching policy. Furthermore, since it is a matrix built from statistical observation, it can converge to a final representative matrix. An additional element, other than the *in cache* probability is eventually required to be shared among small cells: the computational requests popularity. The requests popularity is also a statistical observation and distribution that is built by observation of the requested computation traffic. Its update is not required to be instantaneous at every change of a file's popularity. Thresholds may be put in order to launch popularity information diffusion between small cells. For instance, when $p(c) > p_0(c) + \varepsilon$, where $p(c)$ is the current popularity of

computation request c and $p_0(c)$ is the initial popularity value which is equal to the last diffused request popularity of request c.

In addition, we propose a classification of computational requests that reduces the set of requests to which a search cluster is set up. The idea is to prevent a cluster search set up for requests that are *not worth searching for* because they will, most probably, not be found.

We propose a method to reduce both the number of clusters that are set up, and the size of search clusters from C_s to C_a. The proposed method allows adapting the search cluster size, exploiting the search space vs cache hit probability trade-off.

The novelty of this method is based on patent [P5].

6.7.3 Concepts and Notations

We consider a MEC scenario, where a set of small cells co-exist and can form computational cluster for satisfying mobile users' computational offloading requests. We refer to the set of deployed small cells as the hyper-cluster. We define the following elements and concepts that we use to introduce and explain our search cluster sparsification in the following sections of this chapter.

6.7.3.1 Notations Definition

The connectivity matrix Θ:
In the hyper-cluster, small cells can be connected through different and various technologies (Fiber, WiFi, Microwave, mmW, LTE, etc.). The communication quality and the aggregated throughput on the links between every small cell couple is not necessarily identical over time and space. According to the small cells backhaul technology and topology, the communication link existence and reliability may depend on several factors such as the distance between small cells, the deployment scenario (urban, dense, LOS, NLOS, etc.), the backhaul congestion, and the channel quality in wireless scenarios. The connectivity matrix $\boldsymbol{\Theta}$ at each small cell n is a matrix that contains the connection quality between every small cell couple.
$\boldsymbol{\Theta}$ is a generic of the connectivity matrix $\mathbf{X}$ defined in Section 6.3. $\boldsymbol{\Theta}$ represents normalized connection quality between each small cell couple, whereas $\mathbf{X}$ is a binary matrix that specifies if the small cells are connected or not. $\mathbf{X}$ can be obtained from $\boldsymbol{\Theta}$ by a thresholding operation.

Popularity matrix P:
The popularity matrix is defined in Section 6.3. It gives the popularity of computational requests at each small cell. We do not tackle the problem of building and updating the popularity matrix. We assume that it is built according to statistical observation over time, of the number of each request occurrence at each small cell.

Caching policy metric λ:
Based on the popularity matrix, a metric λ can be built for each couple (request, small cell) in order to choose *what* to cache, and *where* to cache it. This metric can also take into consideration system conditions (wired/wireless connection, available memory, etc.), context, small cells connectivity, etc. λ is associated to a caching policy or algorithm that places the requests in cache memories. Various policies with different objectives can be designed and adopted. *ClusterCaching* and *PropCaching* are two examples of caching policies described in Section 6.5 using $\lambda = \frac{p_{n,c} W_c |\mathcal{N}(n)|}{L_c \sum\limits_{m \in \mathcal{N}(n)} R_{n,m}}$ and p - the files popularity - as a caching metric, respectively.

Distributed cache matrix D:
Caching metric and policy are used at each small cell to decide over the caching of computational requests. A binary matrix **D** defined in Section 6.3 keeps track of the cached requests at each small cell.

In cache probability matrix Γ:
Exploiting cache matrix **D**, and popularity matrix **P**, an *in cache* probability matrix $\mathbf{\Gamma}$ can be computed. Since the caching metric depends on the popularity matrix, a relationship between the requests popularities and their caching decision can be established. It helps locating requested tasks in cache in function of their caching probability, which can be deduced from the popularity. $\mathbf{\Gamma}$ gives for each request popularity (popularity intervals), the probability of the request being cached at each small cell.

$$\mathbf{\Gamma} = \begin{bmatrix} \gamma_{1,1} & \cdots & \gamma_{1,Q} \\ \gamma_{2,1} & \cdots & \gamma_{2,Q} \\ \vdots & \vdots & \vdots \\ \gamma_{N,1} & \cdots & \gamma_{N,Q} \end{bmatrix} \in [0,1]^{N \times Q} \tag{6.10}$$

Where Q is the number of considered intervals of popularity. $\mathbf{\Gamma}$ can represent either *nominative* or *cumulative* In cache probability.
Nominative in cache probability is defined as:

$$\gamma(n,q) = \mathbf{P}(d_{n,c} = 1 | p_1(q) < p_c \leq p_2(q)) \tag{6.11}$$

where $[p_1, p_2]$ is the q popularity interval in which p_c is.
Cumulative in cache probability is defined as:

$$\gamma(n,q) = \mathbf{P}(d_{n,c} = 1 | p_c \leq p_2) \tag{6.12}$$

6.7.3.2 Traffic Classification

Computational requests launched at mobile users' device can be classified according to various characteristics. One classification is based on computation offloadability. This classification is used in the offloading decision algorithm proposed in Chapter 3. Examples of tasks that cannot be offloaded are tasks that use mobile devices hardware such as cameras and microphones. Non offloadable tasks have to be executed locally on the mobile devices. As for the offloadable tasks, we propose to classify them into *Private* and *Common*. By *Private* we refer to computations that are unique for the user that requested them. They may depend on a local metric such as location, or serving small cell. A *Private* computation, can be any computation component that does not yield to the same results if requested by different users. In general, *Private* computations are not able to be shared among different users.

By *Common* we refer to computation that can be requested by any user and yields to the same result (total or partial). We refer as *Common* computations to the ones that have the possibility to be saved in cache memory somewhere in the network.

Private traffic: *Private* computations are not cached. However, an offloading decision process decides if they will be computed locally at the mobile device or offloaded to the small cell where a computation cluster is formed (See Chapter 3).

Common traffic: Common computations have the possibility to be saved in cache memory somewhere in the network. Therefore, before any computation is made, a 'search' for computation

results is launched. If the requested computation is cached, it is retrieved (with the assumption that retrieval costs $<$ computation cost). Otherwise, a computation cluster is formed for computing the request. The popularity matrix can then be updated accordingly.

6.7.4 Proposed Search Cluster Sparsification Method

In this section, we detail the cluster search sparsification method that we propose. After having identified the necessary elements for introducing our proposal, we detail the method in a series of successive steps.

- **Step 0: Initialization**
  We assume that, at each small cell, are available:

  (i) relative popularity matrix $\mathbf{P}_{(1\times C)}$, where C is the number of considered computations. In addition to their own popularity matrix, small cells exchange popularity matrices, and thus, each small cell has access to its own popularity matrix, and to the matrices received from neighboring small cells. The size of the popularity matrix available at each small cell is then equal to $(N_r \times C)$, where $N_r \leq N$ is the number of small cells which popularity matrix is received by the small cell in question.

  (ii) relative connectivity matrix $\mathbf{\Theta}_{(1\times N)}$ that reports normalized connection quality with neighbor small cells.

  (iii) relative distributed cache inventory $\mathbf{D}_{(1\times C)}$ of the cached computations at the small cell in question.

  (iv) *In cache* probability matrices $\mathbf{\Gamma}$ of the small cell in question, and of the neighbor small cells that diffused their own. The size of $\mathbf{\Gamma}$ depends on the popularity quantification size $\delta q = p_2(q) - p_1(q)$. Here, there is a trade-off between the quantification granularity and the overhead. Indeed, the smaller the popularity quantification step δq, the more accurate and reliable is the associated *in cache* probability. On the other hand, small popularity quantification steps result in larger *in cache* matrix, and thus more network overhead, since $\mathbf{\Gamma}$ matrices are shared between network small cells.

- **Step 1: Computations classification**
  In this step, the mobile devices run through the received computational requests and classify them as offloadable or not offloadable. Non offloadable requests are executed on the mobile devices. Local computational resources are allocated for the execution of these tasks.

- **Step 2: Private/Common classification**
  Offloadable tasks are classified as *Private* or *Common*. We distinguish three types of *Private* computations.

  (i) *Least private*: computations are user-private but can be offloaded to the SSC. A computation cluster can be set up by the SSC for the execution of the task.

  (ii) *Medium private*: computations can be offloaded to the SSC, but no computation cluster can be set up for their execution. Tasks can then be computed either locally at the mobile devices, or at the serving small cell.

  (iii) *Utmost private*: Computations cannot be offloaded to the serving small cell. They are executed at the mobile devices. Non offloadable tasks are a subset of the utmost private computations.

Offloading decision algorithm is called for *Medium private* and *Least private* computations. According to the offloading decision and the computation type, either local computation resources are allocated, or joint computation and communication resource allocation takes place in case of cluster set up necessity.

As for *Common* computations, they have the possibility to be found in cache memories of the SSC or neighbor HSCs in the network. If any *Common* computations are requested, they are offloaded to the SSC, and the following algorithm steps are executed.

- **Step 3: In cache probability quantification**
  In cache probability matrix $\mathbf{\Gamma}$ is available at each small cell. For each computation c of the set of *Common* requests $\mathcal{C}$ we identify the popularity p_c. We create a binary matrix $\mathbf{\Gamma'}$ where each line represents a fixed *in cache* probability η, and each column represents small cells whose *in cache* probability matrix is known. The size of $\mathbf{\Gamma'}$ depends on the *in cache* probability quantization step and/or number of levels. Note that the probability quantization can be linear or non-linear. We denote as H the number of probability quantization levels, and thus $\mathbf{\Gamma'}$ will be a matrix of $(H \times N_r + 1)$. $\mathbf{\Gamma'}$ is defined as follows:

$$\mathbf{\Gamma'} = \begin{bmatrix} \gamma'_{1,1} & \cdots & \gamma'_{1,N_r+1} \\ \gamma'_{2,1} & \cdots & \gamma'_{2,N_r+1} \\ \vdots & \vdots & \vdots \\ \gamma'_{H,1} & \cdots & \gamma'_{H,N_r+1} \end{bmatrix} \in \{0,1\}^{H \times N_r + 1} \tag{6.13}$$

  where

$$\gamma'(\eta, s) = \begin{cases} 1 & \text{if } \gamma(s,q) \leq \eta(q), \text{ where } q = \{q / p_1(q) < p_c \leq p_2(q)\} \\ 0 & \textit{otherwise} \end{cases} \tag{6.14}$$

  $\mathbf{\Gamma'}$ identifies, for each target *in cache* probability η, the set of small cells where the probability of finding a cached copy of the request is at least equal to η.

- **Step 4: Reachability weighting**
  In the previous step, the created matrix $\mathbf{\Gamma'}$ allows to identify the set of small cells that hold the requested file with a certain probability. However, the file existence in a small cell's cache is not sufficient for retrieving it. Since computational requests are associated to latency constraints, the connection quality between the SSC, s_s, and HSC ,s, where the computation is cached is important to consider. The cached copy should be retrieved from the cache while respecting the latency constraints. Furthermore, $\mathbf{\Gamma'}$ identifies the possible location of a cached copy of the computation without identifying the *best* location from which it should be downloaded.
  In this step, the elements of $\mathbf{\Gamma'}$ are weighted with the connectivity between small cells. Therefore, we weight the elements of each column of $\mathbf{\Gamma'}$ by the connectivity of the small cell with the SSC. The weights are the elements of the SSC connectivity in matrix $\mathbf{\Theta}$. A matrix $\mathbf{\Gamma''}$ is then defined as follows:

$$\mathbf{\Gamma''} = \begin{bmatrix} \gamma''_{1,1} & \cdots & \gamma''_{1,N_r+1} \\ \gamma''_{2,1} & \cdots & \gamma''_{2,N_r+1} \\ \vdots & \vdots & \vdots \\ \gamma''_{H,1} & \cdots & \gamma''_{H,N_r+1} \end{bmatrix} \in [0,1]^{H \times N_r + 1} \tag{6.15}$$

where

$$\gamma''(\eta,s) = \gamma'(\eta,s)\theta(s_s,s) \tag{6.16}$$

In $\mathbf{\Gamma''}$, lines represent the probability of finding the cached request in small cells, while the matrix element values show the download link quality.

- **Step 5: Truncating**
  According to requests constraints in terms of latency, a minimum downlink quality can be set in order to respect these constraints. In addition to latency, eliminating weak communication link reduces power consumption. Resources management policies at SSCs and HSCs contribute then in setting a threshold on the link quality to be used for cached request download. Setting ζ as the link quality threshold and applying it to $\mathbf{\Gamma''}$ will result in having a binary matrix $\mathbf{\Gamma''_t}$ that identifies the cached request locations with a minimum download link quality of ζ. $\mathbf{\Gamma''_t}$ is defined as follows:

$$\mathbf{\Gamma''_t} = \begin{bmatrix} \gamma''_{t\ 1,1} & \cdots & \gamma''_{t\ 1,N_r+1} \\ \gamma''_{t\ 2,1} & \cdots & \gamma''_{t\ 2,N_r+1} \\ \vdots & \vdots & \vdots \\ \gamma''_{t\ H,1} & \cdots & \gamma''_{t\ H,N_r+1} \end{bmatrix} \in \{0,1\}^{H\times N_r+1} \tag{6.17}$$

where

$$\gamma''_t(\eta,s) = \begin{cases} 1 & \text{if } \gamma''(s,q) \geq \zeta \\ 0 & otherwise \end{cases} \tag{6.18}$$

- **Step 6: Uplink requests set up**
  $\mathbf{\Gamma''_t}$ allows to locate cached requests possible locations. According to the desired search cluster size, and/or latency and power consumption policies, a search cluster can be identified using $\mathbf{\Gamma''_t}$. Indeed, each line of $\mathbf{\Gamma''_t}$ represents a set of small cell that form a search cluster with a defined probability of retrieving the file. For each line q of $\mathbf{\Gamma''_t}$, the search cluster includes the set of small cells defined by $\{s\,|\,\gamma''_t(q,s)=1\}$. The minimum probability of finding the request in at least one of the cluster small cells is:

$$\begin{aligned} \mathbf{P}(c \in cluster) &= 1 - \mathbf{P}(c \notin cluster) \\ &\geq 1 - \prod_{s=1}^{N_r+1} \gamma''_t(q,s)(1-\eta(q)) \\ &\geq 1 - (1-\eta(q))^{||\boldsymbol{\gamma}''_t(q)||_0} \end{aligned} \tag{6.19}$$

where $\boldsymbol{\gamma}''_t(q)$ is the qth line array of $\mathbf{\Gamma''_t}$, and $||.||_0$ is the l_0 norm, which is the number of non-zero elements in the array. $||\boldsymbol{\gamma}''_t(q)||_0$ determines the search cluster size, which can also be an important metric in the cluster set up decision.
If there is a maximal search cluster size limit N_{max}, the cluster q' that maximizes the cache hit probability is defined by the line q' of $\mathbf{\Gamma''_t}$ where

$$q' = \{\min_q |\ ||\boldsymbol{\gamma}''_t(q)||_0 \leq N_{max}\} \tag{6.20}$$

If there is a minimum cache hit probability ξ, then the smallest cluster is defined by

$$q' = \{\max_q |\ (1-\eta(q))^{||\boldsymbol{\gamma}''_t(q)||_0} \geq \xi\} \tag{6.21}$$

Table 6.2: Computation and SSC association

Computation	Serving SC
c_1	SC_1
c_2	SC_3
c_3	SC_4

We note that the smaller the value of q' is, the larger the cluster. When a cluster size limit is set, the proposed method allows to identify the search cluster that achieves the highest perceived *in cache* probability, with a cluster of that size Whereas, when a cache hit minimum probability is set, the method identifies the smallest cluster that can achieve the *in cache* probability target.

6.7.5 Numerical Example of Search Cluster Sparsification

We consider a hyper-cluster of 6 small cells. We consider that computational requests have already been classified and offloading decisions have been made. The offloading computational request of type *Common* are received at the serving small cells. A set of 3 tasks is considered. Table 6.7.5 reports the association of each task with its SSC. The small cells connectivity $\mathbf{\Theta}$ and the computations probability $\mathbf{P}$ matrices are defined as follows:

$$\mathbf{\Theta}_{(6\times6)} = \begin{bmatrix} 1 & 0.9 & 0.3 & 0.4 & 0.2 & 0 \\ 0.9 & 1 & 0.9 & 0.1 & 0.4 & 0.3 \\ 0.3 & 0.9 & 1 & 0.3 & 0 & 0 \\ 0.4 & 0.1 & 0.3 & 10.9 & 0.8 & \\ 0.2 & 0.4 & 0 & 0.9 & 1 & 0.8 \\ 0 & 0 & 0.3 & 0.8 & 0.8 & 1 \end{bmatrix} \tag{6.22}$$

$$\mathbf{P}_{(6\times3)} = \begin{bmatrix} 0.05 & 0.05 & 0.02 \\ 0.01 & 0.02 & 0.01 \\ 0.02 & 0.55 & 0.06 \\ 0.01 & 0.05 & 0.04 \\ 0.03 & 0.07 & 0 \\ 0.08 & 0 & 0.02 \end{bmatrix} \tag{6.23}$$

Small cells SC_1, SC_2, and SC_3 follow the *ClusterCaching* policy and SC_4, SC_5, and SC_6 follow the *PropCaching* clustering policy (See section 6.5). The cumulative *in cache* probability matrix $\mathbf{\Gamma}$ of each of the small cells are represented in Figure 6.2, with a cache memory ratio of $\mu = 1/12$. We adopt a linear *in cache* probability quantization of 10 levels. Therefore, from **Step 3**, we will have

the following matrices:

$$\mathbf{\Gamma'}_{c_1} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{\Gamma'}_{c_2} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{\Gamma'}_{c_3} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (6.24)$$

For **Step 4**, we weight the matrices $\mathbf{\Gamma'}_{c_1}$, $\mathbf{\Gamma'}_{c_2}$, and $\mathbf{\Gamma'}_{c_3}$ with $\boldsymbol{\theta}'_1$, $\boldsymbol{\theta}'_2$, and $\boldsymbol{\theta}'_3$, respectively. We therefore obtain the following $\mathbf{\Gamma''}$ matrices:

$$\mathbf{\Gamma''}_{c_1} = \begin{bmatrix} 1 & 0 & 0.3 & 0 & 0.2 & 0 \\ 1 & 0 & 0 & 0 & 0.2 & 0 \\ 1 & 0 & 0 & 0 & 0.2 & 0 \\ 1 & 0 & 0 & 0 & 0.2 & 0 \\ 1 & 0 & 0 & 0 & 0.2 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{\Gamma''}_{c_2} = \begin{bmatrix} 0.9 & 1 & 0.9 & 0.5 & 0.4 & 0.3 \\ 0.9 & 0 & 0.9 & 0.5 & 0.4 & 0.3 \\ 0.9 & 0 & 0.9 & 0.5 & 0.4 & 0.3 \\ 0.9 & 0 & 0.9 & 0.5 & 0.4 & 0.3 \\ 0.9 & 0 & 0.9 & 0.5 & 0.4 & 0.3 \\ 0.9 & 0 & 0.9 & 0 & 0 & 0 \\ 0 & 0 & 0.9 & 0 & 0 & 0 \\ 0 & 0 & 0.9 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{\Gamma''}_{c_3} = \begin{bmatrix} 0.4 & 0 & 0.3 & 1 & 0 & 0 \\ 0 & 0 & 0.3 & 1 & 0 & 0 \\ 0 & 0 & 0.3 & 1 & 0 & 0 \\ 0 & 0 & 0.3 & 1 & 0 & 0 \\ 0 & 0 & 0.3 & 1 & 0 & 0 \\ 0 & 0 & 0.3 & 1 & 0 & 0 \\ 0 & 0 & 0.3 & 1 & 0 & 0 \\ 0 & 0 & 0.3 & 1 & 0 & 0 \\ 0 & 0 & 0.3 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (6.25)$$

In **Step 5**, we set *zeta*, the truncating threshold for link quality, as $\zeta = 0.3$. The truncated matrices are as follows:

$$\Gamma''_{t\,c_1} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \Gamma''_{t\,c_2} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \Gamma''_{t\,c_3} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (6.26)$$

For the cluster set up in **Step 6**, we consider the following constraints at each of the SSCs:
SC_1**:** Maximum cluster size = 1
SC_3**:** Minimum search cluster hit probability of 80%
SC_4**:** Maximum cluster size = 3

The lines (q) of each matrix represent the small cells that guarantees a minimum hit probability of $\eta = 0.1q$.

Table 6.3: *In cache* probability of search clusters

q	$\mathbf{P}(c_2 \in cluster)$	**q**	$\mathbf{P}(c_2 \in cluster)$
1	0.47	**6**	0.84
2	0.67	**7**	0.7
3	0.83	**8**	0.8
4	0.92	**9**	0
5	0.97	**10**	0

For c_1 at SC_1, the maximum cluster size is equal to 1. Therefore, we identify the search cluster as the lowest line with a single 1 value. This means that we chose the single cell cluster that guarantees the highest cache hit probability. Indeed, η increases with q and thus, $\mathbf{P}(c_1 \in cluster)$ defined in Eq. 6.20 increases. The chosen cluster is showed in the equation below, with: $\mathbf{P}(c_1 \in cluster) \geq 1-(1-0.1q)^1 = 1-(1-0.6) = 0.6$

$$\Gamma''_t c_1 = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{6.27}$$

In the case where the lowest line of the matrix has more small cells than the maximum limit, any subset of small cells can be chosen. Indeed, the lowest line represents the guaranteed locations of the requested file. Even if the small cells are aware of some guaranteed locations, it may happen that a minimum number of download sources are required for achieving lower delivery time.

In addition, we note that if the computation request is cached on the SSC itself, the SSC is always included in the cluster. This is because no connectivity constraints are imposed in this case.

For c_2 at SC_3 a minimum $\mathbf{P}(c_1 \in cluster)$ of $\xi = 0.8$ is required. If we compute this probability for each line of the matrix using Eq. 6.20, we obtain the results showed in table 6.7.5 According to the table, the eligible search clusters are those of $q = 3,4,5,,$, and 8. The proposed method will choose the highest q according to 6.21. The chosen cluster is the smallest cluster that achieves the desired in cache probability target. For $q = 5$ (dark purple highlight), $\mathbf{P}(c_1 \in cluster) > \xi$ as well, but the cluster size is of 5. the chosen cluster achieves $\xi = 0.8$ with cluster size of 1 (light green highlight).

$$\Gamma''_t c_2 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{6.28}$$

6.8 Conclusion

Mobile edge computing is a solution for allowing mobile users' devices to have access to a pool of resources that are at high proximity. Computation offloading to the edge cloud allow faster execution and at lower latencies, power consumption, and energy costs. Edge cloud small cell clustering is proposed in this thesis, and joint computational and communication resource allocation policies are presented, discussed, and evaluated. Proposed policies varied in clustering objectives. While some focused on increasing the cluster delivered QoE by reducing the perceived computation latency from users' perspective, others focused on reducing the computation cluster power consumption. In this chapter, we propose a novel concept that allows further improvement of computation clusters performance and resource allocation efficiency in terms of both perceived latency and power consumption. The novelty of the work presented in this chapter is twofold.

First, we introduce the concept of computation caching in the edge cloud. We exploit the cloud enabled network edge, not only to execute computational tasks using computation resources, but to cache computations using storage space available at the edge cloud. We shift the paradigm of edge cloud clustering, currently used as *demand and compute* into *search and download*. Indeed, the current vision of small cells clustering for computing is to choose the best nodes that can contribute in the execution of computational tasks and guaranteeing a QoE that respects the imposed latency constraints. The cost of the computation offloading and edge cloud computing through clustering is highly dependent on the computational load to be computed, and the size of computational data to be transmitted. Reducing the computations size and the size of data to be exchanged reduces the computation offloading costs. With the computation caching paradigm, tasks will be *searched for* and *downloaded* if possible. This prevents the edge cloud network from computing the same computations repeatedly, and reduces the size of exchanged data. For choosing the computations to be cached, we propose a caching algorithm based on the computations popularity. We exploit the computations popularity at each node of the edge cloud to create a caching metric that depends on the computation popularity, but also on the computation load and the required computational capacity. The caching metric helps identifying the computations that are *worth caching*, since it selects the popular computations that can generate higher costs reduction. The main idea is to store computations that are *popular*, but that also impose high computation and communication load, in the goal of reducing the edge cloud computing costs. We compare the average energy in an edge cloud computing scenario for the cases with no caching, with caching using the proposed algorithm, and with caching that uses files popularity as caching metric. The latter algorithm stores the most popular computations regardless of their load and demanded capacity. Numerical

evaluation showed that not only the size of available cache memory plays a role in reducing the costs, but also the used caching metric. The proposed caching metric proved to achieve lower energy costs for executing and/or downloading computations through an edge cloud platform with small cells clustering.

Second, we propose a search cluster establishment and sparsification method. The goal is to exploit the knowledge of computations popularity at edge cloud small cells for identifying possible locations for cached copies of the computation. Computation cached copies locations are not known by the edge cloud small cells. Indeed, diffusing the cache computation at each small cell in the edge cloud imposes severe overhead. Cached computations are frequently updated, and maintaining knowledge of each small cell cache contents is not practical. Therefore, we introduce the new *in cache* probability concept, which derives from the relationship between the computation popularity and its probability of being cached. This depends of course on the adopted caching policy. We exploit this information at serving small cells and weight it with the connectivity between small cells. This allows to evolve from *in cache* probability to *in cache* probability and reachability. By doing so, we identify possible *reachable* location of cached computation copies. The proposed method reduces the search space for cached elements by exploiting the relationship between request popularities and their possibility of being cached and reachable. Reducing the search space increases the edge computing performance by reducing cluster search delay, and its computational costs as well.

Conclusions and Future Work

In this thesis, we have looked into joint communication and computation resource allocation, and load distribution solutions and algorithms, for local mobile Edge cloud computing.

The Cloud has integrated wireless networks architecture through mobile cloud computing. The Edge cloud paradigm brings cloud-offered functionalities and services to the edge of the cloud, at a close proximity to mobile users. In this thesis, we focused on local Edge cloud, where operator network small cells, usually used for delivering communication services, are also able to deliver cloud services, such as computing and storage. Unlimited services and applications, in numerous domains, are henceforth accessible through mobile networks. Consequently, the foreseen increase in the number of connected devices and in generated traffic reveals a challenge to mobile networks for coping with the ever-increasing requirements.

First, we discussed requirements, and enabling technologies of future 5G networks, the next evolution in wireless networks. In order to keep delivering high QoS and QoE, 5G networks design is based on breakthrough technologies, including the integration of cloud-based technologies. We presented the evolution of cloud-enabled networks, and showed the progressive integration of the cloud concept into wireless networks. We discussed advantages and challenges of each of the cloud-based architectures including Cloud-RAN, mobile cloud computing, mobile edge cloud, and fog computing.

In this thesis, we adopted mobile edge cloud architecture. We proposed the novel concept of local cloud computing through small cell clustering. In literature, the cloud is mostly considered as an established entity that can deliver computational services to mobile users. Our approach is based on forming a local cloud where small cells cooperate for delivering cloud services to mobile users. We discussed the existing trade-offs in this architecture. More precisely, we tackled energy efficiency, from both devices and system perspectives, and delay related trade-offs. A deeper investigation on the impact of the adopted backhaul technology and topology for intra-cluster communication is presented. By analyzing different backhaul models, we showed that both computation and communication resource allocation are affected by the adopted backhaul models. As the backhaul has an impact on both cluster perceived latency and power consumption, communication and computation resources, the cluster size, and the load distribution, should be jointly allocated and optimized in order not to violate imposed delay constraints, or power budgets.

We proceeded to focus on the joint resource allocation in local mobile edge computing, in the

context of small cell computation clustering. We considered a system where mobile users launch computational requests, consisting of an instruction block to be computed in a maximum imposed time delay. We considered QoE as a performance metric throughout our work. Mobile users are satisfied if the results of their computational requests are delivered without violating latency constraints. After the analysis of the trade-offs between energy efficiency, perceived latency, and cluster size, among others, we have identified three research directions to increase the efficiency of mobile computation offloading, and improve the number of satisfied users requests. We divide the problem into two sub-problems, that we tackled respectively.

First, we considered the single user - single cloud case, where a mobile handset decides between executing computational requests locally and offloading the computations to the cloud. Computation offloading from mobile handsets to their serving small cell represents the first-hop communication in the adopted local edge cloud computing paradigm. We assumed that mobile handsets offload the computational requests to the cloud by sending the request to their associated serving small cell. We investigated the computation, handsets, and system parameters that affect the computation offloading decision. More precisely, we proposed a multi-parameter computation offloading decision that is executed at mobile handsets, SM-POD. The algorithm takes into account, not only energy consumption comparison of computation offloading and local computing, but also the offloadability of the computational request, handsets available resources (battery level, available memory space, computational capacity), tasks urgency, and communication channel quality. The algorithm joins offloading decision and scheduling while taking into account all the listed parameters, in order to take the offloading decision that guarantees users QoE. Simulation results showed how the proposed algorithm, SM-POD, helps increase the mobile handsets battery lifetime, and prevents any handset CPU capacity outage and memory overflow. This result can be justified by the adaptation of the offloading decision to handsets and system status. In opportunistic communication channel quality, the handset offloads *non-urgent* requests in order to save energy consumption. As a result of this research, we concluded that the computation offloading decision to edge cloud should be based on the multitude of parameters that affect not only the offloading decision energy consumption, but that also contribute in delivering high QoE to mobile users.

A second research direction to improve mobile users QoE and the efficiency of the local edge cloud computing paradigm, led us to investigate the small cells cluster set up to efficiently distribute computational load, and allocated computational and communication resources. In particular, we considered the case where serving small cells have to form small cell clusters to execute the received computational requests from mobile users. Serving small cells distribute the load on neighbor small cells, through a second-hop communication in the edge computing process. The challenge of small cell cluster set up is the joint communication and communication resource allocation that guarantees the respect of tasks latency constraints. Jointly allocating communication and computation resources is indeed required since both affect the perceived overall computation process latency. We start by studying the single-user multi-cloud scenario case. We proposed various small cells cluster optimizations with different objectives: latency minimization, cluster power consumption minimization, small cell centric power consumption minimization, and a cluster size reducing strategy. Cluster size reduction, or cluster sparsification, is based on exploiting the latency/power consumption trade-off. We proposed to modify the computation load distribution on less small cells, in order to allow the switching off of excluded small cell. Cluster sparsification increases the system energy efficiency, but results in increased perceived latency. We compared the proposed strategies and showed their impact on the perceived cluster latency and power con-

sumption.

We extended the problem formulation to the multi-user multi-cloud case. As the joint resource allocation and load distribution optimization problem is non-convex, we re-wrote an equivalent problem and proved it to be convex The solution of the optimization is derived. Simulation results proved that the proposed solution is able to serve a higher number of users comparing to a successive single user clustering, and static clustering. However, the proposed solution does not take into account tasks or resource scheduling, which can result in non-optimal resource allocation efficiency. With an objective of decreasing the cluster set up complexity, we proposed two different heuristic algorithms. A first algorithm consists of tasks scheduling at serving small cells, and assigning serving small cell computational resources as a first step. As a second step, unserved requests are sent to a centralized cluster manager unit that sets up computational clusters, using one of the proposed single-user multi-cloud clustering solutions. Three variants of the algorithms are proposed. The second proposed algorithm is based on an iterative approach, where each serving small cell sets up computational clusters, and ask for the small cell manager validation. The small cell manager verifies the possibility of the cluster establishment, and reports back to the serving small cells. In case of an excess allocation of computational resources at any of the small cells, the load excess is reported to the serving small cell for reassignment. Simulation results compared our three proposed multi-user cluster set up strategies and evaluated the loss of performance for heuristic solutions. The results proved that the joint resource allocation and load distribution in small cells cluster can guarantee service QoE with an outage probability of less than 5%, for up to 8 users per small cell. A gain of 55% comparing to successive clustering with no scheduling is achieved. As for the proposed scheduling aware and iterative heuristics, they can achieve up to 40% and 35 % gain of satisfaction ratio, respectively. These results prove that small cell clustering for mobile edge computing increases the cloud capabilities, and creates pooled resources at a close proximity to mobile users. However, in order to efficiently exploit available resources, small cells cluster should be optimized, communication and resources jointly allocated, computational tasks scheduled, and computational load adequately distributed.

Finally, in order to further reduce the costs of small cell cloud computing, we proposed a novel concept of *computation caching* for edge computing. After exploiting small cells computing resources for executing mobile users' requests, we proposed to exploit small cells storage space, to cache users' computational requests. In order to identify the computations to be cached at small cells cache memories, we proposed a caching algorithm, *ClusterCaching*. The algorithm identifies requests to be cached, while aiming at reducing the cost of small cell cloud computing. The proposed concept shifts the small cell edge cloud computing paradigm from a *demand and compute* approach, to *search and download*. Then, in order to reduce the search space, i.e. small cells that are searched for finding a cached copy of the requested computation, we propose a search cluster set up method. The proposed method exploits the caching policy in order to establish a relationship between computations popularity and their probability of being cached. This relationship is then exploited, along with the small cells connectivity level, in order to identify possible locations with a reachable cached version of the requested computation. The proposed contributions allow caching the right files at the right small cell, and propose a method for identifying the search space for guaranteeing a minimum cache hit probability. Numerical results proved that computation caching in edge cloud can bring major benefits, especially in terms of computation energy cost reduction. We conclude, that it is important to adapt the caching policy to the system storage capacity, and to cache the right files in accessible cache locations, for further increasing the edge cloud computing performance.

This thesis tackled the clustering techniques in the small cell local edge cloud computing paradigm. Three main research axes were investigated: (i) computation offloading decision; (ii) small cells clustering solutions; (iii) computational caching and cache cluster solutions.

Much research can be carried out in the following directions.

In Chapter 3, the proposed offloading decision algorithm takes into account a multitude of handset and applications parameters. It would be interesting to include an access control at the cloud side into the offloading decision. Possible ways of integrating the cloud in the offloading decision, without increasing the decision complexity, is to consider statistical value of the offered computational capacity by the edge cloud. This information could have a positive impact on the algorithm performance, especially in bad communication channels quality where the proposed algorithm offloads computation without guaranteeing their execution on the cloud.

In Chapter 4, we formulated the joint resource allocation optimization problem, without including tasks or resources scheduling. A challenging step would be to formulate the problem, and design a solution proposal for the joint resource allocation, load distribution, and tasks scheduling. In doing so, computational resources are more efficiently used, and thus, more computational tasks can be served, increasing the overall satisfaction ratio.
Further investigation can also be done in intra-cluster communication, especially the multi-user case, for integrating interference in the cluster set up process. Intra-cluster interference sets a novel challenge in edge cloud small cell clustering, and unblocks a series of possible solutions. For example, the cluster set up process can designed to minimize intra-cluster interference by using colored graph or interference classification techniques.

The proposed clustering solution in Chapters 4 and 5 are based on centralized approaches that include a small cell managing unit to compute or control the clusters set up parameters. A very interesting research area to investigate is the design of a decentralized small cell clusters set up algorithm. The algorithm should be light in terms of signaling and time complexity. Special forms of signaling can be used in order to reduce the intra-cluster overhead. We investigated, during this thesis, a special form on signaling, based on signaling response time, in order to pass cluster set up parameters without increasing signaling overhead. The idea has been investigated, and evolved to be the subject of a patent proposal [P3].

Mobile edge computing is the subject of a vivid research activity. Whereas the majority of the work focuses on the computing functionalities optimization of local cloud, the proposed computation caching concept is a novel paradigm that will gain attention in the near future. Computation caching is a research area that we would like to keep investigating in the near future. Studying both reactive and pro-active caching in small cell cloud networks, designing adaptive caching algorithms, and investigating cache search and computation retrieval algorithms, are possible leads for future work.

Finally, in this thesis, we proposed small cells clustering concepts and solutions in the edge cloud computing paradigm in cellular networks. A very interesting future work direction is to extend the proposed concepts of clustering for cloud functionalities into different contexts such as sensor networks, internet of things, device to device, and smart cities paradigms. The potential brought by the general approach brought by this thesis is indeed exploitable in various scenarios and paradigms. As millions of resource limited devices and sensors will be connected per cell,

clustering solutions can be seen as a very promising solutions for data aggregation, processing, caching, and computation services delivery. As a further step, computing clusters can also include hybrid types of computing entities, such as base stations, end devices, routers, and switches, which opens the possibility of a new form of inter-network cooperation for cloud services.

Bibliography

[1] GSMA Intelligence, "Understanding 5G: Perspectives on Future Technological Advancements in Mobile," https://gsmaintelligence.com/research/?file=141208-5g.pdf, December 2014.

[2] Lu Lu, G.Y. Li, A.L. Swindlehurst, A. Ashikhmin, and Rui Zhang, "An Overview of Massive MIMO: Benefits and Challenges," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 8, no. 5, pp. 742–758, Oct 2014.

[3] Ericsson, "Ericsson Mobility Report," http://www.ericsson.com/res/docs/2014/ericsson-mobility-report-november-2014.pdf, November 2014.

[4] P.A. Kotwal and A.R. Singh, "Evolution and Effects of Mobile Cloud Computing, Middleware Services on Cloud, Future Prospects: A Peek Into the Mobile Cloud Operating Systems," in *Computational Intelligence Computing Research (ICCIC), 2012 IEEE International Conference on*, Dec 2012, pp. 1–5.

[5] Sebastian Feld Thomas Schimper Michael Till Beck, Martin Werner, "Mobile Edge Computing: A Taxonomy," *Advances in Future Internet (AFIN 2014), The Sixth International Conference on*, January 2014.

[6] Ericsson, "Ericsson Mobility Report, on the Pulse of the Networked Society," http://www.ericsson.com/res/docs/2015/ericsson-mobility-report-june-2015.pdf, June 2015.

[7] A. Fehske, G. Fettweis, J. Malmodin, and G. Biczok, "The Global Footprint of Mobile Communications: The Ecological and Economic Perspective," *Communications Magazine, IEEE*, vol. 49, no. 8, pp. 55–62, August 2011.

[8] A. Nasir, M.Z. Shakir, K. Qaraqe, and E. Serpedin, "On the Reduction in Specific Absorption Rate Using Uplink power adaptation in heterogeneous small-cell networks," in *GCC Conference and Exhibition (GCC), 2013 7th IEEE*, Nov 2013, pp. 474–478.

[9] Antonio De Domenico, Emilio Calvanese Strinati, and Antonio Capone, "Review: Enabling Green Cellular Networks: A Survey and Outlook," *Comput. Commun.*, vol. 37, pp. 5–24, Jan. 2014.

[10] S. Barbarossa, S. Sardellitti, and P. Di Lorenzo, "Joint Allocation of Computation and Communication Resources in Multiuser Mobile Cloud Computing," in *Signal Processing*

Advances in Wireless Communications (SPAWC), 2013 IEEE 14th Workshop on, June 2013, pp. 26–30.

[11] 3GPP, "TR 36.932 V12.0.0; Scenarios and Requirements for Small Cell Enhancements for E-UTRA and E-UTRAN," December 2012.

[12] P. Monti, S. Tombaz, L. Wosinska, and J. Zander, "Mobile Backhaul in Heterogeneous Network Deployments: Technology Options and Power Consumption," in *Transparent Optical Networks (ICTON), 2012 14th International Conference on*, July 2012, pp. 1–7.

[13] 3GPP, "Report of 3GPP RAN Workshop on Release 12 and onwards.," June 2012.

[14] Cisco, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2014 - 2019," http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.pdf, February 2015.

[15] A. Osseiran, F. Boccardi, V. Braun, K. Kusume, P. Marsch, M. Maternia, O. Queseth, M. Schellmann, H. Schotten, H. Taoka, H. Tullberg, M.A. Uusitalo, B. Timus, and M. Fallgren, "Scenarios for 5G Mobile and Wireless Communications: The Vision of the METIS project," *Communications Magazine, IEEE*, vol. 52, no. 5, pp. 26–35, May 2014.

[16] European Project METIS., ," https://www.metis2020.com.

[17] A. Osseiran, V. Braun, T. Hidekazu, P. Marsch, H. Schotten, H. Tullberg, M.A. Uusitalo, and M. Schellman, "The Foundation of the Mobile and Wireless Communications System for 2020 and Beyond: Challenges, Enablers and Technology Solutions," in *Vehicular Technology Conference (VTC Spring), 2013 IEEE 77th*, June 2013, pp. 1–5.

[18] Nokia Networks, "5G Use Cases and Requirements," http://networks.nokia.com/sites/default/files/document/5g_requirements_white_paper.pdf.

[19] ERICCSON, "Ericsson White Paper: 5G Radio ACCESS," http://www.ericsson.com/res/docs/whitepapers/wp-5g.pdf, February 2015.

[20] Huawei Technologies Co., "5G: A Technology Vision," www.huawei.com/5gwhitepaper/, 2013.

[21] NTT DOCOMO, "DOCOMO 5G White Paper, 5G Radio Access: Requirements, Concept and Technologies," https://www.nttdocomo.co.jp/english/binary/pdf/corporate/technology/whitepaper_5g/DOCOMO_5G_White_Paper.pdf, July 2014.

[22] T.L. Marzetta, "Multi-cellular Wireless With Base Stations Employing Unlimited Numbers of Antennas," *UCSD Inf. Theory Applicat. Wprkshop, in Proc.*, February 2010.

[23] T.L. Marzetta, "Noncooperative Cellular Wireless with Unlimited Numbers of Base Station Antennas," *Wireless Communications, IEEE Transactions on*, vol. 9, no. 11, pp. 3590–3600, November 2010.

[24] E. Larsson, O. Edfors, F. Tufvesson, and T. Marzetta, "Massive MIMO for Next Generation Wireless Systems," *Communications Magazine, IEEE*, vol. 52, no. 2, pp. 186–195, February 2014.

[25] M.N. Tehrani, M. Uysal, and H. Yanikomeroglu, "Device-to-device Communication in 5G Cellular networks: Challenges, Solutions, and Future Directions," *Communications Magazine, IEEE*, vol. 52, no. 5, pp. 86–92, May 2014.

[26] A. Gupta and R.K. Jha, "A Survey of 5G Network: Architecture and Emerging Technologies," *Access, IEEE*, vol. 3, pp. 1206–1232, 2015.

[27] RAS (Radio Access and FP7 Future Networks Clusters Spectrum), "5G Radio Network Architecutre," http://fp7-semafour.eu/media/cms_page_media/9/SEMAFOUR_2014_RAScluster%20White%20paper.pdf.

[28] N. Bhushan, Junyi Li, D. Malladi, R. Gilmore, D. Brenner, A. Damnjanovic, R. Sukhavasi, C. Patel, and S. Geirhofer, "Network Densification: The Dominant Theme for Wireless Evolution Into 5G," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 82–89, Feb. 2014.

[29] Zhouyue Pi and F. Khan, "An Introduction to millimeter-Wave Mobile broadband Systems," *IEEE Communications Magazine*, vol. 49, no. 6, pp. 101–107, June 2011.

[30] J.G. Andrews, S. Buzzi, Wan Choi, S.V. Hanly, A. Lozano, A.C.K. Soong, and J.C. Zhang, "What Will 5G Be?," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 6, pp. 1065–1082, June 2014.

[31] T.S. Rappaport, Shu Sun, R. Mayzus, Hang Zhao, Y. Azar, K. Wang, G.N. Wong, J.K. Schulz, M. Samimi, and F. Gutierrez, "Millimeter Wave Mobile Communications for 5G Cellular: It Will Work!," *IEEE Access*, vol. 1, pp. 335–349, 2013.

[32] A. Sabharwal, P. Schniter, Dongning Guo, D.W. Bliss, S. Rangarajan, and R. Wichman, "In-Band Full-Duplex Wireless: Challenges and Opportunities," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 9, pp. 1637–1652, Sept. 2014.

[33] "Mobile-Edge Computing - Introductory Technical White Paper," https://portal.etsi.org/Portals/0/TBpages/MEC/Docs/Mobile-edge_Computing_-_Introductory_Technical_White_Paper_V1%2018-09-14.pdf, September 2014.

[34] T. Grance P. Mell, "The NIST Definition of Cloud Computing," Special Publication 800-145, September 2011.

[35] Ben Kepes, "Understanding The Cloud Computing Stack SaaS, Paas, IaaS," http://www.rackspace.com/knowledge_center/sites/default/files/whitepaper_pdf/Understanding-the-Cloud-Computing-Stack.pdf, 2011.

[36] Intel, "Planning Guide Virtualization and Cloud Computing," http://www.intel.fr/content/dam/www/public/us/en/documents/guides/cloud-computing-virtualization-building-private-iaas-guide.pdf, August 2013.

[37] Anand Vardhan Bhalla Mudit Ratana Bhalla, "Generations of Mobile Wireless Technology: A Survey," *Internation Journal of Computer Applications*, vol. 5, no. 4, 2010.

[38] B. Haberland, F. Derakhshan, H. Grob-Lipski, R. Klotsche, W. Rehm, P. Schefczik, and M. Soellner, "Radio Base Stations in the Cloud," *Bell Labs Technical Journal*, vol. 18, no. 1, pp. 129–152, June 2013.

[39] China Mobile, "C-RAN The Road Towards Green RAN," http://labs.chinamobile.com/cran/wp-content/uploads/CRAN_white_paper_v2_5_EN.pdf, October 2011.

[40] Aleksandra Checko, Aleksandra Checko, Henrik Holm, and Henrik Christiansen, "Optimizing Small Cell Deployment by the Use of C-RANs," in *European Wireless 2014; 20th European Wireless Conference; Proceedings of*, May 2014, pp. 1–6.

[41] Sourjya Bhaumik, Shoban Preeth Chandrabose, Manjunath Kashyap Jataprolu, Gautam Kumar, Anand Muralidhar, Paul Polakos, Vikram Srinivasan, and Thomas Woo, "CloudIQ: A Framework for Processing Base Stations in a Data Center," in *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, New York, NY, USA, 2012, Mobicom '12, pp. 125–136, ACM.

[42] Thomas Werthmann, Heidrun Grob-Lipski, and Magnus Proebster, "Multiplexing Gains Achieved in Pools of Baseband Computation Units in 4G Cellular Networks," in *Personal Indoor and Mobile Radio Communications (PIMRC), 2013 IEEE 24th International Symposium on*, Sept 2013, pp. 3328–3333.

[43] A. Checko, H.L. Christiansen, Ying Yan, L. Scolari, G. Kardaras, M.S. Berger, and L. Dittmann, "Cloud RAN for Mobile Networks ;A Technology Overview," *Communications Surveys Tutorials, IEEE*, vol. 17, no. 1, pp. 405–426, Firstquarter 2015.

[44] S. Abolfazli, Z. Sanaei, E. Ahmed, A. Gani, and R. Buyya, "Cloud-Based Augmentation for Mobile Devices: Motivation, Taxonomies, and Open Challenges," *Communications Surveys Tutorials, IEEE*, vol. 16, no. 1, pp. 337–368, First 2014.

[45] Mohsen Sharifi, Somayeh Kafaie, and Omid Kashefi, "A Survey and Taxonomy of Cyber Foraging of Mobile Devices," *Communications Surveys Tutorials, IEEE*, vol. 14, no. 4, pp. 1232–1243, Fourth 2012.

[46] Paramvir Bahl, Richard Y. Han, Li Erran Li, and Mahadev Satyanarayanan, "Advancing the State of Mobile Cloud Computing," in *Proceedings of the Third ACM Workshop on Mobile Cloud Computing and Services*, New York, NY, USA, 2012, MCS '12, pp. 21–28, ACM.

[47] Niroshinie Fernando, Seng W. Loke, and Wenny Rahayu, "Mobile Cloud Computing: A Survey ," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84 – 106, 2013, Including Special section: AIRCC-NetCoM 2009 and Special section: Clouds and Service-Oriented Architectures.

[48] M. Rosa Palacin, "Recent Advances in Rechargeable Battery Materials: a Chemist's Perspective," *Chem. Soc. Rev.*, vol. 38, pp. 2565–2575, 2009.

[49] K. Kumar and Yung-Hsiang Lu, "Cloud Computing for Mobile Users: Can Offloading Computation Save Energy?," *Computer*, vol. 43, no. 4, pp. 51–56, April 2010.

[50] A. Tuli, N. Hasteer, M. Sharma, and A. Bansal, "Exploring challenges in Mobile cloud computing: An overview," in *Confluence 2013: The Next Generation Information Technology Summit (4th International Conference)*, Sept 2013, pp. 496–501.

[51] Mahadev Satyanarayanan, P. Bahl, R Caceres, and N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing," *Pervasive Computing, IEEE*, vol. 8, no. 4, pp. 14–23, Oct 2009.

[52] M. Yannuzzi, R. Milito, R. Serral-Gracia, D. Montero, and M. Nemirovsky, "Key ingredients in an IoT recipe: Fog Computing, Cloud computing, and more Fog Computing," in *Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2014 IEEE 19th International Workshop on*, Dec 2014, pp. 325–329.

[53] A. Sabharwal, P. Schniter, Dongning Guo, D.W. Bliss, S. Rangarajan, and R. Wichman, "Connected Vehicles, the Internet of Things, and Fog Computing," *The Eightth ACM International Workshop on Vehicular Inter-Networking (VANET)*, Las Vegas, USA 2011.

[54] Cisco, "Fog Computing and the Internet of Things: Extened the Cloud to Where the Things Are," `https://www.cisco.com/web/solutions/trends/iot/docs/computing-overview.pdf`, 2015.

[55] I. Stojmenovic and Sheng Wen, "The Fog Computing Paradigm: Scenarios and Security Issues," in *Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on*, Sept 2014, pp. 1–8.

[56] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli, "Fog Computing and Its Role in the Internet of Things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, New York, NY, USA, 2012, MCC '12, pp. 13–16, ACM.

[57] Cisco, "Cisco Fog Computing Solutions: Unleash the Power of the Internet of Things," `https://www.cisco.com/web/solutions/trends/iot/docs/computing-solutions.pdf`, 2015.

[58] Federico Boccardi, Jeffrey G. Andrews, Hisham Elshaer, Mischa Dohler, Stefan Parkvall, Petar Popovski, and Sarabjot Singh, "Why to Decouple the Uplink and Downlink in Cellular Networks and How To Do It," *CoRR*, vol. abs/1503.06746, 2015.

[59] Electronic Communications Committee (ECC) CEPT, "Asymmetry of Mobile Backhaul Networks," `http://www.cept.org/documents/se-19/6560/se19`, 2012.

[60] S. Singh, Xinchen Zhang, and J.G. Andrews, "Uplink Rate Distribution in Heterogeneous Cellular Networks with Power Control and Load Balancing," in *Communication Workshop (ICCW), 2015 IEEE International Conference on*, June 2015, pp. 1275–1280.

[61] Nokia Solutions and Networks, "Nokia Solutions and Networks TD-LTE Frame Configuration Primer," `http://networks.nokia.com/system/files/document/nsn_td_lte_frame_configuration_wp.pdf`, November 2013.

[62] Ericsson, "Ericsson Mobility Report, On the Pulse of the Networked Society," `http://www.ericsson.com/res/docs/2012/ericsson-mobility-report-november-2012.pdf`, November 2012.

[63] Cisco, "Cisco Global Cloud Index: Forecast and Methodology, 2013-2018," `http://www.digital4.biz/upload/images/10_2013/131028130134.pdf`, 2014.

[64] SENSEI, "SENSEI EU project," `www.sensei-project.eu`, 2010.

[65] Qualcomm, "LTE Advanced: Heterogeneous Networks," `https://www.qualcomm.com/media/documents/files/lte-heterogeneous-networks.pdf`, January 2011.

[66] H. Elshaer, F. Boccardi, M. Dohler, and R. Irmer, "Downlink and Uplink Decoupling: A Disruptive Architectural Design for 5G Networks," in *Global Communications Conference (GLOBECOM), 2014 IEEE*, Dec 2014, pp. 1798–1803.

[67] J.G. Andrews, "Seven Ways that HetNets Are a Cellular Paradigm Shift," *Communications Magazine, IEEE*, vol. 51, no. 3, pp. 136–144, March 2013.

[68] F. Boccardi, R.W. Heath, A. Lozano, T.L. Marzetta, and P. Popovski, "Five Disruptive Technology Directions for 5G," *Communications Magazine, IEEE*, vol. 52, no. 2, pp. 74–80, February 2014.

[69] Lei Li and P. Butovitsch, "Uplink CoMP and the Applications in LTE Heterogeneous Networks: Principles and the Field Trial," in *Communication Technology (ICCT), 2013 15th IEEE International Conference on*, Nov 2013, pp. 309–314.

[70] FP7 European Project, "Distributed Computing, Storage and Radio Resource Allocation Over Cooperative Femtocells," (TROPIC).

[71] B. Waels, "WiFi and Small Cells in the Orange Mobile Network Strategy," September 2012.

[72] Fengming Cao and Zhong Fan, "The Tradeoff Between Energy Efficiency and System Performance of Femtocell Deployment," in *Wireless Communication Systems (ISWCS), 2010 7th International Symposium on*, Sept 2010, pp. 315–319.

[73] Yan Chen, Shunqing Zhang, Shugong Xu, and G.Y. Li, "Fundamental Trade-offs on Green Wireless Networks," *Communications Magazine, IEEE*, vol. 49, no. 6, pp. 30–37, June 2011.

[74] A. De Domenico, E.C. Strinati, and A. Duda, "An Energy Efficient Cell Selection Scheme for Open Access Femtocell Networks," in *Personal Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE 23rd International Symposium on*, Sept 2012, pp. 436–441.

[75] S. Chia, M. Gasparroni, and P. Brick, "The Next Challenge for Cellular Networks: Backhaul," *Microwave Magazine, IEEE*, vol. 10, no. 5, pp. 54–66, August 2009.

[76] D.C. Chen, T.Q.S. Quek, and M. Kountouris, "Backhauling in Heterogeneous Cellular Networks: Modeling and Tradeoffs," *Wireless Communications, IEEE Transactions on*, vol. 14, no. 6, pp. 3194–3206, June 2015.

[77] C. Abgrall, E.C. Strinati, and J.-C. Belfiore, "Distributed Power Allocation for Interference Limited Networks," in *Personal Indoor and Mobile Radio Communications (PIMRC), 2010 IEEE 21st International Symposium on*, Sept 2010, pp. 1342–1347.

[78] A.J. Goldsmith and Soon-Ghee Chua, "Adaptive Coded Modulation for Fading Channels," *Communications, IEEE Transactions on*, vol. 46, no. 5, pp. 595–602, May 1998.

[79] O. Munoz-Medina, A. Pascual-Iserte, and J. Vidal, "Optimization of Radio and Computational Resources for Energy Efficiency in Latency-Constrained Application Offloading," *Vehicular Technology, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2014.

[80] J. Baliga, R.W.A. Ayre, K. Hinton, and RodneyS. Tucker, "Green Cloud Computing: Balancing Energy in Processing, Storage, and Transport," *Proceedings of the IEEE*, vol. 99, no. 1, pp. 149–167, Jan 2011.

[81] Eduardo Cuervo, Aruna Balasubramanian, Dae ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Ch, and Paramvir Bahl, "Maui: Making Smartphones Last Longer With Code Offload," in *In In Proceedings of ACM MobiSys*, 2010.

[82] 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," in *INFOCOM, 2012 Proceedings IEEE*, March 2012, pp. 945–953.

[83] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti, "CloneCloud: Elastic Execution Between Mobile Device and Cloud," in *Proceedings of the Sixth Conference on Computer Systems*, New York, NY, USA, 2011, EuroSys '11, pp. 301–314, ACM.

[84] Guangyu Chen, B.-T. Kang, M. Kandemir, N. Vijaykrishnan, M.J. Irwin, and R. Chandramouli, "Studying Energy Trade Offs in Offloading Computation/Compilation in Java-enabled Mobile Devices," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 15, no. 9, pp. 795–809, Sept 2004.

[85] E. Lagerspetz and S. Tarkoma, "Mobile Search and the Cloud: The Benefits of Offloading," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*, March 2011, pp. 117–122.

[86] Yonggang Wen, Weiwen Zhang, and Haiyun Luo, "Energy-optimal Mobile application execution: Taming Resource-poor Mobile Devices With Cloud Clones," in *INFOCOM, 2012 Proceedings IEEE*, March 2012, pp. 2716–2720.

[87] S. Barbarossa, S. Sardellitti, and P. Di Lorenzo, "Communicating while computing: Distributed mobile cloud computing over 5g heterogeneous networks," *Signal Processing Magazine, IEEE*, vol. 31, no. 6, pp. 45–55, Nov 2014.

[88] Xiaohui Gu, K. Nahrstedt, A. Messer, I. Greenberg, and D. Milojicic, "Adaptive Offloading Inference for Delivering Applications in Pervasive Computing Environments," in *Pervasive Computing and Communications, 2003. (PerCom 2003). Proceedings of the First IEEE International Conference on*, March 2003, pp. 107–114.

[89] Bo Gao, Ligang He, Limin Liu, Kenli Li, and S.A. Jarvis, "From Mobiles to Clouds: Developing Energy-Aware Offloading Strategies for Workflows," in *Grid Computing (GRID), 2012 ACM/IEEE 13th International Conference on*, Sept 2012, pp. 139–146.

[90] D. Kovachev, Tian Yu, and R. Klamma, "Adaptive Computation Offloading from Mobile Devices into the Cloud," in *Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on*, July 2012, pp. 784–791.

[91] E.C. Strinati, G. Corbellini, and D. Kténas, "HYGIENE Scheduling for OFDMA Wireless Cellular Networks," in *Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th*, April 2009, pp. 1–5.

[92] A.R. Jensen, M. Lauridsen, P. Mogensen, T.B. SÃ¸rensen, and P. Jensen, "LTE UE Power Consumption Model: For System Level Energy and Performance Optimization," in *Vehicular Technology Conference (VTC Fall), 2012 IEEE*, Sept 2012, pp. 1–5.

[93] S.Y. Seidel and T.S. Rappaport, "914 MHz Path Loss Prediction Models for Indoor Wireless Communications in Multifloored Buildings," *Antennas and Propagation, IEEE Transactions on*, vol. 40, no. 2, pp. 207–217, Feb 1992.

[94] Murali Annavaram, "Energy per Instruction Trends in. Intel Microprocessors," *Technology Intel Magazine*, 2006.

[95] Michael Smit, Mark Shtern, Bradley Simmons, and Marin Litoiu, "Partitioning Applications for Hybrid and Federated Clouds," in *Proceedings of the 2012 Conference of the Center for Advanced Studies on Collaborative Research*, Riverton, NJ, USA, 2012, CASCON '12, pp. 27–41, IBM Corp.

[96] Cheng Wang and Zhiyuan Li, "Parametric Analysis for Adaptive Computation Offloading," in *Proceedings of the ACM SIGPLAN 2004 Conference on Programming Language Design and Implementation*, New York, NY, USA, 2004, PLDI '04, pp. 119–130, ACM.

[97] Tim Verbelen, Tim Stevens, Filip De Turck, and Bart Dhoedt, "Graph Partitioning Algorithms for Optimizing Software Deployment in Mobile Cloud Computing," *Future Generation Computer Systems*, vol. 29, no. 2, pp. 451 – 459, 2013, Special section: Recent advances in e-Science.

[98] Stefania Sardellitti, Gesualdo Scutari, and Sergio Barbarossa, "Joint Optimization of Radio and Computational Resources for Multicell Mobile-Edge Computing," *IEEE Trans. on Signal and Information Processing over Networks*, June.

[99] L. Yang, J. Cao, and H. Cheng, "Resource Constrained Multi-user Computation Partitioning for Interactive Mobile Cloud Applications," Technical Report, 2012.

[100] Dejan Kovachev and Ralf Klamma, "Framework for Computation Offloading in Mobile Cloud Computing," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 1, no. 7, pp. 6–15, 12/2012 2012.

[101] J. Oueis, E.C. Strinati, and S. Barbarossa, "Multi-parameter Decision Algorithm for Mobile Computation Offloading," in *Wireless Communications and Networking Conference (WCNC), 2014 IEEE*, April 2014, pp. 3005–3010.

[102] M. Reza Rahimi, Nalini Venkatasubramanian, Sharad Mehrotra, and Athanasios V. Vasilakos, "MAPCloud: Mobile Applications on an Elastic and Scalable 2-Tier Cloud Architecture," in *Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing*, Washington, DC, USA, 2012, UCC '12, pp. 83–90, IEEE Computer Society.

[103] M.R. Rahimi, N. Venkatasubramanian, and A.V. Vasilakos, "MuSIC: Mobility-Aware Optimal Service Allocation in Mobile Cloud Computing," in *Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on*, June 2013, pp. 75–82.

[104] Akshat Verma, Puneet Ahuja, and Anindya Neogi, "pMapper: Power and Migration Cost Aware Application Placement in Virtualized Systems," in *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, New York, NY, USA, 2008, Middleware '08, pp. 243–264, Springer-Verlag New York, Inc.

[105] Anton Beloglazov and Rajkumar Buyya, "Adaptive Threshold-based Approach for Energy-efficient Consolidation of Virtual Machines in Cloud Data Centers," in *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science*, New York, NY, USA, 2010, MGC '10, pp. 4:1–4:6, ACM.

[106] M. Cardosa, M.R. Korupolu, and A. Singh, "Shares and Utilities Based Power Consolidation in Virtualized Server Environments," in *Integrated Network Management, 2009. IM '09. IFIP/IEEE International Symposium on*, June 2009, pp. 327–334.

[107] Saurabh Kumar Garg, Srinivasa K. Gopalaiyengar, and Rajkumar Buyya, "SLA-based Resource Provisioning for Heterogeneous Workloads in a Virtualized Cloud Datacenter," in *Proceedings of the 11th International Conference on Algorithms and Architectures for Parallel Processing - Volume Part I*, Berlin, Heidelberg, 2011, ICA3PP'11, pp. 371–384, Springer-Verlag.

[108] Ripal Nathuji, Aman Kansal, and Alireza Ghaffarkhah, "Q-clouds: Managing Performance Interference Effects for QoS-aware Clouds," in *Proceedings of the 5th European Conference on Computer Systems*, New York, NY, USA, 2010, EuroSys '10, pp. 237–250, ACM.

[109] Hongbin Liang, Dijiang Huang, and Daiyuan Peng, "On Economic Mobile Cloud Computing Model," in *Mobile Computing, Applications, and Services*, Martin Gris and Guang Yang, Eds., vol. 76 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 329–341. Springer Berlin Heidelberg, 2012.

[110] Hongbin Liang, L.X. Cai, Dijiang Huang, Xuemin Shen, and Daiyuan Peng, "An SMDP-Based Service Model for Interdomain Resource Allocation in Mobile Cloud Networks," *Vehicular Technology, IEEE Transactions on*, vol. 61, no. 5, pp. 2222–2232, Jun 2012.

[111] V. Di Valerio and F. Lo Presti, "Optimal Virtual Machines allocation in mobile femto-cloud computing: An MDP approach," in *Wireless Communications and Networking Conference Workshops (WCNCW), 2014 IEEE*, April 2014, pp. 7–11.

[112] Lei Yang, Jiannong Cao, Shaojie Tang, Tao Li, and A.T.S Chan, "A Framework for Partitioning and Execution of Data Stream Applications in Mobile Cloud Computing," in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, June 2012, pp. 794–802.

[113] picoChip Designs Vodafone 3GPP TSG-RAN451, Alcatel-Lucent, "R4-092042, Simulation Assumptions and Parameters for FDD HENB RF Requirements," May 2009.

[114] Stephen Boyd and Lieven Vandenberghe, *Convex Optimization*, Cambridge University Press, New York, NY, USA, 2004.

[115] F.; Vonfra M.; Dolezal J.; Mach P.; Becvar Z.; LoPresti F.; Wibowo A. Puente, M.A.; Lobillo, ," `http://www.ict-tropic.eu/documents/deliverables/TROPIC_D52ATOSf.pdf`, June 2014.

[116] D. Niyato, P. Wang, E. Hossain, W. Saad, and Zhu Han, "Game Theoretic Modeling of Cooperation Among Service Providers in Mobile Cloud Computing Environments," in *Wireless Communications and Networking Conference (WCNC), 2012 IEEE*, April 2012, pp. 3128–3133.

[117] 3GPP TS 36.300, 2013, ," http://www.3gpp.org/ftp/Specs/html-info/36300.htm.

[118] F.M. Chiussi and V. Sivaraman, "Achieving High Utilization in Guaranteed Services Networks Using Early-Deadline-First Scheduling," in *Quality of Service, 1998. (IWQoS 98) 1998 Sixth International Workshop on*, May 1998, pp. 209–217.

[119] K. Norlund, T. Ottosson, and A. Brunstrom, "Fairness Measures for Best Effort Traffic in Wireless Networks," in *Personal, Indoor and Mobile Radio Communications, 2004. PIMRC 2004. 15th IEEE International Symposium on*, Sept 2004, vol. 4, pp. 2953–2957 Vol.4.

[120] V. Chandrasekhar, J.G. Andrews, and Alan Gatherer, "Femtocell Networks: a Survey," *Communications Magazine, IEEE*, vol. 46, no. 9, pp. 59–67, September 2008.

[121] Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-László Barabási, "Limits of Predictability in Human Mobility," *Science*, vol. 327, no. 5968, pp. 1018–1021, 2010.

[122] Vincent Etter, Mohamed Kafsi, and Ehsan Kazemi, "Been There, Done That: What your Mobility Traces Reveal about your Behavior.," *the Procedings of Mobile Data Challenge by Nokia Workshop at the Tenth International Conference on Pervasive Computing*, June 2012.

[123] D. Wessels and K. Claffy, "Internet cache protocol and microsoft proxy server version 2," .

[124] M. Kataoka, K. Toumura, H. Okita, Junji Yamamoto, and T. Suzuki, "Distributed Cache System for Large-Scale Networks," in *Computing in the Global Information Technology, 2006. ICCGI '06. International Multi-Conference on*, Aug 2006, pp. 40–40.

[125] Jing Zhang, Gongqing Wu, Xuegang Hu, and Xindong Wu, "A Distributed Cache for Hadoop Distributed File System in Real-Time Cloud Services," in *Grid Computing (GRID), 2012 ACM/IEEE 13th International Conference on*, Sept 2012, pp. 12–21.

[126] G. Anandharaj and R. Anitha, "A Distributed Cache Management Architecture for Mobile Computing Environments," in *Advance Computing Conference, 2009. IACC 2009. IEEE International*, March 2009, pp. 642–648.

[127] Peter J. Denning, "The Working Set Model for Program Behavior," *Commun. ACM*, vol. 11, no. 5, pp. 323–333, May 1968.

[128] E. Bastug, J.-L. Guenego, and M. Debbah, "Proactive Small Cell Networks," in *Telecommunications (ICT), 2013 20th International Conference on*, May 2013, pp. 1–5.

[129] E. Bastug, M. Bennis, and M. Debbah, "Living on the Edge: The Role of Proactive Caching in 5G Wireless Networks," *Communications Magazine, IEEE*, vol. 52, no. 8, pp. 82–89, Aug 2014.

[130] Jingxiong Gu, Wei Wang, Aiping Huang, Hangguan Shan, and Zhaoyang Zhang, "Distributed Cache Replacement for Caching-enable Base Stations in Cellular Networks," in *Communications (ICC), 2014 IEEE International Conference on*, June 2014, pp. 2648–2653.

[131] C. Tschudin and M. Sifalakis, "Named Functions and Cached Computations," in *Consumer Communications and Networking Conference (CCNC), 2014 IEEE 11th*, Jan 2014, pp. 851–857.

[132] Amos Waterland, Elaine Angelino, Ekin D. Cubuk, Efthimios Kaxiras, Ryan P. Adams, Jonathan Appavoo, and Margo Seltzer, "Computational Caches," in *Proceedings of the 6th International Systems and Storage Conference*, New York, NY, USA, 2013, SYSTOR '13, pp. 8:1–8:7, ACM.

Abstract: Mobile Edge Cloud brings the cloud closer to mobile users by moving the cloud computational efforts from the internet to the mobile edge. We adopt a local mobile edge cloud computing architecture, where small cells are empowered with computational and storage capacities. Mobile users' offloaded computational tasks are executed at the cloud-enabled small cells. We propose the concept of small cells clustering for mobile edge computing, where small cells cooperate in order to execute offloaded computational tasks. A first contribution of this thesis is the design of a multi-parameter computation offloading decision algorithm, SM-POD. The proposed algorithm consists of a series of low complexity successive and nested classifications of computational tasks at the mobile side, leading to local computation, or offloading to the cloud. To reach the offloading decision, SM-POD jointly considers computational tasks, handsets, and communication channel parameters. In the second part of this thesis, we tackle the problem of small cell clusters set up for mobile edge cloud computing for both single-user and multi-user cases. The clustering problem is formulated as an optimization that jointly optimizes the computational and communication resource allocation, and the computational load distribution on the small cells participating in the computation cluster. We propose a cluster sparsification strategy, where we trade cluster latency for higher system energy efficiency. In the multi-user case, the optimization problem is not convex. In order to compute a clustering solution, we propose a convex reformulation of the problem, and we prove that both problems are equivalent. With the goal of finding a lower complexity clustering solution, we propose two heuristic small cells clustering algorithms. The first algorithm is based on resource allocation on the serving small cells where tasks are received, as a first step. Then, in a second step, unserved tasks are sent to a small cell managing unit (SCM) that sets up computational clusters for the execution of these tasks. The main idea of this algorithm is task scheduling at both serving small cells, and SCM sides for higher resource allocation efficiency. The second proposed heuristic is an iterative approach in which serving small cells compute their desired clusters, without considering the presence of other users, and send their cluster parameters to the SCM. SCM then checks for excess of resource allocation at any of the network small cells. SCM reports any load excess to serving small cells that re-distribute this load on less loaded small cells. In the final part of this thesis, we propose the concept of computation caching for edge cloud computing. With the aim of reducing the edge cloud computing latency and energy consumption, we propose caching popular computational tasks for preventing their re-execution. Our contribution here is two-fold: first, we propose a caching algorithm that is based on requests popularity, computation size, required computational capacity, and small cells connectivity. This algorithm identifies requests that, if cached and downloaded instead of being re-computed, will increase the computation caching energy and latency savings. Second, we propose a method for setting up a search small cells cluster for finding a cached copy of the requests computation. The clustering policy exploits the relationship between tasks popularity and their probability of being cached, in order to identify possible locations of the cached copy. The proposed method reduces the search cluster size while guaranteeing a minimum cache hit probability.

Résumé: Cette thèse porte sur le paradigme "Mobile Edge cloud" qui rapproche le cloud des utilisateurs mobiles et qui déploie une architecture de clouds locaux dans les terminaisons du réseau. Les utilisateurs mobiles peuvent désormais décharger leurs tâches de calcul pour qu'elles soient exécutées par les femto-cellules (FCs) dotées de capacités de calcul et de stockage. Nous proposons ainsi un concept de regroupement de FCs dans des clusters de calculs qui participeront aux calculs des tâches déchargées. A cet effet, nous proposons, dans un premier temps, un algorithme de décision de déportation de tâches vers le cloud, nommé SM-POD. Cet algorithme prend en compte les caractéristiques des tâches de calculs, des ressources de l'équipement mobile, et de la qualité des liens de transmission. SM-POD consiste en une série de classifications successives aboutissant à une décision de calcul local, ou de déportation de l'exécution dans le cloud. Dans un deuxième temps, nous abordons le problème de formation de clusters de calcul à mono-utilisateur et à utilisateurs multiples. Nous formulons le problème d'optimisation relatif qui considère l'allocation conjointe des ressources de calculs et de communication, et la distribution de la charge de calcul sur les FCs participant au cluster. Nous proposons également une stratégie d'éparpillement, dans laquelle l'efficacité énergétique du système est améliorée au prix de la latence de calcul. Dans le cas d'utilisateurs multiples, le problème d'optimisation d'allocation conjointe de ressources n'est pas convexe. Afin de le résoudre, nous proposons une reformulation convexe du problème équivalente à la première puis nous proposons deux algorithmes heuristiques dans le but d'avoir un algorithme de formation de cluster à complexité réduite. L'idée principale du premier est l'ordonnancement des tâches de calculs sur les FCs qui les re coivent. Les ressources de calculs sont ainsi allouées localement au niveau de la FC. Les tâches ne pouvant pas être exécutées sont, quant à elles, envoyées à une unité de contrôle (SCM) responsable de la formation des clusters de calculs et de leur exécution. Le second algorithme proposé est itératif et consiste en une formation de cluster au niveau des FCs ne tenant pas compte de la présence d'autres demandes de calculs dans le réseau. Les propositions de cluster sont envoyées au SCM qui évalue la distribution des charges sur les différentes FCs. Le SCM signale tout abus de charges pour que les FCs redistribuent leur excès dans des cellules moins chargées. Dans la dernière partie de la thèse, nous proposons un nouveau concept de mise en cache des calculs dans l'Edge cloud. Afin de réduire la latence et la consommation énergétique des clusters de calculs, nous proposons la mise en cache de calculs populaires pour empêcher leur réexécution. Ici, notre contribution est double : d'abord, nous proposons un algorithme de mise en cache basé, non seulement sur la popularité des tâches de calculs, mais aussi sur les tailles et les capacités de calculs demandés, et la connectivité des FCs dans le réseau. L'algorithme proposé identifie les tâches aboutissant à des économies d'énergie et de temps plus importantes lorsqu'elles sont téléchargées d'un cache au lieu d'être recalculées. Nous proposons ensuite d'exploiter la relation entre la popularité des tâches et la probabilité de leur mise en cache, pour localiser les emplacements potentiels de leurs copies. La méthode proposée est basée sur ces emplacements, et permet de former des clusters de recherche de taille réduite tout en garantissant de retrouver une copie en cache.