Alma Mater Studiorum – Università di Bologna

DOTTORATO DI RICERCA IN

Ingegneria elettronica, telecomunicazioni,
e tecnologie dell'informazione

Ciclo XXIX

Settore Concorsuale: 09\F2

Settore Scientifico Disciplinare: ING-INF\03

Energy and Delay Efficient Computation Offloading Solutions for Edge Computing

Presentata da: Arash Bozorgchenani

Coordinatore Dottorato	**Supervisore**
Prof. Alessandra Costanzo	**Prof. Daniele Tarchi**

Esame finale anno 2020

To my loving family . . .

for their encourage and assistance

. . . **To my respected Supervisor Daniele** . . .

for his continuous support and unlimited trust

. . . **To my colleagues and friends** . . .

who were my companions throughout PhD journey

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Arash Bozorgchenani
February 2020

Abstract

This thesis collects a selective set of outcomes of a PhD course in Electronics, Telecommunications, and Information Technologies Engineering and it is focused on designing techniques to optimize computational resources in different wireless communication environments. Due to the increase in number of mobile devices accessing the internet, and requesting for computational resources, Mobile Edge Computing (MEC) as a novel and distributed computational paradigm has emerged to address this high demand in 5G. In MEC, edge devices are able to share their resources to collaborate in terms of storage and computation. This technique has also been extended later to Fog Computing (FC) where the end users are able to share their resources even without exploiting the servers at the edge.

One of the techniques that enables computational sharing is known as computation offloading. Computation offloading brings a lot of advantages to the network edge, from lower communication latency in comparison with cloud computing, to lower energy consumption for computation. However, the communication among the devices should be managed such that the resources are exploited efficiently in order to meet the goals of energy consumption and delay minimization. To this aim, in this dissertation, computation offloading in different wireless environments with different number of users, network traffic, resource availability and devices' location are analyzed in order to optimize the resource allocation at the network edge. To better organize the presentation of all the activities that have been carried out in my PhD period, some of the research papers have been selected to be presented in this dissertation, however, list of all the activities can be found in the publication list. In this dissertation, the studies are classified in four main sections.

In the first section, an introduction on Mobile Cloud Computing (MCC), MEC and FC is given where each technology is introduced in chronological order. Later, the problem of computation offloading is defined and the challenges are introduced.

In the second section, two partial offloading techniques are proposed. While in the first one, centralized and distributed architectures are proposed for computation offloading in MEC, in the second work instead, an Evolutionary Algorithm (EA) for task offloading is proposed and the problem is formulated as a multi-objective optimization problem.

In the third section, the offloading problem is seen from a different perspective where the end users are able to harvest energy from either renewable sources of energy or through Wireless Power Transfer (WPT).

In the fourth section, the MEC in vehicular environments is studied. In one work a heuristic is introduced in order to perform the computation offloading in Internet of Vehicles (IoVs) and in the other a learning-based approach on the basis of bandit theory is proposed.

Publications

- Journal Papers

1. Bozorgchenani, Arash; Tarchi, Daniele, Corazza, Giovanni Emanuele, “Centralized and Distributed Architectures for Energy and Delay Efficient Fog Network based Edge Computing Services,” IEEE Transactions on Green Communications and Networking, Vol. 3 , Issue 1, pp 250-263, March 2019.
2. Bozorgchenani, Arash; Jahanshahi, Mohsen; Tarchi, Daniele, “Gateway selection and clustering in multi-interface wireless mesh networks considering network reliability and traffic,” Transactions on Emerging Telecommunications Technologies, Volume 29, Issue 3, March 2018.
3. Bozorgchenani, Arash; Mashhadi, Farshad; Tarchi, Daniele; Salinas, Sergio, “Multi-Objective Computation Sharing in Energy and Delay Constrained Mobile Edge Computing Environments,” IEEE Transactions on Mobile Computing, 2019 (under second round of revision).
4. Bozorgchenani, Arash, Sabato, Simone, Tarchi, Daniele, Roveri, Manuel, “Smart Energy Management in Fog Computing Networks,” IEEE Transactions on Green Communication and Networking, 2019 (under review).
5. Bozorgchenani, Arash, Maghsudi, Setareh, Hossain, Ekram, Tarchi, Daniele, Mobile Edge Computing Partial Offloading Technique: A Multi-armed Bandit Solution,” IEEE Transactions on Mobile Computing 2020 (under submission).
6. Mashhadi, Farshad, Salinas Monroy, Sergio, Bozorgchenani, Arash, Tarchi, Daniele, "Optimal Task Offloading in Mobile Edge Computing: A Deep-learning Based Approach", 2020 (under submission).

- Conference Papers

7. Bozorgchenani, Arash; Tarchi, Daniele, Corazza, Giovanni Emanuele, “Mobile Edge Computing Partial Offloading Techniques for Mobile Urban Scenarios,” IEEE Globecom December 9-13 December, Abu Dhabi, UAE, 2018.
8. Bozorgchenani, Arash; Tarchi, Daniele, Corazza, Giovanni Emanuele, “A Control and Data Plane Split Approach for Partial Offloading in Mobile Fog Networks,” IEEE Wireless Communications and Networking Conference, 15-18 April, Barcelona, 2018.

9. Bozorgchenani, Arash; Tarchi, Daniele, Corazza, Giovanni Emanuele, “An Energy and Delay-Efficient Partial Offloading Technique for Fog Computing Architectures,” IEEE Globecom, 4-8, Singapore, December 2017.

10. Bozorgchenani, Arash; Tarchi, Daniele, Corazza, Giovanni Emanuele, “An Energy-Aware Offloading Clustering Approach (EAOCA) in Fog Computing,” 14th International Symposium on Wireless Communication Systems, 28-31 August, Bologna, Italy, 2017.

11. Bozorgchenani, Arash; Tarchi, Daniele, Corazza, Giovanni Emanuele, “Computation Offloading Decision Bounds in SWIPT-based Fog Networks,” IEEE Globecom, 9-13 December, Waikoloa, HI, USA, 2019.

12. Moallemi, Raheleh, Bozorgchenani, Arash; Tarchi, Daniele, “An Evolutionary-based Algorithm for Smart-living Applications Placement in Fog Networks,” IEEE Globecom, 9-13 December, Waikoloa, HI, USA, 2019.

Acknowledgements

This PhD dissertation is the result of three years of my life dedicated to research. In these years not only I have broadened my knowledge, but also I have widened my personality from various perspectives. Throughout these years to carry out these research activities, I have received a great deal of support and assistance that I would like to acknowledge.

I would first like to thank my supervisor Prof. Daniele Tarchi for his continuous support during my PhD and all the research activities carried out. His guidance, knowledge and patience in every single research activity helped me a lot to become much more experienced.

My sincere thanks also goes to Prof. Ekram Hossain who provided me the opportunity to join the Wireless Communication, Networks and Services group at the University of Manitoba, Canada for a 6-month research activity. This period was indeed a unique and precious experience for me where I had the opportunity to learn from other fellows and opened a new professional window in my academic life.

During these three years of research I have had the opportunity to collaborate with different professors from different universities. I am deeply grateful to Prof. Sergio A. Salinas Monroy from Wichita state University, USA, Prof. Setareh Maghsudi from Technical University of Berlin, Germany, and Prof. Manuel Roveri from university of the Politecnico di Milano, Italy for their collaborations in different research activities. I have learnt a lot from all of them, thanks to their valuable guidance and insightful comments.

I am also thankful to Prof. Alessandro Vanelli Coralli and Prof. Alessandra Costanzo, the former and the current PhD coordinator of the program in Electronics, Telecommunications and Information Technologies Engineering, and also Prof. Giovanni Emanuele Corazza for the opportunity to carry out the research activities in the Digicomm research laboratory.

In addition, I would like to thank the reviewers of this thesis, Dr. Emilio Calvanese Strinati, the Smart Devices & Telecommunications Scientific and Innovation Director, CEA, France, and Prof. Dziong Zbigniew, from École de Technologie Supérieure (University of Quebec), Montreal, Canada.

I would finally like to acknowledge my colleagues Mohammad, Carlos and Xiao from University of Manitoba for the moments we created during the breaks and in the laboratory. Moreover, special thanks goes to my dearest friends Dr. Vahid Kouhdaragh, Farshad

Mashhadi and Mehdi Naseh for their wonderful support and valuable guidance in many aspects.

Table of contents

List of figures **xv**

List of tables **xvii**

Acronyms **xxii**

1 Introduction **1**

2 Partial Offloading Solutions **9**

2.1 State of the art on Partial Offloading 10

2.2 Partial Offloading Estimation in Centralized Vs. Distributed Architectures . 14

2.2.1 System Model 16

2.2.2 Centralized and Distributed Partial Offloading Approaches 22

2.2.3 Numerical Results 34

2.2.4 Summary 41

2.3 Multi-Objective Computation Sharing in MEC 43

2.3.1 System Model and Problem Formulation 44

2.3.2 An Evolutionary Algorithm for Task Offloading in Edge Computing 52

2.3.3 Numerical Results 59

2.3.4 Summary 67

3 Energy Harvesting Solutions **69**

3.1 State of the arts on Harvesting solutions on Edge Computing 70

3.2 SWIPT-based Computation Offloading 72

3.2.1 System Model and Problem Formulation 73

3.2.2 Offloading Decision-Making Approach 76

3.2.3 Numerical Results 81

3.2.4 Summary 85

3.3 Smart Energy Management in Fog Networks 86

3.3.1 System Setting . . . 87
3.3.2 Harvesting Solutions for Cluster based Fog Computing systems . . 87
3.3.3 The clustering algorithm . . . 98
3.3.4 Experimental Results . . . 99
3.3.5 Summary . . . 105

4 Vehicular Environment Solutions 109
4.1 State of the arts on Vehicular Environment Solutions . . . 110
4.2 D2D Control Plane With and Without Relaying . . . 113
4.2.1 System Model and Problem Formulation . . . 114
4.2.2 D2D assisted partial offloading . . . 117
4.2.3 Numerical Results . . . 120
4.2.4 Summary . . . 123
4.3 Multi Armed-Bandit Solution for Vehicular Edge Computing . . . 124
4.3.1 System Model . . . 126
4.3.2 Network Selection . . . 130
4.3.3 Problem Formulation for Network Selection . . . 131
4.3.4 Learning-based Solution for Network Selection . . . 131
4.3.5 BS Selection . . . 134
4.3.6 Problem Formulation for Packet loss Minimization . . . 136
4.3.7 Relaying Mechanism . . . 137
4.3.8 Simulation Results . . . 137
4.3.9 Impact of Bandit Parameters . . . 140
4.3.10 Comparison with other approaches . . . 141
4.3.11 Summary . . . 144

5 Conclusion and future works 149
5.1 Final Conclusions . . . 149
5.2 Directions for future works . . . 150

References 153

List of figures

1.1 Computation domain of Cloud Computing, MEC, and FC [1] 5

2.1 The considered two-layer Fog Network architecture 15
2.2 The 3-quantile function of the remaining energy level distribution 23
2.3 Centralized Architecture 25
2.4 Distributed Architecture 28
2.5 Task delay for offloaded and local portions. 35
2.6 Average Task Delay 39
2.7 FN Energy Consumption 40
2.8 Network Lifetime (20%) 42
2.9 An Architecture for Task Offloading in MEC 46
2.10 Task portion distribution 48
2.11 Crossover operation on parent solutions. 58
2.12 Mutation operation on the selected solution (Child 2) 58
2.13 Energy-based pareto selection 62
2.14 Delay-based pareto selection 62
2.15 Energy&Delay-based pareto selection 63
2.16 Impact of Number of Iterations and number of Edge Clients on latency and energy consumption 65
2.17 Impact of Number of Iterations on latency and Energy Consumptions of the Pareto fronts 66
2.18 Pareto fronts for different number of iterations and different number of Edge Clients. 66

3.1 Difference between the offloading and local thresholds (T_{off}-T_{loc}) 78
3.2 The offloading and local computation thresholds. 79
3.3 Threshold Areas for different bandwidth 82
3.4 Threshold Areas for different packet size 83

3.5 Average Task Delay 84
3.6 Node Lifetime 85
3.7 The energy harvesting curves for four different sensors in four different months 91
3.8 Power consumption levels for CM and CH in an interval 91
3.9 Remaining Energy profiles for CM and CH in an interval 92
3.10 Network Life time of the FNs going off with $\overline{E}_{bc} = 5000\,\mathrm{J}$ for the 3 approaches. 103
3.11 FNs off time in Scenario 1 with $\overline{E}_{bc} = 2000\,\mathrm{J}$ 104
3.12 FNs off time in Scenario 1 with $\overline{E}_{bc} = 1000\,\mathrm{J}$ 106
3.13 FNs off time in Scenario 2 with $\overline{E}_{bc} = 8000\,\mathrm{J}$ 107

4.1 Partial offloading mobile urban scenario. 115
4.2 Outage Probability of 11 fixed F-APs with high capacity 122
4.3 Outage Probability of 11 fixed F-APs with low capacity 123
4.4 Outage Probability of 5 fixed F-APs with low capacity 124
4.5 Outage Probability of 5 fixed F-APs with high capacity 125
4.6 Average task delay without relaying through 11 fixed F-APs 126
4.7 Average task delay with relaying through 11 fixed F-APs 127
4.8 Partial offloading mobile urban scenario. 128
4.9 Offloading cases considering locations of devices 135
4.10 Impact of τ on Average Regret 141
4.11 Impact of β on Average Regret 142
4.12 Impact of ξ on Average Regret 143
4.13 Average regret over rounds 144
4.14 Network Selection by agent 145
4.15 Average regret over intervals 145
4.16 Average packet waiting time 146
4.17 number of lost packets 147

List of tables

2.1 System Model Parameters Definition 17
2.2 Comparison of the centralized and distributed approaches 31
2.3 Parameters Definition for Partial Offloading 32
2.4 Simulation Parameters for the Partial Offloading Approach 36
2.5 Nomenclature (in order of appearance) 45
2.6 Simulation Parameters for NSGA2 approach 61
2.7 Task Parameters for NSGA2 Approach 61
2.8 Device Parameters for NSGA2 Approach 61
2.9 Impact of proposed initialization approach 61

3.1 Threshold differences in seconds with different bandwidth and packet size . 77
3.2 Simulation Parameters for SWIPT Approach 81
3.3 Harvesting Scenario Definition 100
3.4 Harvesting Simulation Parameters 101
3.5 Energy Harvesting Parameters 101

4.1 Simulation Parameters for Vehicular Environment 121
4.2 Simulation Parameters for the Bandit Approach 139
4.3 MAB Setting 139
4.4 Task Parameters for Bandit Approach 140

Acronyms

BS Base Station. 1, 3, 6, 12, 110–112, 124–130, 132, 134–139, 143, 144, 150, 151

CEC Computing Edge Clients. 53, 54, 59, 64

CFN Computing Fog Node. 14, 17, 19–26, 28, 29, 31, 32, 34, 37, 38, 41

CH Cluster Head. 88–90, 93–99, 101, 105

CM Cluster Member. 88–90, 93–99, 105

CMOP Constrained Multi-Objective Problem. 43, 51, 52, 149

D-UCB Discounted-Upper Confidence Bound. 137, 141, 142

D2D Device to Device. 10, 16, 43, 113, 117, 119–121, 123, 150

EA Evolutionary Algorithm. vii, 7, 14, 43, 44

EC Edge Client. 4, 43–45, 47–51, 53–55, 59–61, 63, 64, 67

EN Edge Node. 4, 43–45, 47, 48, 50, 54, 59, 60, 63, 67

ETSI European Telecommunication Standardization Institute. 3, 6, 110

EU Edge Unit. 126–136

F-AP Fog-Access Point. xvi, 10, 14–16, 18, 19, 21, 22, 24–26, 28, 29, 31, 34, 36–38, 41, 43, 72–76, 80, 81, 84, 85, 113–127

FC Fog Computing. vii, xv, 4–6, 10–12, 14, 16, 70, 73, 75, 85–87, 149, 152

FCH Fog Cluster Head. 23–26, 31, 37, 38, 41

FCM Fog Cluster Member. 24–26, 31, 38, 41

FLOPS Floating-Point Operation Per Second. 18, 36, 47, 50, 61, 81, 87, 101, 115, 121, 129

FN Fog Node. xv, xvi, 10–26, 28, 29, 31, 33, 34, 36–38, 40, 41, 43, 70–77, 79–90, 93–107, 113–123

HPFN High Power Fog Node. 22, 25, 26, 29, 31, 37, 38, 41

IaaS Infrastructure as a Service. 2, 4, 7

ICT Information and Communication Technology. 6, 110

IoT Internet of Things. 1, 4, 12, 70, 72, 102, 113, 149

IoV Internet of Vehicles. viii, 6, 149, 150

LPFN Low Power Fog Node. 22, 25, 26, 29, 31, 37, 38

LTE Long Term Evolution. 3, 70

MAB Multi-Armed Bandit. xvii, 112, 125, 131–134, 139

MCC Mobile Cloud Computing. vii, 1, 3, 4, 10

MEC Mobile Edge Computing. vii, viii, xv, 3–7, 11–14, 16, 43, 44, 46, 67, 71, 110, 111, 124, 149, 151, 152

MOEA Multi-Objective Evolutionary Algorithm. 52, 56

NH No Harvesting. 100, 102, 105

NSGA Non-dominating Sorting Genetic Algorithm. 59, 60, 62–64, 67, 149

PaaS Platform as a Service. 2, 4, 7

QoE Quality of Experience. 2, 3, 124

REC Requesting Edge Clients. 53, 54, 64

RF Radio Frequency. 73, 81

RFN Requesting Fog Nodes. 14, 17, 19–24, 26, 28, 29, 31–34, 37, 38

RL Reinforcement Learning. 71, 112, 151

SaaS Software as a Service. 2, 4, 7

SW-UCB Sliding Window- Upper Confidence Bound. 112, 133, 137, 140–143

SWIPT Simultaneous Wireless Information and Power Transfer. 73, 74, 76, 83–85, 149

UCB Upper Confidence Bound. 112, 132, 133, 137, 140–142

V2V Vehicle to Vehicle. 112

VEC Vehicular Edge Computing. 110–113, 124–126, 144, 150

VM Virtual Machine. 2

WPT Wireless Power Transfer. viii, 7, 70, 71, 73, 81

Chapter 1

Introduction

By the development of mobile Internet and the advances in the hardware of the smart devices, the last decades have been characterized by an increasing number of pervasive devices that have been deployed in the environments. These mobile devices provide us with a powerful platform enabling to support a wide range of applications, from environmental monitoring [2] and in-home automation [3] to Smart-cities [4] and Industry 4.0 [5].

These mobile applications that run on the mobile devices require some resources, e.g., processor, battery and storage. On the other hand, these huge amount of devices with their limited resources and capabilities, have a large demand for resources that can not be responded by themselves. Consequently, the data traffic generated by both mobile devices or other smart devices, e.g., Internet of Things (IoT) devices, require other resources for storage and computation. This conflict between the resource hungry applications and the limited capability of the devices has brought many challenges for energy-efficient data processing [6]. Therefore, computing support is crucial for mobile communications networks by the increase in the demand from the users.

MCC, that was first introduced in 2007, is a network architecture that has the potential to address the storage and computation challenges of the mobile users. MCC can improve the performance of the mobile applications by enabling the mobile devices to migrate their computational tasks to the infrastructure-based cloud servers [6]. Later, the concept of cloud computing was extended to cloudlet. In 2009, Satyanarayanan et al. in [7] introduced the term cloudlet, where they proposed a two-tier architecture. The first tier is known as cloud with high latency, and the second as cloudlets with lower latency. The latter is a distributed internet architecture and composed of widely dispersed internet infrastructure components. Their computational and storage resources can be leveraged by nearby devices. As a result, the cloud providers do not necessarily need to be business-level providers, while any resource-rich device having internet connectivity, such as a vehicular Base Station (BS),

has the potentiality to provide cloud-like services to the mobile devices via different wireless connections [8].

Cloud computing can be seen as the use of computing hardware and virtualization technologies for forming a shared infrastructure enabling web-based value added services [9]. Among the many types of cloud computing services delivered internally or by third party service providers, the most common are:

Infrastructure as a Service IaaS can be defined as the use of servers, storage, and virtualization to enable utility like services for users. The infrastructure is composed of the facility, communication networks, physical compute nodes, and the pool of virtualized computing resources that are managed by a service provider. The service aspect consists of components within the user's domain of control and would include the Virtual Machines (VMs) and their operating systems, storage, and their management [10]. IaaS provides users with a web-based service enabling them to create, destroy, and manage VMs and storage. Moreover, it alleviates the responsibility of users in managing the physical and virtualized infrastructure, while still having control over the operating system, configuration, and software running on the VMs.

Platform as a Service PaaS providers offer access to application programming interfaces, programming languages and development middleware allowing subscribers to develop custom applications without installing or configuring the development environment. PaaS, which is built on top of IaaS, features many of the same benefits, such as utility computing, hardware virtualization, dynamic resource allocation, and low investment costs. Using the tools included with the cloud platform, developers are able to build applications and services taking advantage of virtualized hardware, data redundancy, and high availability. Some of the examples of PaaS providers are Google App Engine, Microsoft Azure, and SalesForce.com [11].

Software as a Service SaaS gives subscribed users access to software or services that reside in the cloud and not on the user's device. The consumer of a SaaS application only requires thin client software such as a web browser to access the cloud-hosted application. This reduces the burdens of maintenance, support, and operations by having the application run on computers belonging to the vendor. Hotmail, Gmail, and Google Apps are examples of popular SaaS applications.

Offloading computational load from mobile devices to other resources is a tool for delivering high Quality of Experience (QoE) for the mobile devices. Computation offloading

enables the mobile users to perform sophisticated computations of their application with the assistance of other resource-rich devices [12]. Among several parameters characterizing the effectiveness of an energy-limited Network, the network lifetime, i.e., the time span a certain amount of nodes or the whole network is stopping to work due to energy shortage, seems to be one of the most important [13]. Computation offloading allows the devices to consume less and as a result, prolong the battery lifetime of their own devices. Moreover, in the presence of different nodes, a task can be further divided into some portions and each one allocated to a different node. This technique is called partial offloading [14].

Nevertheless, there exists still challenges in computation offloading to the mobile cloud environment (MCC or cloudlet). Offloading invokes extra transmission and reception energy consumption from the mobile devices. Moreover, this communication brings up additional delay to the task processing, which does not make it suitable for real-time applications, e.g., self-driving cars or industrial IoT scenarios.

On the other hand, the daily increase in number of devices demanding internet access and the huge traffic of data to be managed has brought a huge bottleneck to the networks. Technological innovations in life, entertainment, transportation, medical and other industries have constantly demanded higher QoE in terms of latency and energy consumption, which has motivated the continuous innovation and upgrading of network technology to 5G [15]. The development of mobile communication technology in 5G, has made the means of information interaction more efficient and convenient. To guarantee the high QoE for users new technologies other than cloud computing were needed.

5G is not only a new access technology but also a user-centric network concept aiming to address the application requirements of all people in the connected world. 5G is expected to bring revolutionary impacts on mobile communications. More importantly, it will be built upon the current wireless technologies including Long Term Evolution (LTE), global system for mobile communications, WiFi, and etc.. However, to provide a platform for the existing technologies to coexist, one crucial thing is specifying the basic building blocks of 5G. One of the key building blocks for 5G is Mobile Edge Computing (MEC) [16].

MEC is a promising computing platform that places computing resources near to the end-users, e.g., MEC servers can be co-located with cellular BSs. Compared to the existing MCC infrastructure [17, 18], where computing resources are centralized in data centers far away from end-users, MEC offers a comparable amount of computing resources but with lower communication delays. The increasing interest in MEC is also evident by looking at the standardization effort in European Telecommunication Standardization Institute (ETSI), where some possible use cases have been identified: active device location tracking, augmented reality content delivery, video analytics, radio access network aware content

optimization, distributed content and DNS caching and application-aware performance optimization [19].

This new distributed computing paradigm can support delay-sensitive applications with high computing requirements that the current MCC architecture is unable to handle. In particular, in MEC networks, resource-poor end-user devices, called Edge Clients (ECs), can reduce their computing time by offloading their tasks to nearby resource-rich edge servers, called Edge Nodes (ENs), which are able to complete the task computations much faster. ECs can also reduce their computing time by offloading their tasks to other ECs that are willing to share their idle resources with them.

MEC has been successfully used in several industries to handle applications with low-latency and high-throughput requirements. In Industry 4.0, manufacturers equip their workers with augmented reality headsets that require significant computing resources to render three-dimensional images [20, 21]. In smart healthcare, physiological data collected by IoT devices needs to be quickly analyzed to provide timely diagnoses [22, 23].The oil and gas industry uses MEC to optimize its operations [24].

To further increase the amount of computing resources available to end-users, it is possible to compliment MEC's edge resources with the computing capabilities of the end-users' devices, also known as Fog Computing (FC) [25, 1]. In 2012, Cisco introduced this new computational paradigm to overcome the issues in MEC. The novel approach brought by FC is the possibility to move computation as close as possible to where data are generated, i.e., the IoT devices, hence providing a distributed framework of computing resources. This allows to acquire and process information locally, hence supporting prompt reactions/decisions and making FC units autonomous without requiring a permanent connection with the Cloud computing platform. For these reasons, FC enables pervasive processing of different applications and technological scenarios. This would bring several advantages, from the exploitation of more powerful nodes, up to load balancing and energy savings, while introducing several challenges in terms of management complexity, energy distribution, nodes coordination, just to name a few [26, 27].

In order to have a better view of the differences between the computation domain of the technologies Fig. 1.1 is presented. In traditional cloud computing scheme, the computation is offloaded to the centralized cloud. Similarly in MCC the offloading is to the cloud while the users are mobile. Unlike Edge computing, and similar to MEC and MCC, FC can extend cloud based services like Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS), etc. to the network edge. However, FC can also be extended to the core network as well, and this is why it is more potential for IoT comparing to the other computing technologies [1].

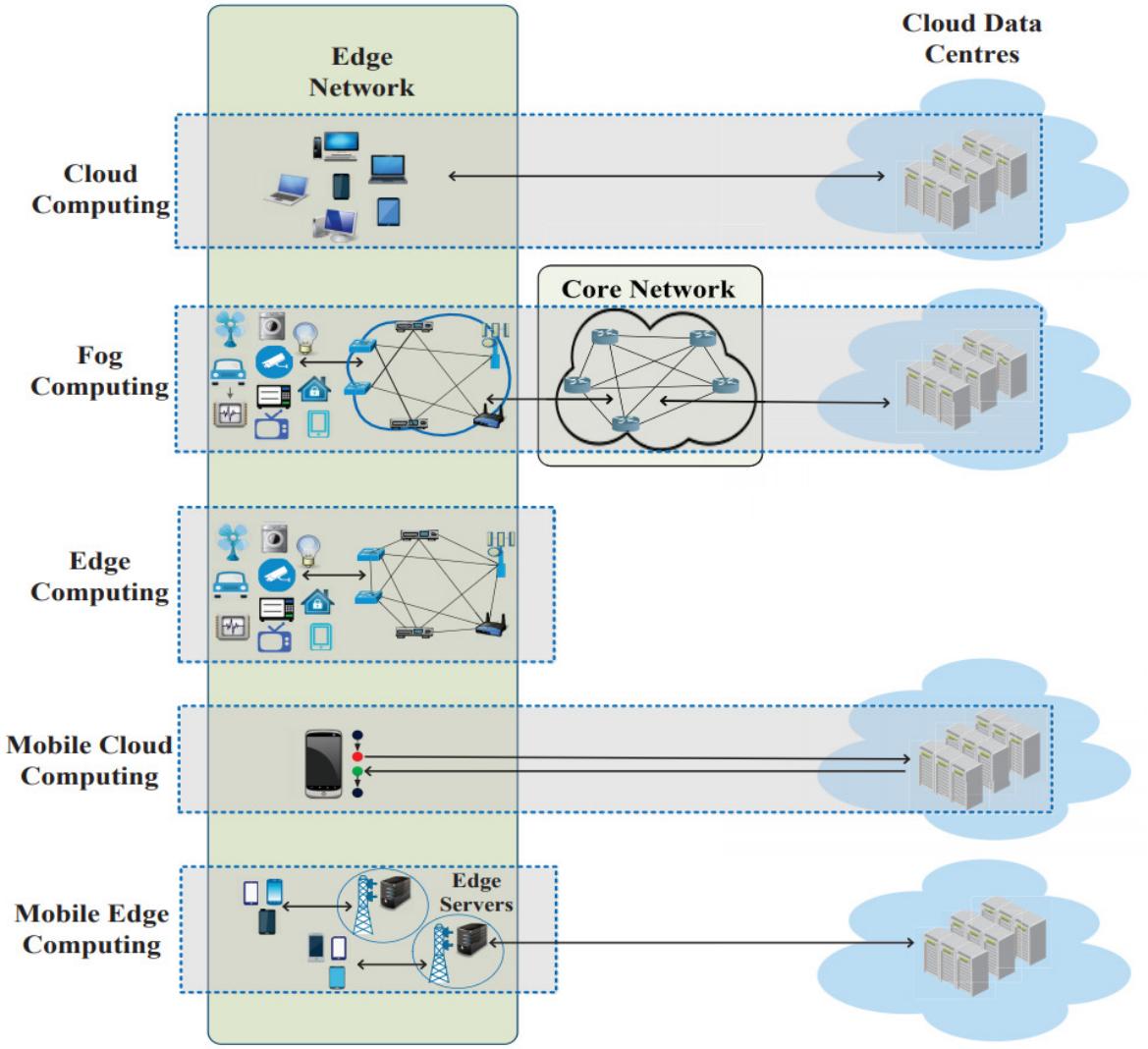


Fig. 1.1 Computation domain of Cloud Computing, MEC, and FC [1]

Due to their high potential, both MEC and Fog computing have attracted broad interest from the research community and from international organizations, such as the ETSI and IEEE, that have started standardization efforts for these technologies [19, 28]. Among many use cases that MEC and FC have, we can name the following:

Connected vehicles Vehicles have been evolving since the second industrial revolution and their role in the modern life is imperative. With the rapid technological advancements in Information and Communication Technologies (ICTs), vehicles are equipped with wireless communication capabilities for both intra-vehicle and inter-vehicle communications to support plenty of applications such as road safety, smart transportation and location-dependent services. Connected vehicles refer to the vehicles with wireless connectivity that can communicate with their internal and external environments, and is considered as the basis of the emerging Internet of Vehicles (IoV). IoV is a dynamic mobile communication system that features gathering, sharing, processing, computing, and secure release of information and enables the evolution to next generation intelligent transportation systems [29]. Further, the development and deployment of fully connected vehicles requires a combination of various emerging technologies, among which IoT, Edge and FC and networking are of great importance.

Smart Grid ICT, is composed by a huge number of smart devices that may be deployed in different geographical regions for data collection. For instance, smart meters and sensors might be deployed across different areas to collect electricity consumption data from different homes and to monitor household activities. The collected data from different homes can be transferred to the nearest BS where it can be used to estimate demand and supply so that a unified policy can be designed for smooth operation of power scheduling and redistribution in the ICT environment [30]. As the amount of data to be managed and analyzed increase, the need for a distributed computation paradigm rises. This computational processing at the edge can sharply reduces the delay and energy consumption and boosts the performance.

Smart cities Smart cities are seen as the paradigm where the wireless communications improve urban services and quality of life for citizens and visitors. The smart city scenario is composed by several elements: the wireless infrastructure, the user devices, sensing devices, machine devices, access points, and cloud/edge infrastructures. Moreover, the citizens of a smart city request for some services by their smart devices. To deliver the requested services, a lot of data are needed to be exchanged and elaborated [31]. On one side this ever-growing demand for services from citizens and institutions, and on the other side the plan to improve

the quality of life of the communities, has encouraged the concept of heterogeneous wireless communication systems and has extended the concept of cloud architectures for providing IaaS, PaaS, and SaaS, towards the integration with novel communications approaches such as edge networking [32, 33].

However, there are some challenges that are brought with offloading, one of which is the intermittent wireless connectivity between the mobile devices and the cloud or edge infrastructure due to the mobility of the devices [8]. Besides, the rejections of the users' requests for offloading due to the capacity limitations of the rich-resource nodes is the other issue. Furthermore, due to the long distance between the nodes, low computational resources, or long waiting time in the queue, local computation might be a better option. Thus, offloading is not always the best choice for the users, while they have to make a decision on whether to process the task locally or offload to an available resource.

To this aim, this has motivated me to do research on this technology, and more specifically on computation offloading in Edge/Fog networks. In this thesis, best effort is made to formulate the computation offloading problem in different technological scenarios and propose heuristic, meta-heuristic and optimization-based solutions to tackle the challenges in resource allocation and offloading decision in Edge/Fog networks.

To have a better organization of all the research activities that have been carried out in my PhD period, in this dissertation a selection of the works among which some have been done in cooperation with other universities is presented. The list of all research activities can be found in the publication section. The studies have been put into three key areas. In the first part, computation offloading in static environments is investigated where both centralized and distributed architectures have been studied. Moreover, clustering schemes have been introduced and the problem has been formulated as a multi-objective optimization, for which Evolutionary Algorithms (EAs) have been proposed. For the second part of the studies, energy harvesting solutions have been proposed, where we have investigated the effect of harvesting energy from renewable energy sources and also Wireless Power Transfer (WPT) technology on computation offloading decisions. In the end, MEC in vehicular environments has been studied where heuristics and machine learning solutions have been proposed.

Chapter 2

Partial Offloading Solutions

The content of the following chapter was extracted from publications [1], [3] [9], [10] in the publications list.

Within the FC scenario, and the related Fog networking scenarios that aim at considering the communication and networking challenges introduced by the Fog architecture, the aim of this chapter is that of proposing different solutions for partial computation offloading in Edge/Fog networks.

Within the scenarios in this chapter, we envisage to consider a two-layer architecture composed of Fog Nodes (FNs), battery operated nodes with lower computing capabilities, and Fog-Access Point (F-APs), fixed nodes usually connected to the electrical power network with higher computational capabilities. These two types of nodes are logically organized into two interconnecting layers; to this aim two communication paradigms are usually considered: Device to Device (D2D) communication among FNs, and infrastructure communications between FNs and F-APs [34].

Due to the limited FNs capabilities, the FC requests cannot be completely fulfilled with a requested target (e.g., delay or energy consumption). To overcome this problem, a joint exploitation of both FNs and F-APs is here considered. In D2D communications, FNs are able to share their resources with the neighboring FNs, while in infrastructure communications, requests are sent from FNs to F-APs for being computed. This allows to exploit the advantages of both when offloading a task computation.

Although F-APs can complete computing tasks faster than the FNs, offloading tasks to a remote node comes with additional energy consumption and time delays, which can, in some cases, exceed those of computing the task locally. In particular, when FNs offload their tasks, they must first spend energy to transmit their data to the remote node, and then wait for the data to be uploaded, and for the task results to be downloaded. Thus, the FNs face a trade-off between the overall task completion time and their energy consumption when deciding whether to offload their tasks to other nodes, which is the target of studies in this chapter.

2.1 State of the art on Partial Offloading

Although FC and networking has been recently introduced, the research community is very active in this field. In this section, we briefly summarize the existing approaches for task partial offloading.

The task offloading problem has been formulated in [35] as a joint radio and computational resources problem. From the architectural point of view, the network in [36] is broken down into several layers in a way that some cloudlets for Mobile Cloud Computing (MCC) are considered. To enhance network capacity and offloading probability, in this work a D2D-based heterogeneous MCC network is proposed. Clustering in edge networking has

also been proposed in some works. In [37] clustering was performed among the access points considering channel and caching status. A clustering algorithm was also proposed in [12] for the radio access points dealing with joint computation and communication resource allocation inside the cluster. Mobile devices send their tasks to a gateway, which acts like an access point, and the gateways in the same cloudlet send the tasks to a master device. The tasks which can not be responded by the master device are sent to the cloud. In the system model presented in [38], some of the FNs are able to act as a relay to help the interaction among FNs which are not in their coverage.

There are mainly two design objectives when offloading tasks to remote nodes. First, the time to complete the task should be minimized. Second, the energy consumption of mobile nodes, which are energy constrained, should also be minimized.

Some existing works exclusively focus on minimizing the task completion time. To target a fog server from the user point of view, [39] considers both communication and computing delays. However, an assumption made in [39] was that all the users can access all the fog servers. In [40], the authors have studied the multi-user computation partitioning problem between mobile devices and cloud. They have proposed an offline heuristic and considered a large-scale mobile cloud application with the aim of minimizing the average completion time. Task completion time has also been considered by Jia et al. [41] that propose an online task offloading algorithm for mobile devices that minimizes the task completion time at the cloud. This work also minimize the task completion latency in cloud computing while considering the finite energy resources at the mobile devices as optimization constraints. Chen et al. [42] minimize the task computing time by modeling the problem as a Semi-Markovian Decision Process. Liu et al. [43], consider a MEC network and a task scheduling policy that minimizes the task computing delays.

On the other hand, some works have considered the energy consumption as the goal of their optimization having either homogeneous or heterogeneous devices in their network. The authors in [44] study the impact of offloading on reducing the energy consumption by focusing on an intensive communication scenario. An Energy-Efficient Computation Offloading (EECO) algorithm is proposed in [6] based on three main phases: classifying the nodes considering their energy and cost feature, prioritizing them by giving a higher offloading priority to the nodes which cannot meet the processing latency constraint, and the radio resource allocation of the nodes considering the priority. The proposed EECO algorithm allows to decrease the energy consumption by up to 15% in comparison with computation without offloading. The authors in [45] have proposed a matching game approach to solve the problem of resource allocation of the cached content in a heterogeneous FC environments. The aim of the work is minimizing the energy consumption for the content access for which

they have considered the energy consumption of the requesting node, embedded computing on the node, baseband processing at the edge, radio interface, transmission through core network and internet, and processing in the cloud. The authors in [46] have introduced an energy efficient FC framework in homogeneous fog networks and proposed a low complexity maximal energy efficient task scheduling algorithm to derive the optimal scheduling decision for a task node. In [47] a node discovery approach in IoT-fog environment was proposed. The authors mainly worked on the impact of the dynamicity of the advertiser nodes on device discovery success, which was proved to be 100%, and also sustainability of the battery-powered IoT nodes.

In [48] the challenges of energy efficiently multimedia sensing as a service at the cloud edges IoT was investigated. The authors proposed a resource allocation approach to achieve optimal multimedia transmission quality in addition to guaranteeing wireless communication energy efficiency. Chen et al. [49] use an online peer offloading framework based on Lyapunov optimization to maximize the long-term network performance while keeping the energy consumption of small-cell BSs in a MEC network low. Zhang et al. [50] develop a scheme that optimally selects between local computation or offloading to the cloud to minimize the energy consumption. In [51], You et al. propose optimal cloud outsourcing mechanisms for mobile devices capable of transferring and harvesting power. The authors in [52] formulate the problem of minimizing energy consumption in a collaborative task execution between a mobile device and a cloud as a constrained shortest path problem. In [53], Mahmoodi et al. propose an optimal computation offloading schedule of a mobile application to the cloud subject to the saved energy at the mobile users. Cao et al. [54] present a computation and communication cooperation scheme for MEC systems. The authors considered multi-hop task offloading where intermediate nodes between the offloading nodes and the edge servers can participate in task execution. [55–57] have also considered energy consumption as the optimization goal. Song et al. [58] propose a pricing mechanism and a Lyapunov optimization scheme that can guarantees fair energy consumption between mobile devices.

However, computation offloading affects both FN energy consumption and task delay. Some of the existing works have considered both metrics. The authors in [59] proposed energy-efficient offloading policies for transcoding tasks in a mobile cloud system. With the objective of minimizing the energy consumption while meeting the latency constraint, they introduced an online offloading algorithm which decides whether the task should be offloaded to the cloud or executed locally. Task processing in [60] is based on a decision of either local processing or total offloading. The authors aimed at minimizing the local execution energy consumption for applications with strict deadline constraints. Authors in

[61] study the problem of network energy minimization while satisfying applications' delay requirement in cloud radio access networks. A joint optimization of beamforming design and power allocation with a decision making strategy is considered. A heuristic offloading decision algorithm is proposed in [62] with the aim of maximizing system utility which considers task completion time and the FN energy consumption. However, in [62] only a single MEC server is considered. Energy consumption and latency have also been targeted in [63] for an offloading approach. In this work, the authors targeted energy consumption and response time for the offloading scenario to the centralized cloud.

In [64], Kao et al. propose a task offloading algorithm that balances energy consumption costs and latency in latency-sensitive applications. Wang et al. in [65] consider mobile device energy consumption, and application execution latency but formulate separate minimization problems. They first obtain an optimal solution for the energy consumption minimization problem and then a locally optimal solution for the latency minimization problem. Hong et al. [66] consider both energy consumption and latency by formulating an aggregate objective function. Jiang et al. [67] propose a Lyapunov optimization approach for cloud offloading scheduling, where multiple applications are running on multi-core CPU mobile devices. Dinh et al. [68] jointly minimize task execution latency and energy consumption of mobile devices by optimally choosing task offloading decisions and the CPU-cycle frequency of the mobile devices. In [69], Wang et al. propose a framework that optimally offloads computation offloading and assigns wireless resources. The MEC server first makes the offloading decision based on the estimated overhead for mobile devices and itself. Then, by solving a graph coloring problem, the framework assigns wireless channels. The authors in [70] propose an energy-aware offloading approach that minimizes the weighted sum of energy consumption and latency by optimally choosing the computation offloading decision, local computation frequency scheduling, and allocation of power and wireless channels. The work in [27] studies a fog-based mobile cloud computing, where mobile devices are modeled with queues, fog nodes act as access points, and a central cloud is available for computations. The authors model the energy consumption and task delay offloading model as a multi-objective optimization that minimizes energy consumption, latency and cloud payments. To solve this problem, the authors relax the multi-objective optimization into a single-objective problem and solve it using the interior point method. Although these works aim to simultaneously optimize task completion delay and mobile device energy consumption, their proposed solution methods form a single objective, e.g., [27], or finally consider the objectives separately, which yields only one trade-off between the two objectives and ignores a large part of the trade-off space. There have also been works targeting two

objectives of latency and power consumption [71], however, working on a different network architecture.

To better explore the trade-off space, researchers have considered meta-heuristic solution approaches. In [72], Midya et al. propose an algorithm that combines a Genetic Algorithm (GA) with adaptive particle swarm optimization to offload tasks in a vehicular cloud. Cui et al. [73] employ an EA to minimize both objectives in a MEC network where mobile devices offload their tasks to the edge servers. However, these works only consider mobile devices that offload their tasks to either the cloud or edge servers while ignoring the possibility of offloading tasks to other mobile devices with idle computing and energy resources.

It can be seen that a lot of works, have seen the problem of task offloading as either performing a local computation or offloading the whole task. While, task sharing in our works benefit from the parallel computation among the devices in an MEC environment.

On the other hand, some works have proposed mathematical schemes or optimal results in a small-scale network with few number of nodes, however, the applicability of such proposals might be in doubt in large-scale networks.

In this chapter, we propose two solutions for partial offloading in a Fog network considering both energy consumption and delay. While in the first work, we propose a heuristic algorithm for estimating the offloading portions to available devices on two layers in both a centralized and distributed architecture. In the second work, we propose a meta-heuristic based approach for the offloading decisions.

2.2 Partial Offloading Estimation in Centralized Vs. Distributed Architectures

In this work we have considered a Fog architectural model as shown in Fig. 2.1. It is possible to see the FNs and the F-APs that interact in forming the FC infrastructure. The FNs in this figure are divided into two types: the Requesting Fog Nodess (RFNs), the devices offloading the computational tasks, and the Computing Fog Nodes (CFNs), the devices accepting tasks to be computed from the RFNs.

The goal of the work is the optimal distribution of the computational effort among the nodes by jointly minimizing the overall task processing delay and the energy consumption while maximizing the network lifetime [13]. To this aim, the optimization has been carried out by resorting to two different approaches, a distributed and a centralized. While in the distributed each RFN is selecting autonomously the nodes to be used for computing, in the centralized approach we foresee to optimize the system by centrally driving the offloading requests of the RFNs.

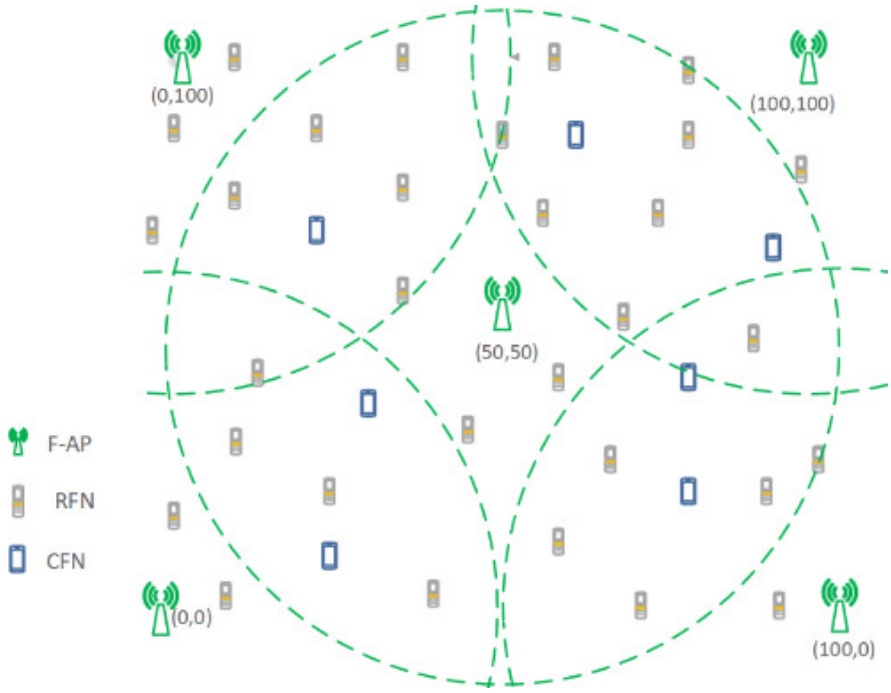


Fig. 2.1 The considered two-layer Fog Network architecture

We have considered both centralized and distributed architectures for the partial offloading problem. Moreover, we have introduced two approaches working on both FN and F-AP layers considering the FN energy consumption and the task processing delay for a sub-optimal solution to the partial offloading problem. The proposed approaches estimate the amount to be offloaded to the available nodes in both layers such that average task delay, FN energy consumption and network lifetime are optimized. This work mainly features the following characteristics:

1. Network design: Both F-APs in the second layer and FNs in the first layer are available for partial computation offloading.

2. Architecture comparison: One of the major goals of this work is that of comparing the centralized and the distributed architectures.

3. Partial offloading estimation: By the availability of all types of nodes for computation offloading in the proposed architectures, a new offloading estimation is proposed. For the estimation of the portion to be offloaded to the available nodes, we have considered data rate and the computational power of the available nodes.

2.2.1 System Model

In this work a two-layer architecture for FC is considered. On one hand $\mathcal{U} = \{u_1, \ldots, u_i, \ldots, u_N\}$ represents the set of FNs in the first layer. The FNs, characterized by limited computational capabilities, are battery powered and are the sources of the computational requests in the system. FNs are able to communicate among themselves for enabling the direct offloading through direct link technologies (e.g., D2D or WiFi-Direct). On the other hand, the second layer is composed by F-APs, whose set is indicated as $\mathcal{A} = \{a_1, \ldots, a_m, \ldots, a_M\}$, characterized by a higher computation capability. The F-APs are plugged to the electrical network, resulting in a virtual unlimited energy, and could also be used for the connection of the FNs with the centralized cloud. F-APs can be exploited by the FNs for computation offloading. In our system we focus on both layers by limiting the offloading requests to these layers and avoiding any offloading requests to outer clouds. The F-APs act as MEC resources, providing the ability of running multiple computations at the same time [6]. In the following, the FNs are considered to be steady and able to offload their tasks to the neighboring FNs or to the F-APs.

The goal of this work is to optimally estimate the task portion to be offloaded by each FN having some tasks to be computed in order to jointly minimize the energy consumption and the task processing delay. For pursuing such optimization we resort to a partial offloading technique that allows to select the amount of data to be offloaded to each of the possible candidates among the available FNs and F-APs, while the remaining can be processed locally. For the sake of readability, in Tab. 2.1 the parameters definitions to be used in the following equations are listed.

Each FN can be considered in one of four possible states during its life: transmitting, receiving, computing or idle, i.e., $\mathcal{S} = \{tx, rx, com, id\}$. The transmitting and receiving states refer to the interaction with other FNs or F-APs and the computing state refers to the computation performed in the FN itself (either for a local task or for an offloaded task); the idle state is considered the remaining time. In the rest of the this work, nodes refer to both FNs or the F-APs, unless otherwise stated. The overall energy consumed by the generic ith FN can be defined as:

$$E^i_{FN} = E^i_{tx} + E^i_{rx} + E^i_{com} + E^i_{id} \tag{2.1}$$

where E^i_{tx}, E^i_{rx} and E^i_{com} are, respectively, the energy consumed during transmission, reception and computation states and E^i_{id} is the energy the ith FN spends during its idle state. The energy spent by the ith FN in a certain state s can be defined as:

$$E^i_s = P^i_s T^i_s, \qquad s \in \mathcal{S} \tag{2.2}$$

Table 2.1 System Model Parameters Definition

Parameter	**Definition**
E_s^i	The energy consumption of the ith FN in state s
$E_r^i(t)$	The remaining energy of the ith FN at time instant t
P_s^i	The consumed power of the ith FN in state s
B_{ij}	The bandwidth of the link between two nodes i and j
h_{ij}	The channel coefficient between two nodes i and j
P_{N_j}	The noise power at receiver side
T_s^l	The time spent by the ith FN in state s
$T_{w_j}^l$	The waiting time of the lth task in the queue of the jth node
r_{ij}	Data rate of the link between the ith and jth node
L_{s_l}	Size of the lth task
L_{r_l}	Size of the lth task result
O_l	Number of operations to process the lth task
η_{comp_j}	The computational power of the jth node
$\alpha_{loc,i}^l$	Local portion of the lth task of the ith node
$\alpha_{off,i}^l$	Offloading portion of the lth task of the ith node
$\alpha_{off,ij}^l$	Offloading portion of the lth task of the ith node to the jth node
E_{CFN}^j	Energy consumption of the jth CFN
E_{RFN}^i	Energy consumption of the ith RFN
$T_{loc,i}^l$	Local computation time for the lth task of the ith node
$T_{off,ij}^l$	Offloading time of the lth task of the ith node to the jth node
D_i^l	Total delay of the lth task of the ith node

where P_s^i represents the power and T_s^i the time spent by the ith FN in the state s.

We suppose that the initial energy of the ith node is $E_r^i(0)$. All the FNs consume a certain amount of energy when they transmit, receive or compute tasks or even when they are idle. Therefore, by a certain time t, the ith FN has consumed $E_c^i(t)$ Joule of energy. Thus, the remaining energy of the ith FN at certain time instant t can be calculated as:

$$E_r^i(t) = E_r^i(0) - E_c^i(t) \tag{2.3}$$

where,

$$E_c^i(t) = \int_0^t E_{FN}^i(\tau) d\tau \tag{2.4}$$

In general, the time spent by the jth node, whether it is an FN or an F-AP, for processing the lth task can be defined as:

$$T_{comp_j}^l = \frac{O_l}{\eta_{comp_j}} \tag{2.5}$$

where O_l represents the number of processing operations related to the lth task and η_{comp_j} is the Floating-Point Operation Per Second(FLOPS) depending on the CPU of the jth processing node, which can be an FN or an F-AP.

In case of offloading, the lth task should be transmitted; hence, the transmission time from the ith FN to the jth node for the lth task can be written as:

$$T_{tx,ij}^l = \frac{L_{s_l}}{r_{ij}} \tag{2.6}$$

where L_{s_l} is the size of the lth task offloaded by the ith FN and r_{ij} is the data rate of the link between the ith FN and the jth node. Following this, the result of the processed task should be sent back from the jth node to the ith FN, leading to a reception time defined as:

$$T_{rx,ij}^l = \frac{L_{r_l}}{r_{ij}} \tag{2.7}$$

where L_{r_l} is the size of the result of the requested task sent back to the requesting FN, by supposing a symmetric channel in terms of data rate between the ith FN and the jth node. By considering the Shannon capacity formula, the data rate between the ith FN and the jth node can be written as:

$$r_{ij} = B_{ij} \log_2 \left(1 + \frac{|h_{ij}|^2 P_{tx}^i}{P_{N_j}}\right) \tag{2.8}$$

where B_{ij} is the bandwidth of the link, P_{tx}^{i} represents the transmission power of the ith FN, h_{ij} is the channel coefficient between the ith FN and the jth node and P_{N_j} is the noise power at the receiver side, defined as $P_{N_j} = N_T B_{ij}$.

In our system we are supposing that the F-APs are able to process the tasks received from other FNs, while the FNs can process both tasks generated by themselves or received from other nearby FNs. To this aim, in reference to the role they are having, in the following we will refer to the RFNs as those FNs asking other nodes to process a task, and CFNs as those FNs computing a task on behalf of other FNs.

Each CFN and F-AP in our work is supposed to have a queue holding the tasks of the RFNs to be processed. The waiting time of the lth task at the jth node can be defined as:

$$T_{w_j}^{l}(p) = \sum_{\pi=1}^{p-1} T_{comp_j}^{\pi} \tag{2.9}$$

where p is the number of tasks already in the queue of the jth node at a given time instant.

The concept behind partial offloading is to delegate only a portion of the computation load to another node. This allows to have a higher flexibility and optimize the energy consumption and the time spent for processing the tasks. We define $\alpha_{loc,i}^{l}$ as the portion of the lth task that can be processed locally by the ith RFN generating that task, and $\alpha_{off,i}^{l}$ as the amount that can be offloaded by the ith RFN, where $\alpha_{off,i}^{l} = 1 - \alpha_{loc,i}^{l}$. We are considering that the offloaded portion can be further split among the available nodes. In this case, the offloaded portion can be written as:

$$\alpha_{off,i}^{l} = \sum_{j \in \mathcal{N}(i)} \alpha_{off,ij}^{l} \tag{2.10}$$

where $\mathcal{N}(i)$ is the set of the neighbor nodes available for accepting the offloaded portion, and $\alpha_{off,ij}^{l}$ is the portion offloaded by the ith RFN to the jth node.

The time required for offloading the portion of a task from the ith RFN to the jth node can be written as the sum of the time for offloading the portion of the task, the time the task should wait in the jth node queue, the time for computing that task and the time needed for having the result back:

$$T_{off,ij}^{l}(\alpha_{off,ij}^{l}) = \alpha_{off,ij}^{l} T_{tx,ij}^{l} + T_{w_j}^{l} + \alpha_{off,ij}^{l} T_{comp_j}^{l} + \alpha_{off,ij}^{l} T_{rx,ij}^{l} \tag{2.11}$$

while the time for local computation, can be defined as the time needed for computing the remaining portion of the task:

$$T_{loc,i}^{l}(\alpha_{off,i}^{l}) = \alpha_{loc,i}^{l} T_{comp_i}^{l} = (1 - \alpha_{off,i}^{l}) T_{comp_i}^{l} \tag{2.12}$$

By assuming that the local and the offloaded portions can be performed in parallel, the total delay for processing a task can be rewritten as the maximum of all the offloading times and the local time, i.e.,

$$D_i^l(\alpha_{off,i}^l) = \max_{\forall j \in \mathcal{N}(i)} \left\{ T_{off,ij}^l(\alpha_{off,ij}^l), T_{loc,i}^l(\alpha_{off,i}^l) \right\} \tag{2.13}$$

On the other hand, the energy consumption of the jth CFN, in case of partial offloading, could be rewritten as:

$$E_{CFN}^j = \alpha_{off,ij}^l (E_{rx}^{ji} + E_{com}^j + E_{tx}^{ji}) + E_{id}^j \tag{2.14}$$

where E_{rx}^{ji} and E_{tx}^{ji} are the energy amounts spent by the jth node for receiving from and transmitting to the ith node, respectively; it corresponds to the sum of the energy of reception, computation and transmission of the portion that is offloaded plus the idle energy of the jth CFN. On the RFN side the energy consumption can be rewritten as:

$$E_{RFN}^i = \sum_{j \in \mathcal{N}(i)} \left(\alpha_{off,ij}^l \left(E_{tx}^{ij} + E_{rx}^{ij} \right) \right) + \alpha_{loc,i}^l E_{com}^i + E_{id}^i \tag{2.15}$$

corresponding to sum of the energy required for offloading the portion to the nearby nodes, performing the computation of the rest of the task locally, and the energy of idle state. We introduce now the following Boolean variable:

$$U_{FN}^i = \begin{cases} 1 & \text{if the } i\text{th FN is a CFN} \\ 0 & \text{if the } i\text{th FN is an RFN} \end{cases} \tag{2.16}$$

representing the two possible conditions in which an FN can be.

In this work, the goal is to minimize the average FN energy consumption in the network and the overall delay. This leads to a formulation of the partial offloading problem as:

$$\begin{aligned} &\min_{\alpha_{\text{off}}} \left\{ \sum_{i=1}^{N} \left(U_{FN}^i \sum_{j \in \mathcal{N}(i)} \left(\alpha_{off,ij}^l \left(E_{rx}^{ji} + E_{com}^j + E_{tx}^{ji} \right) + E_{id}^j \right) + \left(1 - U_{FN}^i\right) \right. \right. \\ &\qquad \left. \left. \cdot \left(\sum_{j \in \mathcal{N}(i)} \left(\alpha_{off,ij}^l \left(E_{tx}^{ij} + E_{rx}^{ij} \right) \right) + \alpha_{loc}^l E_{com}^i + E_{id}^i \right) \right) \right\} \\ &\min_{\alpha_{\text{off}}} \left\{ \frac{\sum_{i=1}^{N} \sum_l D_i^l(\alpha_{off,i}^l)}{\sum_{i=1}^{N} \sum_l \Lambda_l^i} \right\} \end{aligned} \tag{2.17}$$

subject to:

$$\begin{cases} \eta_{comp_m} > \eta_{comp_i} \quad \forall m,i & \text{(2.18a)} \\ L_{s_l} > L_{r_l} & \text{(2.18b)} \\ P^i_{com} \geq \{P^i_{tx}; P^i_{rx}\} & \text{(2.18c)} \\ d_{i,\iota} \leq R \quad u_\iota \in \mathcal{U} & \text{(2.18d)} \\ d_{i,m} \leq F \quad a_m \in \mathcal{A} & \text{(2.18e)} \\ \alpha^l_{loc,i} + \sum\limits_{j \in \mathcal{N}(i)} \alpha^l_{off,ij} = 1 & \text{(2.18f)} \end{cases}$$

where $\alpha_{\mathbf{off}}$ is the set of the offloaded portions of all tasks at a given time instant, and

$$\Lambda^i_l = \begin{cases} 1 & \text{if the } i\text{th FN generates a task to be processed} \\ 0 & \text{otherwise} \end{cases} \tag{2.19}$$

allows to consider the total number of tasks generated by the FNs. Hence, the minimization of the FN energy consumption corresponds to finding the optimal partial offloading parameter $\alpha_{\mathbf{off}}$ allowing to minimize the energy consumed by all the FNs, i.e., including both RFN and CFN, or task delay, corresponding to finding the optimal partial offloading parameter $\alpha_{\mathbf{off}}$ allowing to minimize the task processing delay, defined as the ratio between the time needed for processing all the tasks generated at a certain time instant and the overall number of tasks generated within the same time interval; they are shown in (2.17).

Constraint (2.18a) introduces the hypothesis that the processing speed of F-APs is higher than FNs', that is at the basis of every Fog Network deployment. Constraint (2.18b) shows that the length of the requested packet is higher than the packet result. It is shown in Constraint (2.18c) that computing power for an FN is higher than transmission and reception power, leading the offloading as a feasible solution. Constraint (2.18d) ensures that the distance between two FNs should not exceed threshold R, which is the FN coverage area. Likewise, the distance between an F-AP and an FN should be smaller than threshold F as shown in Constraint (2.18e). The constraint that the local computation plus the offloading of the ith FN should be equal to one is shown in (2.18f). In the following section we propose a low complexity solution, by decomposing the problem in three steps, and exploiting two architectural hypotheses.

2.2.2 Centralized and Distributed Partial Offloading Approaches

In order to solve the problem we resort to a decomposition in three steps. At first, we will classify the nodes based on their energy status. This selection allows to divide the nodes into groups where the higher energy nodes are able to process the tasks for other nodes, while the lower energy nodes benefit from offloading. In the second step, each FN belonging to the lower energy group selects the potential nodes for offloading the processing task; this step is performed in two possible ways by resorting to a centralized and a distributed approach. Finally, in the third step, each RFN is optimally selecting the portion to be offloaded to each of the selected nodes.

On one hand we are dealing with a centralized approach where the offloading decision is taken from a central entity supposed to be able to know the status of each node, while in the distributed approach each FN is selfishly deciding its offloading policy by optimizing the offloaded portion among the nearby nodes.

We are considering in this work that the tasks can be computed by both F-APs and FNs, and we have also categorized the FNs in CFNs, those that can perform the computation, and RFNs, those that are offloading a task; the basic idea of the centralized architecture is that the RFNs and the related offloaded portion are centrally selected, while in the distributed architecture the RFNs select the CFNs and the F-APs for offloading.

FN Classification

Since one of the two objectives we are pursuing is related to the minimization of the energy, the first step aims at classifying the FNs based on their energy level. We suppose that the FNs having a higher remaining energy are better candidates for performing the computation of the incoming tasks. On the other hand, FNs with a lower remaining energy are preferred to offload the computation to others to save energy. FN classification is updated every time a new task is generated by FNs, so that the classification is based on the most updated remaining energy level. To this aim, we exploit a 3-quantile function to classify the FNs considering their remaining energy level. All FNs are classified into two lists, High Power Fog Nodes (HPFNs) and Low Power Fog Nodes (LPFNs), as shown in Fig. 2.2. The FNs whose remaining energy is higher than the upper quantile index, $E_{r,Q2}$, of the energy level distribution of all FNs are considered as HPFNs, and the rest are the LPFNs. We define the LPFN and HPFN lists, $\Im_{LPFN}$ and $\Im_{LPFN}$, at time instant t respectively as:

$$\Im_{LPFN}(t) = \left\{u_i | E_r^i(t) \leq E_{r,Q2}(t)\right\} \tag{2.20}$$

$$\Im_{HPFN}(t) = \left\{u_i | E_r^i(t) > E_{r,Q2}(t)\right\} \tag{2.21}$$

Fig. 2.2 The 3-quantile function of the remaining energy level distribution

The upper quantile index is:

$$E_{r,Q2}(t) = \inf\left\{E_r^i(t), i = 1,\ldots,N | p \leq F_{E_r}(i)\right\} \tag{2.22}$$

where p is equal to 2/3, in case of upper 3-quantile, and $F_{E_r}(i)$ represents the distribution of the remaining energy of all FNs. The pseudo-code of the FN classification is shown in Algorithm 1, where, for each FN (line 3), the remaining energy at time instant t is compared with the upper quantile index (line 4) in order to classify the FNs into one of the two lists, i.e. $\Im_{LPFN}$ and $\Im_{LPFN}$ (lines 5-7).

Algorithm 1 3-Quantile Function

1: **Input:** E_r^i $i = 1,\ldots,N$
2: **Output:** $\Im_{LPFN}$ and $\Im_{HPFN}$
3: **for** each $u_i \in \mathcal{U}$ **do**
4: **if** $E_r^i(t) \geq E_{r,Q2}$ **then**
5: $\Im_{HPFN} \leftarrow u_i$
6: **else**
7: $\Im_{LPFN} \leftarrow u_i$
8: **end if**
9: **end for**

Architectural Approaches

The second step deals with the offloading decision. To this aim, two approaches have been considered.

Centralized Offloading Approach

In the centralized approach, the idea is that of primarily selecting the nodes able to process the tasks, i.e, the CFN; such nodes will select the nearby RFNs. To this aim we resort to a cluster architecture where the FNs can be classified in two types: Fog Cluster Head (FCHs)

and Fog Cluster Member (FCMs). Each cluster can be composed of one FCH and several FCMs. The FCHs are selected in a way that they are able of performing the computations of the tasks requested by the FCMs within their cluster [74]. Hence, the FCHs result to be the CFN while the FCMs are the RFNs.

The cluster formation is started by the FCHs (or CFNs), which are taken from $\Im_{HPFN}$, due to their higher energy amount. Each FCH considers potential FCM candidates, taken from $\Im_{LPFN}$, for the cluster formation as long as they are within its coverage area. FCMs are better candidates for becoming RFN due to their lower energy amount and, hence, asking for offloading to the CFN represented by the FCH. We have considered two policies based on the two layers for the computation offloading:

1. FN layer
2. FN and F-AP layers

In the first policy, we are only considering the presence of the FNs in the network. Hence, FCMs partially offload their tasks to the associated FCHs. In this case, the set of clusters is $\mathcal{C}^{\mathcal{FN}} = \{c_1^{FN}, \ldots, c_g^{FN}, \ldots, c_G^{FN}\}$ and $|\mathcal{C}^{\mathcal{FN}}| \leq |\Im_{HPFN}|$, where the gth cluster is defined as:

$$c_g^{FN} = \{u_i | u_i \in \Im_{LPFN}, |c_g^{FN}| < c_{cap}^{FN}, d_{g,i} \leq R\} \tag{2.23}$$

where c_{cap}^{FN} is the capacity of a cluster corresponding to the maximum number of cluster members. Similar to the FNs classification step, this procedure is updated every time a new task is generated, since the FNs' energy level change in a different way depending on the role they have. Indeed, through the clusters updating, the FNs having consumed more energy, i.e., FCHs, might change their role to FCMs, and the reverse. This approach results also in increasing the overall life time of the network by indirectly equalizing the FNs energy level by allowing a higher consumption for those nodes having a higher amount of energy and a lower consumption for the FNs having a lower amount of energy.

However, the FCMs not being associated to any cluster, are inserted into the set $\mathcal{L}_1$, which is the set of nodes performing the computation locally including all the FCHs generating their own tasks; it is defined as:

$$\mathcal{L}_1 = \left\{u_i | \{u_i \in \Im_{LPFN}, u_i \notin c_g^{FN}\}; \{u_i \in \Im_{HPFN}\}\right\} \forall g \tag{2.24}$$

On the other hand, in the second policy, both FNs and F-APs layers are available. Hence, FCMs can partially offload to the associated FCH and F-APs. In this case, the FCMs which are not in any clusters can still exploit the F-APs if available. Likewise, the FCHs can partially offload to the F-APs within their coverage area. As shown in Fig. 2.3, the FCMs

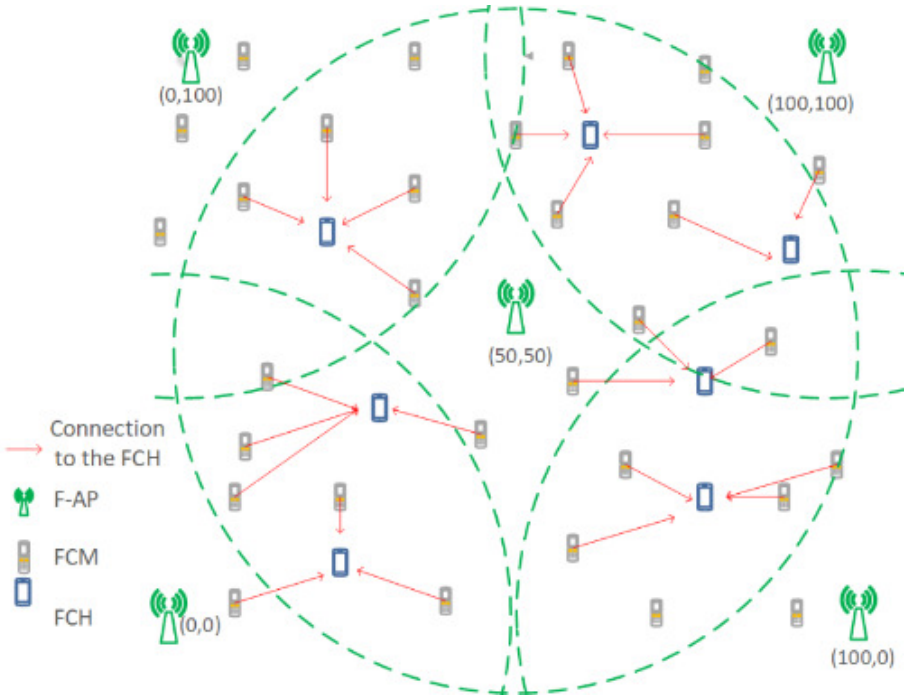


Fig. 2.3 Centralized Architecture

can be connected to the FCHs and the nearby F-APs. Likewise, the FCHs (or the CFNs) are also able to offload to nearby F-APs their own tasks. The set of F-APs cluster having FNs in their coverage area is $\mathcal{C}^{\mathcal{F}-\mathcal{AP}} = \{c_1^{F-AP}, \ldots, c_m^{F-AP}, \ldots, c_M^{F-AP}\}$ and $|\mathcal{C}^{\mathcal{F}-\mathcal{AP}}| \leq M$, where the set of FNs connected to the mth F-AP is defined as:

$$c_m^{F-AP} = \left\{u_i | u_i \in \{\Im_{LPFN}; \Im_{HPFN}\}, |c_m^{F-AP}| < c_{cap}^{F-AP}, d_{m,i} \leq F\right\} \tag{2.25}$$

where c_{cap}^{F-AP} is the capacity of an F-AP corresponding to the maximum number of nodes that an F-AP can manage. However, the FNs not able to offload to any neighboring nodes belong to the local list, $\mathcal{L}_2$, defined as:

$$\mathcal{L}_2 = \left\{u_i | \{u_i \in \Im_{LPFN}, u_i \notin c_g^{FN}, u_i \notin c_m^{F-AP}\}; \{u_i \in \Im_{HPFN}, \notin c_m^{F-AP}\}\right\}, \ \forall g, m \tag{2.26}$$

The pseudocodes of the centralized architecture when using the policy limited to the first layer (a) or both layers (b) are shown in Algorithm 2 and 3, respectively. Algorithm 2 has as input the two sets of nodes $\Im_{HPFN}$ and $\Im_{LPFN}$ (line 1), having the HPFNs and LPFNs previously categorized, while the output (line 2) is represented by the set of clusters $\mathcal{C}^{\mathcal{FN}}$, including one FCH and at least one FCM each, and $\mathcal{L}_1$, the set of nodes performing the local computation. The algorithm starts by considering each HPFN as an FCH candidate (lines

3-4), and, for each of them, the FCMs, selected among the LPFN, having a distance with respect to the selected FCH lower than the coverage range, are put into its cluster, up to the cluster maximum capacity (lines 5-9). In the end, the remaining LPFNs and all the HPFNs are put into the set $\mathcal{L}_1$, the list of the nodes performing the local computation (lines 11-16). Similarly, Algorithm 3 has as input the two sets of nodes $\Im_{HPFN}$ and $\Im_{LPFN}$ (line 1), while the output is represented by the set of clusters $\mathcal{C}^{\mathcal{FN}}$, the set of clusters $\mathcal{C}^{\mathcal{F}-\mathcal{AP}}$, including one F-AP and at least one FN each, and $\mathcal{L}_2$, the set of nodes performing the local computation (line 2). The algorithm starts by first populating the FN clusters, each one composed by one FCH, selected among the HPFNs, and at least one FCM, selected among the LPFNs. The selected FCMs should have a distance with respect to the FCH lower than the coverage range, and each cluster can be composed by a maximum number of FNs (lines 3-10). Moreover, due to the presence of the F-APs, the FNs are put into the F-AP clusters, if respecting the same constraints, i.e, the distance with respect to the F-AP and the F-AP cluster capacity (lines 11-23); this is performed for both LPFNs (lines 12-17) and HPFNs (lines 18-22). In the end, if there are FNs not belonging to any cluster, they are put into the set $\mathcal{L}_2$, the list of nodes performing a local computation (lines 24-29).

Algorithm 2 Centralized architecture (a)

1: **Input:** $\Im_{HPFN}, \Im_{LPFN}$
2: **Output:** $\mathcal{C}^{\mathcal{FN}}$ and $\mathcal{L}_1$
3: **for** each $u_i \in \Im_{HPFN}$ **do**
4: $\quad c_g^{FN} \leftarrow u_i$
5: $\quad$ **for** each $u_\iota \in \Im_{LPFN}$ **do**
6: $\quad\quad$ **if** $d_{i,\iota} \leq R$ **and** $|c_g^{FN}| < c_{cap}^{FN}$ **then**
7: $\quad\quad\quad c_g^{FN} \leftarrow u_\iota$; $|c_g^{FN}| = |c_g^{FN}| + 1$; remove u_ι from $\Im_{LPFN}$
8: $\quad\quad$ **end if**
9: $\quad$ **end for**
10: **end for**
11: **for** each $u_i \in \Im_{HPFN}$ **do**
12: $\quad \mathcal{L}_1 \leftarrow u_i$
13: **end for**
14: **for** each $u_\iota \in \Im_{LPFN}$ **do**
15: $\quad \mathcal{L}_1 \leftarrow u_\iota$
16: **end for**

Distributed Offloading Approach

Unlike centralized approach, in the distributed approach the idea is that the RFNs, belonging to the set $\Im_{LPFN}$, select the available nodes for task computation. In this architecture, the RFNs can offload to multiple available CFNs within their coverage area [75]. Differently from the centralized approach, where the RFNs were selected by the CFNs, in the distributed

Algorithm 3 Centralized architecture (b)

Input: $\Im_{HPFN}$, $\Im_{LPFN}$
Output: $\mathcal{C}^{\mathcal{F}-\mathcal{AP}}$, $\mathcal{C}^{\mathcal{FN}}$ and $\mathcal{L}_2$
for each $u_i \in \Im_{HPFN}$ **do**
  $c_g^{FN} \leftarrow u_i$
  for each $u_\iota \in \Im_{LPFN}$ **do**
    if $d_{i,\iota} \leq R$ **and** $|c_g^{FN}| < c_{cap}^{FN}$ **then**
      $c_g^{FN} \leftarrow u_\iota$;$|c_g^{FN}| = |c_g^{FN}| + 1$
    end if
  end for
end for
for each $a_m \in \mathcal{A}$ **do**
  $c_m^{F-AP} \leftarrow a_m$
  for each $u_\iota \in \Im_{LPFN}$ **do**
    if $d_{\iota,m} \leq F$ **and** $|c_m^{F-AP}| < c_{cap}^{F-AP}$ **then**
      $c_m^{F-AP} \leftarrow u_\iota$; $|c_m^{F-AP}| = |c_m^{F-AP}| + 1$;remove u_ι from $\Im_{LPFN}$
    end if
  end for
  for each $u_i \in \Im_{HPFN}$ **do**
    if $d_{i,m} \leq F$ **and** $|c_m^{F-AP}| < c_{cap}^{F-AP}$ **then**
      $c_m^{F-AP} \leftarrow u_i$; $|c_m^{F-AP}| = |c_m^{F-AP}| + 1$;remove u_i from $\Im_{HPFN}$
    end if
  end for
end for
for each $u_i \in \Im_{HPFN}$ **do**
  $\mathcal{L}_2 \leftarrow u_i$
end for
for each $u_\iota \in \Im_{LPFN}$ **do**
  $\mathcal{L}_2 \leftarrow u_\iota$
end for

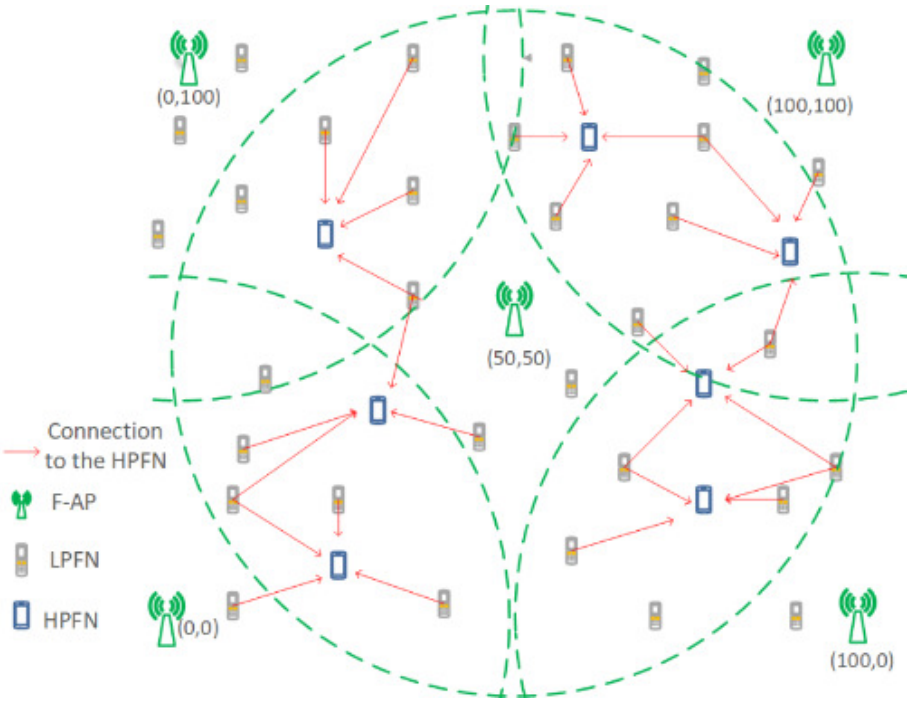


Fig. 2.4 Distributed Architecture

approach the RFNs autonomously select the CFNs among the $\Im_{HPFN}$. Likewise, the RFNs can select several F-APs in the second layer as long as they are within their coverage area. The scenario is represented in Fig. 2.4.

By taking into account the presence of FNs and F-APs, two policies have been considered for the distributed approach by exploiting the two layers for the computation offloading:

1. FN layer
2. FN and F-AP layers

In the first layer, RFNs partially offload to the nearby CFNs while performing the remaining computation locally. Unlike the centralized method, where the RFNs were able to offload only to one associated CFN, in the distributed architecture they can offload to multiple CFNs at the same time. The CFNs available for one RFN are arranged into clusters. The set of clusters centered on RFNs is shown as $\mathcal{B}^{RFN,1} = \{b_1^{RFN,1}, \ldots, b_x^{RFN,1}, \ldots, b_X^{RFN,1}\}$, where $|\mathcal{B}^{RFN,1}| \leq |\Im_{LPFN}|$. Moreover, the RFNs that can be connected to the zth CFN are put in the group b_z^{CFN}; the set of groups compose $\mathcal{B}^{CFN} = \{b_1^{CFN}, \ldots, b_z^{CFN}, \ldots, b_Z^{CFN}\}$. The xth RFN when the F-APs are not available is defined as:

$$b_x^{RFN,1} = \left\{u_i | u_i \in \Im_{HPFN}, |b_z^{CFN}| < b_{cap}^{CFN}, d_{x,i} \leq R\right\} \tag{2.27}$$

where b_{cap}^{CFN} is the computing capacity of a CFN. On the other hand, in the second policy, RFNs can partially offload to both CFNs in the first layer and F-APs in the second layer. Moreover, the CFNs perform a local computation for their own tasks in the first policy while they can partially offload to the F-APs in the second policy. In case of working on both layers, the set of RFNs cluster in which both CFNs and F-APs are available is shown as $\mathcal{B}^{RFN,2} = \{b_1^{RFN,2}, \ldots, b_x^{RFN,2}, \ldots, b_X^{RFN,2}\}$. The xth RFN when both layers are available is defined as:

$$b_x^{RFN,2} = \big\{u_i | \{u_i \in \Im_{HPFN}, d_{x,i} \leq R, |b_z^{CFN}| < b_{cap}^{CFN}\}; \\ \{a_m \in \mathcal{A}, d_{x,m} \leq F\}, |c_m^{F-AP}| < c_{cap}^{F-AP}\big\} \quad (2.28)$$

The pseudocodes of the distributed architecture for layer one (policy (a)) and both layers (policy (b)) are shown in Algorithms 4 and 5, respectively. In both cases the inputs are represented by the sets $\Im_{HPFN}$ and $\Im_{LPFN}$, including all the HPFNs and LPFNs previously categorized (line 1). Algorithms 4 has, as outputs, the set of RFNs centered clusters $\mathcal{B}^{RFN,1}$ and the list of nodes performing the local computation, $\mathcal{L}_1$ (line 2). In this algorithm, the RFNs, belonging to the LPFN set, select as many CFNs as possible as long as the distance and capacity constraints are respected (lines 3-10). In the end, the remaining LPFNs, not able to select any CFNs among the HPFNs list, and all the CFNs, are put into the set $\mathcal{L}_1$, the list of the nodes performing local computation (lines 12-17). In Algorithm 5, the outputs are instead the set of RFNs centered clusters $\mathcal{B}^{RFN,2}$, composed of both HPFNs and F-APs, the F-AP based clusters $\mathcal{C}^{F-AP}$ including the FNs within their coverage range, and the list of nodes performing the local computation, $\mathcal{L}_2$ (line 2). In this algorithm, the RFNs select as many CFNs (lines 4-9) and F-APs (lines 10-14) as possible respecting the distance and capacity constraints. Moreover, due to the presence of the F-APs, the HPFNs are put into the F-AP clusters, if respecting the same constraints, i.e, the distance with respect to the F-AP and the F-AP cluster capacity (lines 18-23). Finally, the remaining LPFNs and HPFNs are put into the list of the nodes performing local computation, $\mathcal{L}_2$ (lines 25-30).

A comparison between the centralized and distributed architectures is briefly shown in Tab. 2.2.

Partial Offloading Estimation

Even if the problem in (2.17) cannot be solved in a closed way, in the previous sections we introduced two steps that allow to relax the problem. The problem relaxation allows to simplify the problem and scale it down by optimizing separately the amount of data to be

Algorithm 4 Distributed architecture (a)

Input: $\Im_{HPFN}$, $\Im_{LPFN}$
Output: $\mathcal{B}^{RFN,1}$ and $\mathcal{L}_1$
for each $u_\iota \in \Im_{LPFN}$ **do**
  for each $u_i \in \Im_{HPFN}$ **do**
    if $d_{\iota,i} \leq R$ **and** $|b_z^{CFN}| < b_{cap}^{CFN}$ **then**
      $b_x^{RFN,1} \leftarrow u_\iota$
      $b_x^{RFN,1} \leftarrow u_i; |b_z^{CFN}| = |b_z^{CFN}| + 1$
    end if
  end for
end for
remove u_ιs which are in $\mathcal{B}^{RFN,1}$ from $\Im_{LPFN}$
for each $u_\iota \in \Im_{LPFN}$ **do**
  $\mathcal{L}_1 \leftarrow u_\iota$
end for
for each $u_i \in \Im_{HPFN}$ **do**
  $\mathcal{L}_1 \leftarrow u_i$
end for

Algorithm 5 Distributed architecture (b)

Input: $\Im_{HPFN}$, $\Im_{LPFN}$
Output: $\mathcal{C}^{\mathcal{F}-\mathcal{AP}}$,$\mathcal{B}^{RFN,2}$ and $\mathcal{L}_2$
for each $u_\iota \in \Im_{LPFN}$ **do**
  for each $u_i \in \Im_{HPFN}$ **do**
    if $d_{\iota,i} \leq R$ **and** $|b_z^{CFN}| < b_{cap}^{CFN}$ **then**
      $b_x^{RFN,2} \leftarrow u_\iota$
      $b_x^{RFN,2} \leftarrow u_i; |b_z^{CFN}| = |b_z^{CFN}| + 1$
    end if
  end for
  for each $a_m \in \mathcal{A}$ **do**
    if $d_{\iota,m} \leq F$ **and** $|c_m^{F-AP}| < c_{cap}^{F-AP}$ **then**
      $b_x^{RFN,2} \leftarrow a_m; |c_m^{F-AP}| = |c_m^{F-AP}| + 1$
    end if
  end for
end for
remove u_ιs which are in $\mathcal{B}^{RFN,2}$ from $\Im_{LPFN}$
for each $a_m \in \mathcal{A}$ **do**
  for each $u_i \in \Im_{HPFN}$ **do**
    if $d_{\iota,m} \leq F$ **and** $|c_m^{F-AP}| < c_{cap}^{F-AP}$ **then**
      $c_m^{F-AP} \leftarrow u_i; |c_m^{F-AP}| = |c_m^{F-AP}| + 1$
    end if
  end for
end for
remove u_is which are in $\mathcal{C}^{\mathcal{F}-\mathcal{AP}}$, from $\Im_{HPFN}$
for each $u_\iota \in \Im_{LPFN}$ **do**
  $\mathcal{L}_2 \leftarrow u_\iota$
end for
for each $u_i \in \Im_{HPFN}$ **do**
  $\mathcal{L}_2 \leftarrow u_i$
end for

Table 2.2 Comparison of the centralized and distributed approaches

	Centralized	**Distributed**
Selection Policy	RFNs are selected by CFNs and F-APs	RFNs select the CFNs and F-APs
RFNs	FCMs taken from $\Im_{LPFN}$	LPFNs taken from $\Im_{LPFN}$
CFNs	FCHs taken from $\Im_{HPFN}$	HPFNs taken from $\Im_{HPFN}$
FN Layer	connection with one CFN	connection with multiple CFNs
F-AP Layer	FCMs and FCHs	LPFNs and HPFNs

offloaded. In this section, by exploiting this relaxation we will calculate in a closed form the optimal amount of data to be offloaded based on the constraints.

In order to evaluate the amount of data to be offloaded we proceed in a two step method. First of all, we estimate the portion to be offloaded to each of the available nodes; in order to do this we will consider that each RFN is aware of the processing power of the nearby computing nodes and data rate of each link. Then, considering the offloaded portion and the characteristics of the neighboring nodes, we estimate the portion that should be performed locally. With the proposed approach the estimation of the local and offloaded portion is performed in the same way in both centralized and distributed architectures.

Partial Offloading Portion estimation

To estimate the amount that should be offloaded to each of the available nodes that are going to perform the computation of a task portion, we have considered both data rate and computational power of the available neighboring nodes.

When the ith FN decides to offload a task, there are $j \in \mathcal{N}(i)$ available nodes, where $\mathcal{N}(i)$ is a set of neighboring nodes of the ith FN available for computing. As a result, the task can be divided into several portions to be offloaded to each of those available nodes. We define β_{ij}^{l} as the portion of lth task to be offloaded from the ith node to the jth node. Due to the impact of the offloaded portion on task processing delay and energy consumption of all FNs, we have considered two goals for the estimation of partial offloading portion, β_{ij}^{l}:

1. Task processing delay
2. Node energy consumption and task processing delay

If the task processing delay is considered, the amount of the lth task to be offloaded to the jth node can be defined as:

$$\dot{\beta_{ij}^{l}} = \gamma \cdot \frac{r_{ij}}{\sum_{j \in \mathcal{N}(i)} r_{ij}} + (1-\gamma) \cdot \frac{\eta_{comp_j}}{\sum_{j \in \mathcal{N}(i)} \eta_{comp_j}} \tag{2.29}$$

Table 2.3 Parameters Definition for Partial Offloading

Parameter	Definition
β_{ij}^{l}	Offloaded portion of the lth task from the ith node to the jth node
$\dot{\beta}_{ij}^{l}$	Offloaded portion of the lth task from the ith node to the jth node considering goal (a)
$\ddot{\beta}_{ij}^{l}$	Offloaded portion of the lth task from the ith node to the jth node considering goal (b)
γ	Estimation weight coefficient
$\tilde{T}_{off,i}^{l}$	Offloading time for the lth task of the ith node considering each of the estimation goals
$\tilde{\alpha}_{loc,i}^{l}$	Estimated local portion of the lth task of the ith node considering each of the estimation goals
$\tilde{E}_{CFN}^{j}$	The estimated energy consumption of the jth CFN
$\tilde{E}_{RFN}^{i}$	The estimated energy consumption of the ith RFN

where γ is a coefficient giving a weight to the importance of the data rate and computational power in the estimation. This estimation considers the data rate of the link and the computational power of the node, for offloading a higher portion to the nodes with better characteristic. If both task processing delay and energy consumption are considered the amount of the lth task to be offloaded to the jth node can be defined as:

$$\ddot{\beta}_{ij}^{l} = \gamma \cdot \frac{\frac{r_{ij}}{E_{tx}^{ij}+E_{rx}^{ij}}}{\sum_{j\in\mathcal{N}(i)} r_{ij}} + (1-\gamma) \cdot \frac{\frac{\eta_{comp_j}}{E_{id}^{i}}}{\sum_{j\in\mathcal{N}(i)} \eta_{comp_j}} \tag{2.30}$$

In this formula the ith RFN's energy is affecting the delay and computational power metrics to estimate the portion to be offloaded to each available node; in particular, the energy spent in transmission and reception is affecting the data rate metric while the energy spent in idle affect the computational power metric. Hence, a higher portion will be offloaded to those nodes allowing to consume less energy. As a result not only delay is minimized but the energy consumption is also the target of the minimization. For the sake of readability, Tab. 2.3 has been provided presenting the definitions of the parameters used in the equations of Section 2.2.2.

Local Computation Portion estimation

After having estimated the portions to be offloaded to the available neighboring nodes for computation, we have to estimate the amount that should be performed locally considering the characteristics of the available neighboring nodes.

Since one of the objectives is minimizing the delay, the idea is that of imposing that the amount of time spent for the local computation is equal to the amount of time spent for the offloading phase. This corresponds to minimize the idle time for any FN; hence imposing:

$$T^l_{loc,i} = T^l_{off,i} \tag{2.31}$$

where:

$$T^l_{loc,i} = \alpha^l_{loc,i} \frac{O_l}{\eta_{comp_i}} \tag{2.32}$$

and

$$T^l_{off,i} = \max_{j \in \mathcal{N}(i)} \left\{ \alpha^l_{off,ij} \frac{L_{s_l}}{r_{ij}} + T^l_{w_j} + \alpha^l_{off,ij} \frac{O_l}{\eta_{comp_j}} + \alpha^l_{off,ij} \frac{L_{r_l}}{r_{ij}} \right\} \tag{2.33}$$

corresponding to set the local computation time to the maximum among all the times spent for offloading to the neighbor nodes of the ith RFN. The amount of data offloaded to the jth node from the ith node is represented by $\alpha^l_{off,ij}$. By resorting to the estimation of the partial offloaded amount obtained in (2.29) and (2.30), it is possible to write that:

$$\alpha^l_{off,ij} = \beta^l_{ij}(1 - \alpha^l_{loc,i}) \tag{2.34}$$

where $\beta^l_{ij} = \dot{\beta}^l_{ij}$ or $\beta^l_{ij} = \ddot{\beta}^l_{ij}$ depending on the selected goal. By neglecting the queue waiting time, T^l_w, that is assumed to be unknown by FNs, we can rewrite (2.33) as:

$$\tilde{T}^l_{off,i} = \max_{j \in \mathcal{N}(i)} \left\{ \beta^l_{ij}(1 - \alpha^l_{loc,i}) \frac{L_{s_l}}{r_{ij}} + \beta^l_{ij}(1 - \alpha^l_{loc,i}) \frac{O_l}{\eta_{comp_j}} + \beta^l_{ij}(1 - \alpha^l_{loc,i}) \frac{L_{r_l}}{r_{ij}} \right\} \tag{2.35}$$

Hence, by exploiting (2.31), (2.32) and (2.35), it is possible to write:

$$\alpha^l_{loc,i} \frac{O_l}{\eta_{comp_i}} = \max_{j \in \mathcal{N}(i)} \left\{ \beta^l_{ij}(1 - \alpha^l_{loc,i}) \frac{L_{s_l}}{r_{ij}} + \beta^l_{ij}(1 - \alpha^l_{loc,i}) \frac{O_l}{\eta_{comp_j}} + \beta^l_{ij}(1 - \alpha^l_{loc,i}) \frac{L_{r_l}}{r_{ij}} \right\} \tag{2.36}$$

Through simple algebraic operations, it is possible to estimate the amount of local computation for the ith node as:

$$\tilde{\alpha}^l_{loc,i} = \frac{\max_{j \in \mathcal{N}(i)} \left\{ \beta^l_{ij} \left(\frac{L_{r_l}}{r_{ij}} + \frac{L_{s_l}}{r_{ij}} + \frac{O_l}{\eta_{comp_j}} \right) \right\}}{\frac{O_l}{\eta_{comp_i}} + \max_{j \in \mathcal{N}(i)} \left\{ \beta^l_{ij} \left(\frac{L_{r_l}}{r_{ij}} + \frac{L_{s_l}}{r_{ij}} + \frac{O_l}{\eta_{comp_j}} \right) \right\}} \tag{2.37}$$

In the end, any jth node is requested to process an amount equal to $(1 - \alpha^l_{loc,i}) \cdot \beta^l_{ij} \cdot O_l$ related to the lth task of the ith node.

Having estimated the local and offloading portion, considering $\tilde{\alpha}^l_{off,i} = 1 - \tilde{\alpha}^l_{loc,i}$ and exploiting (2.13) we can calculate the total delay, when the ith FN is offloading to the available neighboring nodes, as:

$$D^l_i(\tilde{\alpha}^l_{off,i}) = \max_{j\in\mathcal{N}(i)} \left\{ \beta^l_{ij}\tilde{\alpha}^l_{off,i}\frac{L_{s_l}}{r_{ij}} + T^l_{w_j} + \beta^l_{ij}\tilde{\alpha}^l_{off}\frac{O_l}{\eta_{comp_j}} + \beta^l_{ij}\tilde{\alpha}^l_{off,i}\frac{L_{r_l}}{r_{ij}}, \left(1-\tilde{\alpha}^l_{off,i}\right)\frac{O_l}{\eta_{comp_i}} \right\} \tag{2.38}$$

On the other hand, the energy consumed by the jth CFN, by exploiting (2.14), can be written as:

$$\tilde{E}^j_{CFN} = \tilde{\alpha}^l_{off,i}\beta^l_{ij}\left(E^j_{rx} + E^j_{com} + E^j_{tx}\right) + E^j_{id} \tag{2.39}$$

which is the energy consumption for receiving, computing and transmitting the offloaded part from the ith node to the jth node plus the idle energy of the jth node. Likewise, the energy consumption for the ith RFN, by exploiting (2.15), can be written as:

$$\tilde{E}^i_{RFN} = \sum_{j\in\mathcal{N}(i)} \left(\beta^l_{ij}\tilde{\alpha}^l_{off,i}\left(E^{ij}_{tx} + E^{ij}_{rx}\right)\right) + \left(1-\tilde{\alpha}^l_{off,i}\right)\cdot E^i_{com} + E^i_{id} \tag{2.40}$$

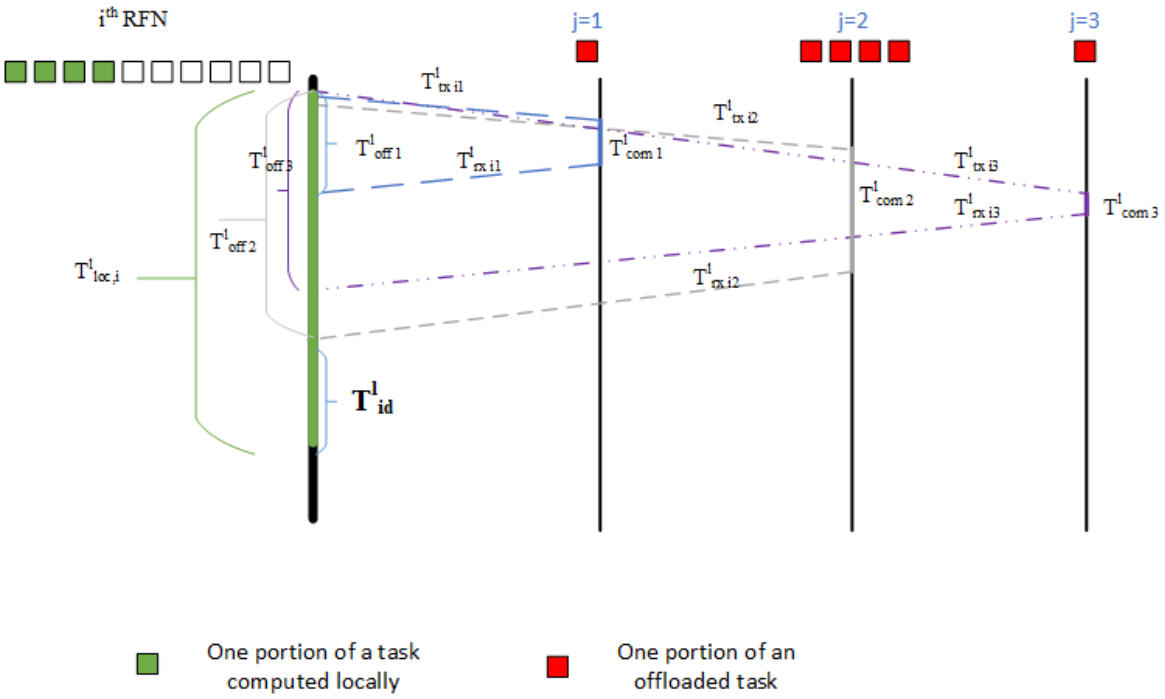
It is worth to be noticed that in (2.38), (2.39) and (2.40), β^l_{ij} could be equal to either (2.29) or (2.30) depending on the minimization goal.

In Fig. 2.5a the total delay for the lth task of the ith RFN is shown, when the $\alpha^l_{loc,i}$ and β^l_{ij} are not optimized. As seen, in this example 4 portions of the lth task are performed locally and the rest are offloaded to the 3 available nodes, which could be CFNs or F-APs. If the offloading portion is not optimized it might lead to having the local delay longer than the offloading delay, or the reverse. However, as shown in Fig. 2.5b, if $\alpha^l_{loc,i}$ and β^l_{ij} are optimized both local and offloading delay are equal by minimizing the idle time leading to a shorter delay.

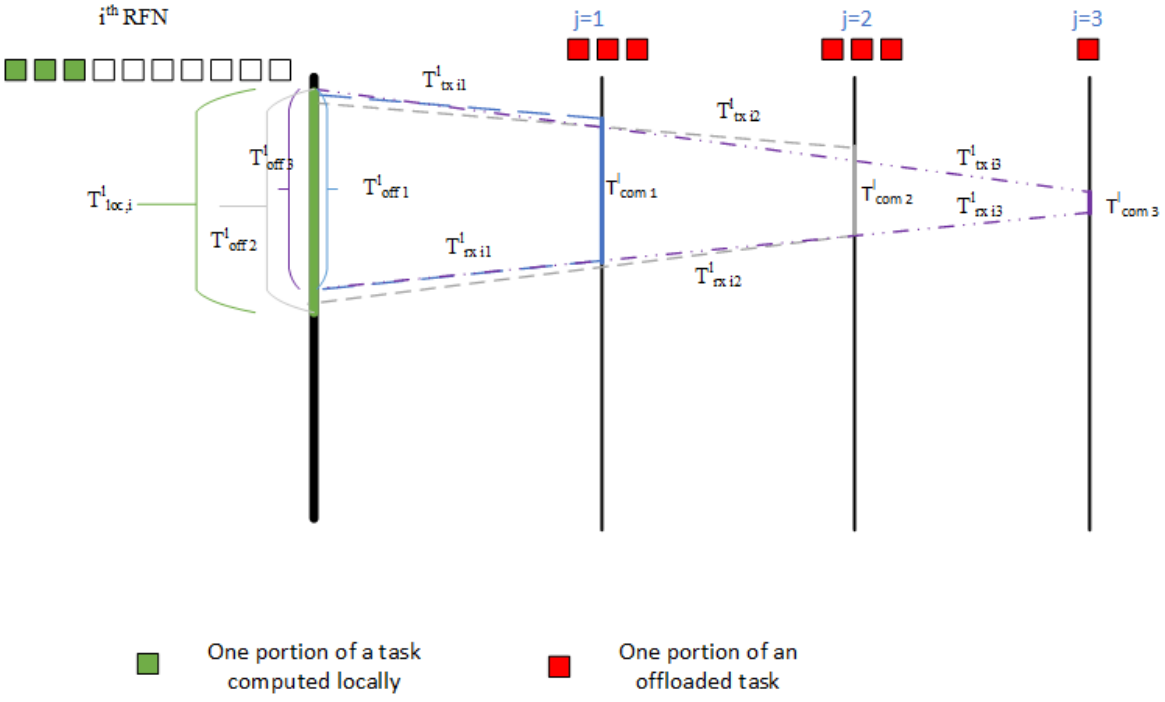
2.2.3 Numerical Results

In this section, the numerical results obtained through computer simulations are presented. In the following we are comparing the performance for the single layer scenario with the performance of the two-layer scenario. Moreover, the comparison is performed between the delay minimization policy and the joint energy and delay minimization policy. These options have been considered in both centralized and distributed architectures.

The computer simulations are performed in Matlab where the considered parameters are listed in Tab. 2.4. The simulation is performed for comparing the performance in terms of average task delay, average FN energy consumption and network lifetime as:



(a) Not Optimized



(b) Optimized

Fig. 2.5 Task delay for offloaded and local portions.

Table 2.4 Simulation Parameters for the Partial Offloading Approach

Parameter	**Value**
Dimension	100m x 100m
Communication Protocol	IEEE 802.11
Task size (L_{s_l})	[1-5] MB
Task result size (L_{r_l})	[0.2-1] MB
h_{ij}	Outdoor RRH/Hotzone, Model 1: Pico to UE [76]
Bandwidth (B)	10 MHz
Noise Density (N_T)	-174 dBm/Hz
FN to FN coverage range (R)	25 m
F-AP coverage range (F)	50 m
Maximum Initial energy ($E_r^l(0)$)	5000 J
Task Operation (O_l)	50G
FN FLOPS	15G FLOPS
F-AP FLOPS	150G FLOPS
Computation power (P_{com})	0.9 W
Idle power	0.01 W
FN Transmission power (P_{tx}^{FN})	1.3 W
F-AP Transmission power (P_{tx}^{F-AP})	1.5 W
FN reception power (P_{rx}^{FN})	1.1 W
F-AP reception power (P_{rx}^{F-AP})	1.3 W

- Average Task Delay: The average time spent by a task for transmitting, waiting, computing and receiving back the result;
- Average Node Energy Consumption: The average energy all FNs have consumed per second;
- Network Lifetime: The time instant beyond which 20% of the FNs deplete their battery [13].

In the following we briefly describe the simulation environment implemented in Matlab. We hypothesize an area of 100×100 meters, with a variable number of FNs randomly positioned in the area, while there are 5 F-APs placed in the locations shown in Fig. 2.1, so that when working on two layers every FN can be always connected to at least one F-AP. Once the FNs are placed in the area, each of them randomly generates tasks with a Poisson distribution having average one task every 50 s[1]; this value have been selected after a careful optimization. The size of tasks generated by each FN has a uniform distribution between 1 MB to 5 MB; the selection of this interval is driven by the application scenario that considers

[1]The parameter has been selected based on our previous work in [74]

the case of nodes offloading heavy computing tasks to the nearby nodes for saving energy and reduce the overall delay. Although all of the FNs are identical in terms of computational power, we have also considered F-APs with higher computational capabilities resulting in a heterogeneous network. We have considered a battery capacity for all the FNs equal to 5000 Jouls; however, each FN has an initial random energy level between 70% and 100% of the battery capacity. When the tasks are generated by each FN, based on the architecture, either centralized or distributed, the available nodes for the task offloading are identified. In the centralized architecture the FCHs, belonging to the set $\Im_{HPFN}$, are selected and their Euclidean distance with all the other LPFNs, belonging to the set $\Im_{LPFN}$, is evaluated. The LPFNs enter the cluster as long as they meet the distance and cluster capacity requirement. In each run of the simulation the clusters are updated and, because the energy level of the FNs changes, the clusters and the connections change as well. In the distributed architecture, instead, the LPFNs identify the HPFNs within their coverage and select them by taking also into account the cluster capacity. The rest of algorithm for both centralized and distributed architecture works based on the policies defined in 2.2.2. In the end, the RFNs offload a task portion to each of the selected nodes based on the estimation they have made considering either, delay or joint delay and energy consumption.

The simulation scenarios are defined based on the connections they have with the layers and their estimation goals; each simulation runs 1000 seconds. Moreover, in order to obtain steady results each simulation run has been carried out 10 times; to this aim, in the following figures, each point on the curves represent the average over the 10 runs, while the error bar represent the variance of the 10 runs. A fifth scenario labeled as *Local*, in which all FNs perform a local computation, has been also considered as a benchmark.

In the figures legend we are considering that the numbers corresponds to the number of layers involved in the computation (i.e., 1 and 2), while the letters *D* and *DE* show, respectively, delay minimization (2.29) and the joint delay and energy minimization (2.30) policies for the $\alpha^l_{loc,i}$ estimation.

First of all, we evaluate the performance of the centralized and distributed architectures in terms of task delay, by comparing the scenario with only one layer and the scenario with both layers.

(a) FN Layer: As seen in both Fig. 2.6a and Fig. 2.6b, scenarios working only on the first layer seem to make smaller changes when the number of FNs is increasing comparing with the scenarios performing on both layers. In centralized scenario, when the number of FNs increases, the delay increases; this is because when there are few FNs, only few of them can be assigned to a CFN (FCH), and as a result more FNs perform a local computation which leads to a lower delay. On the other hand, increasing the number

of FNs, the possibility of having an FCH nearby is higher and the delay for offloading to an FCH is higher than performing a local computation. However, in the distributed scenario that is the reverse. When the number of FNs is not high, there are few CFNs (HPFNs) available but when the number of FNs increases more CFNs (HPFNs) would be available for performing the computation in parallel which leads to a lower delay. The centralized scenario has a higher delay comparing with distributed scenarios due to the fact that when the centralized scenario is limited to the first layer, there is only one FCH in each cluster performing the computation for the RFNs (FCMs). However, in the distributed scenario each RFN (LPFNs) can offload to several CFNs (HPFNs) leading to parallel computation which results in a lower delay.

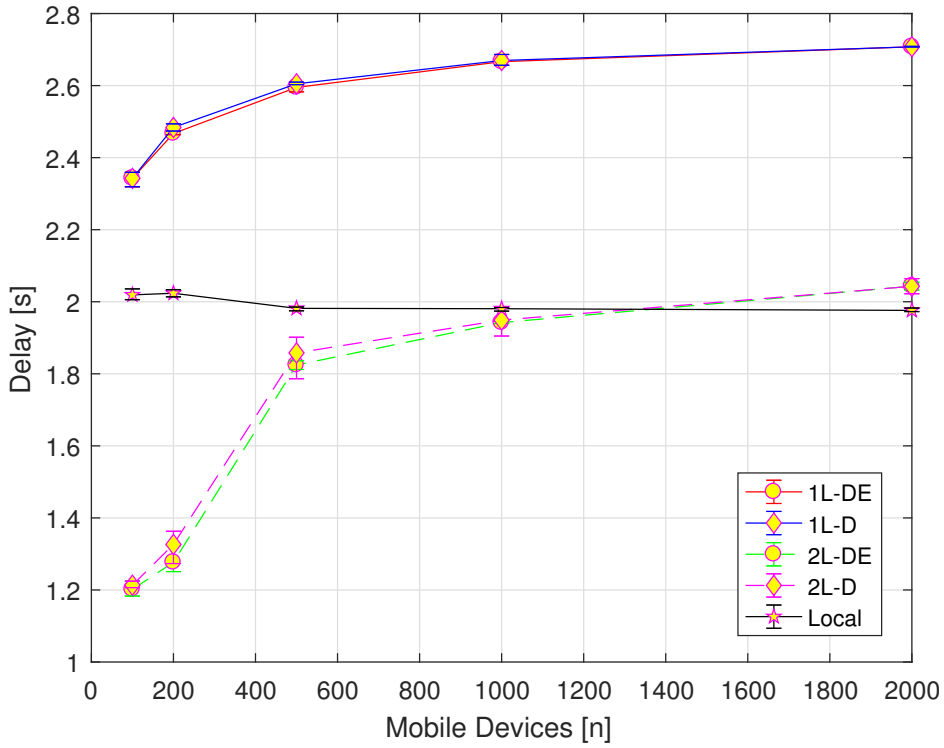
(b) FN and F-AP layers: As depicted in both Fig. 2.6a and Fig. 2.6b, the delay is increasing sharply when the number of FNs increases. When the number of FNs is reduced in both centralized and distributed approaches the FNs have lower possibility of having CFNs nearby to offload; as a result higher portions are offloaded to the F-APs, leading to a lower delay. However, when increasing the number of FNs, more CFNs (FCH for centralized and HPFN for distributed) are available and a lower portion is offloaded to the F-APs in the second layer which leads to a higher delay.

In the end, as the number of FNs performing the computation in first layer increases, delay decreases, due to a parallel computation as depicted in Fig. 2.6b. Furthermore, when computational power is higher (i.e. offloading to F-APs), delay is also lower, as depicted in the scenarios working on both layers in both Figs. 2.6a and 2.6b. When all the FNs perform local computation the delay would be the same for both centralized and distributed scenarios as shown in scenario labeled *Local*.

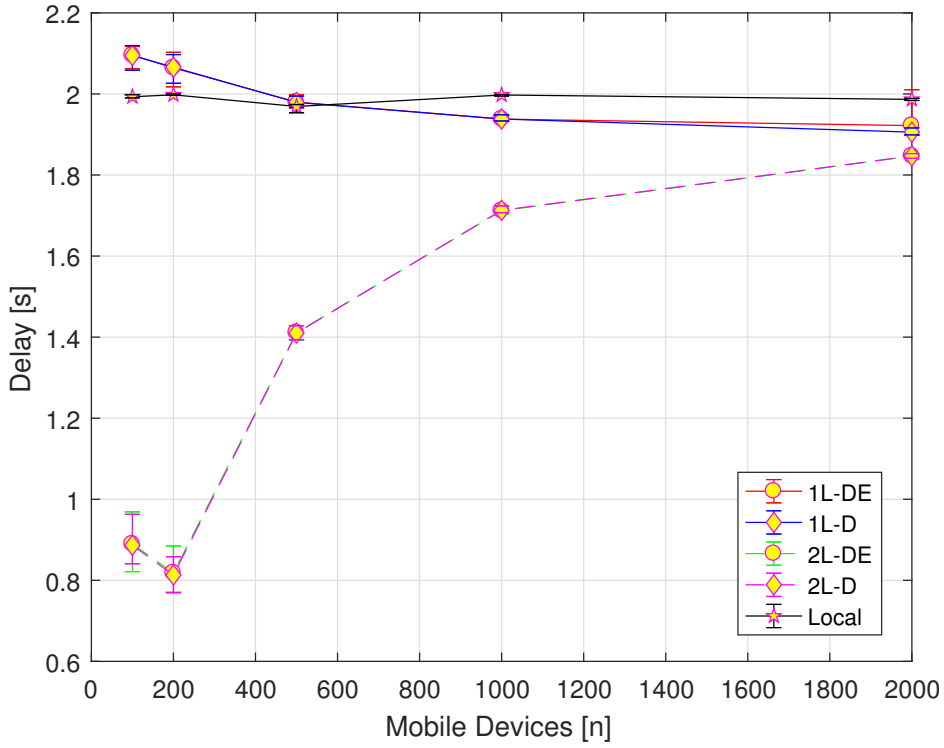
We evaluate then the performance of the centralized and distributed architectures in terms of average FN energy consumption by comparing the system performance in only one layer or both layers.

(a) FN Layer: According to both Fig. 2.7a and Fig. 2.7b, the scenarios limited to the first layer consume more energy in comparison with scenarios working on both layers by showing that offloading only to the nearby FNs results in higher energy consumption. Moreover, there is no significant difference if considering only the delay or both delay and energy optimization because there is no difference in offloading different task portions to different nodes and, in the end, the FNs are consuming energy for performing the computation regardless of the portion that was offloaded.

(b) FN and F-AP layers: When working on two layers, both centralized and distributed architectures result in a lower energy consumption, as seen in both Fig. 2.7a and

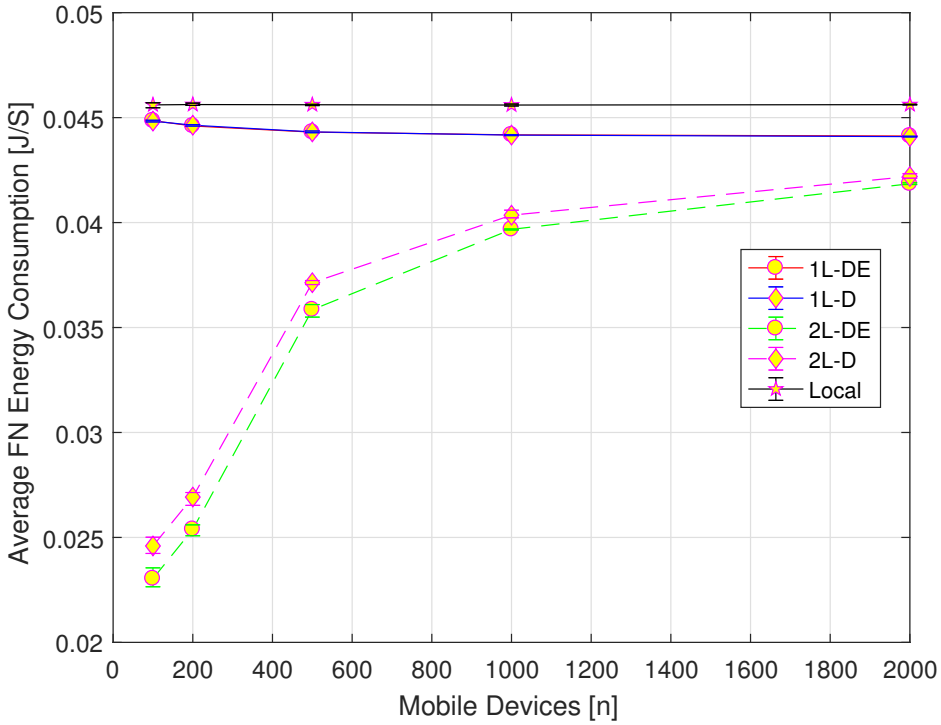


(a) Centralized Architecture

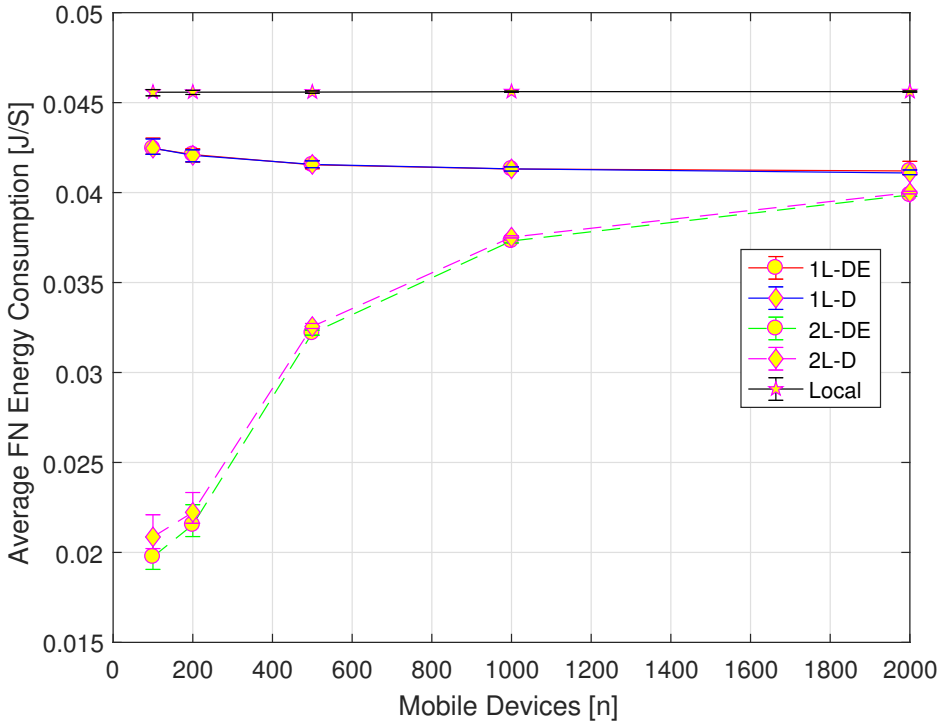


(b) Distributed Architecture

Fig. 2.6 Average Task Delay



(a) Centralized Architecture



(b) Distributed Architecture

Fig. 2.7 FN Energy Consumption

Fig. 2.7b. By exploiting the availability of F-APs some portions are offloaded to the F-APs leading to a lower energy consumption. Moreover, considering both the delay and energy, a higher portion is offloaded to F-APs, resulting in a slight improvement.

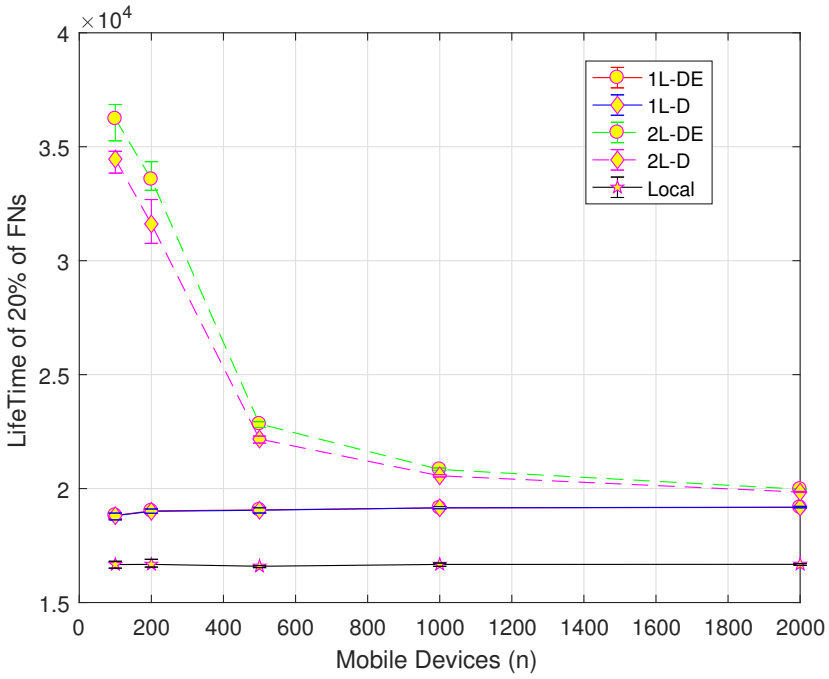
The performance of the centralized and distributed architectures in terms of Network Lifetime is finally evaluated and the results are here compared in case of working only on the first layer or on both layers.

(a) FN Layer: As seen in both Fig. 2.8a and Fig. 2.8b, in the scenarios working on first layer, independently from the parameter considered for the estimation of $\alpha^{l}_{loc,i}$, the nodes go off at the same time and earlier than the scenarios working on two layers in both centralized and distributed architecture. Moreover, it could be seen that number of FNs do not have an impact on the network lifetime; this is due to the fact that all the generated tasks are processed by the FNs regardless of the portions they were offloaded.

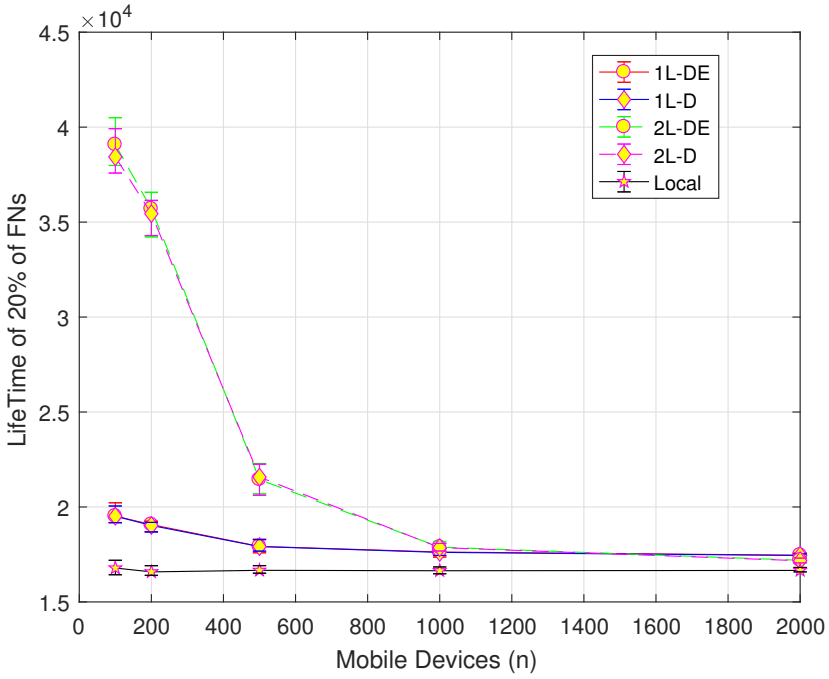
(b) FN and F-AP layers: It can be seen in both Fig. 2.8a and Fig. 2.8b that when number of FNs is reduced in both centralized and distributed scenarios there are fewer options in the first layer and as a result more portions are offloaded to the second layer which results in a higher energy saving and longer network lifetime. However, when the FNs are more, there are also more options available in the first layer for offloading, resulting in higher energy consumption and shorter lifetime. Furthermore, in the centralized scenario in which there is maximum one available FCH for the FCMs, lifetime is slightly higher and this is due to the fact that in the distributed architecture there are more CFNs (HPFNs) involved in the first layer for computation, resulting in consuming more energy comparing with the centralized architecture. Furthermore, considering both delay and energy for the estimation of offloaded portion results in a longer lifetime in the centralized architecture.

2.2.4 Summary

In this work, partial offloading in edge computing has been studied. Two architectures solutions, i.e., centralized and distributed, have been considered for the partial offloading scenario. We have proposed a heuristic solution based on relaxing some of the hypotheses of the partial offloading optimization problem, for minimizing task processing delay and FN energy consumption. Considering these two parameters we have estimated the portion to be offloaded to each of the available nodes at the network edge in order to meet the



(a) Centralized Architecture



(b) Distributed Architecture

Fig. 2.8 Network Lifetime (20%)

objectives. Simulation results demonstrate the impact of different parameters, i.e., delay and joint energy and delay, different layers and different architectures on the performance of the network in terms of FN energy consumption, task processing delay and network lifetime. It is possible to notice that the distributed architecture appears to be more appropriate for partial offloading scenarios when delay has higher priority, due to the fact that it can exploit parallel computation by a larger number of FNs. On the other hand, the centralized architecture appears to be more suitable when priority is given to FNs energy consumption and network lifetime, due to the fact that F-APs are more involved in the computation with respect to the distributed architecture. It is also interesting to note that, even if the presence of F-APs is always an advantage, when the FN density is increasing the performance obtained by using only the FN layer is similar to the scenario with both layers when priority is given to FNs energy consumption and network lifetime. This is an indication in favor of structureless wireless networks.

2.3 Multi-Objective Computation Sharing in MEC

In this work we model the Mobile Edge Computing (MEC) network as a two-layer architecture with the ECs, i.e., the end-user devices, in the bottom layer, and ENs, i.e., the edge servers, in the top layer. While ECs are battery operated with low computing capabilities, the ENs have a higher computational capability and are connected to the electrical network.

There are two types of communications in the network: D2D communication among ECs, and infrastructure communication between ECs and ENs [34]. This results in a scenario where the low-performance & energy-poor ECs share with energy-rich ECs and high-performance ENs the computational process in order to reduce the energy consumption and jointly respect the latency constraints [6].

To offload a task, an EC must spend energy transmitting its task's data, and experience a data uploading and downloading delay. In some cases, the time and energy cost of task offloading can exceed that of computing the task locally by the EC. Thus, ECs face a trade-off between the overall task completion time and their energy consumption when deciding whether to offload their tasks to other nodes.

To find optimal offloading decisions in MEC, we model the problem of task offloading as a Constrained Multi-Objective Problem (CMOP) that jointly minimizes the task processing delay and the energy consumption of the EC nodes.

The solution to the CMOP is characterized by a Pareto front formed by a set of possible solutions where each solution represents a different trade-off between task processing delay and energy consumption of the ECs. To solve the CMOP, we propose an EA that can

efficiently find a high-quality approximation of the Pareto-optimal front. After generating the initial set of solutions, the EA iteratively selects the best ones and combines them using the principles of Darwinian evolution to form new sets of possibly-better solutions [77, 78]. Different to existing EAs that use a set of random solutions for initialization, our EA leverages knowledge about the ECs energy resources and location to generate an initial set of solutions that is closer to the Pareto-optimal front compared to a randomly generated one. Our extensive simulations show that using the proposed initialization technique allows our EA to find a high-quality approximation of the Pareto-optimal in fewer iterations than existing EAs.

To this aim, an EA that can simultaneously minimize task completion delays and energy consumption while considering task offloading to both edge servers, i.e., ENs, and to other mobile devices, i.e., ECs. Compared to previous works, our approach efficiently explores the trade-off space and yields a set of different solutions with different trade-offs between the two objectives. Besides, our approach scales well in the number of nodes in the MEC network and can handle partial task offloading, which results in fine-grained control of the offloading process and lower completion delays and energy consumption.

2.3.1 System Model and Problem Formulation

We consider a two-tier MEC architecture formed by a set $\mathcal{U} = \{u_1, \ldots, u_i, \ldots, u_N\}$ of heterogeneous ECs, with limited computational capabilities and acting as sources of the computation requests, and a set of ENs $\mathcal{F} = \{f_1, \ldots, f_m, \ldots, f_M\}$, characterized by a higher computation capability. The heterogeneous ECs have different computational capabilities and energy requirements. ECs can offload their computational tasks through wireless links to other ECs and to the ENs. ECs are battery powered, and, thus, have limited energy resources, while ENs are connected to the electrical network, and have access to virtually unlimited energy. A system operator manages the MEC network by collecting information about the ECs and ENs and estimating the offloading amounts that minimize the energy consumption and task processing delay of the ECs.

To reduce both task completion delay and energy consumption, ECs can offload their computational tasks through wireless links to other nearby ECs or ENs. Allowing task offloading between ECs is particularly important to meet the task completion delay requirements of low latency communications in battery operated edge environments. Thus, we assume that each EC can split its tasks into multiple unique portions and simultaneously offload each portion to different ECs or ENs. Fig. 2.9 shows the considered MEC architecture. For a better clarity of the system to be presented in the following, we list the key notations in Table 2.5.

Table 2.5 Nomenclature (in order of appearance)

Notation	**Description**
$\mathcal{N}_{EC}(i)$	Set of EC nodes within u_i's range.
$\mathcal{N}_{EN}(i)$	Set ofENnodes within u_i's range.
$R_{\mathcal{U}}$	Coverage range of ECs
$R_{\mathcal{F}}$	Coverage range of ENs
v_k^i	kth device belonging to the neighborhood of u_i
δ_{tx}, δ_{rx}	Minimum transmitted and received task portion size
α	Ratio of δ_{tx} to δ_{rx}
γ_i	The number of portions in u_i's task
η_{c_j}	computational capability of device u_j
η_{c_m}	computational capability of device f_m
O_l^i	number of processing operations for lth task of u_i
$\phi_{u_i}^{u_j}$	Amount of task portions of u_i shared with u_j
$\phi_{f_m}^{u_j}$	Amount of task portions of u_i shared with f_m
E_{off}^{i,v_k^i}	Task portion offloading energy consumption by u_i to v_k^i
$\bar{E}_{off}^i$	Partial offloading energy consumption by u_i
T_{off}^{i,v_k^i}	Task portion offloading delay by the u_i
$\bar{T}_{off}^i$	Overall offloading delay by the u_i
$\bar{n}$	Number of ECs in the REC subset.
$\hat{n}$	number of CECs within reach of u_i.
$\hat{f}$	Number of ENs within range of u_i.
E_r^i	Remaining energy of EC u_i.
$F_{E_r}(i)$	Distribution of the remaining energy of all ECs.
K_i	Maximum number of devices that can accept offloaded tasks.
$\tilde{w}_{u_i}^{v_k^i}$	Sum of energy and delay for possible offloading decision of node u_i.
$\tilde{E}_{off}^{i,v_k^i}$	Normalized value of E_{off}^{i,v_k^i} in range of [0,1].
$\tilde{T}_{off}^{i,v_k^i}$	Normalized value of T_{off}^{i,v_k^i} in range of [0,1].
$\phi_{u_i}^{v_k^i}$	Initial offloading decision for node u_i.
Φ_0	The initial solution population.
Φ_z	Solution matrix, where $\Phi_z \in \Phi_0$.
E_{Φ_z}	Overall energy consumption of solution Φ_z.
T_{Φ_z}	Overall processing delay of solution Φ_z.
I_z	Proximity of solution z with other solutions in the objective space.
W	Maximum number of crossover operations.
P_C	Probability of performing crossover operation.
P_M	Probability of performing mutation operation.
$\mathcal{Q}_t$	Generated offspring set.

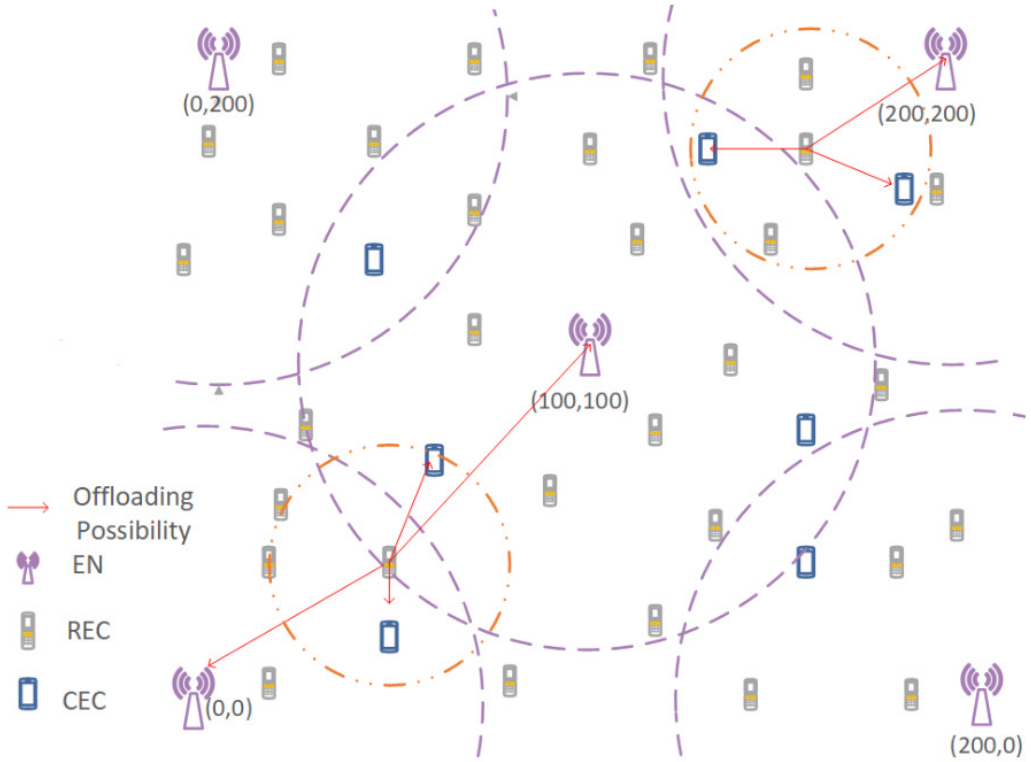


Fig. 2.9 An Architecture for Task Offloading in MEC

Consider an EC u_i having a computing task that needs to be processed, and its set of neighboring devices, which is given by

$$\begin{aligned}\mathcal{N}(i) &= \mathcal{N}_{EC}(i) \cup \mathcal{N}_{EN}(i) \\ &= \left\{u_j | d(u_i,u_j) \leq R_{\mathcal{U}}, \forall j\right\} \cup \left\{f_m | d(u_i,f_m) \leq R_{\mathcal{F}}, \forall m\right\} \\ &= \left\{v_1^i, \ldots, v_k^i, \ldots, v_{N_i}^i\right\} \end{aligned} \tag{2.41}$$

where $d(\cdot,\cdot)$ denotes the Euclidean distance operator between devices, $R_{\mathcal{U}}$ is the coverage range of ECs, $R_{\mathcal{F}}$ is the coverage range of ENs, v_k^i identifies the generic kth device belonging to the neighborhood of u_i, whether u_i is an EC or an EN, and N_i is the total number of neighbors of EC u_i. Note that the neighbor set cardinality is variable across the nodes, and that a device can belong to different neighbor sets if it can be served by more than one device.

The task offloading operation can be performed in two ways. The first way is to offload the computing task from an EC as a whole to a single device. The second way is to first partition the EC's computing task into several segments that can be computed in parallel, and then offload each task to a different device. We call the second approach task partitioning. Since task partitioning offers more opportunities to utilize the idle computing resources in the network, we adopt this approach in our system model.

Each EC can offload a set of portions of a single task to another EC or EN. We assume that each set of offloaded portions can be individually outsourced and computed in parallel. For instance, in Fig. 2.10, EC u_i has one task with ten portions, and three available devices to offload, out of which two are other ECs and one is an EN. In this example, five fixed-size portions are offloaded to u_1, two are offloaded to u_2 and three are offloaded to f_1. The set of portions to be estimated and shared from EC u_i to EC u_j and from EC u_i to EN f_m, is denoted as $\phi_{u_i}^{u_j}$ and $\phi_{u_i}^{f_m}$, respectively.[2]

To partition the tasks into meaningful portions, the portion sizes are set equal to a multiple of a minimum size. The minimum portion size is given by the smallest number of bytes that are needed to convey an instruction of the task's application, e.g., a given number of floating point operations (FLOPS). Moreover, we assume the task result size is smaller than size of the offloaded task. Let δ_{tx} be the minimum task portion size which can be transmitted, and δ_{rx} the minimum task portion size which can be received. Thus, the ratio between one offloaded portion and one downloaded portion is defined as $\alpha = \delta_{tx}/\delta_{rx}$, where $\alpha > 1$, and depends on the application type. Now, let D_s^i be the size of u_i's task to be offloaded, and D_r^i be the size of u_i's task to be downloaded. Then, the number of portions in u_i's task is given

[2] Since some of the neighbors of EC may offer a long task processing delay, or result in high energy consumption, the EC may decide to not outsource any task portion to some of its neighbors.

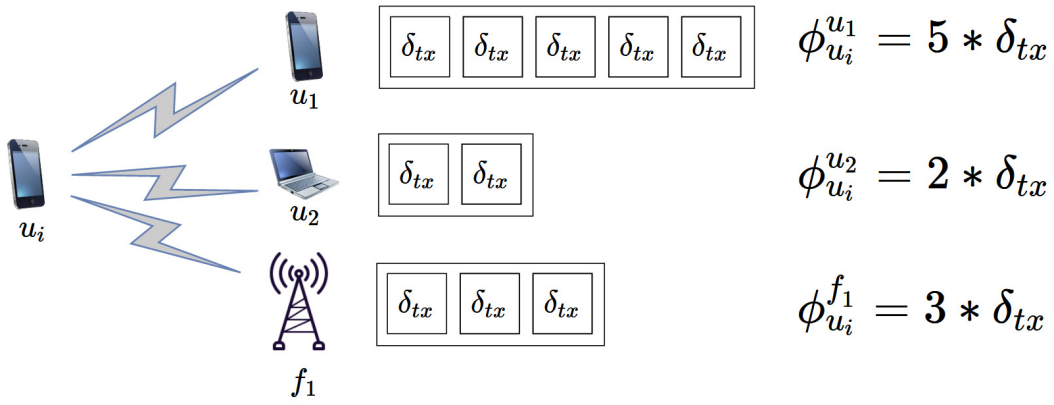


Fig. 2.10 Task portion distribution

by:

$$\gamma_i = \frac{D_s^i}{\delta_{tx}} = \frac{D_r^i}{\delta_{rx}} \tag{2.42}$$

where we assume that we can pad the task size to make D_s^i a multiple of δ_{tx} and, similarly, D_r^i a multiple of δ_{rx}. Let $\phi_{u_i}^{u_j}$ and $\phi_{u_i}^{f_m}$ be the portion of task shared from EC u_i to EC u_j and from EC u_i to EN f_m, respectively. Both $\phi_{u_i}^{u_j}$'s and $\phi_{u_i}^{f_m}$'s are positive integers less than γ_i. For all the offloaded portions we have the following:

$$\sum_{j=1}^{N} \phi_{u_i}^{u_j} + \sum_{m=1}^{M} \phi_{u_i}^{f_m} = \gamma_i. \tag{2.43}$$

To offload tasks, the ECs collaborate with other ECs and ENs in three phases. First, the originating EC transmits the task's data to a neighboring EC or EN. Second, the neighboring node completes the computational task. Third, the originating EC downloads the task result from the neighboring node. Thus, the originating EC's energy consumption when offloading a single portion[3] of size δ_{tx} to a single neighboring device is:

$$E_{off}^{i,v_k^i} = E_{tx}^{i,v_k^i} + E_{rx}^{i,v_k^i}, \qquad \forall v_k^i \in \mathcal{N}(i) \tag{2.44}$$

[3]For a single portion, $\phi_{u_i}^{u_j}$ or $\phi_{u_i}^{f_m}$ equals to one. Note that the sum of these two parameters, which represent the total number of offloading portions, should be equal to γ_i as defined in (2.43).

where E_{tx}^{i,v_k^i} and E_{rx}^{i,v_k^i} are the energy spent for transmitting a single task portion and for receiving the results from [4] remote node $v_k^i \in \mathcal{N}(i)$, respectively, Now, the total energy consumption of the EC is given by

$$\bar{E}_{off}^{i} = \sum_{u_j, f_m \in \mathcal{N}(i)} \left(\phi_{u_i}^{u_j} \left(E_{tx}^{i,u_j} + E_{rx}^{i,u_j} \right) + \phi_{u_j}^{u_i} \cdot E_{com}^{i} + \phi_{u_i}^{f_m} \left(E_{tx}^{i,f_m} + E_{rx}^{i,f_m} \right) \right) + E_{id}^{i}, \quad (2.45)$$

where E_{com}^{i} represents the energy spent by the EC u_i to compute a single portion of a neighboring client, while E_{id}^{i} denotes the energy consumption of EC u_i when it is idle, i.e., neither transmitting, receiving or computing.

Similarly, we model the offloading delay for a single portion as:

$$T_{off}^{i,v_k^i} = T_{tx}^{i,v_k^i} + T_{rx}^{i,v_k^i} + T_{com}^{i,v_k^i}, \qquad \forall v_k^i \in \mathcal{N}(i) \quad (2.46)$$

where T_{tx}^{i,v_k^i} and T_{rx}^{i,v_k^i} are the time needed to transmit a single task portion to a remote device and the time needed to receive the results, respectively, and T_{com}^{i,v_k^i} is the computation time of the task portion at the remote device v_k^i. The time period T_{com}^{i,v_k^i} is also the time that the EC u_i waits to between uploading the task portion and receiving the result. Given a task portion with size δ_{tx}, the transmission time is defined as:

$$T_{tx}^{i,v_k^i} = \frac{\delta_{tx}}{r_{i,v_k^i}} \quad (2.47)$$

and the corresponding receiving time as:

$$T_{rx}^{i,v_k^i} = \frac{\delta_{rx}}{r_{i,v_k^i}} \quad (2.48)$$

where r_{i,v_k^i} is the data rate of the link between the EC u_i and the remote device v_k^i in the set of neighboring nodes $\mathcal{N}(i)$.

In this work, we consider that ECs generate tasks with variable size. We define the index of a single task as l. Hence, the time spent by device v_k^i to compute one portion of the lth task produced by the EC u_i corresponds to:

$$T_{com}^{i,v_k^i} = \begin{cases} \frac{O_l^i/\gamma_i}{\eta_{c_j}}, & \text{if } v_k^i \in \mathcal{N}_{EC} \\ \frac{O_l^i/\gamma_i}{\eta_{c_m}}, & \text{if } v_k^i \in \mathcal{N}_{EN} \end{cases} \quad (2.49)$$

[4] Note that the reception energy is smaller than the transmission energy because $\delta_{tx} > \delta_{rx}$.

where O_l^i represents the number of processing operations related to the lth task produced by the EC u_i, and η_{c_j} and η_{c_m} are the FLOPS depending on the CPU of the EC u_j or the EN f_m, respectively. Thus, O_l^i/γ_i represents the number of processing operations for a portion.

A remote device v_k^i may receive multiple task portions from other ECs or ENs. Therefore, the arriving task portion of EC u_i at the remote device v_k^i may experience a waiting time before it is processed. Thus, we can write the overall offloading time as follows:

$$\bar{T}_{off}^i = \max_{u_j, f_m \in \mathcal{N}(i)} \left(\phi_{u_i}^{u_j} \cdot T_{off}^{i,u_j} + T_w^{i,u_j}, \phi_{u_i}^{f_m} \cdot T_{off}^{i,f_m} + T_w^{i,f_m} \right) \tag{2.50}$$

where T_w^{i,u_j} is the task portion waiting time at the remote node due to other processes.

Our goal is to efficiently find the task portions that ECs should offload to their neighboring ECs, and to their neighboring ENs, i.e., $\phi_{u_i}^{u_j}$'s and $\phi_{u_i}^{f_m}$'s, respectively, that simultaneously minimize the EC's overall energy consumption and task processing delay.

A Constrained Multi-objective Optimization Problem for Task Offloading in Edge Computing

$$\min_{\Phi} \left\{ \sum_{i=1}^{N} \left(\sum_{u_j, f_m \in \mathcal{N}(i)} \left(\phi_{u_i}^{u_j} \left(E_{tx}^{i,u_j} + E_{rx}^{i,u_j} \right) + \phi_{u_j}^{u_i} \cdot E_{com}^i + \phi_{u_i}^{f_m} \left(E_{tx}^{i,f_m} + E_{rx}^{i,f_m} \right) \right) \right), \right.$$
$$\left. \sum_{i=1}^{N} \left(\max_{u_j, f_m \in \mathcal{N}(i)} \left\{ \phi_{u_i}^{u_j} \left(\frac{\delta_{tx}}{r_{i,u_j}} + \frac{\delta_{rx}}{r_{i,u_j}} + \frac{O_l^i/\gamma_i}{\eta_{c_j}} \right) + T_w^{i,u_j}, \phi_{u_i}^{f_m} \left(\frac{\delta_{tx}}{r_{i,f_m}} + \frac{\delta_{rx}}{r_{i,f_m}} + \frac{O_l^i/\gamma_i}{\eta_{c_m}} \right) + T_w^{i,f_m} \right\} \right) \right\} \tag{2.51}$$

To define the task offloading problem, we first oberve that an EC aiming to minimize its task processing delay would attempt to offload task portions to as many neighboring devices in order to exploit their computing resources in parallel.

To define the task offloading problem, we observe that an EC aiming to minimize its task processing delay would attempt to offload task portions to as many neighboring devices as possible. The reason is that this would allow it to exploit the neighbors' computing resources in parallel. However, increasing the number of devices for offloading incurs in an increased energy consumption due to the additional energy needed for transmission and reception to devices that are further away. Indeed, in the considered task offloading operation, delay is minimized when

$$\max \sum_{k=1}^{|\mathcal{N}(i)|} \mathbb{1}\{\phi_{u_i}^{v_k^i} > 0\}$$

for every single task of u_i that we have. This drives to the observation that the EC's energy consumption and task processing delays are competing objectives.

To efficiently handle these two competing objectives, we formulate the task offloading problem in edge computing as a CMOP in (2.51), subject to the constraints:

$$Eq.(2.43), \tag{2.52a}$$
$$\phi_{u_i}^{u_j} = 0, \quad \text{if } u_j \notin \mathcal{N}(i), \tag{2.52b}$$
$$\phi_{u_i}^{f_m} = 0, \quad \text{if } f_m \notin \mathcal{N}(i), \tag{2.52c}$$
$$\phi_{u_i}^{u_j} \geq 0, \quad \text{if } u_j \in \mathcal{N}(i), \tag{2.52d}$$
$$\phi_{u_i}^{f_m} \geq 0, \quad \text{if } f_m \in \mathcal{N}(i), \tag{2.52e}$$

where Φ is the variable vector denoting the task offloading decisions for the ECs, i.e.,

$$\Phi = \begin{pmatrix} \phi_{u_1}^{u_1} & \cdots & \phi_{u_1}^{u_j} & \cdots & \phi_{u_1}^{u_N} & \phi_{u_1}^{f_1} & \cdots & \phi_{u_1}^{f_m} & \cdots & \phi_{u_1}^{f_M} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi_{u_i}^{u_1} & \cdots & \phi_{u_i}^{u_j} & \cdots & \phi_{u_i}^{u_N} & \phi_{u_i}^{f_1} & \cdots & \phi_{u_i}^{f_m} & \cdots & \phi_{u_i}^{f_M} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi_{u_N}^{u_1} & \cdots & \phi_{u_N}^{u_j} & \cdots & \phi_{u_N}^{u_N} & \phi_{u_N}^{f_1} & \cdots & \phi_{u_N}^{f_m} & \cdots & \phi_{u_N}^{f_M} \end{pmatrix}. \tag{2.53}$$

In (2.51), the first objective minimizes the EC's energy consumption, and the second objective minimizes the task processing delay. Constraint (2.52a) guarantees that all task portions are computed either locally or at a remote node, as already defined in (2.43). Constraints (2.52b), and (2.52c) prevent the ECs from offloading tasks to non-neighboring nodes, and (2.52d) and (2.52e) constraints the offloading decisions to non-negative values.

Characterization of the Pareto-optimal Front for Task Offloading CMOP in Edge Computing

Unlike single-objective optimization problems where there is a unique solution, the task offloading CMOP in edge computing is characterized by a Pareto front of solutions.

In the presence of multiple conflicting objectives, any solution point has to be gauged along multiple dimensions. Hence, the quality of a solution is determined by its Pareto-dominance with respect to other solutions. In particular, let $\Phi = \{\Phi_1, \Phi_2, ..., \Phi_Z\}$ be the set of solutions, where Φ_z is the zth solution as represented in (2.53), and Z is the total number of generated solutions. Considering two solutions, say Φ_1 and Φ_2, for a given problem with C conflicting objectives, say ω_c (for all $c \in [1, C]$), we define Pareto-dominance as follows:

Definition 2.3.1. *Let $\omega_c(\Phi)$ be the value of the objective function for the cth objective evaluated at some solution Φ. Then Φ_1 is said to Pareto-dominate Φ_2 (i.e., $\Phi_1 \succ \Phi_2$) if $\omega_c(\Phi_1) \leq \omega_c(\Phi_2)$ for all $c \in [1,C]$, and there exists some $p \in [1,C]$ such that $\omega_p(\Phi_1) < \omega_p(\Phi_2)$.*

Although the above Pareto-dominance definition allows to classify solutions based on their quality, it treats feasible and unfeasible solutions equally. To favor feasible solutions and penalize those that violate the constraints, we adopt the constraint-dominance definition proposed in [77], i.e.,

Definition 2.3.2. *A solution vector Φ_1 is said to constraint-dominate another solution vector Φ_2 if any of the following conditions is true:*

1. *Φ_1 is feasible, i.e., it satisfies all constraints, but Φ_2 is not.*
2. *Both Φ_1 and Φ_2 are feasible and Φ_1 Pareto-dominates Φ_2.*
3. *Both Φ_1 and Φ_2 are infeasible, but Φ_1 has lower overall constraint violation.*

Consequently, we can define the set of non-dominated solutions as:

$$S^P = \{\Phi_a \mid \nexists \Phi_b \succ \Phi_a, \text{ for } 1 \leq a,b \leq Z\} \tag{2.54}$$

In the following section, we design an evolutionary algorithm that can find high-quality approximations of the solution to the CMOP in (2.51).

2.3.2 An Evolutionary Algorithm for Task Offloading in Edge Computing

To solve the task offloading CMOP described in Section 2.3.1, we propose a Multi-Objective Evolutionary Algorithm (MOEA). The main idea of a MOEA is to use the evolutionary principles of crossover, mutation, and selection of Darwinian evolution to find the Pareto front. Crossover and mutation probabilistically combine solutions to find possibly better new solutions, while selection deterministically discards low-quality solutions and keeps high-quality ones. Compared to other methods for multi-objective optimization, MOEAs offer an efficient way to find a high-quality approximation of the Pareto-optimal front in a single run.

Specifically, our proposed MOEA operates in four steps: initialization, selection, reproduction, and population update. First, the proposed algorithm randomly generates an initial solution population, and ranks these solutions based on their quality using selection. In

the reproduction step, the proposed algorithm probabilistically combines the high-quality solutions with each other to generate possibly better new ones. Then, the proposed algorithm repeats the selection and reproduction steps until it reaches a maximum number of iterations. In the following, we explain these steps in details.

Initialization

To reduce the number of iterations needed to find a high-quality approximation of the Pareto-optimal front, we develop a two-phase initialization procedure that leverages the structure of the task offloading problem to find a high-quality initial solutions set. The proposed method generates initial solutions with better quality compared to the ones generated by random initialization, which allows us to find a high-quality approximation of the Pareto-optimal front in a low number of iterations. We measure the quality of our initial solutions in the numerical result section.

In the first phase, we discard the devices that have low energy levels. The reason is that selecting these devices to compute the tasks from the ECs could deplete their energy. In addition, discarding low energy devices significantly reduces the solution space. In the second phase, we propose a weighted random solution generation that favors offloading decisions that delegate tasks between nearby nodes, resulting in initial offloading decisions with low energy consumption and task processing delay.

Phase 1: EC Classification We divide EC's into two subsets depending on their energy level. The first subset contains ECs that have enough energy to perform the computing tasks on behalf of other ECs. We call this subset the Computing EC's (CEC) subset. The second subset contains EC's that lack enough energy to perform their own computations, and thus offload their tasks to other EC's. We call this subset the Requesting EC's (REC) subset. In order to perform this classification there could be several methods aiming at sorting the devices and dividing the set into two subsets. In this work, we classify the ECs based on the distribution of their remaining energy at the moment the classification is performed. This approach allows us to take into account the actual energy level of the ECs. In particular, the two sets are selected based on a 3-quantile function, so that they are always balanced in numerosity [79]. To this aim, we formally define the CEC and REC subsets as follows:

$$\mathcal{U}_{REC} = \left\{u_i | E_r^i \leq E_{r,Q2}\right\} \tag{2.55a}$$

$$\mathcal{U}_{CEC} = \left\{u_i | E_r^i > E_{r,Q2}\right\} \tag{2.55b}$$

$$\mathcal{U} = \mathcal{U}_{REC} \cup \mathcal{U}_{CEC} \tag{2.55c}$$

where E_r^i is the remaining energy of the EC u_i and:

$$E_{r,Q2} = \inf\left\{E_r^i, i = 1,\ldots,N | p \leq F_{E_r}(i)\right\} \tag{2.56}$$

is the quantile function, where $\inf\{\cdot\}$ is the *infimum* operator, p is equal to 2/3 for the upper 3-quantile index and $F_{E_r^i}$ represents the distribution of the remaining energy of all the ECs. Equation (2.56) aims to select the remaining energy value that divides the set of ECs in 1/3 having a remaining energy higher than $E_{r,Q2}$ and 2/3 having a remaining energy lower than $E_{r,Q2}$. Note that we have considered one of the indexes (upper index) for the classification mainly due to two reasons. First it allows to classify the devices into two subsets. Moreover, the upper index allows to select higher number of users to act as the REC, due to the fact that each CEC can perform the computation for several RECs. RECs can only offload tasks to CEC or EN that are within their transmission range. Hence, EC u_i's set of CEC that are within its reach is given by

$$\mathcal{N}_{CEC}(i) = \left\{u_j | d(u_i,u_j) \leq R_{\mathcal{U}}, \forall u_j \in \mathcal{U}_{CEC}\right\}, \tag{2.57}$$

and the set of ENs that are within its reach is defined as

$$\mathcal{N}_{EN}(i) = \{f_m | d(u_i,f_m) \leq R_{\mathcal{F}}, \forall f_m \in \mathcal{F}\}$$

for all $u_i \in \mathcal{U}_{REC}$.

Besides, we denote the number of ECs in the REC subset, the number of CECs within reach of u_i, and the number of ENs within reach of u_i by $\bar{n} = |\mathcal{U}_{REC}|$, $\hat{n} = |\mathcal{N}_{CEC}(i)|$, and $\hat{f} = |\mathcal{N}_{EN}(i)|$, respectively.

Phase 2: Initial Solution Set Generation The main idea of our initial solution generation algorithm is to form solutions that prioritize offloading tasks to EC nodes, or ENs, with low energy consumption and low task processing delay relative to other nodes.

In particular, suppose a requesting EC $u_i \in \mathcal{U}_{REC}$ is seeking to offload one task with one portion. Then, to find an initial offloading decision for u_i,

we first calculate the sum of energy consumption and task processing delay that u_i would experience by offloading its task portion to one of its neighboring CEC or EN nodes, i.e.,

$$\tilde{w}_{u_i}^{v_k^i} = \tilde{E}_{off}^{u_i,v_k^i} + \tilde{T}_{off}^{u_i,v_k^i} \tag{2.58}$$

for all $k \in [1,K_i]$, where $\tilde{E}_{off}^{i,v_k^i}$ and $\tilde{T}_{off}^{i,v_k^i}$ are obtained by normalizing E_{off}^{i,v_k^i} and T_{off}^{i,v_k^i} to the range $[0,1]$ using min-max re-scaling [80], and K_i is the number of devices that can accept

node u_i's task portions. By further normalizing $\tilde{w}_{u_i}^{v_k^i}$ to be in the range $[0, \gamma_i]$, we define the upper bound on the number of the task portions for node u_i that should be offloaded to node v_k^i, by $w_{u_i}^{v_k^i} \in [0, \gamma_i]$.

Based on these upper bounds, the initial offloading decisions is as follows:

$$\phi_{u_i}^{v_k^i} \sim \text{unif}\{0, w_{u_i}^{v_k^i}\} \tag{2.59}$$

for all $k \in [1, K_i]$, and $u_i \in \mathcal{U}_{REC}$, where unif denotes the discrete uniform distribution between zero and $w_{u_i}^{v_k^i}$

However, since the offloading decisions in (2.59) may not satisfy constraint (2.52a), i.e., the sum of an EC's offloading decisions may not be equal to the total number of task portions $\sum_{k=1}^{K_i} \phi_{u_i}^{v_k^i} \neq \gamma_i$, we replace the offloading decisions in (2.59) as follows:

$$\phi_{u_i}^{max} \leftarrow \phi_{u_i}^{max} + \gamma_i - \sum_{k=1}^{K_i} \phi_{u_i}^{v_k^i} \tag{2.60}$$

where $\phi_{u_i}^{max} = \max_{\forall k \in [1,K_i]} \{\phi_{u_i}^{v_k^i}\}$.

The initial solution population $\Phi_0 = \{\Phi_1, \Phi_2, ..., \Phi_Z\}$ is generated by repeating the above procedure for each offloading decision $\phi_{u_i}^{v_k^i} \in \Phi_z$ in every solution matrix Φ_z for all $\Phi_z \in \Phi_0$. We summarize the proposed weighted initial solution generator in Algorithm 6.

Algorithm 6 Weighted initial solution generator

Require: $\tilde{E}_{off}^{u_i, v_k^i}$, $\tilde{T}_{off}^{u_i, v_k^i}$

Find $\tilde{w}_{u_i}^{v_k^i}$ using (2.58) for all $k \in [1, K_i]$ and $u_i \in \mathcal{U}_{REC}$
for $k = 1$ to K_i **do**
  Find $w_{u_i}^{v_k^i}$ by normalizing $\tilde{w}_{u_i}^{v_k^i}$ to the range $[0, \gamma_i]$, and rounding to the nearest integer.
  Set $\phi_{u_i}^{v_k^i}$ equal to a random integer drawn from the distribution unif$\{0, w_{u_i}^{v_k^i}\}$.
end for
Adjust the largest $\phi_{u_i}^{v_k^i}$ to meet constraint (2.52a) using (2.60)

Ensure: $\phi_{u_i}^{v_k^i}$

Selection

After generating the initial population, the proposed algorithm ranks the initial solutions based on their quality, and their distance to their nearest neighbors in the objective space. Specifically, the selection procedure first calculates the overall energy consumption and processing delay of solution $\Phi_z \in \Phi$ as:

$$E_{\Phi_z} = \sum_{u_i \in \mathcal{U}_{REC}} \bar{E}^i_{off} \tag{2.61}$$

$$T_{\Phi_z} = \max_{u_i \in \mathcal{U}_{REC}} \left\{ \bar{T}^i_{off} \right\} \tag{2.62}$$

for all $z \in [1, Z]$, where $\bar{E}^i_{off}$ and $\bar{T}^i_{off}$ are given by (2.45) and (2.50), respectively.

Then, using the constraint-dominance relationship explained in *Definition 2*, all individuals in Φ_0 are compared to each other and assigned a rank to determine the number of times each individual is dominated by the others. For example, an individual that is dominated by 10 other individuals is assigned a rank of 11, and a non-dominated individual receives a rank of 1.

An MOEA is desired to have a good diversity of non-dominated solutions converging to the Pareto-optimal front. To maintain this diversity, we further classify solutions based on the crowding distance which measures the proximity of an individual with others in the objective space, and denote it using I_z [77]. Crowding-distance calculation procedure is described in Algorithm 7.

Algorithm 7 Crowding-distance

Require: ϕ_z for $z \in [1, Z]$, C
  for each ϕ_z, calculate the objective values $\omega_{z,1}, ..., \omega_{z,c}$ **do**
    Define individual I_z and set it to 0 for each individual ϕ_z
    for c=1 to C **do**
      Sort individuals ϕ_z in Z in ascending order according to $\omega_{z,c}$
      The crowding distance of the first and the last individual
      are set to infinity
      for z=2 to $Z-1$ **do**
        $I_z = I_z + \frac{\omega_{z-1,c} - \omega_{z+1,c}}{\max_z\{\omega_{z,c}\} - \min_z\{\omega_{z,c}\}}$
      end for
    end for
  end for
Ensure: Crowding-distance of individual I_z

Reproduction

To generate a new set of possibly better solutions, the proposed algorithm performs the reproduction step through the following genetic operations: binary tournament, crossover, and mutation. The algorithm first applies the binary tournament operation to the mating pool, which is the set of solutions that will be combined by the crossover operation to form new ones. We denote the mating pool by Φ^M_0. The binary operation consists in randomly

selecting two solutions, Φ_i and Φ_j, from the current solution set Φ_0, and then adding the solution with the better dominance rank assigned during the selection phase to Φ_0. If both Φ_i and Φ_j have the same rank, the solution with larger crowding distance is selected. If both have equal crowding distance, one is chosen at random.

The algorithm repeats the binary tournament until the maximum number m of solutions has been added to the mating pool.

Next, the crossover operation combines the solutions in the mating pool to generate new ones. The main idea of crossover is to randomly choose two solutions Φ_i and Φ_j from the mating pool Φ_0^M, and generate two child solutions Φ'_i and Φ'_j formed by combining parts of the parent solutions. Specifically, the crossover operation first randomly chooses the set of crossover points $\mathcal{D} = \{d_1, d_2, ..., d_D\}$ from the interval $[1, \bar{n}]$, where $d_i < d_{i+1}$ (for all $i \in [1, D]$). Then, children Φ'_i and Φ'_j are formed by alternating columns from Φ_i and Φ_j according to the crossover points, that is, $\Phi'_i = [\phi_1^i, ..., \phi_{d_1}^i, \phi_{d_1+1}^j, ..., \phi_{d_2}^j, ..., \phi_{d_{D-1}}^i, ..., \phi_D^i]$, and $\Phi'_j = [\phi_1^j, ..., \phi_{d_1}^j, \phi_{d_1+1}^i, ..., \phi_{d_2}^i, ..., \phi_{d_{D-1}}^j, ..., \phi_D^j]$, as shown in Fig. 2.11. The child solutions are added to the set of child solutions $\mathcal{Q}_0$. The crossover operation selects two solutions from the mating pool W times, where W is the maximum number of crossovers. However, the crossover operation is only performed with probability P_C.

To apply the mutation operation, we first randomly choose a child solution $\Phi'_i \in \mathcal{Q}_0$ with probability P_M. If a child solution is chosen, we replace a randomly chosen column with randomly generated offloading decisions. To ensure the new solution remains feasible, we apply equation (2.60). For example, in Fig. 2.12, the child solution Φ'_j has been chosen for mutation, and the offloading decisions in the 4th column have been replaced by random decisions, where $\gamma_4 = 30$.

Population Update

Once the offspring population $\mathcal{Q}_0$ has been generated and updated with mutations, we form the new solution population Φ_1 by discarding the low-quality solutions in the initial population Φ_0, and in the offspring population $\mathcal{Q}_0$. To this end, we first form an aggregate solution population $\mathcal{A}_0 = \Phi_0 \cup \mathcal{Q}_0$, and calculate the rank and crowding distance of the solutions in $\mathcal{A}_0$ as described in Section 2.3.2. Then, we form the new population set Φ_1 by adding solutions from $\mathcal{A}_0$ in descending rank order until the maximum size of Φ_1 has been reached. In other words, we add solutions with rank 1 first, then, if there are unfilled positions in Φ_1, we add solutions with rank 2, and so on.

In the tth iteration of the algorithm, we apply the selection, and reproduction steps to the solution population Φ_t, generate the offspring set $\mathcal{Q}_t$, and the aggregate population $\mathcal{A}_t = \Phi_t \cup \mathcal{Q}_t$. Then, the new solution population Φ_{t+1} is filled with the top-ranked solutions

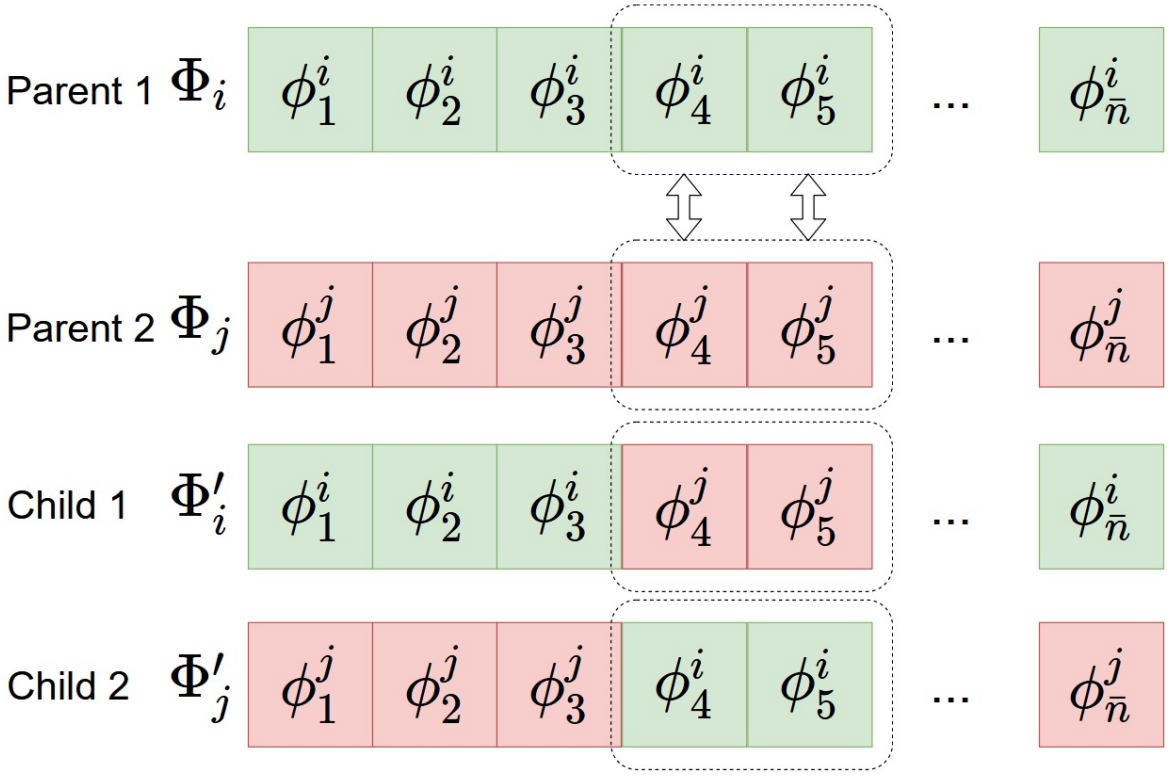


Fig. 2.11 Crossover operation on parent solutions.

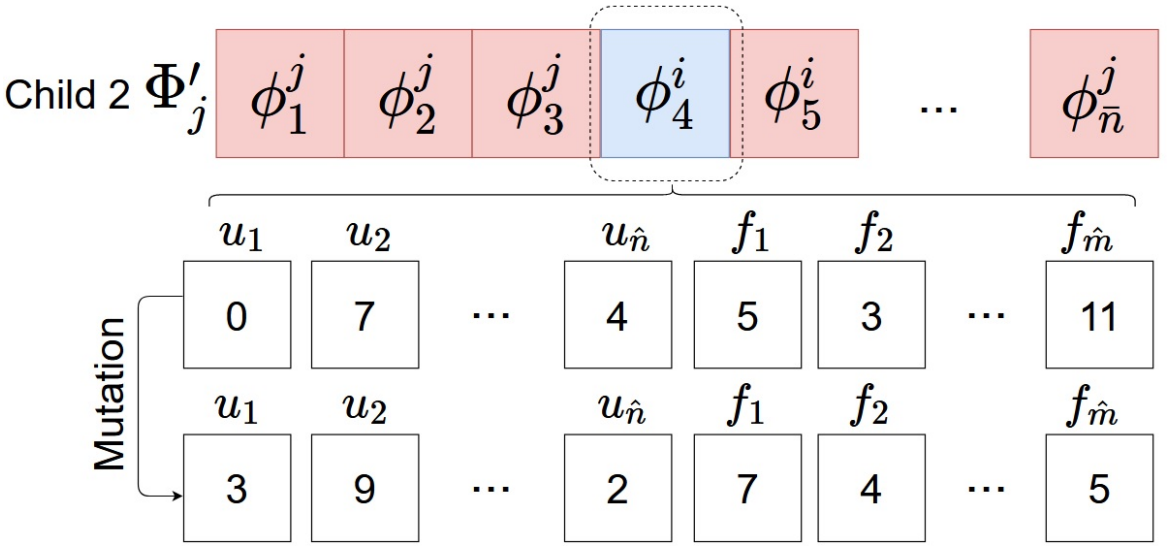


Fig. 2.12 Mutation operation on the selected solution (Child 2)

in $\mathcal{A}_t$ as described above. The iteration continues until $t = X$. We summarize the proposed Non-dominating Sorting Genetic Algorithm (*NSGA2*)-based algorithm in Algorithm 8.

Algorithm 8 Multi-objective offloading decision algorithm

Require: REC and CEC nodes
  Execute the initialization procedure
  Generate the initial population Φ_0 of size Z, given the
  constraints (9)
  for each $\phi_z \in \Phi_0$ **do**
    for each $u_i \in \mathcal{U}_{REC}$ **do**
      calculate E_{ϕ_Z} using (18)
      calculate T_{ϕ_Z} using (19)
    end for
    $\mathcal{E} = \mathcal{E} + E_{\phi_Z}$
    $\mathcal{T} = \mathcal{T} + T_{\phi_Z}$
    Sorting Φ_0 based on Pareto dominance relationship
    Determining crowding distance to maintain the diversity
    Using binary tournament to fill the mating pool
    Apply crossover on Φ_0 and fill $\mathcal{Q}_0$
    Apply mutation on $\mathcal{Q}_0$
    Forming $\mathcal{A}_0 = \Phi_0 \cup \mathcal{Q}_0$
    for t = 1 to X **do**
      Form Φ_t by adding sorted solutions from $\mathcal{A}_{t-1}$
      Generate $\mathcal{Q}_t$ by performing selection and reproduction on Φ_t
      Forming $\mathcal{A}_t = \Phi_t \cup \mathcal{Q}_t$
    end for
  end for
Ensure: Pareto Fronts to the MO computation sharing problem

2.3.3 Numerical Results

In this section, we present the numerical results obtained by our computer simulations. Specifically, we compare our *NSGA2*-based approach to an algorithm that assigns a task portion of equal size to each of the available devices (i.e., CECs and ENs). We call this straightforward solution the *fair allocation algorithm*. By comparing to the fair allocation algorithm, we can show that our proposed genetic-based approach can reduce both the energy consumption of ECs and task completion delay.

The computer simulations are performed in Matlab, where the considered parameters are listed in Table 2.6. The simulation is performed for 25000 seconds, aiming at comparing the performance in terms of average task delay, average EC energy consumption over time, and network lifetime, defined as:

- *Average Task Delay*: The average time spent for a task for transmitting, waiting, computing and receiving back the result over the number of generated tasks;

- *Average EC Energy Consumption Over Time*: The average energy all ECs have consumed per second;
- *Network Lifetime* $\kappa\%$: The time instant beyond which $\kappa\%$ of the ECs deplete their battery [13].

We hypothesize an area of 200×200 meters, with a variable number of heterogeneous ECs distributed randomly, while there are 5 ENs placed as seen in Fig. 2.9, in a way that each EC can be always served by at least one EN; the task generation is a Bernoulli process with average 0.02 tasks per second for each EC. ECs generate tasks according to a Bernoulli distribution with average $p = 0.1$ tasks per each simulation run. We have considered a battery capacity for all ECs equal to 5000 Jouls; however, each EC has a starting random energy level uniformly distributed between 50% and 100% of the battery capacity. We have considered two applications: a processing application generating tasks requiring a higher number of processing operations (e.g., image processing), and a collecting application, requiring a lower amount of processing operations, (e.g., sensor data analysis). In Table 2.7 the numerical values are reported, expressed in terms of Floating Point Operations (FLOP) per task data size. In order to have a realistic scenario, we have considered three device types of ECs nodes with different capabilities, defined in Table 2.8.

In the *NSGA2*-based solution, the algorithm runs for 1000 iterations to find the Pareto results (i.e., $X = 1000$). In order to analyze the impact of the number of iterations on the energy and delay, we introduced a third approach in the simulation results, named *Constrained NSGA2*, with a lower computational complexity, where $X = 500$.

We first investigate the impact of the proposed initialization approach on the quality of the generated final solutions. To do so, we run the simulation for a single run for five different sets of ECs, and two cases: when the *NSGA2*-based algorithm uses the Proposed Initialization (PI) approach, as introduced in Section 2.3.2, and when it uses the Random Initialization (RI) approach. Then, we compare the average energy consumption and average processing delay values obtained from these two cases and show them in Table 2.9. As shown in Table 2.9, the PI approach shows better results in both energy consumption and task processing delay, when compared to the RI approach. Increasing the number of ECs in the network adds more complexity to the network and creates a bigger solution space, making it more time consuming and harder to find good quality solutions. However, the good quality initial solutions generated using the PI approach, helps the *NSGA2*-based algorithm to find better solutions even when the network complexity increases.

At each iteration of the simulation, the output of *NSGA2*-based approaches is composed by 15 non-dominated pareto optimal solutions. To explore the different attributes of the

Table 2.6 Simulation Parameters for NSGA2 approach

Parameter	**Value**
Simulation Scenario Area	200m x 200m
Channel model	Outdoor RRH/Hotzone, Model 1: Pico to UE [76]
Channel Bandwidth	10 MHz
Max. number of solutions in mating pool (m)	100
Max. number of devices computing the offloaded task (K_i)	10
Crossover prob. (P_C)	0.8
Mutation prob. (P_M)	0.2
Max. number of crossovers (W)	100

Table 2.7 Task Parameters for NSGA2 Approach

Task Parameter	**Value**
Task size (D_s^i)	[1 5] MB
δ_{tx}, δ_{rx}	100 KB, 20 KB
Offloaded to downloaded portion (α)	5
Processing Application Operations	10 G FLOP per MB
Collecting Application Operations	1 G FLOP per MB

Table 2.8 Device Parameters for NSGA2 Approach

Parameter	**Coverage Range**	**Battery Capacity**	**Initial Energy**	**Computational Capability**	**Computational Power**	**Idle Power**	**Transmission Power**	**Reception Power**
High End EC	25 m	5000 J	[50 100]% battery capacity	25 G FLOPS	0.9 W	1.1 W	1.3 W	1.1 W
Low End EC	25 m	5000 J	[50 100]% battery capacity	15 G FLOPS	1.2 W	1.1 W	1.6 W	1.3 W
Heavy Duty EC	25 m	5000 J	[50 100]% battery capacity	20 G FLOPS	1.2 W	1.1 W	1.6 W	1.3 W
EN	100 m	-	-	150 G FLOPS	-	-	-	-

Table 2.9 Impact of proposed initialization approach

Number of ECs	**200**	**400**	**600**	**800**	**1000**
Energy-PI (J/s)	0.0204	0.0211	0.0224	0.0237	0.0245
Energy-RI (J/s)	0.0209	0.0217	0.0232	0.0249	0.0260
Improvement (%)	6.1	5.3	3.5	3.2	2.6
Delay-PI (s)	0.7719	0.6510	0.5309	0.4711	0.4515
Delay-RI (s)	0.8244	0.6959	0.5702	0.5258	0.5098
Improvement (%)	12.9	11.6	7.4	6.9	6.8

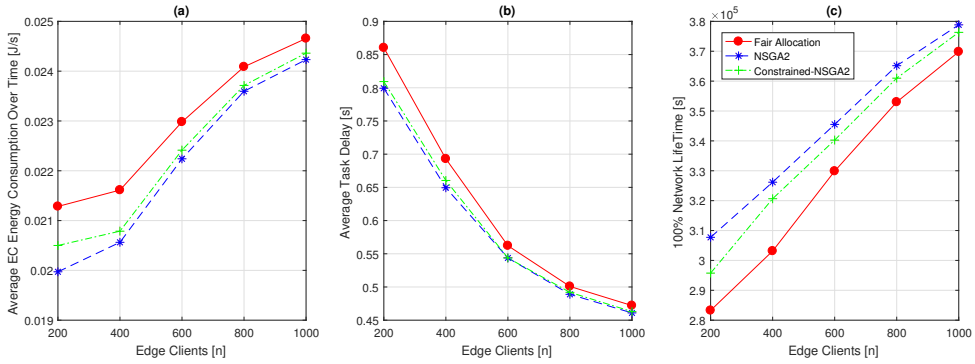


Fig. 2.13 Energy-based pareto selection

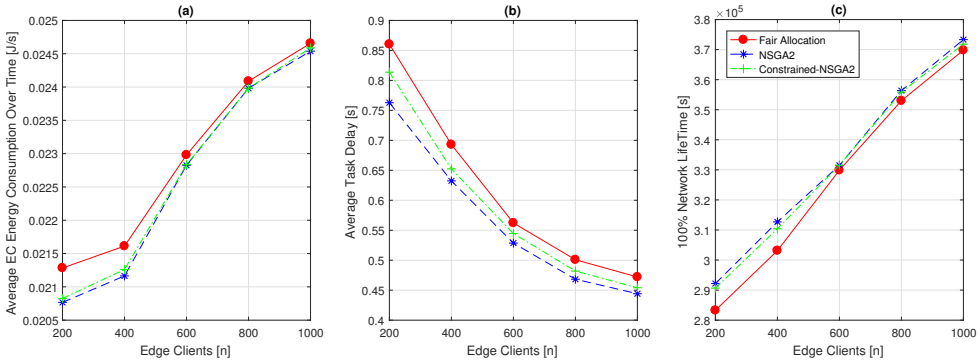


Fig. 2.14 Delay-based pareto selection

pareto optimal solutions, we considered three scenarios where once the best pareto solution in terms of energy consumption is selected (Fig. 2.13), once the solution with the best delay is selected (Fig. 2.14), and once the tradeoff solution with both relatively good delay and energy is selected (Fig. 2.15). Then, we analyze the average energy consumption (Figs. 2.13(a), 2.14(a), 2.15(a)), task delay (Figs. 2.13(b), 2.14(b), 2.15(b)), and the network life-time (Figs. 2.13(c), 2.14(c), 2.15(c)) for each of these three scenarios.

For the first scenario in which the selected pareto solution has the best energy consumption (Fig. 2.13), we observe the lowest average energy consumption compared to the other two scenarios. Although as a result of giving preference to energy the delay attribute should be sacrificed (Fig. 2.13(b)), the *NSGA2*-based approaches still show better delay compared to the Fair allocation approach. The same case is valid when we select the pareto solution with the lowest delay (Fig. 2.14). In this case, the *NSGA2* approach shows the lowest average task delay comparing to *Constrained NSGA2* and Fair allocation approaches. However, we still observe that the *NSGA2* approach shows an acceptable average energy consumption, very close to the *Constrained NSGA2* approach. It must be noted that for these scenarios, the selected solution for the *Constrained NSGA2* is the tradeoff point, where there is no

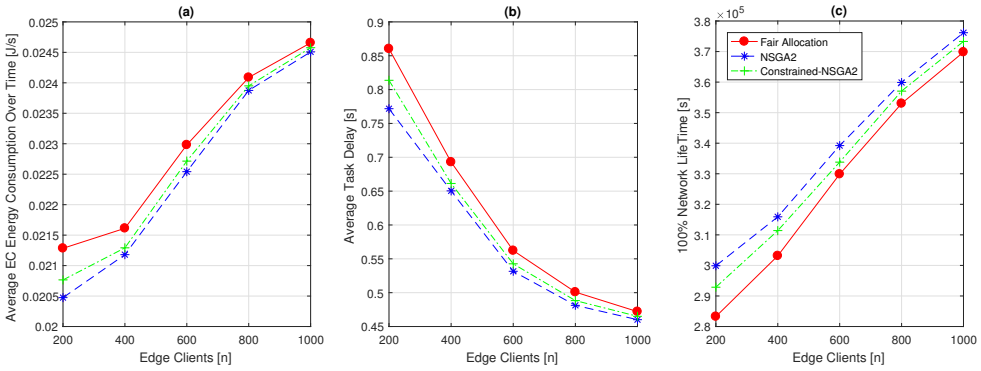


Fig. 2.15 Energy&Delay-based pareto selection

preference between energy and delay. Finally, when the tradeoff solution with relatively good delay and energy consumption is selected (Fig. 2.15), in both average task delay and energy consumption, *NSGA2*-based approaches perform better. However, the *NSGA2* approach performs better than the *Constrained NSGA2* approach, showing the impact of higher number of iterations that leads to higher variety of solutions generated when the number of iteration increases.

Furthermore, it must be noted that as the number of ECs increases, the number of interactions among ECs increases which is why the energy consumption figures tend to rise. In contrast to energy consumption, increasing the number of ECs will decrease the average task delay. Due to the high computational capability of the ECs and ENs for a single task a small delay is obtained, and therefore the ratio of average task delay over number of generated tasks becomes smaller by the increase in number of ECs. To overcome the complexity of the algorithm due to higher number of ECs, we have used the energy-based ECs classification and weighted probabilistic solution generation, which helps the algorithm to generate approximately high-quality solutions faster with fewer number of solutions. Although the *constrained NSGA2* approach has half of the time complexity of the proposed *NSGA2* approach, it fails to deeply investigate the solution domain and this is why it shows comparably higher values for energy and delay.

Figures (c) in Figs. 2.13, 2.14 and 2.15 show the network lifetime. The figures are closely related with the energy consumption figures. As shown in the figures, exploiting NSGA2 for optimization results in less energy consumption, and therefore prolonging the network lifetime. Moreover, the best performance is gained when the pareto solution is selected based on the lowest energy consumption.

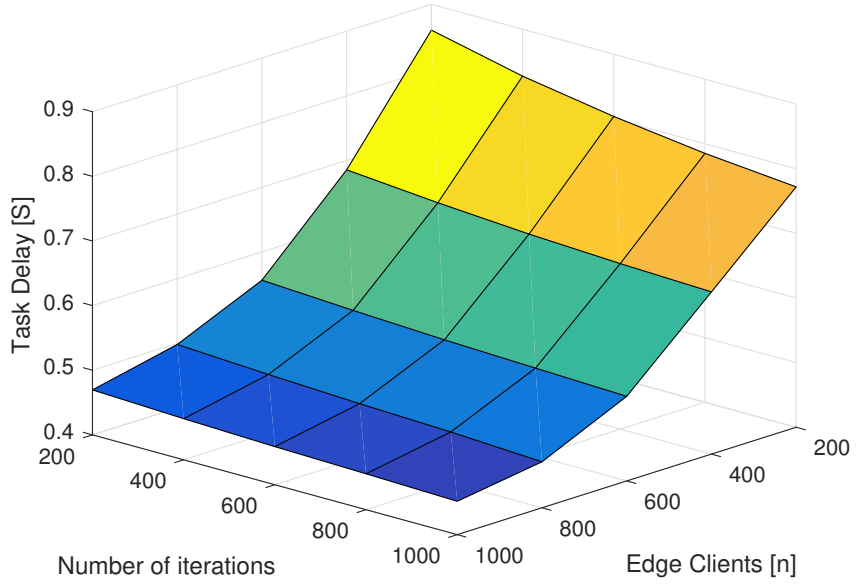
It is worth mentioning that number of ECs and iterations have huge impact on the estimated portions to be offloaded to the available nodes for computation. As seen in the previous figures, average task delay and average EC energy consumption are largely

influenced by these two important factors. Hence, to show the trend of the changes in the solutions obtained by *NSGA2*-based algorithms, we have conducted the simulation for a single simulation run (i.e., 5 seconds) and analyzed the impact of the number of ECs and iterations on latency and energy consumption.

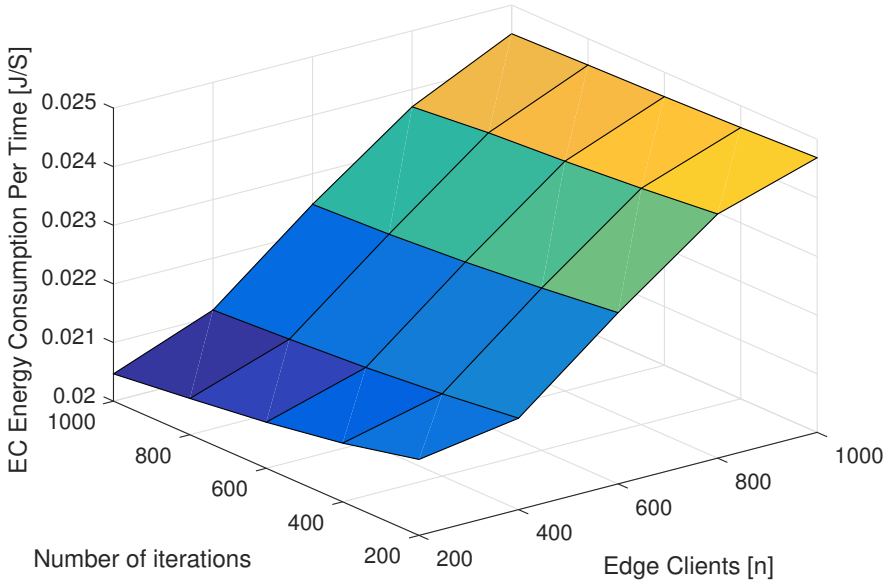
As seen in both Figs. 2.16a and 2.16b, as the number of ECs increases, the delay decreases and energy consumption increases. In particular, the higher number of task requests from RECs causes more interaction between the RECs and CECs, and subsequently, more energy consumption. In addition, the increase in the density of the RECs and CECs, leads to lower task processing delay. Furthermore, we can observe that increasing the number of iterations leads to improvement on energy consumption and processing delay of the tasks. This is due to the ability of the *NSGA2*-based to discover a variety of solutions, and generate possibly better ones using crossover and mutation operators.

However, the impact of number of iterations on the diversity of generated solutions degrades gradually and becomes less observable after some iterations. To analyze this impact, we fix the number of ECs to 1000 and plot the Pareto solutions of *NSGA2*-based for a single run, in Fig. 2.17. We can see from the figure that Pareto fronts for 200 iterations are more diverse and cover a higher range of values for energy consumption and task processing delay. As the number of iterations increases, this range becomes smaller, and reaches at its lowest for 1000 iterations.

We further analyze the generated Pareto fronts for different number of ECs and iterations, and show the results in Fig. 2.18. We can see the values of energy consumption and delay increase with the growth in number of ECs, while increasing the number of iterations helps slowing down this growth to some extent. We can see from the figure that increasing the number of iterations has more impact on the Pareto fronts, for the lower number of ECs in the network. Taking the case with 200 ECs as example, we see major improvement in energy consumption and task processing delay by increasing the number of iterations. On the other hand, and for the case of 1000 ECs, changing from 200 to 1000 iterations does not lead to much improvement in the objectives' values. The increase in network complexity (due to higher number of ECs) leads to an extremely large solutions space, which negatively impacts the ability of the *NSGA2*-based approach to generate diverse solutions and possibly better ones. However, we still observer that the *NSGA2*-based approach shows better results for energy consumption and task processing delay for all the 5 cases, when compared to the Fair allocation approach.



(a) Latency behavior by different number of Edge Clients and Iterations



(b) Energy consumption behavior by different number of Edge Clients and Iterations

Fig. 2.16 Impact of Number of Iterations and number of Edge Clients on latency and energy consumption

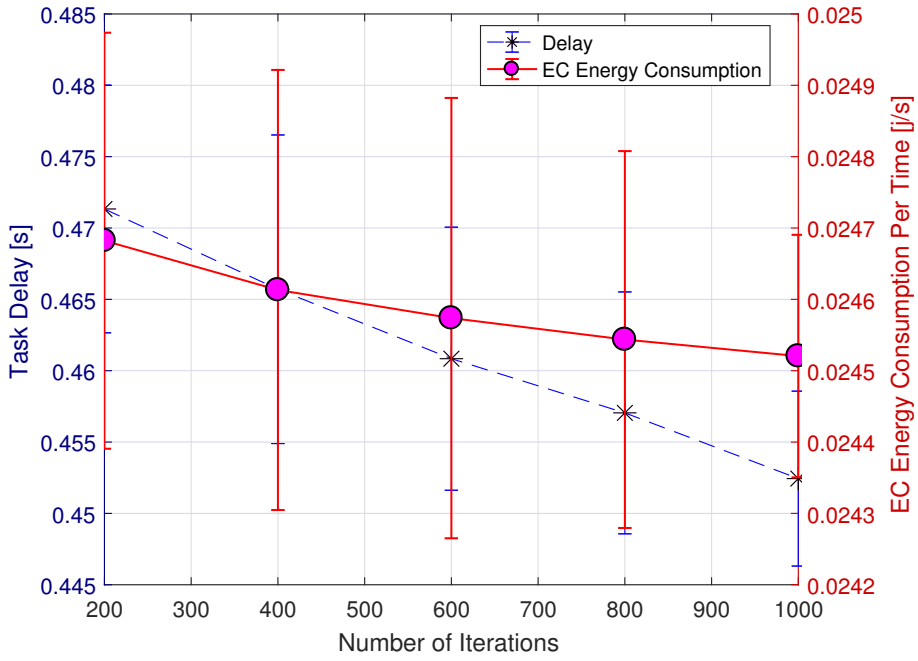


Fig. 2.17 Impact of Number of Iterations on latency and Energy Consumptions of the Pareto fronts

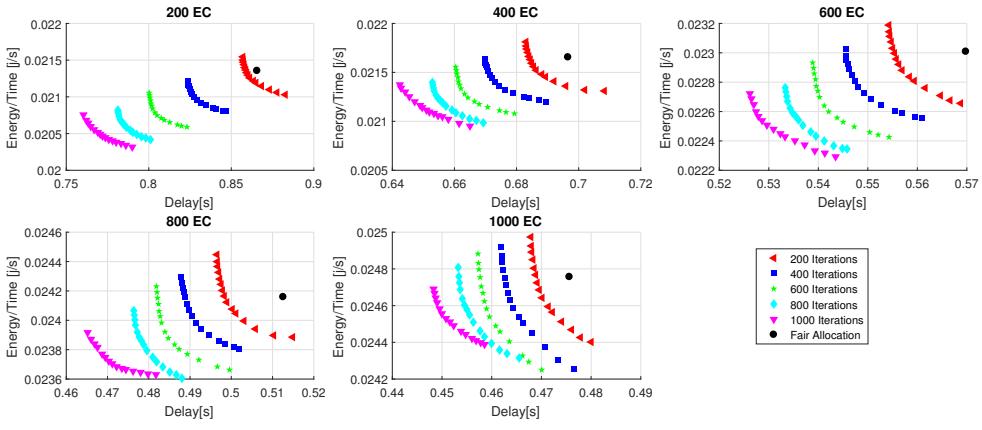


Fig. 2.18 Pareto fronts for different number of iterations and different number of Edge Clients.

2.3.4 Summary

In this work, we have considered a computation sharing problem in MEC. We have taken into consideration two layers in the network where ECs, in the first layer, can offload their computational tasks to the ENs, in the second layer, and also the other ECs. We have exploited NSGA2 for optimizing the computation sharing problem. The proposed multi-objective solution considers both average task delay and average EC energy consumption. We have demonstrated in the numerical results, that the results obtained through the evolutionary steps of the NSGA2 lead to optimize both energy and delay comparing with a benchmark solution considering an equal allocation of portions to the available nodes for computation offloading. Moreover, the proposed NSGA2-based approaches result in prolonging the network lifetime by optimizing the amount to be offloaded by the ECs. Furthermore, we have analyzed the effect of number of iterations in the genetic algorithm and have observed the convergence of the Pareto fronts in different scenarios with variable number of ECs. In the end, it can be noticed that the *Constrained-NSGA2* approach with a quite low time complexity can have results close to the *NSGA2* approach.

Chapter 3

Energy Harvesting Solutions

The content of the following chapter was extracted from publications [4] and [11] in the publications list.

last decades have been characterized by an increasing number of pervasive devices that have been deployed in the environments to support a wide range of applications, from environmental monitoring [2] and in-home automation [3] to Smart-cities [4] and Industry 4.0 [5]. This technology, i.e., IoT, that enables ubiquitous information acquisition and exchange among the devices, facilitates the evolution of cellular network from long-term evolution-advanced (LTE-A) systems to future 5G [81]. In an IoT scenario, where there are huge amount of devices characterized by constraints on memory, computation and energy, there is a large demand for resources that can not be responded by the devices themselves. This collected information are typically transmitted to the Cloud for storage and processing.

Despite being very popular, this computational paradigm, i.e., IoT devices collect the information and Cloud processes the information, is not suitable when the latency in making a decision or activating a reaction is very strict (or at least constrained in time) or when the connection between the IoT devices and the Cloud is intermittent or limited in bandwidth.

To overcome these issues, FC, which is a computational paradigm, can be exploited. Due to the limited battery of the edge devices, energy consumption is an additional issue to be addressed in a FC scenario. Among several parameters characterizing the effectiveness of an energy-limited Fog Network, the network lifetime, i.e., the time span a certain amount of nodes or the whole network is stopping to work due to energy shortage, seems to be one of the most important [13]. One possible solution is to exploit technologies enabling energy harvesting from other edge devices.

The issue on the energy consumption of the devices can be addressed by the exploitation of microwave links for transferring also energy apart from the information. WPT technology is a promising candidate for energy related issues in a wireless network.

On the other hand, renewable energy resources can be exploited in an IoT scenario to tackle the energy consumption issues.

To this aim, in this chapter harvesting solutions are proposed. While, in one of the works, FNs are assumed to be harvesting energy from a promising technology enabling WPT, in the other work, FNs are considered to be equipped with a small solar panel enabling them to harvest energy from sunshine.

3.1 State of the arts on Harvesting solutions on Edge Computing

FC has been recently investigated by taking into account different aspects and points of view. Due to the importance of the energy issues in this area, many of the works have taken it into consideration.

Looking at the literature, many works can be found studying the WPT, among which we could mention [82–87]. These works mostly studied the WPT architecture and design, channel efficiency and beamforming analysis on WPT. Some other works have studied the energy harvesting methodologies in edge networks. In [88], mobile devices are equipped with an energy harvesting component and powered by renewable energy. The authors have proposed a dynamic computation offloading algorithm setting the offloading decision by considering execution latency and task failure as the performance metrics. The offloading decision is system operator-centered by deciding the amount of workload to be offloaded from the edge server to the central cloud. A similar work in this area is [89], in which the authors propose a Reinforcement Learning (RL)-based resource management algorithm for dynamic offloading to the centralized cloud. Another work in the same context is [90] where both local computing and offloading are powered by WPT.

An Energy-Efficient Computation Offloading (EECO) algorithm is introduced in [6] and relies on three main phases: classifying the nodes considering their energy and cost feature, prioritizing them by giving a higher offloading priority to the nodes which cannot meet the processing latency constraint, and the radio-resource allocation of the nodes considering the priority. The proposed EECO algorithm allows to decrease the energy consumption up to 18% w.r.t. computation without offloading. A heuristic offloading decision algorithm was proposed in [62] with the aim of maximizing system utility which considers task completion time and the FN energy consumption in a single server MEC scenario. The authors in [59] proposed energy-efficient offloading policies for transcoding tasks in a mobile cloud system. With the objective of minimizing the energy consumption while meeting the latency constraints, they introduced an online offloading algorithm which decides whether the task should be offloaded to the Cloud or executed locally. Task processing in [60] was based on a decision of either local processing or total offloading. The authors aimed at minimizing the local execution energy consumption for applications with strict deadline constraints. Energy consumption has also been targeted in [63] for an offloading approach. In this work, the authors targeted energy consumption and response time for offloading to the Cloud.

Clustering in edge networking has also been proposed in some works. In [37] clustering was performed among the access points considering channel and caching status. A clustering algorithm was also proposed in [12] for the radio access points dealing with joint computation and communication resource allocation inside the cluster. The same problem has been also addressed in [91]. In this work, whenever a user has a packet to offload, the computing cluster is assigned to it. The main idea is to jointly compute clusters for all active users' requests simultaneously in order to distribute computation and communication resources among the users. A similar approach has also been proposed in [71]. A mobile edge computing

clustering algorithm is also proposed in [92], which aims at maximizing the traffic handled inside the clusters and reduce the traffic going out to the core network data servers. Channel conditions of the IoT devices have been considered for creating clusters in [93], where they considered the IoT devices with best and worst channel condition to be placed in one clusters. In addition, the authors proposed a power allocation method for each cluster. In [74], different clustering mechanisms have been studied considering the energy consumption of the nodes. Moreover, the impact of cluster updating has been investigated. A centralized and distributed architecture has been introduced in [79] where, in the centralized architecture, the nodes are clustered having the possibility of offloading to the edge devices. The aim of the work is to estimate the offloading portions to the available devices for computation.

The solutions concerning energy harvesting, especially with solar panels, have been widely studied in literature. The authors in [94, 95] extensively studied all the technological details and challenges involving the adoption of solar panels (and/or other energy harvesting techniques). The authors in [96] studied the problem of unpredictability associated with renewal energies (including the solar one) to ensure the quality of a service in an edge computing system. The authors in [97] developed a real wireless sensor network with nodes capable of acquiring data at high frequencies and, at the same time, are equipped with solar panels. Similar works can be found in [98–100]. In [101], a dynamic algorithm is proposed, aiming at minimizing energy costs by leveraging dual energy sources, solar and grid power, to support the fog nodes. The Lyapunov technique has been considered to design algorithms in Cloud of Things system.

3.2 SWIPT-based Computation Offloading

Many mobile applications run on devices with possible services that can be deployed through a fog network, aiming at migrating the computation from the hungry edge devices, FNs, to the resource-rich devices, F-APs. However, when offloading, there is an extra transmission and reception time for which the FN also consumes a certain amount of energy. As a result, computation offloading in battery powered devices involves a trade-off between energy consumption and processing delay, impacting the offloading decision. There are some parameters involved in the offloading decision; among them, bandwidth and packet size are worth to be mentioned. For instance, in an IoT scenario with small packet size, it might be better to perform a local computation. Moreover, when high bandwidth is available, computation offloading might be more favorable. As a result, perfect knowledge about these parameters have high impact on the offloading decision that is worth to be investigated.

As mentioned earlier, WPT is a promising technology that can be exploited in FC for energy harvesting. Several WPT applicability experiments were conducted in its early stage; among them, it is worth to mention a wireless powered helicopter, flying at 60 feet above the ground level by William C.Brown in 1963 [102].

In 2008, the idea of simultaneously transmitting power and data through a wireless link was firstly proposed at MIT [103]. In microwave WPT systems, direct current power can be converted to Radio Frequency (RF) power using an amplifier, while at the receiver side the RF power can be converted back to the direct current power using a rectifier [104]. As a result, a wireless interaction between the F-AP and the FN can be done not only for sharing computational resources, but also for sharing energy. This wireless charging technique is called Simultaneous Wireless Information and Power Transfer (SWIPT) which is expected to extend the battery life time [105].

In SWIPT the trade-off between energy consumption and task offloading latency is moved from the devices to the links requesting a different approach in designing the system. This is particularly important in a FC scenario that is characterized by heterogeneous applications having different characteristics in terms of task sizes, processing load, link bandwidth. In this work, we exploit the SWIPT technology in a FC scenario. The FN can offload its task to the energy beacon enabled F-APs. The FN can harvest energy from the F-AP when the communication channel is idle. We have analyzed the computation offloading under different bandwidth and packet sizes in order to find the bounds allowing the FN to make an offloading decision such that the system is stable from the energy point of view.

3.2.1 System Model and Problem Formulation

The system is composed by two types of edge nodes, named F-APs and FNs. Both edge nodes (either the FN or the F-AP) have computational and storage capabilities. However, the F-AP is supposed to have higher computational capabilities than the FN. On the other side, the FN can either perform the computation of a given task locally, or offload it to the nearby F-AP. In particular we are focusing on a single link between one FN and one F-AP, interacting for offloading the computational effort and harvesting energy through the SWIPT technology.

We have assumed that a power beacon is integrated with the F-AP so that the deployed power beacon can radiate power to the FN. Thus, the FN is able to harvest some amount of energy from the F-AP by using the SWIPT technology. We have considered a time division approach between wireless and power transfer so that the power can be transferred only during communication idle periods [106].

The focus of this work is to define appropriate bounds for the offloading decision with the aim of remaining in an energy stability region by properly exploiting the SWIPT technology. To this aim, when an FN has a task to be processed, it can either compute it locally or offload the processing to the F-AP, as long as the FN is within its coverage area.

The overall energy consumed by the FN up to the time instant t is:

$$E_c^{FN}(t) = P_{tx}^{FN} t_{tx}^{FN} + P_{rx}^{FN} t_{rx}^{FN} + P_{com}^{FN} t_{com}^{FN} + P_{id}^{FN} t_{id}^{FN} \tag{3.1}$$

where P_{tx}^{FN}, P_{rx}^{FN}, P_{com}^{FN} and P_{id}^{FN} are, respectively, the power consumption when in transmission, reception, computation and idle, while, t_{tx}^{FN}, t_{rx}^{FN}, t_{com}^{FN} and t_{id}^{FN} are, respectively, the amount of time the FN is in transmission, reception, computation and idle up to the time t. It is worth to be noticed that the communication circuitry is separated from the computational circuitry, hence, an FN can transmit/receive while computing; this means that in general:

$$t \leq t_{tx}^{FN} + t_{rx}^{FN} + t_{com}^{FN} + t_{id}^{FN}$$

where the equality occurs if and only if the transmit/receive phase and the computing phase are completely disjoint. Otherwise, the sum of the four terms is higher than the considered interval.

By supposing to use the SWIPT technology between the F-AP and the FN with the time division approach, we can consider that the FN can harvest energy [105]. We define the received power at the FN as [102]:

$$P_h^{FN} = \eta^h P_{tx}^{F\text{-}AP} |h|^2 \tag{3.2}$$

where η^h is the power transfer efficiency, $|h|^2$ is the channel gain between the FN and the F-AP and $P_{tx}^{F\text{-}AP}$ is the power transmitted by the F-AP to be harvested by the FN.

We assume that the FN can harvest energy when its communication circuitry is not used (i.e., when it is neither transmitting nor receiving). Thus, at a certain time instant t, the overall harvested energy can be defined as:

$$E_h^{FN}(t) = P_h^{FN} \cdot (t - t_{tx}^{FN} - t_{rx}^{FN}) \tag{3.3}$$

If we suppose that the initial energy of the FN is $E_r^{FN}(0)$, the remained energy of the FN at certain time instant t, considering the harvested and consumed energy can be calculated as:

$$E_r^{FN}(t) = E_r^{FN}(0) - E_c^{FN}(t) + E_h^{FN}(t). \tag{3.4}$$

In the considered FC scenario, the FN consumes some amount of energy during each of the states defined in (3.1), and harvests some amounts while the antenna is free. Thus, in order to have the system in a stable state from the energy point of view, the following should hold:

$$E_h^{FN}(t) \geq E_c^{FN}(t) \tag{3.5}$$

which means the total harvested energy up to the time t by the FN should be greater than the total consumed power. If the above condition is true, the network is alive at least up to time instant t.

Let us focus on a time interval T between two task generations at the FN to be computed, by considering that the packet generation rate follows a Poisson distribution with average λ generated packets per second. Let us define the time for locally computing at the FN a task as $T_{com} = O/\eta_{com}^{FN}$, where O and η_{com}^{FN} are the number of operations to process a task and the computational power of the FN, respectively. On the other side, in case of offloading, we define $T_{tx} = L_d/r$ and $T_{rx} = L_r/r$ as the transmission and reception time interval for a single task, where L_d, L_r and r are the transmitted packet size for the related task generated at the FN, the packet size of the result of the offloaded processing, and the data rate of the link between the FN and the F-AP, respectively. Now, by considering (3.1) and (3.3), if we focus on a single inter-arrival time interval T we can rewrite (3.5) as:

$$\begin{aligned} P_h^{FN}\left(T - \alpha\frac{L_d}{r} - \alpha\frac{L_r}{r}\right) \geq P_{id}^{FN}\left(T - \alpha\frac{L_d}{r} - \alpha\frac{L_r}{r} - (1-\alpha)\frac{O}{\eta_{com}^{FN}}\right) \\ + \alpha P_{tx}^{FN}\frac{L_d}{r} + \alpha P_{rx}^{FN}\frac{L_r}{r} + (1-\alpha)P_{com}^{FN}\frac{O}{\eta_{com}^{FN}} \end{aligned} \tag{3.6}$$

where α equals to 1 if the FN is offloading to the F-AP and 0 if it is performing a local computation.

Now, we are interested in finding the minimum packet inter-arrival time to harvest enough energy in order to respect (3.5). We define T_{loc} as the packet inter-arrival time when the FN performs a local computation (i.e., $\alpha = 0$), and rewrite (3.6) as:

$$P_h^{FN} \cdot T_{loc} \geq P_{id}^{FN}\left(T_{loc} - \frac{O}{\eta_{com}^{FN}}\right) + P_{com}^{FN}\frac{O}{\eta_{com}^{FN}} \tag{3.7}$$

Through simple algebraic operations, it is possible to set a lower bound for having respected the energy stability condition in (3.5) in case of local processing, as:

$$T_{loc} \geq \frac{\frac{O}{\eta_{com}^{FN}}(P_{com}^{FN} - P_{id}^{FN})}{P_h^{FN} - P_{id}^{FN}} \tag{3.8}$$

The obtained bound can be seen as a threshold for understanding if the packet generation inter-arrival time allows to remain in the energy stability condition when performing a local processing. In case the inter-arrival time is higher we have sufficient amount of energy that is harvested, otherwise it is not possible to harvest sufficient amount of energy for processing that packet locally.

On the other hand, if we consider the case in which the FN offloads the computation (i.e., $\alpha = 1$), it is possible to set a lower bound to the time interval for having respected the energy stability condition. We define T_{off} as the packet inter-arrival time when the FN offloads the packet; hence, (3.6) can be written as:

$$P_h^{FN}\left(T_{off} - \frac{L_d}{r} - \frac{L_r}{r}\right) \geq P_{id}^{FN}\left(T_{off} - \frac{L_d}{r} - \frac{L_r}{r}\right) + P_{tx}^{FN}\frac{L_d}{r} + P_{rx}^{FN}\frac{L_r}{r} \tag{3.9}$$

Through simple algebraic operations, it is possible to set a lower bound for having respected the energy stability condition in (3.5) in case of offloaded processing, as:

$$T_{off} \geq \frac{\frac{L_d}{r}(P_{tx}^{FN} + P_h^{FN} - P_{id}^{FN}) + \frac{L_r}{r}(P_{rx}^{FN} + P_h^{FN} - P_{id}^{FN})}{P_h^{FN} - P_{id}^{FN}} \tag{3.10}$$

The obtained bound can be seen as a threshold for understanding if the packet generation inter-arrival time allows to remain in the energy stability condition when offloading the process. In case the inter-arrival time is higher we have sufficient amount of energy that is harvested, otherwise it is not possible to harvest sufficient amount of energy for offloading the computation.

The obtained bounds give two thresholds that can be exploited for deciding whether offloading or not in order to remain in energy stability.

3.2.2 Offloading Decision-Making Approach

In this work we are studying the offloading decision of an FN to an F-AP exploiting SWIPT technology. We have defined two thresholds in Section 3.2.1 allowing an FN to understand if the harvested energy is sufficient to respect the energy stability condition. The first threshold (3.8) is the minimum required inter-arrival time for an FN for remaining in the energy stability region, if performing a local computation; while the second threshold, (3.10), set the same inter-arrival time when the FN offloads the packet for F-AP computation.

Following the bounds definition in (3.8) and (3.10), it is possible to relate them with the packet interarrival time for estimating the decision to be taken. Hence, we make an estimation of the arrival time of the next packet and, based on the estimated time, we can

Table 3.1 Threshold differences in seconds with different bandwidth and packet size

	500 B	1 kB	2 kB	5 kB	10 kB	20 kB	50 kB	100 kB	200 kB	500 kB	1 MB	2 MB	5 MB
200 Hz	42	84	168	420	840	1680	4200	8401	16803	42007	84015	168029	420074
500 Hz	17	34	68	172	344	689	1724	3449	6898	17247	34494	68988	172469
1 kHz	8	17	35	87	175	351	879	1758	3517	8794	17589	35177	87943
500 kHz	0.001	0.003	0.006	0.017	0.034	0.068	0.17	0.34	0.68	1.7	3.4	6.8	17
1 MHz	-0.008	-0.017	-0.035	-0.088	-0.177	-0.354	-0.886	-1.7	-3.5	-8.8	-17	-35	-88
10 MHz	-0.018	-0.037	-0.075	-0.187	-0.375	-0.751	-1.87	-3.75	-7.51	-18	-37	-75	-187

decide whether the FN can perform a local computation or offload in order to respect the condition in (3.5).

We consider that the arrival time of the packets follows a Poisson distribution with average arrival λ packets per second. We define $\bar{T}^l(t)$ as the estimated inter-arrival time of the lth packet at time instant t, as:

$$\bar{T}^l(t) = \sum_{i=1}^{N} \alpha_i T_A^{l-i} \tag{3.11}$$

where T_A^{l-i} indicates the measured inter-arrival time of the $(l-i)$th packet, and α_i is an opportunely set parameter allowing to consider a window of N previously measured inter-arrival intervals.

The offloading decision is based on comparing the estimated inter-arrival time with the two thresholds in (3.8) and (3.10), i.e.,

$$\begin{cases} \bar{T}^l(t) \gtrless T_{loc} \\ \bar{T}^l(t) \gtrless T_{off}. \end{cases} \tag{3.12}$$

To this aim it is worth to be noticed that the greater between T_{loc} and T_{off} depends on several parameters, among which bandwidth and the packet length, while considering fixed the power related terms. By analyzing both (3.8) and (3.10), it is possible to notice that higher is the bandwidth, the lower is the interaction time, favoring to offload the tasks. However, if the bandwidth is smaller due to the longer communication time, a local computation might be better in terms of time. On the other hand, the length of a packet impacts on the computation, reception and transmission times. To this aim, this motivates us to analyze the impact of these two parameters on the two thresholds.

We have considered the difference between the offloading and local thresholds (T_{off}-T_{loc}) by considering variable bandwidth and packet sizes. The result is depicted in Fig. 3.1 and Tab. 3.1.

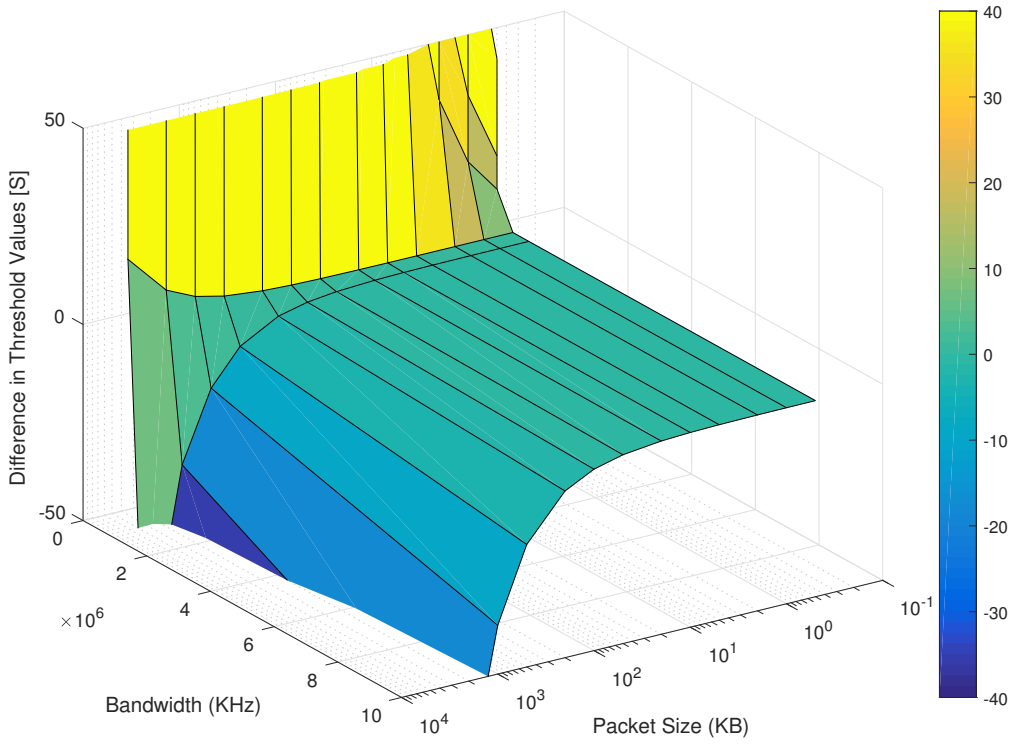


Fig. 3.1 Difference between the offloading and local thresholds (T_{off}-T_{loc})

As seen in Tab. 3.1, when the bandwidth is low and packet size is small, the offloading threshold is slightly greater than local threshold; however, as the packet size increases, the offloading threshold gets much larger than the local threshold. This means in low bandwidth and large packet size performing a local computation is more beneficial. On the other hand, when the bandwidth is high, with small packet size local threshold is slightly greater than the offloading threshold, however, when the packet size increases, this difference also rises. This indicates that with high bandwidth and different packet sizes, offloading is desirable because it takes a shorter time to harvest sufficient amount of energy to make up for the consumed energy required for the processing the task.

The previous analysis allows to individuate three areas. Area 1 represents the inter-arrival times lower than the lowest threshold. Area 2 is the intermediate area, representing the inter-arrival times where it is possible to make an offloading decision allowing to harvest sufficient amount of energy, while in Area 3 both offloading decisions allow to harvest a sufficient amount of energy.

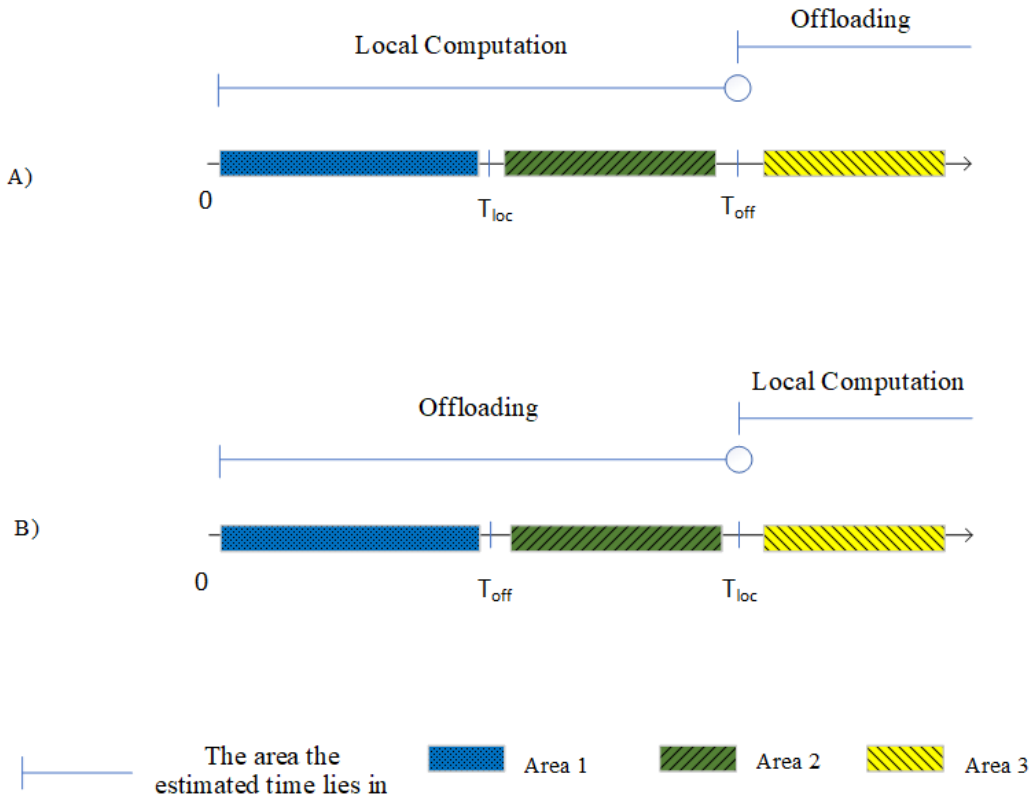


Fig. 3.2 The offloading and local computation thresholds.

The offloading decision, hence, leads to two cases depending on the order of the thresholds, represented in Fig. 3.2. If the estimated arrival time is lower than the lowest of the two thresholds the FN cannot harvest sufficiently respecting (3.5), and it is supposed to make the decision based on the lower threshold, meaning local computation in case (A), and offloading in case (B). On the other hand, if the estimated arrival time is greater than the lowest threshold, the FN is able to harvest sufficiently if it makes the decision only based on the lowest threshold; meaning local computation in case (A), and offloading in case (B). However, if the estimated inter-arrival time is greater than both thresholds the FN can harvest sufficiently regardless of the decision it makes, either offload or not. In this case, we further improve the decision by opting the solution resulting in minimizing the task processing time. To this aim we define the task processing time, when the FN performs a local computation, as:

$$D_{loc}^{l} = \frac{O}{\eta_{com}^{FN}} \tag{3.13}$$

while if the FN offloads the computation to the F-AP, the task delay is:

$$D_{off}^{l} = \frac{L_d}{r} + \frac{L_r}{r} + \frac{O}{\eta_{com}^{F\text{-}AP}} \tag{3.14}$$

The overall decision Algorithm 9 can be summarized in this way. Let us focus on a task Δ^l transmitted by the considered FN; the offloading process can be identified by a tuple $\langle L_d, L_r, O, r \rangle_l$. Following the parameters of the lth task it is possible to calculate the related local and offloading thresholds T_{loc}^{l} and T_{off}^{l}, following (3.8) and (3.10). The comparison between the two thresholds leads to one of the two cases represented in Fig. 3.2; the decision ζ_l can be performed by comparing the estimated interarrival time with the obtained thresholds.

Algorithm 9 The Offloading Decision Algorithm

1: **Input:** $\langle L_d, L_r, O, r \rangle_l, \bar{T^l}(t)$
2: **Output:** ζ_l
3: Calculate T_{loc}^{l} and T_{off}^{l}
4: **if** $T_{loc}^{l} \geq T_{off}^{l}$ **then**
5: $\quad \Theta \leftarrow T_{loc}^{l}$
6: **else**
7: $\quad \Theta \leftarrow T_{off}^{l}$
8: **end if**
9: **if** $\bar{T^l}(t) < \Theta$ **then**
10: $\quad$ **if** $\Theta \equiv T_{loc}^{l}$ **then**
11: $\quad\quad \zeta^l \leftarrow$ **Offload**
12: $\quad$ **else**
13: $\quad\quad \zeta^l \leftarrow$ **Local**
14: $\quad$ **end if**
15: **else**
16: $\quad$ **if** $D_{loc}^{l} < D_{off}^{l}$ **then**
17: $\quad\quad \zeta^l \leftarrow$ **Local**
18: $\quad$ **else**
19: $\quad\quad \zeta^l \leftarrow$ **Offload**
20: $\quad$ **end if**
21: **end if**

As seen, in the pseudo-code, the inputs of the algorithm are three sets of $\mathcal{L}$, Ω, and Ψ and the output is the decision which is made by the FN for each task, whether to offload or compute locally. The algorithm first finds the upper threshold (Lines 4-8) and then the decision is made based on the lower threshold (Lines 10-14). When the estimated arrival time of next packet is greater than both thresholds, the offloading decision is made by considering the action resulting in a lower delay (Lines 16-20).

Table 3.2 Simulation Parameters for SWIPT Approach

Parameter	**Value**
Task result size (L_r)	$L_d/5$
Path loss exponent	2.7 (urban area)
F-AP coverage range	50 m
Task Operations (O)	10000 FLOPS per Byte
FN Processing Speed (η_{com}^{FN})	15 GFLOPS
F-AP Processing Speed($\eta_{com}^{F\text{-}AP}$)	150 GFLOPS
FN Computation power (P_{com}^{FN})	0.9 W
FN Idle power (P_{id}^{FN})	0.01 W
FN Transmission and reception power (P_{tx}^{FN}, P_{rx}^{FN})	1.3 W

3.2.3 Numerical Results

In this section, the numerical results obtained through computer simulations in Matlab are presented, where the considered parameters are listed in Tab. 3.2. Each simulation is supposed to run for 3000 s and the initial energy of the FN, $E_r^{FN}(0)$, is considered to be 20 J.

By resorting to [102], in an urban environment, the power transfer efficiency η^h can be on the order of 1%, when the distance between the beacon and the FN is on the order of 10 meters; to this aim we set $\eta^h = 1\%$ and the FN is 10 m away from the power beacon.

Remark 1: (Safety Levels of Human Exposure toRF Electromagnetic Fields): According to the IEEE Standard C95.1-2005, for safety levels with respect to human exposure toRF electromagnetic fields, the permissible exposure level from 2 GHz to 100 GHz in a public environment is 10 W m^{-2} [107, p. 27]. Due to the fact that typical frequencies that are used for far-field WPT systems development are 2.45 GHz and 5.8 GHz, we need to define the minimum distance between the receiver and the beacon power in order to respect the standard. If effective radiated power at the power beacon is (P_{erp}) and Euclidean distance between the power beacon and receiver is d and considering the power density formula the minimum distance between the two nodes respecting the mentioned permissible exposure level, is defined as:

$$P_{erp}/(4\pi d^2) < 10 \tag{3.15}$$

In this work we have set the radiated power to 1.5 W; hence, the minimum distance in meters is:

$$d > \sqrt{1.5/40\pi} \simeq 0.1092 \tag{3.16}$$

Therefore, the radius of exclusion zone is set to this minimum distance considering a far-field WPT.

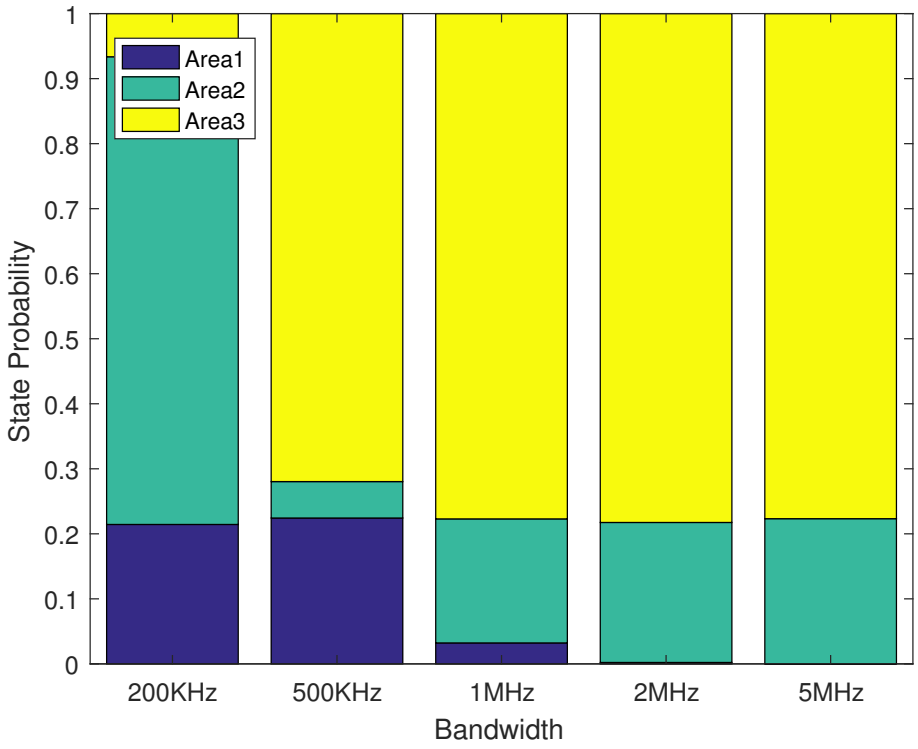


Fig. 3.3 Threshold Areas for different bandwidth

We have performed the simulation results for variable bandwidth, packet length and λ, while the estimation of the interarrival time has been performed by considering a constant averaging window over the past $N = 5$ packets (i.e, α_i=0.2)=. In the following, we will be briefly studying the results for each parameter.

In Fig. 3.3 the impact of bandwidth is analyzed for different bandwidth values. The packet size is fixed to 1 kB and λ is set to 5.5 packets per second. As seen in the figure, as the bandwidth increases the estimated arrival time of the next packet is larger than both thresholds and it falls in the 3rd area. That is due to the fact that by having higher bandwidth, the transmission and reception time decrease as well and the overall task delay gets smaller so that the FN can harvest in a shorter time the consumed energy. Therefore, by the arrival of next packet, the FN has harvested the consumed energy with a high probability (i.e., about 80%) when the bandwidth is higher than or equal to 1 MHz. However, when the bandwidth is smaller, due to the longer interaction time, a higher delay is experienced and the arrival time usually falls in the second and first area. This result is in accordance with Fig. 3.2.

To see the impact of the packet size on the two thresholds, numerical results for variable packet lengths have been considered by fixing the bandwidth to 500 kHz and λ to 5.5 packets per second. We have selected 500 kHz for the bandwidth, because it allows to have the

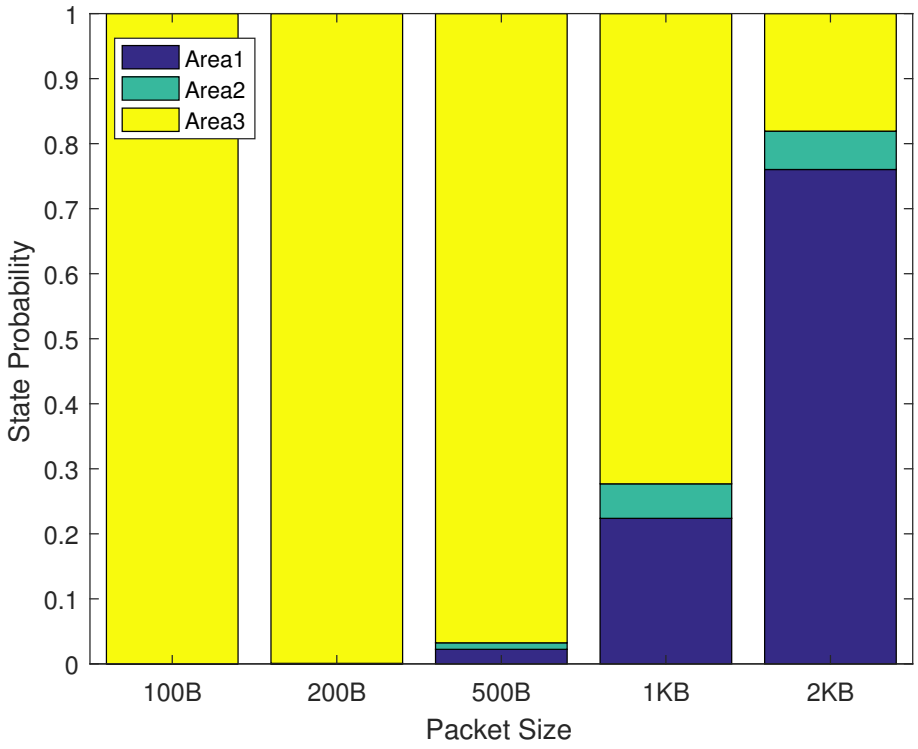


Fig. 3.4 Threshold Areas for different packet size

inter-arrival time falling in all areas based on Fig. 3.3. As seen in Fig. 3.4, when the packet size is smaller the processing time is smaller as well and there is sufficient time for the FN to harvest; however, when the packet size is large it takes a longer time to process the task resulting in a shorter time for harvesting. This is why the arrival of next packet is earlier than the time required for harvesting and the next packet inter-arrival time mostly lies in area 1 and 2. The other consideration is that due to the value of the bandwidth, i.e, 500 kHz, the difference between the two thresholds is reduced; this is in line with Fig. 3.2 and Tab. 3.1.

Finally, the impact of variable λ on the network delay and lifetime is investigated. We have considered a bandwidth equal to 500 kHz and a packet size equal to 1 kB allowing to have the arrival time in the all areas according to Figs. 3.3 and 3.4.

The comparison is performed by considering the proposed decision algorithm, where the packet inter-arrival time is estimated and compared with the decision thresholds. The proposed algorithm, labeled *SWIPT & Opt.Thre*, is compared with 4 benchmarks: *SWIPT & Loc* and *SWIPT & Off* consider SWIPT technology with always doing local processing and SWIPT technology when always offloading, respectively. Finally, *Loc* and *Off* consider the cases in which the FN is always performing local and always offloading without harvesting with SWIPT technology.

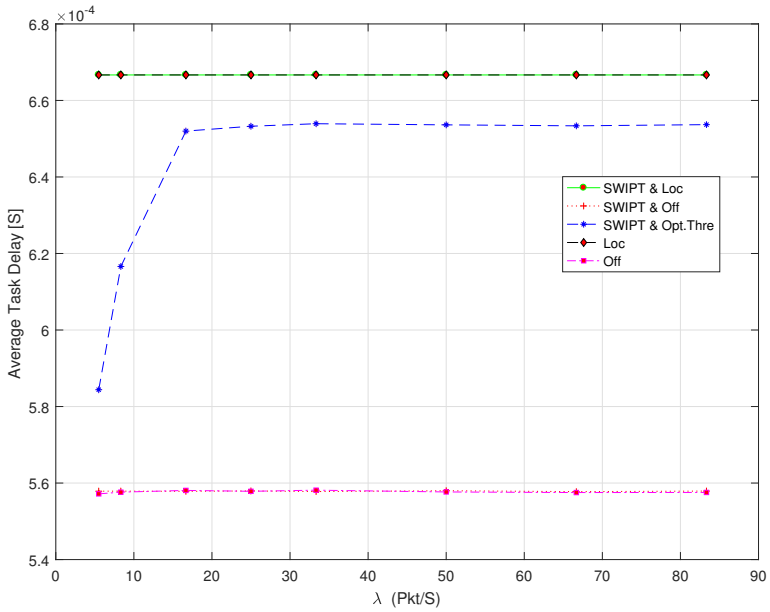


Fig. 3.5 Average Task Delay

Fig. 3.5 depicts the average task delay for the 5 scenarios. As seen, the local scenarios are overlapping and having the highest delay due to the fact that packet size and computational power of an FN are fixed in this experiment. On the other hand, the offloading scenarios also overlap and have the lowest delay due to the higher computational power of the F-AP. However, the proposed solution, in which the decision is done based on the arrival of the next packet and the calculation of the two thresholds, lies in the middle of the other curves. As seen, when the λ is small, it takes a longer time to have the next packet for computation, therefore there is a longer time for harvesting. As a result next packet arrival time is usually in area 3 which results in offloading decision to be made based on task delay. Therefore, because offloading takes a shorter time, the algorithm opts offloading. However, as λ increases the arrival of next packet gets shorter and it lies mostly in area 2 (and sometimes in area 3) in which the lower threshold is selected. In the end, for packet generation rates higher than around 16 pkt/s, the arrival time is always early, so that the FN opts the local computation and gets closer to the local computation scenarios.

Fig. 3.6 depicts the node lifetime for the 5 scenarios corresponding to the time instant the node goes off. As seen in the figure, scenarios exploiting SWIPT technology have a longer lifetime due to the energy harvesting, and the FN does not even go off when the λ is small. As λ increases more packets are generated for computation which results in more

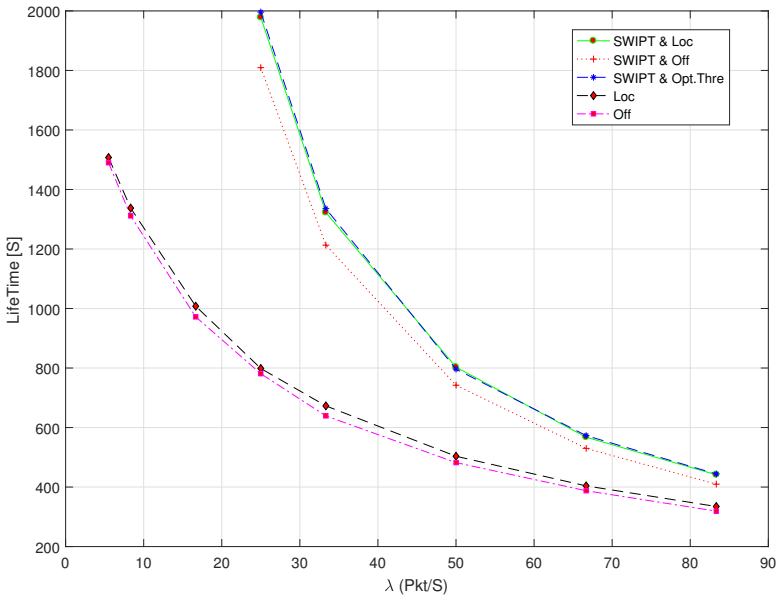


Fig. 3.6 Node Lifetime

consumption, therefore, lifetime decreases for all scenarios. Moreover, the proposed solution has the longest lifetime.

As seen in Figs. 3.5 and 3.6, the proposed solution allows to select the offloading decision in a way that it benefit from both offloading and local computation. The proposed approach has a lower delay than the *SWIPT & local*, where it benefits from the low delay of offloading, and a slightly higher lifetime with respect to the *SWIPT & local* where it benefits from its lower energy consumption.

3.2.4 Summary

In this work, we have studied the impact of packet size and bandwidth on defining some thresholds which are later used for a computation offloading decision from an FN to an F-AP in a FC scenario. We have shown that exploiting SWIPT technology enables the FN to harvest energy and by estimating the arrival time of the following packet, considering the Poisson distribution, the FN is able to make the best offloading decision in order to shorten the task latency and extend the network lifetime. In the future, we will be investigating a multi-user and multi-F-AP scenario in which the problem is not only for offloading decision but also on the selection of devices for offloading.

3.3 Smart Energy Management in Fog Networks

In this work, we are interested in considering the energy saving aspects for prolonging the network lifetime of battery-powered FC nodes by leveraging the presence of energy harvesting devices (e.g., solar panels) when they offload data/code to be processed to other nodes. This could be particularly important for those applications in which the human intervention could be an issue.

The aim of this work is to introduce a novel smart energy management approach for FC, based on the prediction of harvested energy to optimize the offloading of the codes so as to prolong the FC network lifetime. In particular, we are proposing to exploit the prediction of the required and harvested energy to be spent/acquired by each FC node over time to support the optimization of the clustering algorithm so as to prolong the network lifetime. The considered application scenario foresees the presence of pervasive FC nodes able to organize themselves into clusters. This organization has been demonstrated to be effective in terms of both distributed processing and energy saving allowing also to reduce the overall delay [79].

The proposed solution has been tested and evaluated in a synthetic scenario but with real data about the energy harvesting profiles in Northern Italy [108].

After studying the previous works we have noticed that, to the best of our knowledge, most of the works have been focusing on the offloading decision, whether to offload or not, or where to offload in order to minimize the energy consumption. On the other hand, some works considered different parameters for cluster formation in different environments, however, no work has considered a joint exploitation of energy harvesting design and offloading decision. Our contributions in this work can be summarized as:

- Designing a green FC environment where the FNs are equipped with solar panels and capable of recharging their batteries from the solar energy.
- A clustering algorithm is considered based on the energy status of the devices, so that the cluster members in each cluster offload their tasks to the cluster head for computation, by taking into account both consumed and harvested energy.
- Differently from previous works, we have considered the energy management among the devices by changing the role of the nodes in each cluster between requesting devices and computing devices when necessary in order to prolong the network lifetime;
- A harvesting prediction method is proposed in order to avoid the network to go off in the future by predicting the energy consumption of the nodes based on their harvesting pattern.

3.3.1 System Setting

In this work, we are focusing on a scenario composed of N FNs identified by the set $\mathcal{U} = \{u_1, \ldots, u_i, \ldots, u_N\}$. The generic ith FN is supposed to periodically generate some data to be processed,[1] where l_i stands for a data unit generated by the ith FN and δ_{l_i} accounts for the memory occupation in Byte. Each FN is supposed to have a certain processing capability η_c^j characterized by the FLOPS it can handle; to this aim we suppose that the l_ith data unit requests a certain amount of floating point operations equal to ω_{l_i}. The data processing operation impacts the energy consumption of the FN, and, to this aim, we suppose that the power consumption for computing is P_c^i.

The operative scenario is supposed to be an area having a dimension of A m^2, in which the FNs are scattered in random positions, where the generic ith FN is placed in (x_i, y_i). By resorting to the FC paradigm, we are supposing that data can be processed at the originating FN, or can be offloaded to any other surrounding FN for remote processing. We suppose that the FNs can communicate among themselves within a range R, i.e, any couple of nodes u_i and u_j can interact if and only if $d(u_i, u_j) \leq R$, where $d(\cdot, \cdot)$ is the Euclidean distance operator and R is the coverage range depending on the considered wireless technology. The transmission technology is shared among all the FNs. This operation impact also on the power consumption of the FNs, where we suppose that the ith FN is characterized by a transmission power P_{tx}^i and a reception power P_{rx}^i.

The objective of this study is designing an energy-sustainable solution able to prolong the network lifetime characterized as the amount of time the network can operate without any external intervention. To this aim each FN is supposed to be equipped with an energy harvesting device. In particular, in this work, we have considered the FNs to be equipped with a small solar panel able to instantaneously harvest $E_h^i(t)$ Joule at the time instant t, by the generic ith FN. The behaviour of $E_h^i(t)$ depends on several factors; among them, the FN location, the size of the solar panel, the time of the day, as well as the relative orientation between the solar panel and the sun can be mentioned. With respect to this, in this work we have considered an energy harvesting model based on the data provided by the ENEA agency [108] that averages daily acquisitions from 1995 to 1999 in the city of Milan, Italy.

3.3.2 Harvesting Solutions for Cluster based Fog Computing systems

In order to implement an effective energy sustainable Fog Network, a communication organization should be considered for allowing data exchange among the FNs and implementing

[1] The data can be also generated by external sensor nodes connected to the FN; however, from the processing point of view, this corresponds to have data generated directly by the FN.

the computation offloading scheme. By relying on previous studies we resort on a cluster organization, considered as a feasible network structure for energy efficient Fog Network design [74, 79]. In particular, a clustering scheme has been introduced in [74, 79] for optimizing the energy management of FNs scattered in a given area, when they are exchanging data to be processed.

In the following, we will refer to FNs acting as Cluster Head (CH) as those nodes processing data received from the FNs acting as Cluster Members (CMs). The sets of CHs and CMs can be defined as $\mathcal{U}_{CH}$ and $\mathcal{U}_{CM}$, where

$$\mathcal{U} = \mathcal{U}_{CH} \cup \mathcal{U}_{CM}.$$

By focusing on the generic ith FN acting as CM and the generic jth FN acting as CH, having defined the size of the l_ith data unit generated by the ith FN as δ_{l_i}, and the data rate of the link between the ith FN and the jth FN as r_{ij}, we can write the energy spent for transmitting the l_ith data unit as:

$$E^i_{tx,l_i} = P^i_{tx} \frac{\delta_{l_i}}{r_{ij}}. \tag{3.17}$$

Similarly, by focusing on the generic jth FN acting as CH with respect to the ith CM, we can write that the energy spent for receiving the l_ith data unit to be processed as:

$$E^j_{rx,l_i} = P^j_{rx} \frac{\delta_{l_i}}{r_{ij}}. \tag{3.18}$$

In each cluster, the CMs are supposed to offload the generated data to the associated CH for being processed; hence, the energy spent by the j-th FN for processing can be written as:

$$E^j_{p,l_i} = P^j_c \frac{\omega_{l_i}}{\eta^j_c}. \tag{3.19}$$

It is worth to be noticed that the computing energy is spent by the jth FN also for processing its own generated data when it is a CH or, in case of an isolated FN, when it is not connected to any CH[2].

In case the ith FN is not transmitting, receiving, nor computing, the energy spent in idle mode corresponds to:

$$E^i_{id} = P^i_{id} t^i_{id} \tag{3.20}$$

[2]We can consider the case of an isolated node, i.e., not connected to any other node, as a singular case of a cluster having only one FN and hence working as a CH.

where P^i_{id} is the power spent for remaining in an idle state and t^i_{id} is the time interval in which the ith FN is in idle.

Due to the offloading process, the energy consumption among CHs and CMs results to be unbalanced due to the higher processing work by the CHs with respect to the CMs. For coping with this issue, we foresee the possibility that, periodically, a reclustering operation is performed with the aim of optimizing the selection of the CHs and the related CMs among the available FNs. In the following, we refer to the reclustering period as $\bar{t}$.

Within the reclustering period $\bar{t}$, the generic ith FN role cannot change while being either a CM or a CH. In the following we focus our analysis on a single reclustering period, within which the FNs role is fixed. The overall analysis can be performed by considering consecutive reclustering intervals, at the beginning of which the FNs role is optimized with the aim of prolonging the network lifetime.

By supposing that the generic ith FN is generating N_i data units to be processed within a reclustering period $\bar{t}$, the energy spent when acting as a CM can be written as:

$$\bar{E}^i_{CM} = N_i E^i_{tx,l_i} + P^i_{id}\left(\bar{t} - \frac{\delta_{l_i}}{r_{ij}} N_i\right) \tag{3.21}$$

corresponding to the sum of the energy spent for offloading the N_i generated data units plus the energy spent in idle; it is worth to be noticed that the energy spent in idle is evaluated on a time interval equal to the reclustering interval minus the time needed for transmitting the N_i tasks to be offloaded.

On the other side, when the ith FN is operating as a CH, it spends energy for computing the own generated data plus the energy spent for both receiving and computing the data offloaded by the CMs in the same cluster; in addition, the energy spent in idle is considered. This corresponds to an energy consumption as CH equal to:

$$\bar{E}^i_{CH} = N_i E^i_{p,l_i} + \sum_{\substack{u_\iota \in \mathcal{C}_i \\ u_\iota \neq u_i}} N_\iota \left(E^i_{rx,l_\iota} + E^i_{p,l_\iota}\right) + P^i_{id}\bar{t}^i_{id} \tag{3.22}$$

where $\mathcal{C}_i = \{u_\iota | d(u_i, u_\iota) \leq R\}$ is the cluster of the ith FN when acting as a CH, composed by the CH itself and the CMs connected to it, and ι is the index identifying the CMs belonging to the set $\mathcal{C}_i$. Hence, N_ι identifies the data units generated by the ιth FN acting as CM and offloaded to the ith FN acting as CH. Finally, the idle time within a reclustering interval $\bar{t}$, can be written as:

$$\bar{t}^i_{id} \geq \bar{t} - N_i \frac{\omega_{l_i}}{\eta^i_c} - \sum_{\iota \in \mathcal{N}(i)} N_\iota \left(\frac{\delta_{l_\iota}}{r_{\iota j}} + \frac{\omega_{l_\iota}}{\eta^i_c}\right) \tag{3.23}$$

calculated as the reclustering interval minus the time spent for receiving and computing. It is worth to be noticed that the computing and receiving actions can be done also in parallel; this means that when they are disjoint an equality should be considered, otherwise the inequality should be considered.

On the other hand, the amount of energy harvested in a reclustering interval starting at the time t can be defined as:

$$\bar{E}_h^i(t,\bar{t}) = \int_t^{t+\bar{t}} E_h^i(\tau)d\tau \tag{3.24}$$

where $E_h^i(\tau)$ is the instantaneous harvested energy at time instant τ. It is worth to be noticed that the harvested energy depends on both the starting time t and the reculstering interval $\bar{t}$, by supposing the FN is able to harvesting during the whole time. Indeed, the daily acquisition curve for the generic ith FN can be modelled as a Gaussian with mean μ_i (i.e., the acquisition peak), for each FN $u_i \in \mathcal{U}$, and standard deviation σ_i defined in a way the interval $(\mu_i - 3\sigma_i, \mu_i + 3\sigma_i)$ covers the sunshine hours of the considered month [97] (e.g., 9 hours and 15 minutes in January and 15 hours and 40 minutes in June in case of Milan, Italy). Hence, (3.24) can be rewritten as:

$$\bar{E}_h^i(t,\bar{t}) = \int_t^{t+\bar{t}} E_h^i(\tau)d\tau = \int_t^{t+\bar{t}} \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{(\tau-\mu_i)^2}{\sigma_i^2}} d\tau \tag{3.25}$$

for each node $u_i \in \mathcal{U}$.

The way to define μ_i and σ_i are clarified in the Experimental Section and shown in Fig. 3.7. As expected, the sunshine daily hours change during the year and the daily peak is slightly changed for each sensor nodes by at most 30 minutes from the 13:00.

In order to have a better view of the consumption and harvesting patterns in each interval for a CM node (or the offloader) and a CH node (or the offloadee), in Figs. 3.8 and 3.9 the energy patterns have been plotted by considering that the CM generates two tasks to be computed by the CH. Fig. 3.8 represents the power consumption levels for both the CM and CH nodes within one interval; it is possible to notice the different phases during which the FNs can be in transmission, reception, computation or idle. It is possible to notice that while the CH is computing the tasks, the CM is in idle. Fig. 3.9, on the other hand, depicts the time behaviour of the remaining energy level of both CM and CH nodes within the same interval. It is possible to notice that the CM is gaining more from the harvested energy due to a higher idling with respect to the CH, that instead spends a higher amount of time in receiving and computing when the harvested energy is not enough to cover the reception and processing phases. In the depicted example, both nodes are harvesting during the whole interval, however, only during the idle the FNs are able to positively recharge their batteries.

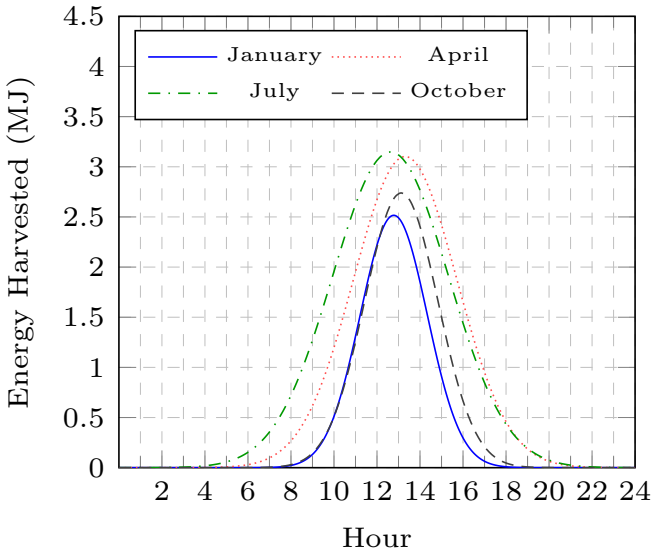


Fig. 3.7 The energy harvesting curves for four different sensors in four different months

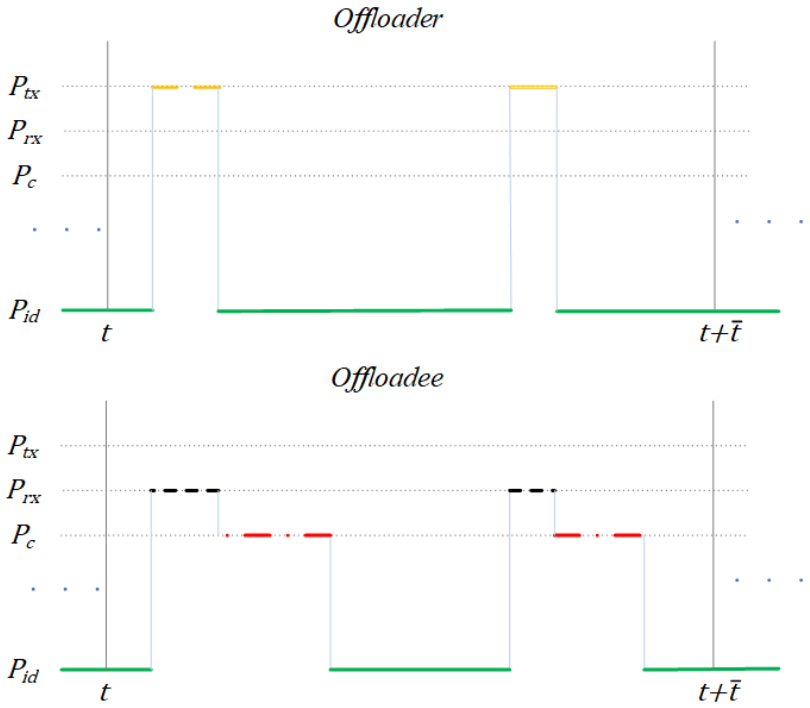


Fig. 3.8 Power consumption levels for CM and CH in an interval

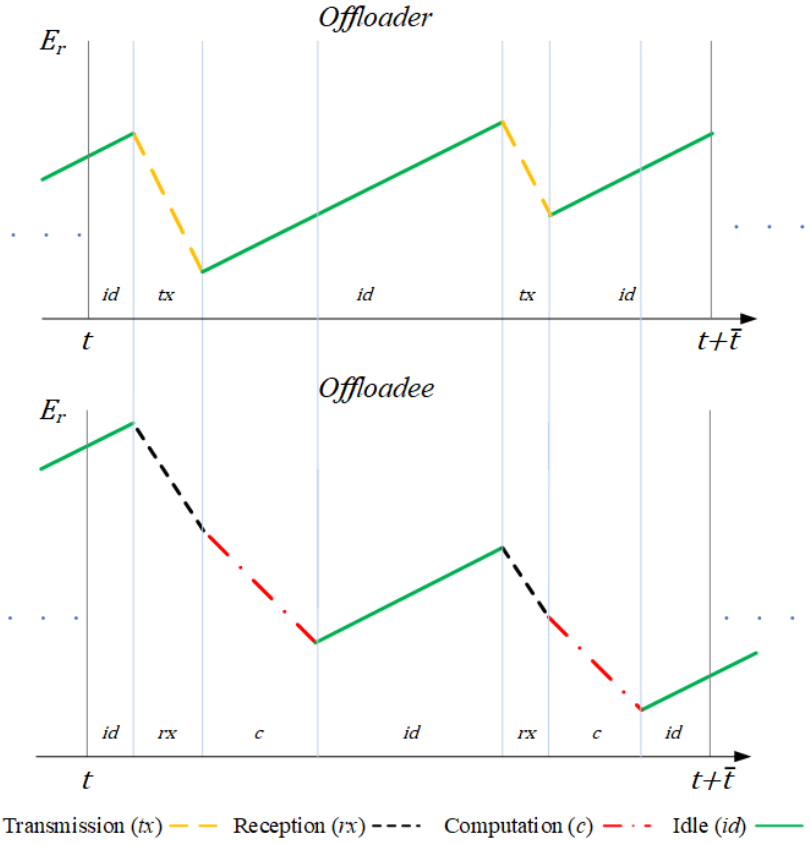


Fig. 3.9 Remaining Energy profiles for CM and CH in an interval

The FNs are supposed to have a rechargeable battery that can store a maximum amount of energy $\bar{E}_{bc}$. By supposing that the energy stored by the ith FN at a generic time instant t is:

$$E_r^i(t) = \gamma_i(t)\bar{E}_{bc}, \tag{3.26}$$

where $\gamma_i(t)$ is a parameter in the range $[0,1]$ modeling the portion of energy in the battery at the time instant t[3], it is possible to derive the energy stored by the FNs at the end of a reclustering interval by exploiting (3.21), (3.22) and (3.24):

$$\bar{E}_r^i(t+\bar{t}) = \begin{cases} E_r^i(t) - \bar{E}_{CM}^i + \bar{E}_h^i(t,\bar{t}), & \text{if } u_i \in \mathcal{U}_{CM} \\ E_r^i(t) - \bar{E}_{CH}^i + \bar{E}_h^i(t,\bar{t}), & \text{if } u_i \in \mathcal{U}_{CH} \end{cases} \tag{3.27}$$

Our goal is to maximize the network lifetime; this corresponds to optimally select the CHs and CMs among the FNs at each reclustering interval, for creating the clusters with the aim of having maximized the number of FNs having the remaining energy at the end of any reclustering interval higher than zero, i.e.:

$$\max_{\mathbf{x}} \left\{ \sum_i x_{ik} p_{ik} \right\}, \tag{3.28}$$

where $\mathbf{x}$ is a two-columns assignment vector where each element $x_{ik} = \{0,1\}$ is the integer assignment variable such that

$$(x_{i1}, x_{i2}) = \begin{cases} (1,0) & \text{if } u_i \in \mathcal{U}_{CM} \\ (0,1) & \text{if } u_i \in \mathcal{U}_{CH} \end{cases} \tag{3.29}$$

and p_{ik} is the element of the objective function to be maximized corresponding to:

$$p_{ik} = \begin{cases} 1 & \text{if } \bar{E}_r^i(t+\bar{t}) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.30}$$

[3] In particular the parameter $\gamma_i(0)$ models the the FNs characterized by different energy levels at the beginning of the working operations.

where $u_i \in \mathcal{U}_{CM}$ when $k = 1$ and $u_i \in \mathcal{U}_{CH}$ when $k = 2$. The maximization problem in Eq. (3.28) is subject to the following constraints:

$$\bar{E}_r^i(t) \leq \bar{E}_{bc} \quad \forall i,t \tag{3.31a}$$

$$\sum_k x_{ik} = 1 \quad \forall i \tag{3.31b}$$

$$\mathcal{C}_i = \{u_i | x_{i1} = 1\} \cup \left\{u_j | \left\{d(u_i,u_j) < R\right\} \wedge \left\{x_{j2} = 1\right\}\right\} \quad \forall i,j \tag{3.31c}$$

where Eq. (3.31a) is an upper bound for the remaining energy set to the battery capacity, i.e, the battery cannot recharge more than its capacity, Eq. (3.31b) allows to limit the allocation of an FN to either the CH or the CM sets, and Eq. (3.31c) defines a cluster set as following the x_{ik} assignment variable introduction.

In order to solve the FNs assignment problem we resort on a two-step approach. In the first step the FNs role is set based on the energy level, while in the second step the cluster formation is performed aiming at setting the connections between CHs and CMs.

Fog Nodes Cluster Assignment

The CH and CM role assignment to the FNs in the system is performed with the aim of maximizing the remaining energy of the FNs at the end of every reclustering interval. By resorting to the approach proposed in [74, 79] it is possible to assign the role to the FNs based on their energy level, with the rationale that the CHs consume more energy than the CMs due to the additional computing requested by the associated CMs. To this aim, the assignment is performed by giving the CH role to those nodes having a higher residual energy.

This approach can be performed by resorting to a 3-quantile classification performed on the energy level distribution of all the FNs. By using a 3-quantile function it is possible to classify the nodes based on the distribution of their remaining energy. At each reclustering period, the nodes energy values are evaluated, and then, the FNs are classified in one of the two groups depending on their residual energy value.

Differently from the solution proposed in [74, 79] that is considered as a benchmark in the Experimental Results in Section 3.3.4, we are here considering also the effect of the energy harvesting on the energy level.

We are here proposing two different solutions for the CM/CH clustering scheme: an Harvesting Clustering Scheme where the harvested energy is considered in the cluster formation and a Predictive-based Harvesting Scheme where the harvested energy is predicted

for optimizing the cluster formation and minimizing the FN energy shortage in the next reclustering period.

Harvesting Clustering Scheme

In the harvesting clustering scheme at the beginning of each reclustering interval we are selecting the FNs role based on their actual energy level, whose value is affected by both the consumed and harvested energy in the previous interval.

As a first step, in order to avoid that FNs with a very low energy level can be selected as CH, a threshold on the remaining energy is considered. To this aim, we suppose that the FNs whose energy level is lower than a given threshold $\beta \bar{E}_{bc}$ are not appropriate candidates for becoming a CH. A result these FNs are potentially selected as CMs. On the other side, the FNs having an energy level higher than or equal to the threshold $\beta \bar{E}_{bc}$ are considered to be CH candidates ($\hat{\mathcal{U}}_{CH}(t_\nu)$, at time instant $t_\nu = \nu \bar{t}$, where ν is an integer index identifying the νth reclustering interval). Hence, at the beginning of the νth reclustering interval, the FNs role selection is performed by defining:

$$\hat{\mathcal{U}}_{CH}(t_\nu) = \left\{ u_i | E_r^i(t_\nu) \geq \beta \bar{E}_{bc} \right\}. \tag{3.32}$$

The parameter β allows to exclude those nodes having a very low amount of energy to be selected as CHs, thus preventing any network failure due to the energy shortage of a CH when processing data offloaded by the CMs.

To perform the energy-aware FN classification [79], we use the 3-quantile function procedure on the nodes belonging to the set $\hat{\mathcal{U}}_{CH}(t_\nu)$. This considers the distribution of energy level of the FNs, for classifying them into three sets. The FNs having a remaining energy higher than the upper quantile index, $E_{r,Q2}$, of the energy level distribution of all FNs are selected as CHs and the rest as CMs. This rule allows to select the FNs having the highest energy level. In the end, it is possible to define $\mathcal{U}_{CM}(t_\nu)$ and $\mathcal{U}_{CH}(t_\nu)$ as:

$$\begin{aligned} \mathcal{U}_{CH}(t_\nu) &= \left\{ u_i \in \hat{\mathcal{U}}_{CH}(t_\nu) | E_r^i(t_\nu) \geq E_{r,Q2}(t_\nu) \right\} \\ \mathcal{U}_{CM}(t_\nu) &= \left\{ u_i \in \hat{\mathcal{U}}_{CH}(t_\nu) | E_r^i(t_\nu) < E_{r,Q2}(t_\nu) \right\} \cup \left\{ u_i \notin \hat{\mathcal{U}}_{CH}(t_\nu) \right\} \end{aligned} \tag{3.33}$$

assigning the FNs in $\hat{\mathcal{U}}_{CH}(t_\nu)$ with a remaining energy lower than the upper quantile index, and all the FNs not selected as CH candidates in $\mathcal{U}_{CM}(t_\nu)$.

The pseudo-code of the classification scheme is shown in Algorithm 10, where the input is the energy level of all FNs and the output is the list of CMs and CHs (Lines 1-2). If the FNs have a remaining energy equal or greater than $\beta \bar{E}_{bc}$, they are put into the CH candidate list,

Algorithm 10 3-Quantile Function

1: **Input:** $E_r^i(t_v)\ \forall u_i \in \mathcal{U}$
2: **Output:** $\mathcal{U}_{CM}$, $\mathcal{U}_{CH}$
3: **for all** $u_i \in \mathcal{U}$ **do**
4: **if** $E_r^i(t_v) \geq \beta \bar{E}_{bc}$ **then**
5: $\hat{\mathcal{U}}_{CH} \leftarrow u_i$
6: **else**
7: $\mathcal{U}_{CM} \leftarrow u_i$
8: **end if**
9: **end for**
10: **for all** $u_i \in \hat{\mathcal{U}}_{CH}$ **do**
11: **if** $E_r^i(t_v) \geq E_{r,Q2}$ **then**
12: $\mathcal{U}_{CH} \leftarrow u_i$
13: **else**
14: $\mathcal{U}_{CM} \leftarrow u_i$
15: **end if**
16: **end for**

$\hat{\mathcal{U}}_{CH}$, otherwise they are put into $\mathcal{U}_{CM}$ (Lines 3-8). It is worth to be noticed that the remaining energy is also affected by the harvested energy during the last reclustering interval. Then, among the FNs in $\hat{\mathcal{U}}_{CH}$, the quantile function is used so that the FNs with the remaining energy higher than the upper quantile index are inserted in $\mathcal{U}_{CH}$, otherwise they are inserted in $\mathcal{U}_{CM}$ (Lines 10-16). In the end, the FNs in $\mathcal{U}_{CH}$ are those with an amount of energy higher than the other FNs at the starting of the reclustering interval.

Predictive-based Harvesting Scheme

In this section, we are proposing a clustering mechanism based on predicting both harvested and consumed energy. The CH selection is performed for avoiding that the selected CHs are running out of energy during the next reclustering interval, preventing that the associated CMs lose their connection with the CH.

The energy harvested by the *i*th FN in the next reclustering interval can be predicted by using the energy harvesting model described above.

$$\tilde{E}_h^i(t_v, \bar{t}) = \int_{v\bar{t}}^{(v+1)\bar{t}} E_h^i(\tau) d\tau. \tag{3.34}$$

By considering the worst case scenario in terms of energy consumption when the FN acts as CH always processing and receiving, it is possible to predict the consumed energy by the *i*th FN in the next reclustering interval as:

$$\tilde{E}_c^i = \left(P_{rx}^i + P_c^i\right) \bar{t} \tag{3.35}$$

Hence, by exploiting (3.27), we can define the predicted remaining energy of the ith FNs at the end of the νth reclustering interval as:

$$\tilde{E}_r^i((\nu+1)\bar{t}) = E_r^i(t_\nu) - \tilde{E}_c^i + \tilde{E}_h^i(t_\nu,\bar{t}). \tag{3.36}$$

Having predicted the remaining energy of all the nodes, we define:

$$\check{\mathcal{U}}_{CH}(t_\nu) = \left\{u_i | \tilde{E}_r^i((\nu+1)\bar{t}) \geq 0\right\},$$

as the CH candidates set, at time instant t_ν. The FNs whose predicted remaining energy at the end of the reclustering interval is positive are considered as the CH candidates. Now, we define $\mathcal{U}_{CM}(t_\nu)$ and $\mathcal{U}_{CH}(t_\nu)$ as:

$$\begin{aligned}
\mathcal{U}_{CH}(t_\nu) &= \left\{u_i \in \check{\mathcal{U}}_{CH}(t_\nu) | E_r^i(t_\nu) \geq E_{r,Q2}(t_\nu)\right\} \\
\mathcal{U}_{CM}(t_\nu) &= \left\{u_i \in \check{\mathcal{U}}_{CH}(t_\nu) | E_r^i(t_\nu) < E_{r,Q2}(t_\nu)\right\} \cup \left\{u_i \notin \check{\mathcal{U}}_{CH}(t_\nu)\right\}
\end{aligned} \tag{3.37}$$

Algorithm 11 Predictable-based 3-Quantile Function

```
1: Input: Ẽ_r^i((ν+1)t̄) ∀u_i ∈ U
2: Output: U_CM and U_CH
3: for all u_i ∈ U do
4:     if Ẽ_r^i((ν+1)t̄) ≥ 0 then
5:         Ǔ_CH ← u_i
6:     else
7:         U_CM ← u_i
8:     end if
9: end for
10: for all u_i ∈ Ǔ_CH do
11:     if E_r^i(t_ν) ≥ E_{r,Q2} then
12:         U_CH ← u_i
13:     else
14:         U_CM ← u_i
15:     end if
16: end for
```

The pseudo-code of the Predictable-based 3-Quantile Function is shown in Algorithm 11, where the input is the estimated remaining energy of the FNs calculated in (3.36), and the output is the $\mathcal{U}_{CH}$ and $\mathcal{U}_{CM}$ (Lines 1-2). In this algorithm, $\tilde{E}_r^i((\nu+1)\bar{t})$ of all FNs is considered as a metric for selecting the CH candidates, $\check{\mathcal{U}}_{CH}(t_\nu)$. The FNs having the estimated energy higher than zero until the next reclustering interval, are considered as the CH candidates while the rest are assumed to be the CMs due to their high probability of running out of energy before the next reclustering (Lines 3-9). The rest of the pseudo-code is

following the same structure of Algorithm 10. In the end, the CHs are selected among those FNs having a sufficient estimated energy at the end of the reclustring interval, under worst conditions, and those having a higher amount of energy in comparison with the other FNs.

3.3.3 The clustering algorithm

After having classified the FNs based on the Harvesting Clustering Scheme or the Predictive-based Harvesting Scheme, the clustering procedure is performed by considering $\mathcal{U}_{CM}$ and $\mathcal{U}_{CH}$ sets as an input. The clustering procedure described here is the one introduced in [79], while it is applied to a different set of FNs including the harvesting effect.

Algorithm 12 Clustering Scheme

1: **Input:** $\mathcal{U}_{CH}$, $\mathcal{U}_{CM}$, $E_r^i(t_v)\ \forall u_i \in \mathcal{U}$
2: **Output:** $\mathcal{C}_i \forall i$
3: **while** $\mathcal{U}_{CH} \neq \emptyset$ **do**
4: Select $u_i \in \mathcal{U}_{CH} | \max_{u_i} \left\{E_r^i(t_v)\right\}$
5: $\mathcal{C}_i \leftarrow u_i$
6: **for all** $u_j \in \mathcal{U}_{CM}$ **do**
7: **if** $d(u_i,u_j) \leq R$ **then**
8: $\hat{\mathcal{C}}_i \leftarrow u_j$;
9: **end if**
10: **end for**
11: **while** $|\hat{\mathcal{C}}_i| < M$ **do**
12: $\mathcal{C}_i \leftarrow u_j | \min_{u_j}(E_r^j(t_v)) \quad \forall u_j \in \hat{\mathcal{C}}_i$;
13: remove u_j from $\mathcal{U}_{CM}$ and $\hat{\mathcal{C}}_i$;
14: **end while**
15: remove u_i from $\mathcal{U}_{CH}$
16: **end while**
17: **if** $\mathcal{U}_{CM} \neq \emptyset$ **then**
18: **for** each $u_j \in \mathcal{U}_{CM}$ **do**
19: $\mathcal{C}_j \leftarrow u_j$;
20: remove u_j from $\mathcal{U}_{CM}$
21: **end for**
22: **end if**

The pseudo-code of the clustering scheme is shown in Algorithm 12, where the lists of CHs and CMs and the remaining energy level of all FNs are the input, while the output is the clusters list (Lines 1-2). Since the goal is that of maximizing the energy lifetime, the clustering algorithm starts from CH set, composed by following one of the Algorithms 10 or 11, by selecting the FN with the highest remaining energy that is put in the *i*th cluster $\mathcal{C}_i$ (Lines 4-5). All the FNs belonging to the CM set are then considered, and among them, those with a distance with respect to the selected CH lower than the coverage range are selected and put in a temporary cluster $\hat{\mathcal{C}}_i$ composed by the candidates CMs of the *i*th CH, as long as the coverage range condition is respected (Lines 6-10). Among the candidates CMs, the

$M-1$ CMs with the lowest remaining energy level are put in the cluster $\mathcal{C}_i$ and removed by both CM and $\hat{\mathcal{C}}_i$ sets (Lines 11-14). The bound M has been considered for taking into account both communications and computing multiple access limitations. The selection of the lowest energy level CMs allows to go in the direction of increasing the network lifetime by limiting the power consumption of the FNs with lower energy values. The selected CHs are removed from their lists once they are assigned (Line 15). In the end, the remaining CMs in $\mathcal{U}_{CM}$ are considered as isolated nodes and hence performing as CHs of a cluster with no CMs (Lines 17-22).

At the beginning of a new reclustering period, the $\mathcal{U}_{CM}$ and $\mathcal{U}_{CH}$ are updated following either the Harvesting Clustering Scheme or the Predictive-based Harvesting Scheme for taking into account the different amount of energy spent by each FN. Then, Algorithm 12 is performed between two consecutive reclustering instants the FNs do not change their role.

3.3.4 Experimental Results

The computer simulations are performed in Matlab for two possible spans of the daily time, as depicted in Table 3.3. We have considered two different starting times and sunshine durations in order to observe the behavior of the network in two different situations. The simulation parameters are listed in Table 3.4.

We hypothesize a squared area A equal to 200×200 m^2, with a variable number of FNs equal to 200, 500, or 700 randomly placed in the area with uniform distribution. The FNs generate data units with a Geometric distribution with p=0.1 and k=10 per reclustering period. The data units are supposed to have always the same size δ_{l_i} and requesting the same amount of operations ω_{l_i}. All FNs are supposed to have the same computational capability and battery capacity; however, in order to have a different initial energy, we have considered that the initial value of the remaining energy of the ith FN is $\gamma_i(0)\overline{E}_{bc}$, where $\gamma_i(0)$ is uniformly distributed in the range $[0.7, 1]$, with different values for different FNs. The reclustering interval $\bar{t}$ has been set equal to 50s; this value has been defined by balancing the need to adapt quickly to the network changes and the management cost. The solar panel is supposed having size equal to 25 cm^2, is south-oriented (i.e., the Azimuth degree is zero), and has an inclination of 36 degrees w.r.t the horizontal ground. Moreover, it is assumed that there are no obstacles, and the ground reflection coefficient is set to 0.20, i.e., the value of stones/rubbles, a value in between the land (0.14), the asphalt (0.10), the roofs (0.13) and the dark buildings (0.27), representing a typical urban landscape with a few clear buildings that have a higher reflection coefficient (0.60).

The parameters of the Guassian modelling the energy acquired by the solar panel, as described in Section 3.3.2, are the peak time μ_i uniformly distributed between 12.30 and

Table 3.3 Harvesting Scenario Definition

Scenario	**Starting Time**	**Length**
1	10:00	4 hours
2	noon	12 hours

13.30, and the variance σ_i such that $(\mu_i - 3 \cdot \sigma_i, \mu_i + 3 \cdot \sigma_i)$ covers the sunshine hours of the given month.

It is noteworthy to point out that the previous equation is valid if and only if the interval $\bar{t}$ is less than 24 hours. However, if one is interested to analyze acquisition periods greater than a day, the Gaussian curve is repeated with a period of 24 hours, hence assuming that the month does not change[4].

For the simulation results, we have considered three different approaches:

- No Harvesting (NH): The FNs are not equipped with a solar panel and energy cannot be harvested. This is the benchmark approach presented in [74].
- Harvesting (H): This is Harvesting clustering scheme presented in Section 3.3.2 where the FNs are able to harvest energy by using a solar panel with the parameters listed in Table 3.5.
- Harvesting and Prediction (*H&P*): This is the approach presented in Section 3.3.2, where the FNs are able to harvest and predict their energy level in the next clustering period.

The NH approach is considered as a benchmark to see the impact of the harvesting in the network lifetime. In the NH approach Algorithm 10 and Algorithm 12 have been used by considering that $E_h^i = 0$ for all FNs. On the other side, in the harvesting approach, the FNs are equipped with solar panel and are able to harvest. Algorithm 10 and Algorithm 12 have been considered for the clustering procedure and (3.27) for the remaining energy of the FNs considering the amount of energy they have harvested. Finally, *H&P* approach considers Algorithm 11 and Algorithm 12 for the clustering procedure and (3.36) for calculating the remaining energy of the FNs. Details about energy harvesting parameters are given in Table 3.5.

In the experimental results we are interested in observing the impact of harvesting solutions on the lifetime of the network measured as the amount of time the FNs deplete their

[4]If needed, for periods significantly longer than a day, it is possible to define more realistic mechanisms. An example sets the sunshine hours equal to the monthly mean only in the first fifteenth day and then progressively changes that value toward the previous/subsequent month's mean

Table 3.4 Harvesting Simulation Parameters

Parameter	**Value**
Dimension	200m x 200m
Data Unit Size (δ_{l_i})	5 MB
Battery Capacity ($\overline{E}_{bc}$)	5000 J
FNs' coverage range (R)	25 m
Data Unit Operations (ω_{l_i})	50 GFLOP
Max number of FNs per cluster (M)	5
min % of remained energy for CH candidates (β)	10%
FN Computation capability (η_c^j)	12G FLOPS
FN Computation power (P_c^i)	0.9 W
FN Idle power (P_{id}^i)	0.3 W
FN Transmission power (P_{tx}^i)	1.3 W
FN reception power (P_{rx}^i)	1.1 W

Table 3.5 Energy Harvesting Parameters

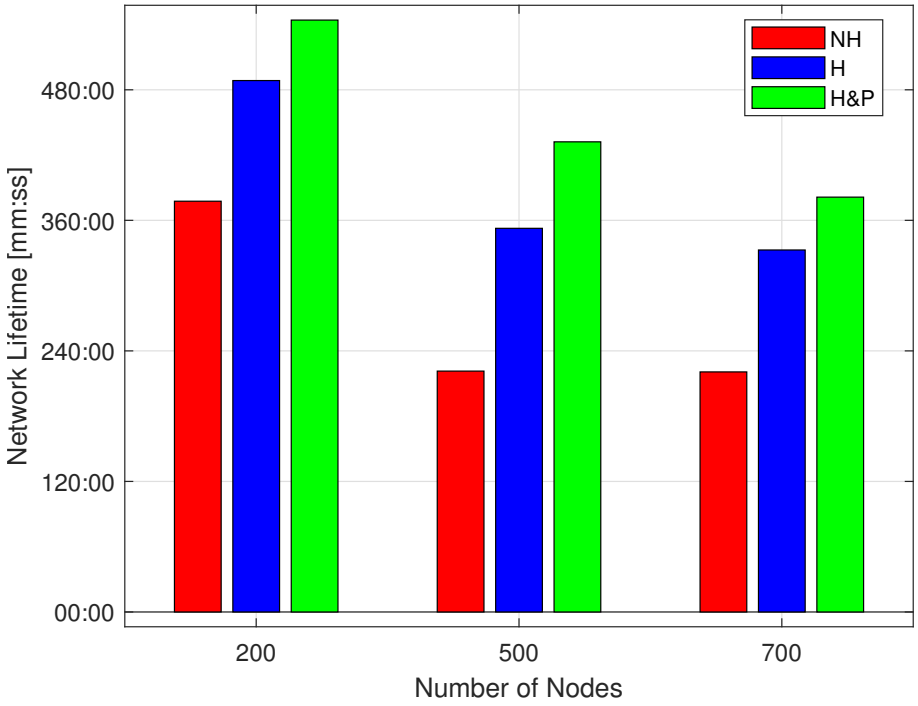
Parameter	**Value**
Solar Panel Area	25 cm^2
Azimut	0 degrees (South-oriented)
Inclination w.r.t. ground	36 degrees
Ground Reflection Coefficient	0.20
Obstacles Occlusion	None
Sun Peak Time Hour	13.00 ($\pm$30 minutes)
Sunshine hours (December–June)	8h52 – 15h42

battery [13]. In particular, in order to show how the *H&P* solution outperforms the others, we are conducting some experiments in the network where FNs have different battery capacities.

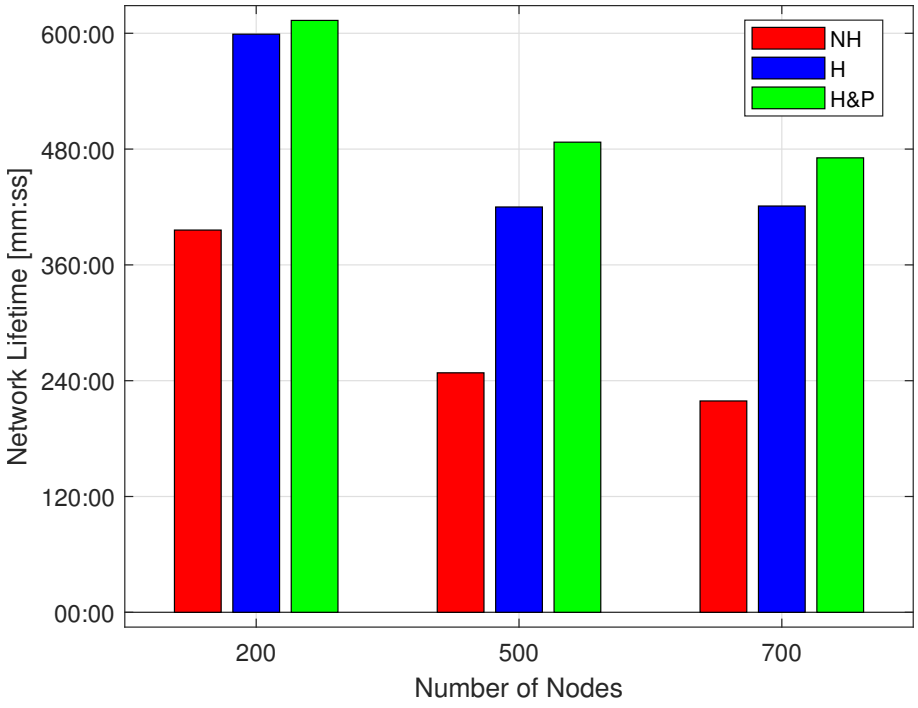
In order to see the pattern of the nodes going off, we have performed an experiment plotting the lifetime expressed in minutes and seconds when all the FNs deplete their energy; the results are shown in Fig. 3.10 for two different starting points until the last node goes off. We have labeled the bars with *NH*, *H*, and *H&P*, which refer, respectively, to the No harvesting, Harvesting and *H&P* based approaches. As seen in both Fig. 3.10a and Fig. 3.10b, in which all the FNs are supposed to have the same battery capacity ($\overline{E}_{bc} = 5000\,\mathrm{J}$), the *H&P* allows to extend the FNs lifetime, as expected. It can also be seen that the *H&P* approach allows to extend the lifetime with respect to the simple harvesting approach thanks to the ability to predict the harvested energy. The other point to be highlighted is that network lifetime in harvesting solutions in Fig. 3.10a is longer than 3.10b and that is due to the higher amount of harvested energy starting at 10:00, owing to the presence of the harvesting peak around noon. In both cases, the NH approach performs in the same way since no harvesting is considered.

In order to identify when the harvesting based solutions have a higher impact on the network lifetime, we are here investigating the impact of the battery capacity on the network lifetime by considering the two different scenarios defined in Tab. 3.3. In particular, the following figures show the cumulative function of the day hour in which the FNs are going off.

In Fig. 3.11, we have compared the results in case of FNs performing NH with a battery capacity $\overline{E}_{bc} = 5000\,\mathrm{J}$ while the battery capacity for H and *H&P* has been set to $\overline{E}_{bc} = 2000\,\mathrm{J}$. It is possible to note that even by reducing the battery capacity to less than one half of the battery capacity used for the NH approach, the nodes show a similar behaviour in terms of lifetime, demonstrating the effectiveness of the harvesting in the network lifetime. This is particularly important showing that if we are able to implement the harvesting solution in the IoT scenario we can reduce the battery size, and consequently its cost, while having a similar behaviour in terms of lifetime. The *H&P* approach allows to reduce even more the number of nodes depleting their energy. Furthermore, it can be seen that when the number of FNs increases, the lifetime decreases; this is due to the fact that when the network is composed by a higher number of FNs, also the clusters are composed by a higher number of nodes reflecting in an increased interaction among FNs and as a result a higher energy consumption. The other interesting point is that the lifetime behaviour in the *H&P* approach for all the FNs scenarios is changing with a lower slope; this is due to the prediction mechanism in which the energy consumption among the FNs is balanced thus reducing the effect of re-clustering.

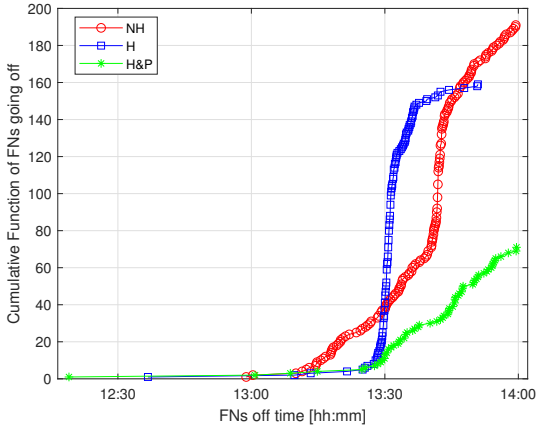


(a) Starting at 10 am

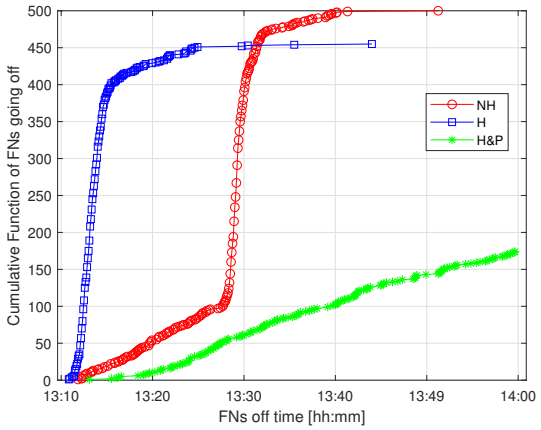


(b) Starting at noon

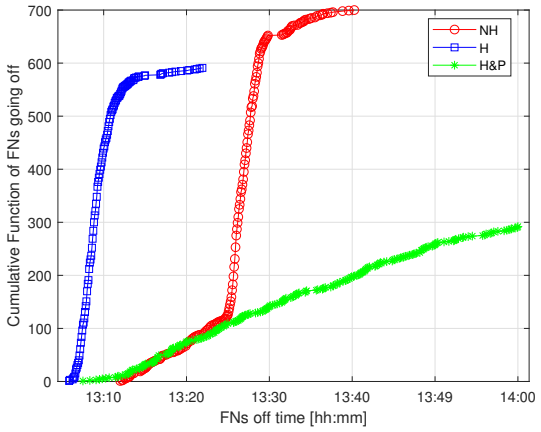
Fig. 3.10 Network Life time of the FNs going off with $\overline{E}_{bc} = 5000\,\mathrm{J}$ for the 3 approaches.



(a) 200 FNs



(b) 500 FNs



(c) 700 FNs

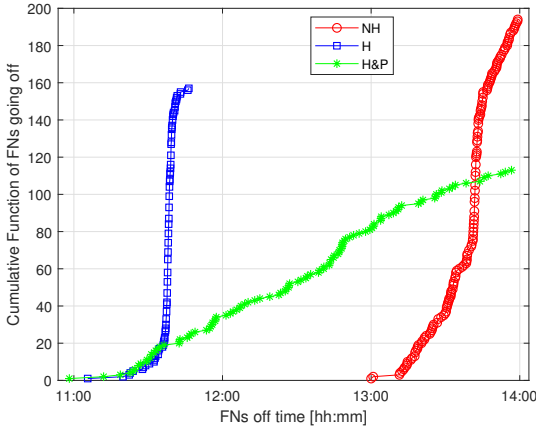
Fig. 3.11 FNs off time in Scenario 1 with $\overline{E}_{bc} = 2000\,\mathrm{J}$

In Fig. 3.12 we have further reduced the battery capacity for FNs with H and *H&P* by setting $\overline{E}_{bc} = 1000\,\mathrm{J}$ while in case of non harvesting $\overline{E}_{bc} = 5000\,\mathrm{J}$. It is possible to notice that in this case the harvested energy is not sufficient, hence the FNs consume their energy earlier than the NH approach in which the nodes are supposed to have a battery capacity of five times larger. However, it is worth to be noticed that a reduced number of FNs still have a certain amount of energy when the nodes with NH are already out. We emphasize the effect of the *H&P* solution, that allows to extend even more the nodes lifetime for about one half of the nodes with respect to the situation of NH with a battery capacity five time larger.

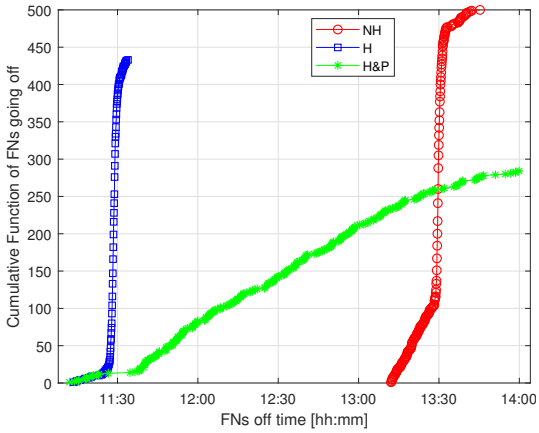
As a final result, we have considered the opposite case in Scenario 2 when the FNs operating the harvesting approach have a battery capacity $\overline{E}_{bc} = 5000\,\mathrm{J}$, while the FNs performing the non harvesting approach have a battery capacity $\overline{E}_{bc} = 8000\,\mathrm{J}$. In Fig. 3.13, the time instant in which the FNs are going off is depicted. It is possible to notice that the three solutions have quite the same performance. However, thanks to the prediction property, FNs in the *H&P* solution consume their energy later than the other two cases. Moreover, it is worth noticing that the FNs operating with harvesting solution have a smaller battery capacity confirming the effectiveness of the proposed approach.

3.3.5 Summary

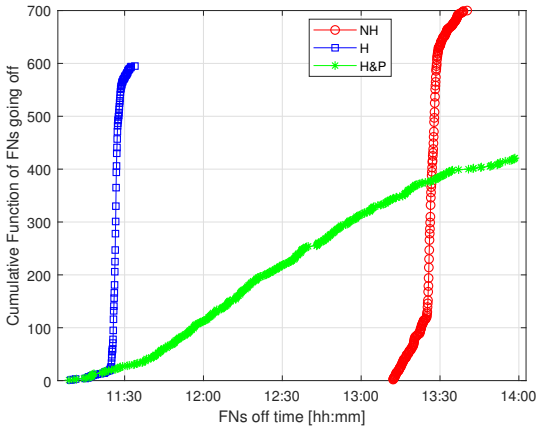
In this work, a clustering mechanism at the network edge in a computation offloading scenario is proposed. We have considered a harvesting scenario where the FNs are able to harvest exploiting on-board solar panels. We have mathematically modeled the energy consumption in the fog networks. Later, we proposed a prediction method in which the CH and CM selection in the clustering solution is affected by the knowledge the FNs have about their consumption and harvesting energy amount until the next clustering period. We have shown in the simulation result how effective the H&P solution is by analyzing different battery capacity.



(a) 200 FNs

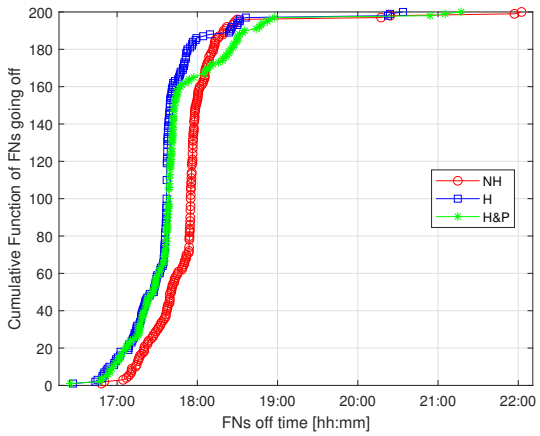


(b) 500 FNs

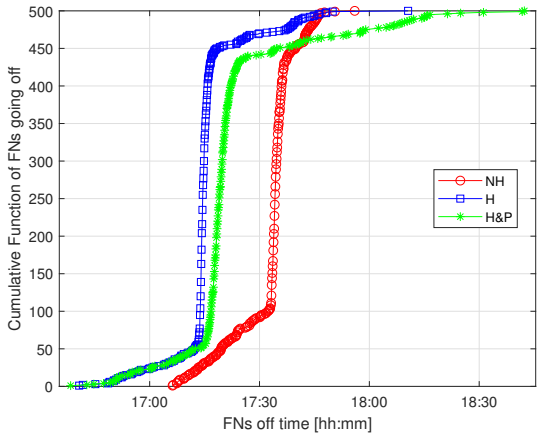


(c) 700 FNs

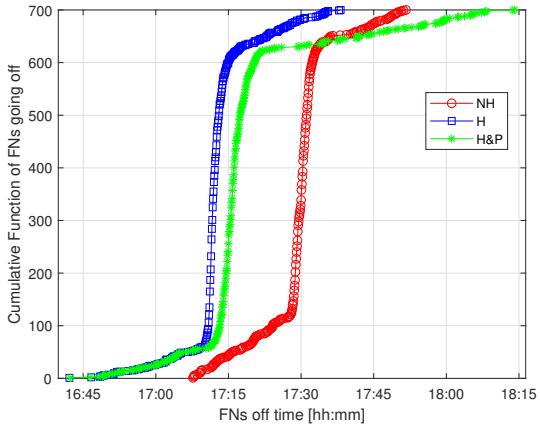
Fig. 3.12 FNs off time in Scenario 1 with $\overline{E}_{bc} = 1000\,\mathrm{J}$



(a) 200 FNs



(b) 500 FNs



(c) 700 FNs

Fig. 3.13 FNs off time in Scenario 2 with $\overline{E}_{bc} = 8000\,\mathrm{J}$

Chapter 4

Vehicular Environment Solutions

The content of the following chapter was extracted from publications [5], [7], and [8] in the publications list.

During the past few years, edge computing has emerged as a distributed computing paradigm that brings the capabilities and resources of the cloud towards the network edge [109]. MEC or as recently renamed by ETSI, multi-access edge computing, offers an ultra-low latency environment with high bandwidth and real-time access to network resources in a mobile cellular network [110].

Vehicles have been evolving since the second industrial revolution and their role in modern life is imperative. With the rapid technological advancements in ICTs, vehicles are equipped with wireless communication capabilities for both intra-vehicle and inter-vehicle communications to support plenty of applications such as road safety, smart transportation, and location-dependent services [111]. Vehicular Edge Computing (VEC) has been widely discussed in the literature [112, 113], where the infrastructure and the vehicles contribute their computing resources to the network. In essence, VEC enables offloading from mobile users to the infrastructure. In performed suitably, task offloading reduces the energy consumption and speeds up the response time of applications in a vehicular environment [114–116].

In vehicular networks, Road Side Units (RSUs), referred to as BSs throughout this chapter, provide reliable wireless access in their coverage ranges; however, vehicles cannot have a continuous connection to the BSs. In this chapter, we study task offloading scenarios from vehicles in a VEC scenario.[1] In vehicular networks, due to the mobility of the vehicles, the network topology and the availability of different networks on the roadside change frequently. Hence the traffic generated in network and the sojourn time of the vehicles in the network is of paramount importance when making offloading decisions in VEC. To this aim, in this chapter we propose two offloading mechanisms in vehicular environments.

4.1 State of the arts on Vehicular Environment Solutions

The research community is very active on computation offloading in MEC. The authors in [8] have considered the effect of mobility, users' local load and availability of cloudlets for developing an optimal offloading algorithm and compared the performance in case of always performing computation locally, always offloading or randomly selecting one of these modes. The idea of exploiting fog networking concepts applied to vehicular environment seems also a promising trend. The authors in [117] propose a vehicular fog computing infrastructure in which vehicles with more resources are considered as the computational infrastructure, to relieve the burden of the congested resource limited vehicles. In [118] a local roadside cloud-based network is proposed to deal with traffic-related data. A mobility-aware offloading

[1] More specifically, the generator of the tasks can be the driving system or the passengers of the vehicle.

decision strategy exploiting genetic algorithm for a single job, multi component is proposed in [119] to improve offloading success rate and decrease energy consumption.

There have been plenty of works done on vehicular networks, such as content downloading and guaranteeing internet access[120, 121], content popularity and cooperative caching policy to reduce latency [122, 123], cooperative offloading [124] and cooperative downloading [125, 126].

On the other hand, there have been a lot of research in computation offloading in a mobile environment considering joint optimizations. In [114], the authors investigate the problem of energy conservation on mobile devices by offloading tasks to the infrastructure-based cloud. In [115], the authors propose a heuristic offloading decision algorithm, which jointly optimizes the offloading decision, and communication and computation resources. In [116], the computation offloading decision, resource allocation and content caching strategy as an optimization problem was investigated. The authors in [127], address jointly optimization of the offloading decisions, the allocation of computation resource, transmit power, and radio bandwidth. In [128], the authors target a joint optimization of offloading decision making, computation resource allocation, resource block assignment, and power distribution in a mixed fog and cloud network. However, in the above works, the authors only considered the performance of processing a single task.

In [129], the authors consider a dynamic environment with different wireless networks among which the decision is to select the network reducing the execution cost. In [130], the authors have suggested an offloading scheme to fog layer in a mobile environment for users. They have modeled the offloading such that if the sojourn time of the mobile users is less than transmission time, they perform a local computation, and for the offloaded tasks if the computation time is less than the sojourn time, the BS sends the result back to the user, otherwise the task will be migrated to the cloud for relaying the result to the destination BS of the user. The objective of the work is minimizing the cloud migration by proposing a generic-based solution. However, no queuing model has been considered in these works.

Furthermore, task offloading in Vehicular Adhoc Networks (VANET) and VEC has also been investigated. In [131] the authors propose a MEC-based computation offloading framework to minimize the vehicles' cost in task offloading, while guaranteeing processing delay. In [132], in order to maximize the economical profit of service providers while guaranteeing the delay tolerance of tasks, the authors develop a game theoretic approach to jointly optimize task offloading decision and computation resource allocation. In [117] the authors study the idea of utilizing vehicles as the infrastructure for communication and computation where both scenarios of moving and parked cars as infrastructure were analyzed. In [133], the authors formulate a dual-side cost minimization, jointly optimizing

the offloading decision and local CPU frequency on the vehicles side, and the radio resource allocation on the server side. First of all, none of these works has considered the minimization of waiting time in the process of network or BS selection. Moreover, although these works studied offloading in VEC, none has discussed the availability of the BS for the overall task offloading. In other words, the selected BS might not be able to process the offloaded task within the sojourn time of the vehicle. Task offloading in a vehicular scenario has also been studied in [134], however, the waiting and reception time have been ignored in the latency model. A graph-based scheduling scheme for Vehicle to Vehicle (V2V) and vehicle to infrastructure communication has also been studied in [135].

By studying the literature, a lot of works can be found proposing learning algorithms for task offloading. In [136], the authors targeted the minimization of delay and utilization of physical machines for task offloading in mobile cloud computing. Due to the high dimension of state and action space for a Deep Neural Network (DNN) solution, they proposed a two-layer RL structure, where in the first layer they propose a Deep RL (DRL) method to select the optimal cluster and in the second layer a Q-learning approach to select the optimal physical machine in the selected cluster. Similarly in [137], the authors aim at minimizing the latency for task assignment to the servers and propose a RL method. However, these works consider simple scenarios without any detailed formulation on the delay model.

Moreover, there can be a recent paper found targeting the minimization of delay in computation offloading scenario exploiting Multi-armed Bandit (MAB) method[138]. The proposed MAB solution is for a V2V scenario and assumes all offloaded tasks are received from the same vehicle when offloaded, however, in reality vehicles might not be able to receive the result while they are within the sojourn time of the processing vehicle. The proposed scenario is suitable under a specific mobility case. Moreover, there is no queue model in the scenario, while one vehicle might be sent multiple tasks from different vehicles. Besides, Reward distribution is stationary while in our scenario, we have considered a non-stationary environment which is more realistic to a vehicular environment. Task offloading in a V2V scenario brings high packet loss, due to the high mobility, if no relaying mechanism is foreseen. To this aim, we have designed an offloading method from vehicles to infrastructure while considering a relaying possibility for the constraint on sojourn time. A similar work to [138], is [139], where the authors considered network load as a parameter in the Upper Confidence Bound (UCB) Function which weighs offloading when the network load is low. The effectiveness of Sliding Window-UCB (SW-UCB) in a piece-wise stationary environment is also analyzed in [140] to manage the users mobility in terms of connection to the available cells.

In the end, task offloading in VEC is a promising area that has received a lot of attention from research community in the literature. In the following, the proposed ideas are presented.

4.2 D2D Control Plane With and Without Relaying

The continuous rise of mobile applications has led to an exponential growth of demand in high computational capability in wireless cellular networks [141]. Edge computing brings this computational capabilities closer to the users and enables a large number of devices to process their tasks at the network edge instead of transmitting to the centralized cloud infrastructure by saving energy consumption, limiting the traffic to the fronthaul, and providing services with faster response. Among different scenarios, mobility and computation offloading, which are largely served within the bound of the network edge, have been adopted in internet of vehicles [142, 143]. Edge computing is also considered one of the fundamental techniques of the Fog Networking, where the focus is more on the architectural point of view, in particular toward IoT applications [144]. In this work a partial offloading technique for edge computing environments is proposed to be used in a mobile urban scenario.

In this work we have considered a partial offloading technique in an urban vehicular environment at the network edge, by considering two main types of device: FNs, smart mobile devices generating the tasks to be processed, and the F-APs, devices able to process the offloaded tasks. In cloud computing, the users are able to offload their tasks to the centralized cloud, however, in some cases, e.g., for real time applications, the delay from centralized cloud might not be acceptable. On the other side, in edge computing, the FNs are able to exploit the other FNs and the F-APs for offloading their computational tasks and reduce the amount of traffic sent to the centralized cloud [145, 38]. Due to the storage and energy limitation of the FNs, it is not always feasible to consider direct FNs to FNs offloading; as a result, in this work, we are considering that FNs are able to offload to the F-APs.

On the other hand, computational offloading in a mobile environment is a challenging issue, mainly due to the devices mobility. To this aim, the idea at the basis of this work is that of exploiting FNs to FNs communications (e.g., through D2D connections) for updating the FNs about the status of the system. By leveraging on a similar concept introduced in [146], an idea could be that of employing the D2D communications among FNs for sharing those parameters needed for optimally estimating the amount of data that can be offloaded to the nearby F-APs while respecting the constraints imposed by the mobility. To this aim the network can be seen as composed by two logical connections: a control plane among FNs and a data plane between FNs and F-AP for implementing the task offloading. We

have here considered the possibility to have two types of F-APs, fixed and mobile. In order to reduce the outage probability due to a delayed response from the F-APs computing the offloaded task, a relaying policy has also been considered between mobile and fixed F-APs. Furthermore, we have investigated the impact of the amount of tasks, that each F-AP can manage, on the task processing delay.

4.2.1 System Model and Problem Formulation

In this work a two layer Fog architecture for edge computing is considered. On one hand, $\mathcal{U} = \{u_1, \ldots, u_i, \ldots, u_N\}$ represents the set of FNs in the first layer. All the FNs have computational and storage capabilities; FNs can communicate among them within a specific range depending on the deployed wireless technology. On the other hand, in the second layer, there are two types of F-APs, fixed and mobile. The set of mobile F-APs is shown as $\mathcal{C} = \{c_1, \ldots, c_m, \ldots, c_M\}$, and fixed F-APs as $\mathcal{F} = \{f_1, \ldots, f_k, \ldots, f_K\}$. Fixed F-APs have higher computational and storage capabilities comparing with mobile F-APs and they both have higher capabilities comparing with the FNs. F-APs are able to communicate with the FNs and compute the offloaded tasks. The fixed F-APs have a wider coverage range comparing with the FNs and the mobile F-APs, and are able to aggregate the FNs' traffic requests, while mobile F-APs and FNs are supposed to have the same coverage range.

Each FN having a task to be computed can have different choices: perform a local computation, offload to either a fixed or mobile F-AP in proximity or partially offload to the F-APs; the goal of the proposed partial offloading technique is to estimate the amount of data to offload in order to minimize the outage probability and the task processing delay. In our work, the outage probability corresponds to the probability that an offloaded task cannot be received back by the offloading FN due to the devices mobility, while the task processing delay, corresponds to the time needed for processing the task by taking into account both local and offloaded amount.

We have considered a street scenario, as shown in Fig. 4.1, where the generic ith car, acting as FN, can move with velocity $\vec{v}_i$ in two directions: left to right or the reverse depending on the lane they are located. Likewise, the mth mobile F-AP, which can be a bus or truck, is moving with a velocity $\vec{v}_m$ in a direction depending on the lane they are located [147]. Moreover, there are some fixed F-APs (e.g., located on light poles) at the roadside with a broader coverage area to cover the street when there is no mobile F-APs available. The priority from each FN is offloading to the mobile F-APs, and, then to the fixed F-APs.

In general, the computational time for the lth task by any device is defined as:

$$T_c^l = O_l / \eta_c \tag{4.1}$$

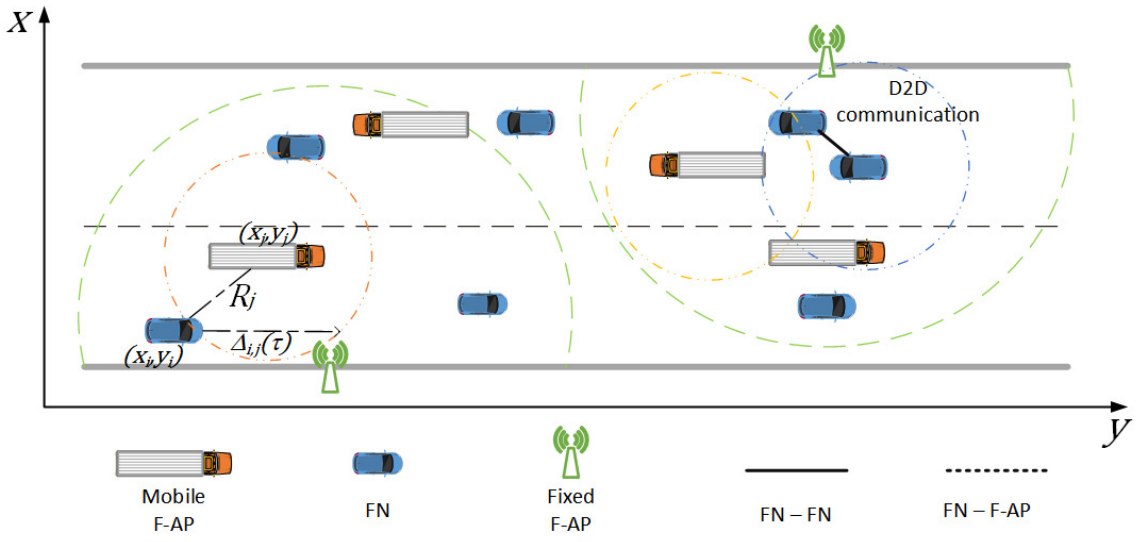


Fig. 4.1 Partial offloading mobile urban scenario.

where O_l represents the number of operations required for computing the lth task and η_c is the FLOPS depending on the CPU of the processing device, which can be an FN or an F-AP.

In case of offloading, each task should be transmitted, hence, the transmission time for the lth task can be written as:

$$T^l_{tx,ij} = L_{s_l}/r_{ij} \tag{4.2}$$

where L_{s_l} is the size of the lth task requested by the ith FN and r_{ij} is the data rate of the link between the ith FN and the jth F-AP which could be either fixed or mobile. Later the result of the processed task should be sent back to the ith FN, leading to a reception time defined as:

$$T^l_{rx,ij} = L_{r_l}/r_{ij} \tag{4.3}$$

where L_{r_l} is the size of the result of the requested task sent back from the F-AP to the offloading FN, when we suppose a symmetric channel in terms of data rate between the ith FN and the jth F-AP. Each F-AP is supposed to have a buffer holding the tasks of the requesting FNs to be processed. The waiting time of the lth task at the jth F-AP can be defined as:

$$T^l_{w_j}(p) = \sum_{\lambda=1}^{p-1} T^{\lambda}_{c_j} \tag{4.4}$$

where p is the number of tasks already in the queue of the jth F-AP. The waiting time for the task to be processed plus the computing time at the F-AP corresponds to the FN idle time when the FN waits for the result back.

The concept behind partial offloading is to delegate only a portion of the computational load to another device to optimize energy and time [14]. We define α_l as the portion of the lth task that is offloaded. As a result, the time required for offloading a task can be written as the sum of the time for sending the portion of the task, the time the task should wait in the F-AP processing queue, the time for computing that task at the F-AP and the time needed for having the result back:

$$T^l_{off,i}(\alpha_l) = \alpha_l T^l_{tx,ij} + T^l_{w_j} + \alpha_l T^l_{c_j} + \alpha_l T^l_{rx,ij} \tag{4.5}$$

while the time for local computation, can be defined as the time needed for computing the remaining portion of the task:

$$T^l_{loc,i}(\alpha_l) = (1-\alpha_l) T^l_{c_i} \tag{4.6}$$

Thus, in case of partial offloading, the total delay for processing a task can be rewritten as the maximum of the two delays, i.e.,

$$D^l_i(\alpha_l) = \max\{T^l_{off,i}(\alpha_l), T^l_{loc,i}(\alpha_l)\} \tag{4.7}$$

In order to estimate the amount of data that can be offloaded we have to estimate the amount of time that the ith FN remains under the coverage of the jth F-AP for avoiding to have the result back when the FN is out of coverage. The remaining distance before going out of the coverage of the jth F-AP at time instant τ, as defined in [146], is equal to:

$$\Delta_{i,j}(\tau) = \sqrt{R_j^2 - (y_j(\tau) - y_i(\tau))^2} \pm (x_j(\tau) - x_i(\tau)) \tag{4.8}$$

where $\{x_i(\tau), y_i(\tau)\}$ and $\{x_j(\tau), y_j(\tau)\}$ are, respectively, the position of the ith FN and the jth F-AP at time τ and R_j is the radius of the jth F-AP's coverage area[2]. Thus, the time that the ith FN remains in the coverage area of the jth F-AP (i.e., sojourn time) can be written as:

$$\bar{T}^{i,j}_{\tau}(\alpha_l) = \frac{\Delta_{i,j}(\tau)}{\partial v_{ij}} \tag{4.9}$$

where $\partial v_{ij} = |\vec{v}_i - \vec{v}_j|$ is the modulo of the vector speeds of the ith FN and jth F-AP taking into account their relative direction. The outage for the lth task of the ith FN can be defined

[2]In case the vehicles are in the lower lane the operator between the first and the second term is +, otherwise is -.

as:

$$\Omega_l^i(\alpha_l) = \begin{cases} 1 & \text{if } \bar{T}_\tau^{i,j}(\alpha_l) < T_{off,i}^l(\alpha_l) \\ 0 & \text{if } \bar{T}_\tau^{i,j}(\alpha_l) \geq T_{off,i}^l(\alpha_l) \end{cases} \tag{4.10}$$

corresponding to the occurrence that, due to the FNs and F-APs mobility, the time needed for offloading a task is higher than the FN sojourn time within the F-AP coverage area. Having the goal of minimizing the outage probability and processing delay, we define our minimization problem as:

$$\begin{cases} \min_\alpha \left\{ \sum_{i=1}^N \sum_l \Omega_l^i(\alpha_l) \right\} \\ \min_\alpha \left\{ \sum_{i=1}^N \sum_l D_i^l(\alpha_l) \right\} \end{cases} \tag{4.11}$$

subject to

$$T_{c_i}^l > T_{c_q}^l > T_{c_k}^l > T_{tx,ij}^l > T_{rx,ij}^l > 0 \tag{4.12}$$

$$R_m < R_k \tag{4.13}$$

$$|\vec{v}_j| < |\vec{v}_i| \tag{4.14}$$

$$0 \leq \alpha_l \leq 1 \tag{4.15}$$

where α is the set of the offloaded portion of all the tasks in a given time instant.Hence, there are two objectives in the formulation, i.e., minimizing the sum of the tasks not successfully received due to devices mobility during the offloading phase and the sum of all the total delays suffered by all of the tasks, respectively shown in (4.11). Constraint (4.12) introduces the hypothesis that the FNs computing time is higher than that of the mobile F-APs, that is even higher than that of the fixed F-APs. All these computational times are supposed to be higher than FNs transmission and receiving times. Constraint (4.13) set the fixed F-APs coverage area higher than the mobile F-APs. Constraint (4.14) means that the velocity of the FNs is higher than that of the mobile F-APs. Finally, the offloaded portion is always between 0 and 1 as shown in constraint (4.15).

In the following, we resort to a suboptimal solution by relaxing some of the hypotheses and employing D2D communications among FNs for sharing information to be used for the partial offloading estimation.

4.2.2 D2D assisted partial offloading

The optimization procedure is based on evaluating a closed form expression for the optimized α by relaxing some of the problem constraints. However, due to the mobility of FNs, some of the parameters cannot be considered as known by FNs. Hence, we aim at exploiting D2D

communications among FNs to exchange information related to the status of the F-APs (i.e., waiting time, node position and direction, velocity). Then, the estimated information is used by the FNs to calculate the amount of data to be offloaded. In the end a relaying method between mobile and fixed F-APs is also proposed in order to reduce the outage probability.

Ideal partial offloading estimation

As a first step for the optimization procedure the F-APs within the coverage area of a given FN are selected as potential candidates for offloading. All FNs prioritize the mobile F-APs in the network edge for offloading, and if there is no mobile F-AP they will offload the task to a fixed F-AP. In order to minimize the outage probability the ith FN having a task to be processed selects the F-AP allowing to maximize the sojourn time within its coverage area. Hence, the jth F-AP is selected such that:

$$\max_j \left\{ \psi_\tau^{i,j}(\alpha_l) \right\} = \max_j \left\{ \bar{T}_\tau^{i,j}(\alpha_l) - T_{w_j}^l \right\} \tag{4.16}$$

corresponding to select the F-AP with the highest available time $\psi_\tau^{i,j}$, that is a function of both sojourn time (4.9) and task waiting time (4.4) in the buffer of the F-APs due to previous ongoing computations. It is worth to be noticed that the sojourn time (4.9) is a function of velocities and directions of both ith and jth devices.

In order to minimize the outage probability, we aim at optimizing the portion of the tasks to be offloaded. To avoid outage, the offloading time of the task portion from an FN should be less than the sojourn time in the coverage area of the selected F-AP for offloading, as shown in the second condition in (4.10). To find the portion of the lth task which can be offloaded considering the offloading time and the velocity, exploiting (4.8) and (4.9), we can rewrite the second condition in (4.10), corresponding to no outage, as:

$$\alpha_l \frac{L_{s_l}}{r_{ij}} + T_{w_j}^l + \alpha_l \frac{O_l}{\eta_{c_j}} + \alpha_l \frac{L_{r_l}}{r_{ij}} \leq \frac{\Delta_{i,j}(\tau)}{\partial v_{ij}} \tag{4.17}$$

that allows to find the optimal α_l parameter, as:

$$\alpha_l \leq \frac{\Delta_{i,j}(\tau) - T_{w_j}^l \cdot \partial v_{ij}}{\partial v_{ij} \cdot \left\{ \frac{L_{s_l}}{r_{ij}} + \frac{O_l}{\eta_{c_j}} + \frac{L_{r_l}}{r_{ij}} \right\}} \tag{4.18}$$

The above condition allows to minimize the outage condition by setting an upper limit on the amount of data to be offloaded. However, the reliability of the calculated α_l parameter

depends on the knowledge of some input information, i.e., direction and velocity, and task waiting time in the F-AP computing buffer.

D2D assisted information sharing

In order to know the parameters to be used for estimating (4.18) we rely on the D2D communications among FNs that is used for sharing information related to waiting time, velocity and direction of movement.

Hence, we suppose that when an FN receives back the result of its offloaded task, it is also able to estimate the amount of time the task has waited in the queue of that specific F-AP, as well as its velocity and direction, and also the time instant this information has been estimated, corresponding to τ. The updated set of information at time instant τ of the information obtained by the ith FN from the jth F-AP corresponds to $\left\{\tilde{T}_{w_j}(\tau), \tilde{\vec{v}}_j(\tau)\right\}$, where $\tilde{T}_{w_j}(\tau)$ corresponds to the waiting time in the jth F-AP and $\tilde{\vec{v}}_j(\tau)$ corresponds to the velocity and direction of the jth F-AP, both estimated by ith FN at time instant τ.

In the proposed idea as two FNs are approaching, they update their set by comparing the time in which the information regarding the corresponding F-AP has been updated in order to record only the most recent values. If the sender's updating time is more recent, the information about that F-AP will be updated in the recipient FN's set. This corresponds to say that the information in the buffer of each FN, can be written as:

$$\mathcal{B}_i = \left\{\tilde{T}_{w_j}(\bar{\tau}), \tilde{\vec{v}}_j(\bar{\tau}) | \bar{\tau} = \max_{\iota}(\tau_\iota), d_{i\iota} \leq R_i\right\} \ \forall j \tag{4.19}$$

where $\bar{\tau}$ is the maximum updating time instant, i.e., the most recent time instant, among all the approaching FNs that are in the D2D coverage area of the ith FN, that is equal to R_i, while $d_{i\iota}$ us the distance between the ith and the ιth FNs. Information related to the waiting time, direction and velocity of each F-AP is spread out through the D2D connections whenever FNs are approaching and used as an input for estimating α_l for minimizing the outage probability and the task processing delay.

In order to see the impact of the parameters in (4.19) on the results, we are considering two types of information spread among the FNs. In case the information related to the velocity and direction of F-APs is spread through the FNs, by exploiting (4.18), we could rewrite the offloading parameter estimation as:

$$\dot{\alpha}_l \leq \frac{\Delta_{i,j}(\tau)}{\partial \tilde{v}_{ij} \cdot \left\{\frac{L_{s_l}}{r_{ij}} + \frac{O_l}{\eta_{c_j}} + \frac{L_{r_l}}{r_{ij}}\right\}} \tag{4.20}$$

while, when the waiting time is also spread through the FNs D2D connection, by exploiting (4.18), we could rewrite the offloading parameter estimation as:

$$\ddot{\alpha}_l \leq \frac{\Delta_{i,j}(\tau) - \tilde{T}^l_{w_j} \cdot \partial\tilde{v}_{ij}}{\partial\tilde{v}_{ij} \cdot \left\{ \frac{L_{s_l}}{r_{ij}} + \frac{O_l}{\eta_{c_j}} + \frac{L_{r_l}}{r_{ij}} \right\}} \tag{4.21}$$

where $\partial\tilde{v}_{ij}$ is the estimated velocity modulo of the vector difference between ith FN and jth F-AP.

F-AP Relaying

After the task computation by the mobile F-AP, the result will be sent to both the FN and the nearest fixed F-AP, so that in case the result can not be received due to the devices mobility, the fixed F-AP with its broader coverage area will send the result back to the requesting FN. This will lead to a significant reduction of outage probability. In this case, the outage becomes:

$$\hat{\Omega}^i_l(\alpha_l) = \begin{cases} 1 & \text{if } \bar{T}^{i,k}_\tau < \hat{T}^l_{off,i} \\ 0 & \text{if } \bar{T}^{i,k}_\tau \geq \hat{T}^l_{off,i} \end{cases} \tag{4.22}$$

by considering the sojourn time of the ith FN in the coverage area of the kth fixed F-AP, $\bar{T}^{i,k}_\tau$, while the delay for the offloading phase becomes:

$$\hat{T}^l_{off,i}(\alpha_l) = \begin{cases} \alpha_l T^l_{tx,im} + T^l_{w_m} + \alpha_l T^l_{c_m} + \alpha_l T^l_{tx,mk} + \alpha_l T^l_{rx,ik} & \text{(4.23a)} \\ \alpha_l T^l_{tx,ij} + T^l_{w_j} + \alpha_l T^l_{c_j} + \alpha_l T^l_{rx,ij} & \text{(4.23b)} \end{cases}$$

where the first equation refers to the case with relaying while the second one for the case with no relaying; the additional term in the relaying delay is due to the transmission time between the mth mobile F-AP and the kth fixed F-AP.

4.2.3 Numerical Results

In this section, the numerical results obtained through computer simulations in Matlab are presented; the parameters used for the scenario are shown in Tab. 4.1. The computer simulations are carried out in terms of average task delay and outage probability, defined as:

- Average Task Delay: The average time spent for offloading or for performing the local computation (See (4.7)).

Table 4.1 Simulation Parameters for Vehicular Environment

Parameter	**Value**
Dimension	500m x 20m
Task size (L_s)	5 MB
Task result size (L_r)	1 MB
Channel Model	Extended Vehicular A model (EVA) [148]
FN and mobile F-AP coverage range (R_i, R_m)	15 m
Fixed F-AP coverage range (R_k)	50 m
Task Operation (O_l)	50G
FN FLOPS (η_{c_i})	15G FLOPS (= 1 CPU)
Mobile F-AP FLOPS (η_{c_j})	30G FLOPS (= 2 CPUs)
Fixed F-AP FLOPS (η_{c_j})	60G FLOPS (= 4 CPUs)
FN Velocity ($\|v_i\|$)	[8-12] m/s
Mobile F-AP Velocity ($\|v_m\|$)	[5-7] m/s

- Outage probability: The average probability of number of unsuccessful receptions by FNs, due to devices mobility, over total number of generated tasks (See (4.10) and (4.22)).

In this section we will compare the performance of the D2D approaches with a benchmark that considers to know perfectly all the needed parameters, and labeled as ideal. The comparison is done with two possible D2D approaches, taking into account the impact of the information spread through the nearby FNs on the performance. In particular, when we suppose that the FNs share the information only about the velocity and direction of movement of the F-APs, as defined in (4.20), the scenario is labeled as D&V, while when the information regarding the waiting time is also known, as defined in (4.21), it is labeled as D&V&T_w.

We have compared the performance of these three approaches in terms of delay and outage probability for different number of FNs and F-APs, by considering a task generation rate equal to 0.1 task per second. In the following we will refer as computational capacity as the amount of task that each F-AP can manage; in particular with low computational capacity each F-AP can backlog an amount of tasks equal to the number of CPUs (i.e., p=2 per CPU), while with high computational capacity can backlog two task for each CPU (i.e., p=3 per CPU). Moreover the effect of the relaying between mobile and fixed F-APs is considered. The location of 20 mobile F-APs is generated randomly; 10 of them are on the right lane while and 10 in the left lane.

Figs. 4.2 and 4.3 depict the average outage probability of the FNs for different scenarios in the presence and absence of relaying among the F-APs. As seen, whenever more information regarding F-APs waiting time, velocity and direction is known, the performance is better because of the better estimation of the portion to be offloaded. Moreover, when there is a relay among the F-APs, the outage probability is reduced, and this is due to the receiving back the result from the fixed F-APs because of the higher coverage area. Furthermore, we can

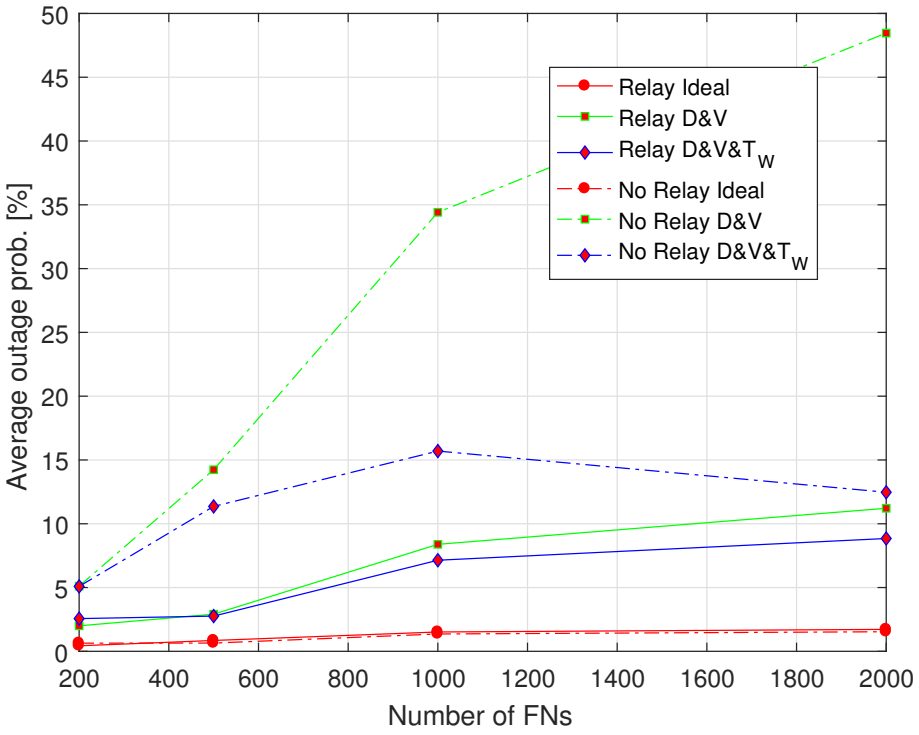


Fig. 4.2 Outage Probability of 11 fixed F-APs with high capacity

notice that in Fig. 4.2 where the F-APs have higher capacity, the average outage probability is slightly decreased comparing with Fig. 4.3 where the F-APs have lower capacity.

The average outage probabilities when there are 5 fixed F-APs are depicted in Figs. 4.4 and 4.5. The performance order of the techniques is the same as for 11 fixed F-APs, however, it can be noticed that when number of fixed F-APs decreases from 11 to 5, the outage probability increases. This is because fixed F-APs have a broader coverage area and higher computational capabilities and by having fewer of them in the scenario more tasks will be offloaded to the mobile F-APs which increases the outage probability. Furthermore, in Fig. 4.5 where the F-APs can manage more tasks, the outage probability is lower comparing with Fig. 4.4 in which the capacity of the F-APs is lower.

Figs. 4.6 and 4.7 depict the average task delay of the network with 11 fixed F-APs where there is, respectively, a relay among the F-APs in the first one and there is no relay in the second one. It can be seen that relaying does not have an impact on the delay, however, delay is highly influenced by the capacity of the F-APs. When the F-APs have higher capacity more tasks can be processed and kept in the queue which will result in parallel computation in F-APs and local computation in FNs, when partial offloading, which will result in a lower delay.

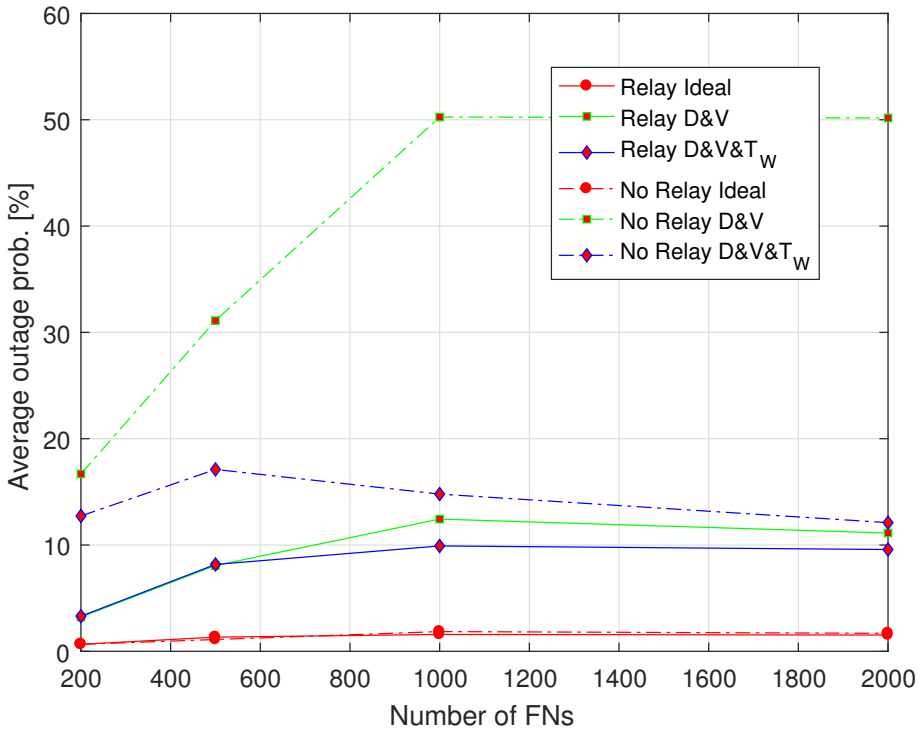


Fig. 4.3 Outage Probability of 11 fixed F-APs with low capacity

The simulation results underscore the impact of the proposed estimation approach and employing the D2D communication on the performance in terms of outage probability and delay. It is proved that the knowledge about waiting time, velocity and direction of the other nodes can greatly impact the accuracy of the estimation of the offloaded portion. By having a D2D communication for informing the other FNs about the status of the F-APs, FNs are able to better estimate how much they can offload in order to have the lowest amount of delay and outage probability.

4.2.4 Summary

The partial offloading problem in mobile edge computing in a mobile urban scenario with FNs and F-APs mobility is considered. FNs consider the remaining time in the coverage for the selection of an F-AP and estimate the portion of task to offload in order to avoid outage. By using a D2D communication the information of the F-APs among FNs is spread, allowing to better estimate the task offloading portion. A relaying technique is also proposed for minimizing the outage. Simulation results demonstrate that by benefiting from the D2D communication and relaying the result among F-APs, outage probability is minimized. Moreover, the impact of the F-APs capacity on the average task delay is shown.

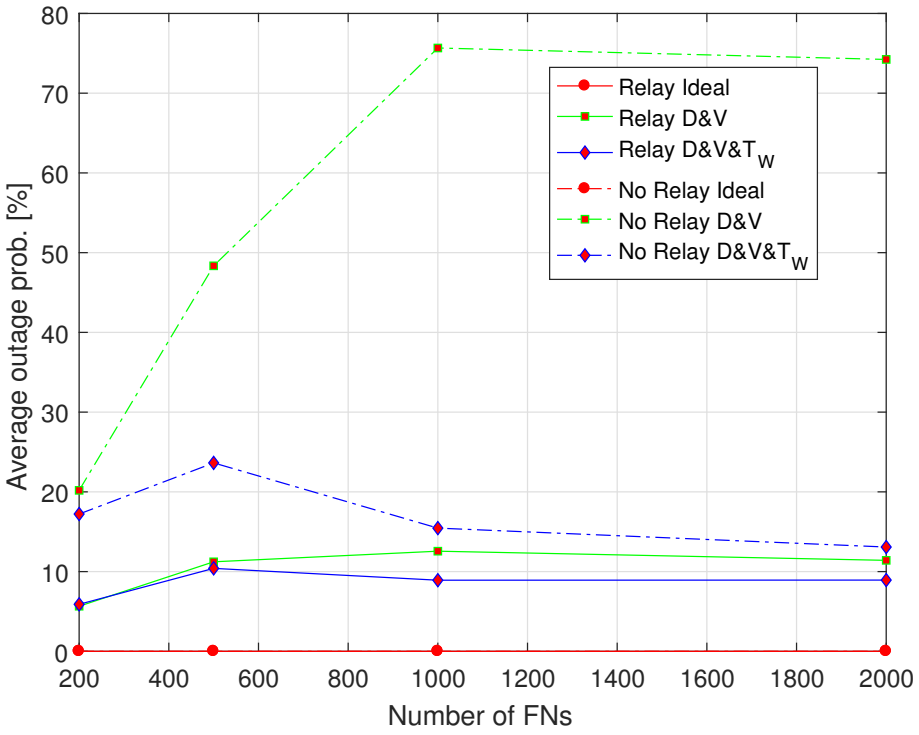


Fig. 4.4 Outage Probability of 5 fixed F-APs with low capacity

4.3 Multi Armed-Bandit Solution for Vehicular Edge Computing

In this work, the proposal is built on the idea of combining cellular networks and MEC. Different cellular networks are deployed in a vehicular environment with different network characteristics and the number of BSs. In such a dynamic wireless environment, vehicles can access different wireless networks, such as cellular networks, Wi-Fi and so on. Thus, they prefer to select the best network and the BS for internet access to perform the offloading.

Despite its great potential, offloading in VEC is associated with several challenges. For example, low latency plays a crucial role in ensuring acceptable QoE; however, it is difficult to achieve in a VEC scenario. It is known that in MEC, waiting times usually dominate the transmission times [149]; Therefore, in this work, we concentrate on the analysis of waiting times. The results, however, are simply extendable to the case that includes the transmission time.

To address the aforementioned problems, a learning-based solution is proposed for making offloading decisions where the vehicles aim at selecting the best network in terms of congestion. Exploiting Bandit theory, the vehicle takes advantage of the historical offloading

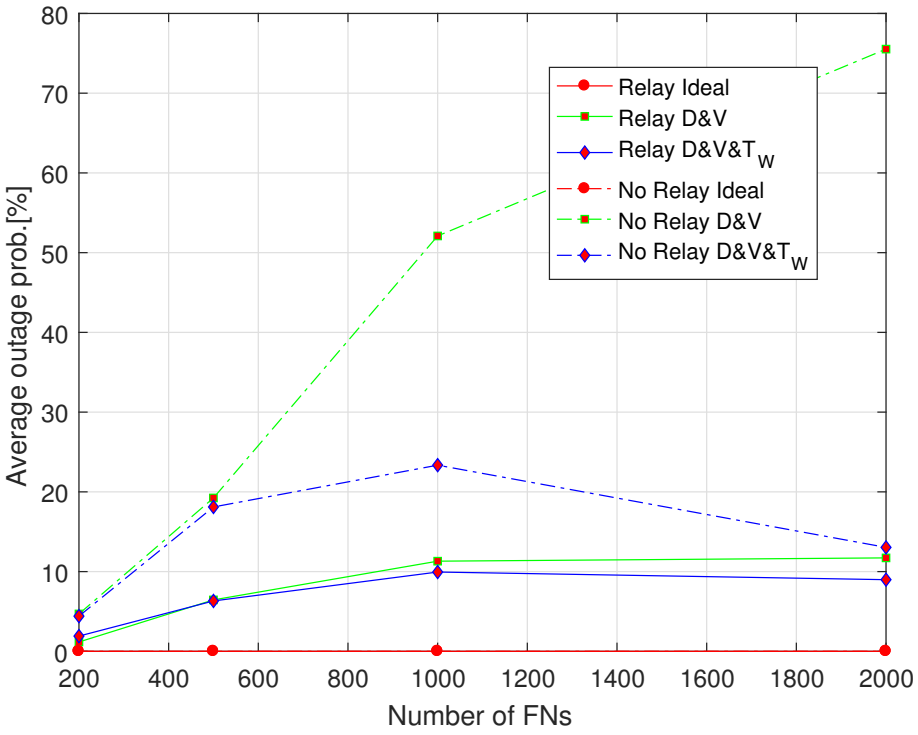


Fig. 4.5 Outage Probability of 5 fixed F-APs with high capacity

records to adapt its decisions over time and achieve the optimality in some sense. Using the proposed solution, the vehicles detect the optimal network even in the scenarios with unpredictable or non-stationary traffic. Our main contributions include:

- We model the offloading problem in the VEC environment in a piece-wise stationary model, which is a good approximation of network dynamicity. To the best of our knowledge, no previous work in the literature has investigated the task offloading problem in a piece-wise stationary VEC scenario;
- We develop a network selection scheme based on congestion and traffic patterns of the networks using *MAB theory*, which is a suitable mathematical framework for problems with no prior information and limited feedback. The proposed solution aims at minimizing the task waiting time;
- We propose a BS selection method based on the sojourn time of the vehicle in the selected network in conjunction with developing a relaying mechanism for offloading to minimize the packet loss. The proposed approach considers the task size and application types as random variables which contributes to the generality of the solution;

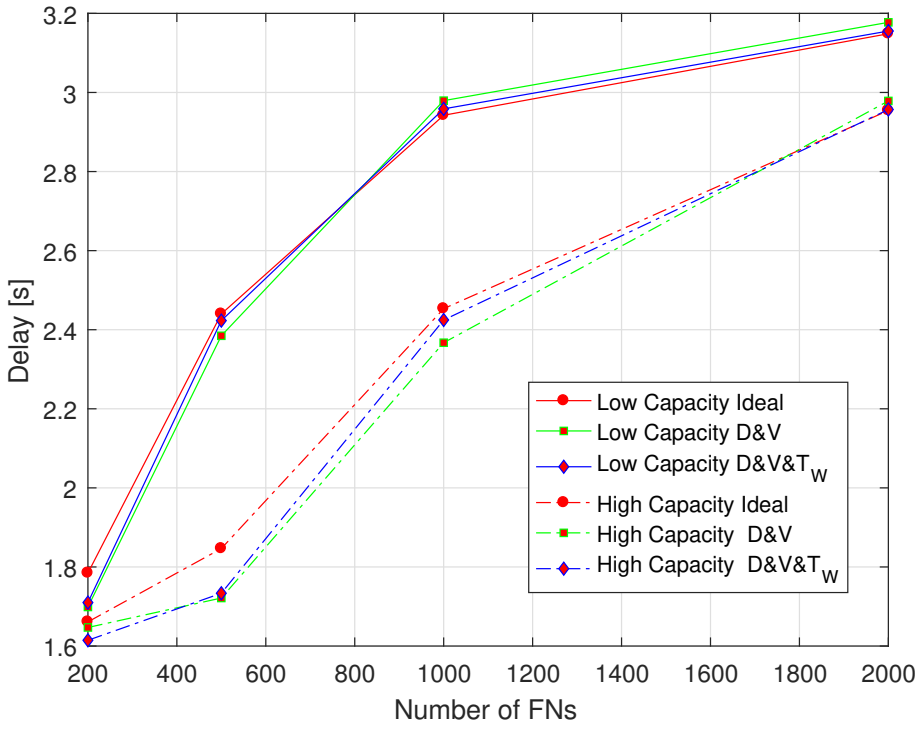


Fig. 4.6 Average task delay without relaying through 11 fixed F-APs

- We perform extensive numerical analysis to demonstrate that our proposed solution adapts to the changes in traffic load in the piece-wise stationary scenario. This leads to a small waiting time for the tasks while guaranteeing a tolerable of packet loss.

4.3.1 System Model

In this section, we first describe the network model and the offloading environment. Afterward, we explain the task computation, communications, and queue models.

System Framework

We consider a two-layer architecture for VEC. By $\mathcal{U} = \{u_1, \ldots, u_i, \ldots, u_N\}$, we represent the set of heterogeneous Edge Units (EUs) in the first layer. All the EUs have certain computational and storage capabilities. The second layer includes M types of wireless networks, each with M_j BSs [129]. $\mathcal{F} = \{F_{m_j}\}_{j=1,2,\ldots,J_m}^{m=1,\ldots,M}$ shows the set of BSs, where m is the index for the network and j is the index for the BS of that specific network. Every EU can communicate with at least one BS by means of cellular-based communication.

Each BS is equipped with a number of computational servers and therefore can process different tasks. In general, the BSs have higher computational capabilities and coverage area

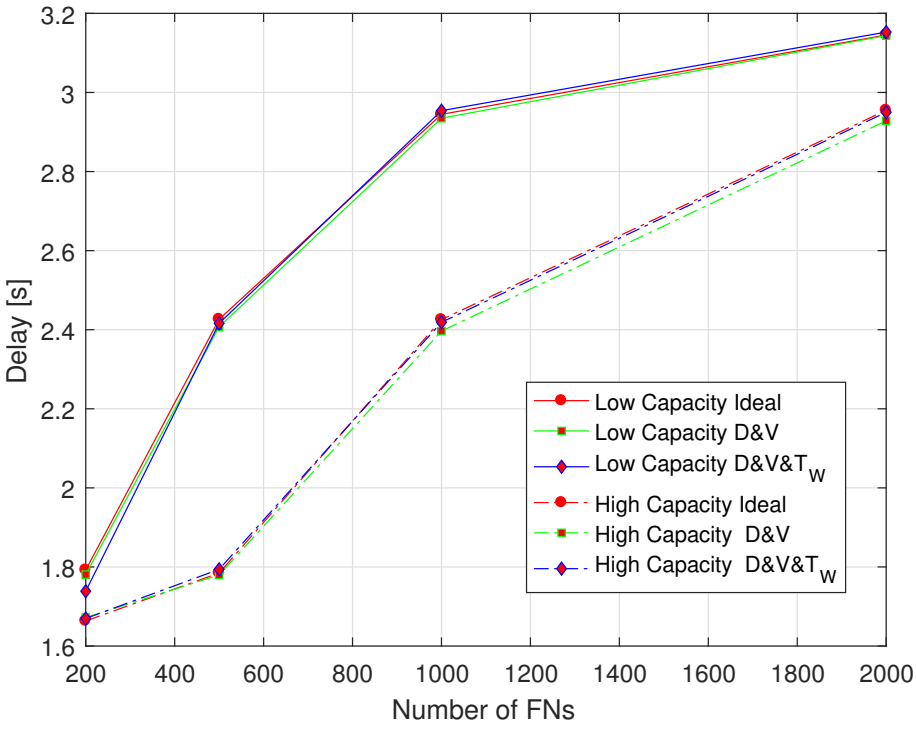


Fig. 4.7 Average task delay with relaying through 11 fixed F-APs

compare to EUs. They can aggregate the EUs' traffic requests and process the offloaded tasks. The BSs of each network type are homogeneous in the sense that they have the same computational and coverage characteristics. In contrast, the type of BSs varies from one network to another.

We consider an urban scenario as shown in Fig. 4.8, where the vehicles act as EUs. Each vehicle i moves with some velocity v_i that is selected uniformly at random. The vehicles can move from left to right or in the opposite direction, depending on their location. The speed and the direction of each vehicle remain fixed through time [126, 131]. An outage occurs if an offloading vehicle is not able to receive the result of the offloaded task due to its mobility, i.e., due to the short sojourn time.

In heterogeneous wireless networks, the networks of different sizes overlap while covering the entire area. In addition to different coverage and computational capacities, heterogeneous networks also face different traffic demands, therefore different congestion probabilities. Another important issue is the packet loss, as some times an EU is not able to receive the task result due to a shortage in the sojourn time.

Based on the discussion above, the offloading decision in dynamic vehicular environments is twofold. First, an EU selects the optimal network to offload its task, based on the probability

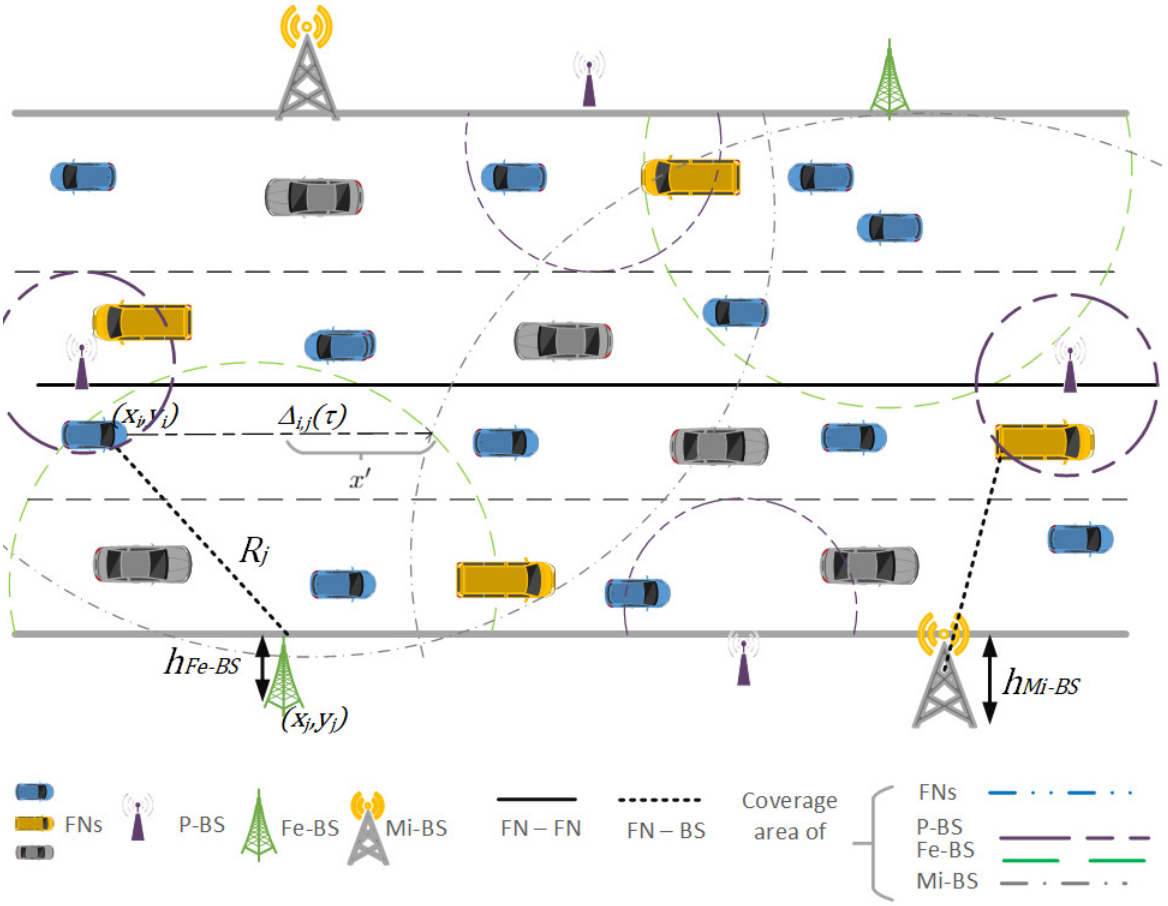


Fig. 4.8 Partial offloading mobile urban scenario.

of congestion. Afterwards, it selects a BSs in the selected network that guarantees the reception of the result of the offloaded task.The goal of the this work is to address the joint delay and outage minimization problem by a suitable selection of the network and BS for computation offloading.

Task Computation and Communication Model

The computational time for the lth task generated by the EU is defined as:

$$T_c^l(t) = \begin{cases} \frac{O \cdot \delta_l(t)}{\eta_i} & \text{local computation} \\ \frac{O \cdot \delta_l(t)}{\eta_{m_j}} & \text{otherwise} \end{cases} \tag{4.24}$$

where O represents the number of operations required for computing one byte, $\delta_l(t)$ is the task size of the lth task generated by the EU in byte, and η_* is the Floating-point Operation

Per Second (FLOPS) depending on the CPU of the device, which could be either a local processor or a processor at the m_jth BS.

In case of offloading, transmission is necessary. The transmission time of the lth task from the EU to the m_jth BS is given by

$$T^l_{tx,m_j}(t) = \frac{\delta_l}{r^{up}_{m_j}(t)} \tag{4.25}$$

where $r^{up}_{m_j}(t)$ is the up-link data rate of the link between the EU and the m_jth BS at time instant t. Later the result of the processed task should be sent back from the m_jth BS to the EU, leading to a reception time as

$$T^l_{rx,m_j}(t) = \frac{\omega_l}{r^{dl}_{m_j}(t)} \tag{4.26}$$

where $r^{dl}_{m_j}(t)$ is the down-link data rate and ω_l is the size of the result of processing.

In urban scenarios, there is often no line-of-sight due to the dense presence of buildings. Moreover, on a highway, the trucks on a communication path may introduce significant signal attenuation and packet loss [150]. Hence we assume that cellular networks have different channels so that there is no inter-cell interference, whereas the users might confront intra-cell interference due to the signal attenuation. Therefore, we make the following assumption.

Assumption 1. *A user i inside a cellular network experiences independent and identically distributed interference signals caused by other users. However, different cells use different channels so that there is no inter-cell interference.*

Focusing on a specific time instant when user i is inside the m_jth cell, we define $\mathbb{I}^{i,\iota} = 1$, where $i, \iota \in \mathcal{U}, i \neq \iota$, if the ith EU is interfering with ιth EU. Thus the interference received by the ith EU yields

$$I^i_{m_j} = \sum_{\iota} \mathbb{I}^{i,\iota} \beta_{\iota,m_j}(t) P^{m_j}_{tx} \tag{4.27}$$

where $\beta_{i,m_j}(t)$ is the path loss attenuation factor of the channel, which is a function of the distance, shown as $d_{i,m_j}(t)$, between the BS and the vehicle. Moreover, $P^{m_j}_{tx}$ is the transmission power of the m_jth BS. The distance between the ith EU and the m_jth BS is given by [134, 151]

$$d_{i,m_j}(t) = \sqrt{h^2_{m_j} + |y_{m_j} - y_i(t)|^2 + |x_{m_j} - x_i(t)|^2} \tag{4.28}$$

where $\{x_i(t), y_i(t)\}$ and $\{x_{m_j}, y_{m_j}\}$ are, the position of the ith EU at time t and the m_jth BS, respectively. Moreover, h_{m_j} is to the height of the m_jth BS.

Let B_j represent the bandwidth of the leased channel for the cellular network. By Shannon capacity formula, the transmission rate in the up-link from the ith EU to the m_jth BS at time instant t yields

$$r^{up}_{m_j}(t) = B_j \log_2 \left(1 + \frac{\beta_{i,m_j}(t) P^i_{tx}}{I^i_{m_j} + \sigma^2} \right) \tag{4.29}$$

where P^i_{tx} is the transmission power of the ith EU. Moreover, σ^2 is the noise power defined as $\sigma^2 = N_T B_j$, where N_T is the noise density. Due to the channel reciprocity, the reception rate of the results in the down-link, i.e., from the m_jth BS to the ith EU, can be similarly calculated by using the BS' transmission power. We assume that the channel is quasi-static during transmission and reception period.

Queue Model

We observe the arrival processes in networks from a bit-level perspective, and consider a Poisson-Exponential queuing model, as extensively used in the literature [152]. The server has a buffer large enough to queue all packets. The task arrival to the BSs belonging to the mth network follows a Pois(λ_m) distribution. Moreover, the service time follows Exp(μ_m) distribution. Therefore, the waiting time of the lth task offloaded to F_{m_j} is given by

$$T^l_{w_{m_j}}(\vartheta_t) = \sum_{\kappa=1, u_\kappa \in \mathcal{U}}^{\vartheta_t - 1} T_c^{l_\kappa} \tag{4.30}$$

where ϑ_t is the queue length of the m_jth BS at time instant t, and κ represent the index of the nodes in the queue. Hence, $\vartheta_t \sim \lambda_m$ and $T_c^{l_\kappa} \sim \mu_m$. As a result, (4.30) represents the task waiting time in the BS of any network. The distribution's characteristic (arrival- and departure rate) for the BSs belonging to each network is the same.

We are interested in analyzing the behavior of a single EU in a vehicular environment when it makes offloading decisions. There are mainly two phases for an offloading scheme: (i) Network selection based on congestion characteristics; and (ii) BS selection in the selected network. In the following, we describe our proposed network and BS selection procedures.

4.3.2 Network Selection

As mentioned earlier, since the BSs of each network have on average the same congestion, the offloading decision boils down to a network selection. To this aim, in this section, we first formulate the network selection problem with the aim of minimizing the waiting time (congestion). Later, based on the traffic of each network, we suggest a learning approach to select the least congested network for every offloading task.

4.3.3 Problem Formulation for Network Selection

We define $\Im_m(t)$ as the set of available networks for offloading at time instant t for the EU as

$$\Im_M(t) = \{m | \exists F_{m_j}, d_{m_j}(t) < R_m\} \quad \forall j \tag{4.31}$$

Let $\mathbb{Q}_t^m$ denote an indicator function that takes the value of 1 if EU offloads to the mth network. Over a horizon including T rounds of decision-making for offloading, our objective is to assign the tasks of the EU to a network such that the expected waiting time in the network is minimized. Ideally,

$$\mathbf{P1}: \min_{m \in \Im_M(t)} \left\{ \sum_{t=1}^{T} T^l_{w_{m_j}}(\vartheta_t) \right\} \tag{4.32}$$

subject to:

$$\mathbf{C1}: \sum_{m \in \Im_M(t)} \mathbb{Q}_t^m \leq 1 \tag{4.33}$$

Constraint (4.33) declares that each EU offloads to only one network.

The waiting time for an offloaded task depends on the number of tasks offloaded to that network. In case of the availability of information, the EU selects the BS such that $\text{argmin}_m\{T^l_{w_{m_j}}(\vartheta_t)\}$. However, the EU is not aware of the offloading decisions of other users and therefore the BSs' queue status. Therefore, in the following, we develop a learning-based solution for network selection for computation offloading in the vehicular environment.

4.3.4 Learning-based Solution for Network Selection

The waiting time of a task depends on several parameters related to either the EU or the BSs. The EU knows the tasks' parameters such as the size and the required number of process operations. However, the traffic load in each network is unknown to the EU as it depends on the cars' arrival and departure and the offloading demands. Hence, we utilize the single-player MAB model, which is suitable to solve the problems with limited information such as **P1**.

In a bandit model, an agent gambles on a machine with a finite set of arms. Upon pulling an arm, the agent receives some instantaneous reward from the reward generating process of the arm, which is a priory unknown. Since the agent does not have sufficient knowledge, at each trial it might pull some inferior arms in terms of reward which results in some instantaneous regret. By pulling arms sequentially at different rounds of the game, the

agents aims at satisfying some optimality conditions [153]. Since in this work the objective is minimizing the waiting time, we map to use the notion of *cost* instead of reward. Therefore, the goal is minimizing the cost. In brief, in our model:

- The EU and networks represent the agent and the arms;
- The instantaneous loss of pulling arm is the difference between the expected waiting time and the waiting time of the optimal arm;

At every round, the player selects an arm (a network), for offloading a packet, observes its loss and updates the estimation of its loss distribution. Each time a network is selected, the player observes the waiting time that is used for cost calculation. The objective is to minimize the loss over time. We define the instantaneous cost function for taking action m (network selection), at round t as:

$$c_t^m = \left\{ T^l_{w_{m_j}}(\vartheta_t) \right\} \cdot \mathbb{1}_{\{\mathbb{Q}_t^m = 1\}} \tag{4.34}$$

The value of offloading delay depends largely on the task queuing time; however, due to the dynamicity of a vehicular network such as vehicles' density, often no information is available about this variable. Moreover, the statistical characteristics of cars' arrival and density, also of offloaded tasks, change over time. Hence we assume that λ_m and μ_m are not identically distributed through time, however, their distribution remain identical only over a specific period of time, and changes from one period to another. Hence, the queue status of the BSs is piece-wise stationary, where the length of the period and the distribution are not known. Therefore, we introduce the following assumption.

Assumption 2. *For all BSs in network m, λ_m and μ_m are Piece-wise constant over intervals of unknown length and suffer ruptures at change points.*

Based on this assumption, BSs of the same network have the same probability distribution for the arrival and departure of the tasks, while they change in each period.

Network selection for task offloading with MAB is a stochastic problem. The previously offloaded tasks provide latency/cost information. However, this information may not be accurate due to insufficient trial of each arm in the window time period. Hence, there exists an exploration-exploitation trade-off to be addressed. One of the most influential works in the literature that considers the exploration-exploitation trade-off is Upper Confidence Bound (UCB) algorithm [154]. In UCB algorithm, at every round of the game, an index is calculated for each arm corresponding to the average gained reward of pulling the arm in all previous

rounds (the exploitation factor) and the tendency in pulling the arm for another round (the exploration factor).

When the UCB algorithm is applied to bandit problems with stationary reward distribution, the entire history of gained reward is considered for the arms' index calculation. However, in a piece-wise stationary setting, the old observations are less important [155]. Hence, to calculate the arms' indexes, the obsolete observations are penalized and only the last τ observations are considered. In our vehicular scenario, we exploit *SW-UCB* [156] algorithm that uses the last τ observations for learning.

The number of times the mth arm has been selected during a window with length τ up to round t is given by

$$C_t^m(\tau) = \sum_{s=t-\tau+1}^{t} \mathbb{1}_{\{\mathbb{Q}_t^m=1\}} \tag{4.35}$$

Let us define the total number of offloaded tasks of the EU, C_t, by round t to all networks as

$$C_t = \sum_{s=1}^{t} \sum_{F_{m_j} \in \Im_F(s)} \mathbb{1}_{\{\mathbb{Q}_s^m=1\}} \tag{4.36}$$

Inspired by the SW-UCB, we propose a learning algorithm in this work. We define the cost index of pulling arm m at round t as

$$\hat{c}_t^m(\tau) = \bar{c}_t^m(\tau) - \beta \sqrt{\frac{\xi \log(\min\{C_t, \tau\})}{C_t^m(\tau)}} \tag{4.37}$$

where the first term on the right side of the equation is the exploitation factor, the second one is the exploration factor, $0.5 < \xi < 1$, is a constant weight, β is an upper bound on exploration factor, and $\bar{c}_t^m(\tau)$ is the average accumulated cost up to round t with window length τ, defined as

$$\bar{c}_t^m(\tau) = \frac{1}{C_t^m(\tau)} \sum_{s=t-\tau+1}^{t} c_s^m \tag{4.38}$$

Each time there is a task to be offloaded, the agent pulls the arm with the minimum $\hat{c}_t^m(\tau)$.

The proposed MAB algorithm is illustrated in Algorithm 13. In lines 5-6 the agent pulls each arm once and calculates the immediate cost. In line 8-9 the cost function considering the exploitation and exploration is calculated. In lines 10-11, the best arm that maximizes the cost function is selected. Lines 12-15 update the total number of turns and selected arms and average accumulated cost.

Algorithm 13 The proposed MAB Algorithm

Input: $\xi > 0.5$, $C_0^m = 0$, $\bar{c}_0^m(\tau) = 0$, $C_0 = 0$
Output: a selected arm for each offloading task
Set the window length τ
for t=1 to T **do**
  if $\exists m$ that has not been pulled yet **then**
    Pull the arm, and update $\bar{c}_t^m(\tau)$, $C_t^m(\tau)$ and C_t
  else
    calculate the cost function $\forall m$
    $\hat{c}_t^m(\tau) = \bar{c}_t^m(\tau) - \beta\sqrt{\frac{\xi \log(\min\{C_t, \tau\})}{C_t^m(\tau)}}$
    Select the arm such that:
    $\arg\min_m \hat{c}_t^m(\tau)$
    calculate c_s^m and update:
    $C_t^m(\tau) \longleftarrow C_{t-1}^m(\tau) + 1$
    $C_t \longleftarrow C_{t-1} + 1$
    $\bar{c}_T^m(\tau) = \frac{1}{C_t^m(\tau)} \sum_{s=t-\tau+1}^{t} c_s^m$
  end if
end for

Let $L^*(t,m) = \min_m \mathbb{E}\left[c_{\psi_t}^m\right]$ represent the expected cost of offloading to the expected optimal network selected by nature during interval[3] ψ_t, and $L(t,m) = \hat{c}_t^m(\tau)$ denote the accumulated cost of offloading to the mth network selected by the proposed MAB method. We define the regret during T rounds as

$$R_T = \mathbb{E}\left[\sum_{t=1}^{T} L(t,m)\right] - \sum_{t=1}^{T} L^*(t,m) \tag{4.39}$$

which is the expected loss of the algorithm compared with the optimal network selection.

4.3.5 BS Selection

Once the least congested network is identified, one of the BSs in this network should be selected for offloading. However, as mentioned earlier, the BSs of the same network have on average the same waiting time. Moreover, small waiting time is not the sufficient condition for a BS to be the best among the available ones, mainly due to the effect of some other factors such as the sojourn time. As a result, we consider the sojourn distance in the coverage of the BS as a parameter when selecting a BS inside the previously-identified least congested network.

The sojourn distance can be calculated by the EU at any time, since it only requires the knowledge about the location of the BSs which are deployed in fix places.

[3]Interval refers to the period between two consecutive break points.

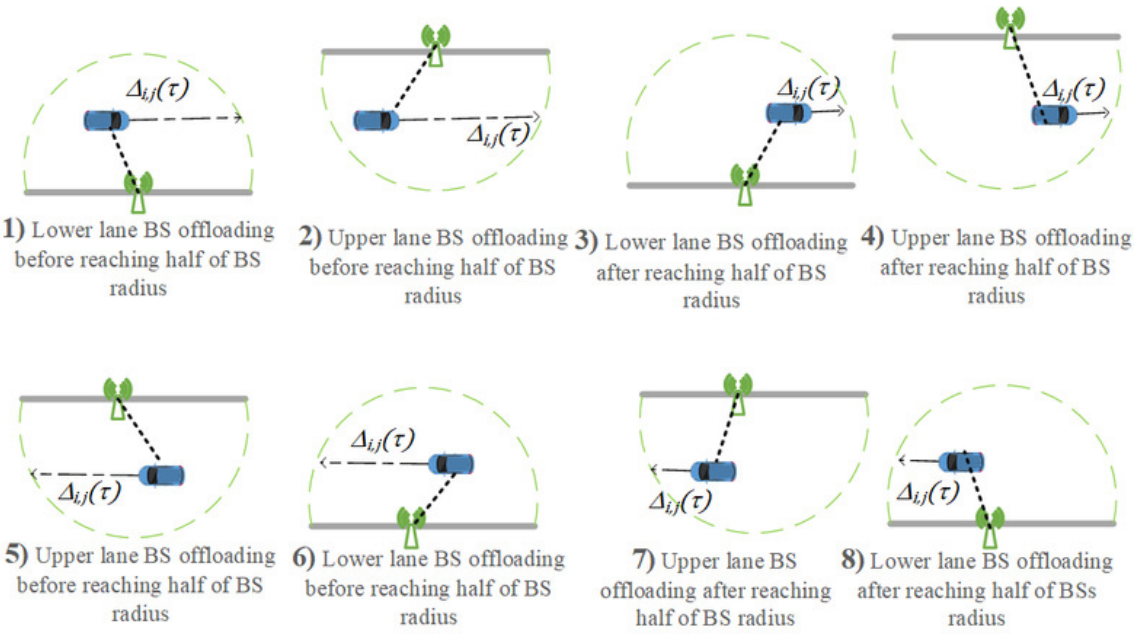


Fig. 4.9 Offloading cases considering locations of devices

As shown in Fig. 4.9, in a task offloading procedure in our framework, there can be eight offloading cases depending on the locations of the devices. Considering all offloading cases for the ith EU, the remaining distance before going out of the coverage of the jth BS at time instant t is equal to [157]:

$$\Phi_{i,m_j}(t) = \begin{cases} |x_{m_j} - x_i(t)| + x' & \text{cases 1,2,5,6 in Fig. 4.9} \\ x' - |x_{m_j} - x_i(t)| & \text{cases 3,4,7,8 in Fig. 4.9} \end{cases} \tag{4.40}$$

and

$$x' = \sqrt{R_m^2 - h_{m_j}^2 - |y_{m_j} - y_i(t)|^2} \tag{4.41}$$

where x' is a distance inside the BS coverage as depicted in Fig. 4.8.

As a first step for the BS selection procedure, the EU identifies the BSs in the selected network, as long as it is within their coverage area, as potential candidates for offloading. Hence, we define the set of BSs candidates, $\Im_F \subset \mathcal{F}$, in the selected network m that are available for the EU for computation offloading at time instant t as

$$\Im_F(t) = \left\{ F_{m_j} | d_{i,m_j}(t) < R_m, \mathbb{Q}_t^m = 1 \right\} \tag{4.42}$$

where $d_{i,m_j}$is the euclidean distance between the EU and the jth BS of the mth network as defined in (4.28), and R_m is the coverage area of every BS in network m.

Considering the sojourn distance for the BS selection phase, the best BS is selected such that:

$$\arg \max_{j \in \Im_F(t)} \left\{ \Phi_{i,m_j}(t) \right\} \tag{4.43}$$

Remark 1. *BS selection problem can also be generalized and constrained with* $\mathbf{C2}: |\Im^i_{m_j}(t)| \leq \bar{N}$ *restricting the number of EUs accessing each BS to* $\bar{N}$*, where* $\Im^i_{m_j}(t) = \{u_i | \mathbb{B}^i_{m_j} = 1\}$ *and* $\mathbb{B}^i_{m_j}$ *is an indicator function which is 1 if EU i offloads to BS* m_j*. We introduce* ρ *as an indicator function which is one if* **C2** *holds and zero otherwise. In this case, the BS can be selected as*

$$\arg \max_{j \in \Im_F(t)} \left\{ \Phi_{i,m_j}(t) \right\} \cdot \mathbb{1}_{\{\mathbb{Q}^m_i = 1\}} \tag{4.44}$$

4.3.6 Problem Formulation for Packet loss Minimization

We define the overall offloading time as

$$T^l_{off,j}(t) = \frac{\delta_l}{r^{up}_{ij}(t)} + \frac{O \cdot \delta_l(t)}{\eta_{c_j}} + T^l_{w_j}(t) + \frac{\omega_l}{r^{dl}_{ji}(t)} \tag{4.45}$$

In this stage, we aim at minimizing the outage probability of the tasks. In our work, the outage probability corresponds to the probability that an offloaded task cannot be received back by the offloading EU due to the mobility of the devices. We first introduce the amount of time that the ith EU remains under the coverage of the jth BS for avoiding to have the result back when the EU is out of coverage. Thus, the time that the ith EU remains in the coverage area of the jth BS (i.e., sojourn time) yields

$$\widetilde{T}^{i,j}(t) = \frac{\Phi_{i,j}(t)}{v_i} \tag{4.46}$$

Hence, the outage for the lth task generated at time instant t by the EU can be defined as

$$\Omega^l_j(t) = \begin{cases} 1 & \text{if } \widetilde{T}^{i,j}(t) < T^l_{off,j}(t) \\ 0 & \text{if } \widetilde{T}^{i,j}(t) \geq T^l_{off,j}(t) \end{cases} \tag{4.47}$$

that means an outage occurs when the time needed for offloading a task is higher than the EU sojourn time within the coverage area of the BS. This brings us to our second objective function defined as:

$$\mathbf{P2}: \min \sum_{t=1}^{T} \Omega^l_j(t) \tag{4.48}$$

In order to minimize the overall packet loss for the selected BS, we propose the following relaying mechanism.

4.3.7 Relaying Mechanism

Once the best BS is selected considering the sojourn distance, it might be the best among the available ones, however, the overall offloading delay still might be more than the sojourn time and as a result having a packet loss, $\Omega_j^l(t) = 1$.

To this aim, we propose a relaying mechanism in order to solve **P2**. In the relaying mechanism, once the packet is transmitted successfully to the selected BS (original BS), even if the packet can not be received by the original BS, it can be received by the destination BS where the agent is located at the time of receiving the packet. As a result, we consider the packet is relayed through wireless backhaul deployed in the infrastructure, where the BSs can communicate with each other[131]. Hence, we define the relaying offloading time as:

$$\hat{T}_{off,j}^l(t) = \frac{\delta_l}{r_{ij}^{up}(t)} + \frac{O \cdot \delta_l(t)}{\eta_{c_j}} + T_{w_j}^l(t) + T_{tx,Op}^l + \frac{\omega_l}{r_{ki}^{dl}(t)} \tag{4.49}$$

where $T_{tx,Op}^l$ is the transmission time for relaying through backhaul, and $\frac{\omega_l}{r_{ki}^{dl}(t)}$ the reception time from the destination BS. Now, we redefine the outage for relaying as:

$$\hat{\Omega}_j^l(t) = \begin{cases} 1 & \text{if } \widetilde{T}^{i,j}(t) < T_{tx,m_j}^l(t) \\ 0 & \text{if } \widetilde{T}^{i,j}(t) \geq T_{tx,m_j}^l(t) \end{cases} \tag{4.50}$$

which signifies by relaying a packet is lost only if the transmission time to the original BS is not sufficient due to the shortage of sojourn time.

4.3.8 Simulation Results

In this section, the numerical results obtained through computer simulations are presented. In the following we are comparing the performance of the proposed *SW-UCB* solution with the benchmarks presented as the following:

- Discounted UCB (*D-UCB*): D-UCB where the feedback value of each round is discounted by the factor $\gamma = 0.9$ in the UCB function;
- *UCB without SW*: Similar to *SW-UCB* in the cost calculation, however, there is no SW and the current round of the game is considered;

- ε-*greedy*: At each round of the game ε-*greedy* selects with probability 1-ε the arm having the highest empirically computed average cost, and with probability ε a randomly selected arm. To this aim, the ε has been selected equal to $1/t$;
- *Random*: At each round of the game, the algorithm selects one arm uniformly at random.

The computer simulations are performed in Matlab, where the considered parameters are listed in Table 4.2. We define a round of the game in this work as the simulation runs where all three networks are available. The simulation is performed for 40000 seconds, where only the rounds of the game are considered in the result. The simulation results aim at comparing the performance in terms of arm selection, average regret over interval, average regret over rounds, average task waiting time, and average packet loss, defined as:

- *Arm Selection*: the number of times each of the arms has been pulled in each of the intervals defined in Table 4.3;
- *Average Regret Over Interval*: the average regret in each interval calculated as defined in (4.39);
- *Average Regret Over Rounds*: the average regret over number of rounds played;
- *Average Task Waiting Time*: the average task waiting time over total number of rounds;
- *Packet Loss*: Total number of packets that have been lost for offloading in all rounds of the game.

We hypothesize an area of 1000×50 meters, with an agent located randomly in the area, while there are three networks of Micro, Femto and Pico available in the area. The Micro BSs are placed such that the whole area is covered by one of the Micro BSs. The number of Femto and Pico BSs is based on Poisson distribution where the density of Femto and Pico BSs are $\Lambda_{Mi} = 40$ and $\Lambda_{Fe} = 60$ respectively. The Femto BSs are placed in the upper and lower side of the road and the Pico BSs in the upper, lower and middle of the road both considering uniform distribution for the placement. Having placed the BSs as explained, the agent can be always served by at least one network, and in the presence of all three networks the Bandit functions.

The traffic generated in each network in different intervals is presented in Table 4.3. We have considered that in each round of the game there is a task generated by the agent. We have considered two applications: a processing application generating tasks requiring a higher number of processing operations (e.g., image processing), and a collecting application,

Table 4.2 Simulation Parameters for the Bandit Approach

Parameter	**Value**
Dimension	1000×50
Height of Mi-BS (h_{Mi-BS})	15
Height of Fe-BS (h_{Fe-BS})	10
Height of P-BS (h_{P-BS})	8
Path loss attenuation factor β_{ij}	$140.7 + 36.7\log_{10}(\frac{d_{i,j}}{1000})$[76]
Bandwidth (B_{ji})	10 MHz [138]
Noise power (σ^2)	$10^{-13}B_{ji}$ [138]
Mi-BS coverage range (R_i)	50
Fe-BS coverage range (R_m)	25
P-BS coverage range (R_k)	15
EU computational power (η_{c_i})	15 GHz
Mi computational power ($\eta_{c_{F-Mi}}$)	$4 * \eta_{c_i}$
Fe computational power ($\eta_{c_{F-Fe}}$)	$3 * \eta_{c_i}$
P computational power ($\eta_{c_{F-P}}$)	$2 * \eta_{c_i}$
EU Velocity (v_i)	10-20 meters/second

Table 4.3 MAB Setting

	Micro Network	**Femto Network**	**Pico Network**
Change Points	t=1 t=4000 t=10000 t=16000 t=30000 t=40000	t=1 t=4000 t=10000 t=16000 t=30000 t=40000	t=1 t=4000 t=10000 t=16000 t=30000 t=40000
cost Parameters	λ=7 , μ=0.5 λ=1 , μ=0.5 λ=10 , μ=0.5 λ=1 , μ=0.5 λ=7 , μ=0.5	λ=1 , μ=0.5 λ=10 , μ=0.5 λ=7 , μ=0.5 λ=10 , μ=0.5 λ=10 , μ=0.5	λ=10 , μ=0.5 λ=7 , μ=0.5 λ=1 , μ=0.5 λ=7 , μ=0.5 λ=1 , μ=0.5
Expected cost	$r_t^{a_b} = 3.5$ $r_t^{a_b} = 1.5$ $r_t^{a_b} = 5$ $r_t^{a_b} = 1.5$ $r_t^{a_b} = 3.5$	$r_t^{a_b} = 1.5$ $r_t^{a_b} = 5$ $r_t^{a_b} = 3.5$ $r_t^{a_b} = 5$ $r_t^{a_b} = 5$	$r_t^{a_b} = 5$ $r_t^{a_b} = 3.5$ $r_t^{a_b} = 1.5$ $r_t^{a_b} = 3.5$ $r_t^{a_b} = 1.5$

Table 4.4 Task Parameters for Bandit Approach

Task Parameter	**Value**
Task size (D_s^i)	[1 5] MB
Offloaded to downloaded portion (α)	5
Processing Application Operations	10 G FLOP per MB
Collecting Application Operations	1 G FLOP per MB

requiring a lower amount of processing operations, (e.g., sensor data analysis). In Table 4.4 the task-related parameters are reported.

4.3.9 Impact of Bandit Parameters

We first investigate the impact of some of the Bandit parameters on the performance of the proposed *SW-UCB*. One of the most important parameters in a SW algorithm is the decision on the size of the window length. Fig. 4.10 shows the impact of window length on average regret over rounds.

As seen in the figure, when τ is small and set to 10, the number of wrong arm selection increases and this is due to the fact that in order to start the game in each window length, all arms should be pulled once and this bring at least 2 trials of wrong arm which in the end increases the overall average regret over rounds. However, the agent is able to detect the expected optimal arm after each break point fast. On the other hand, when τ is set to 350, it takes longer to detect the expected optimal arm after each break point and the number of wrong arm selection in each interval is small in comparison with the total number of rounds. In the end, it can be observed, as the window length increases the variance of the changes after each break point is high and it takes a longer time to detect the expected optimal arm. To this aim, we have set $\tau = 50$ for rest of the simulation results because it has faster adaptation to the changes while maintaining low average regret.

The other bandit parameter that has been investigated is the impact of β, which is the coefficient of the exploration factor in UCB function, on the average regret as shown in Fig. 4.11. As seen in the figure, no significant difference can be observed between the values. However, the best value which is $\beta = 0.7$ has been selected for the rest of the simulation results.

Furthermore, the effect of ξ on the average regret as shown in Fig. 4.12. As seen in the figure, no significant difference can also be observed between the values. However, the value of $\xi = 0.6$ has been set for the rest of the simulations.

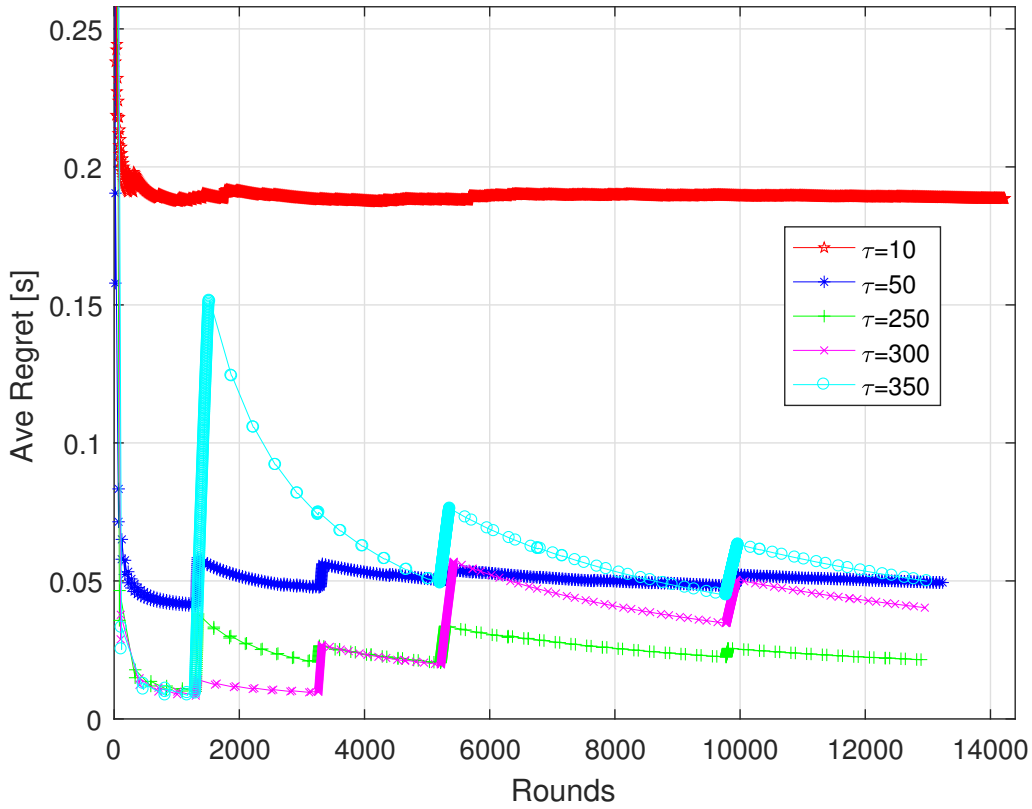


Fig. 4.10 Impact of τ on Average Regret

4.3.10 Comparison with other approaches

In order to further analyze the impact of the proposed *SW-UCB* approach we have compared its performance with the benchmarks introduced earlier. In Fig. 4.13 the performance of different approaches can be observed in terms of average regret over number of rounds. As seen the proposed *SW-UCB* approach is performing the best in comparison with other methods. Then the *D-UCB* approach performs the best by discounting the value of the feed backs received from the networks, however, it has a higher regret when considering a *SW-UCB* with $\tau = 50$. *E-greedy* and *UCB without SW* have quite the same performance and higher than the others. *Random* approach on ther other hand, selects the arms on average equally leading to a high and consistent regret.

Upon closer look at each interval, the superiority of the proposed approach can also been observed. Fig. 4.14 represents the number of networks selections by the agent in each interval in different approaches. In the first interval, three approaches of *SW-UCB*,*UCB without SW*

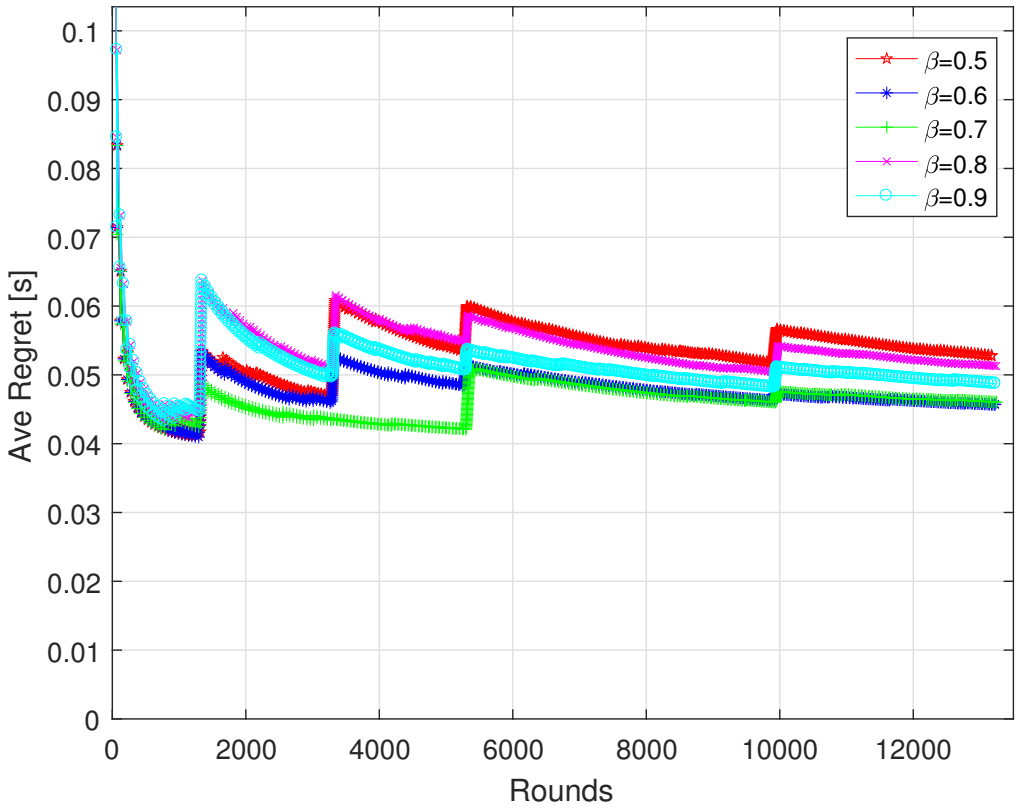


Fig. 4.11 Impact of β on Average Regret

and *E-greedy* have quite the same performance. However, after the break point when the expected optimal network changes to Micro, the *SW-UCB* is able to detect it based on what it learnt from history. On the other hand, *UCB without SW* and *E-greedy* still stick to the previous expected optimal arm and mostly pull arm 2. *E-greedy* method at the beginning in the first interval explores different arms and later only sticks to exploitation of the expected optimal arm, and third is why it mostly pulls arms 2, which is the expected optimal arm in the first interval. *UCB without SW* on the other hand, considers the UCB function however, taking an average over whole simulation run. To this reason, it takes a long time to change the selected arm and is not updated at proper time in each interval to select the best arm. *D-UCB* has the closest performance to the propose *SW-UCB*. *Random* method on ther other hand, tries the arms always quite at the same amount and does not take into account the cost distribution changes.

Fig. 4.15 depicts the result of the arms selection in the previous figure. Depending on the number of wrong arm selection and also the selected wrong arm, which can be the worst

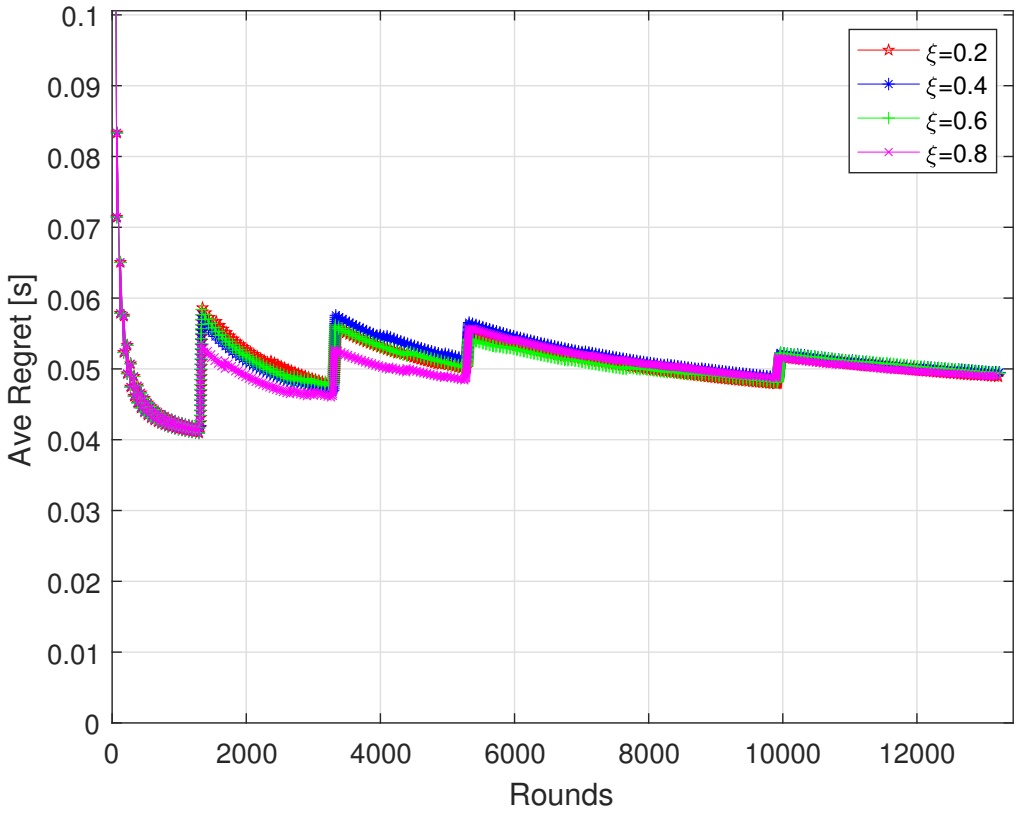


Fig. 4.12 Impact of ξ on Average Regret

among the three or the second one, the regret in each interval can be seen in Fig. 4.15. As seen average regret of the proposed *SW-UCB* approach is the smallest comparing with the other methods.

Fig. 4.16 depicts the average waiting time each packet has in the selected BS. Clearly, if a network with the lowest congestion is selected in each interval, the packets of the agent suffer lower waiting time. As seen in the figure, the proposed *SW-UCB* approach leads to the lowest waiting time for the packets, which is the results of best arm selection in different congestion patterns of the networks.

In the end, to see the impact of the relaying mechanism in the packet loss, Fig. 4.17 has been plotted. As seen, the relaying mechanism leads to less than % 10 of packet loss. This is mainly due to the small sojourn time that does not allow even for transmitting the packet which is unavoidable.

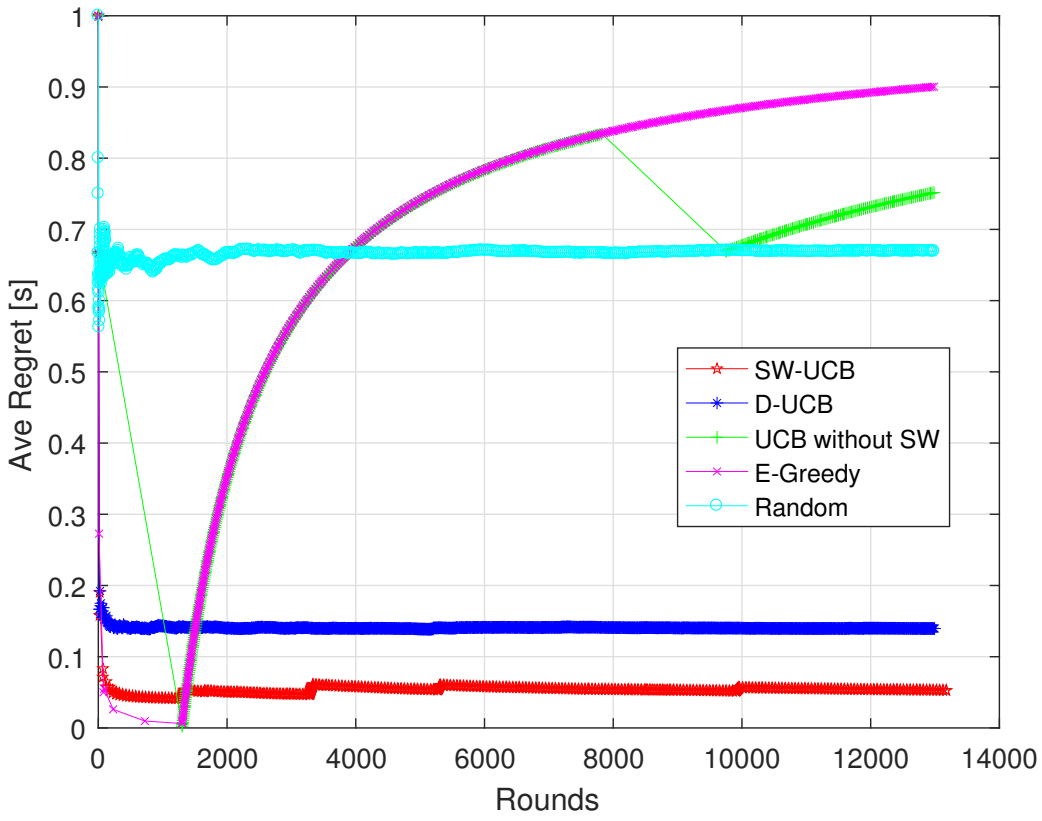


Fig. 4.13 Average regret over rounds

4.3.11 Summary

In this work, we have studied task offloading in VEC. We have modeled a vehicular environment deployed and covered by different wireless networks with different traffic patterns where the arrival and departure of the cars changes in order to depict a good approximation of the network dynamicity. We further proposed a multi-armed bandit approach to solve the problem of network selection. Later, a BS selection and a relaying mechanism through backhaul was proposed in order to reduce the task waiting time and packet loss. Through simulation results we have demonstrated the effectiveness of the proposed approach in selecting the least congested network and adapting to the changes of the network traffic.

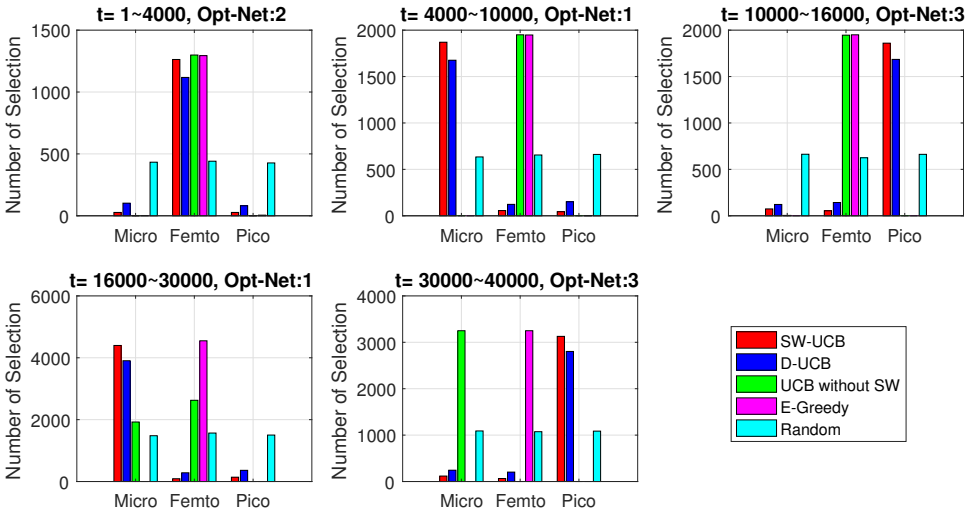


Fig. 4.14 Network Selection by agent

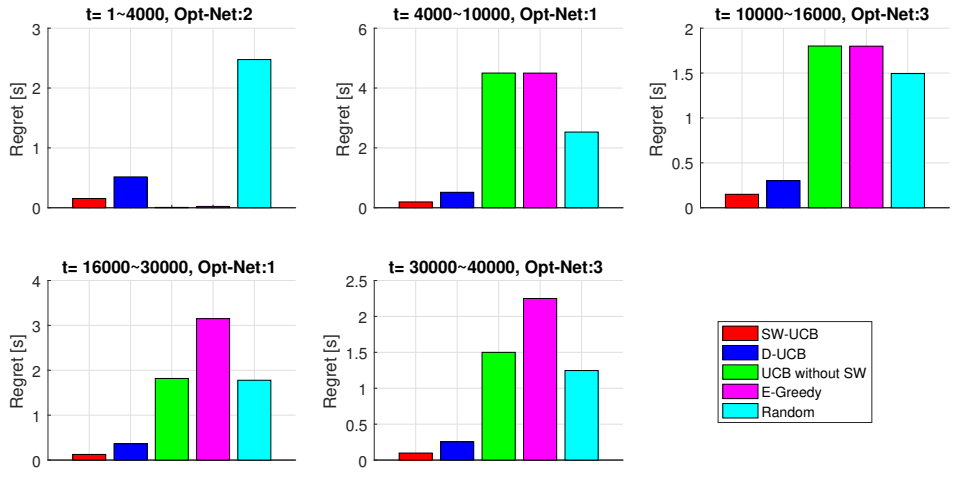


Fig. 4.15 Average regret over intervals

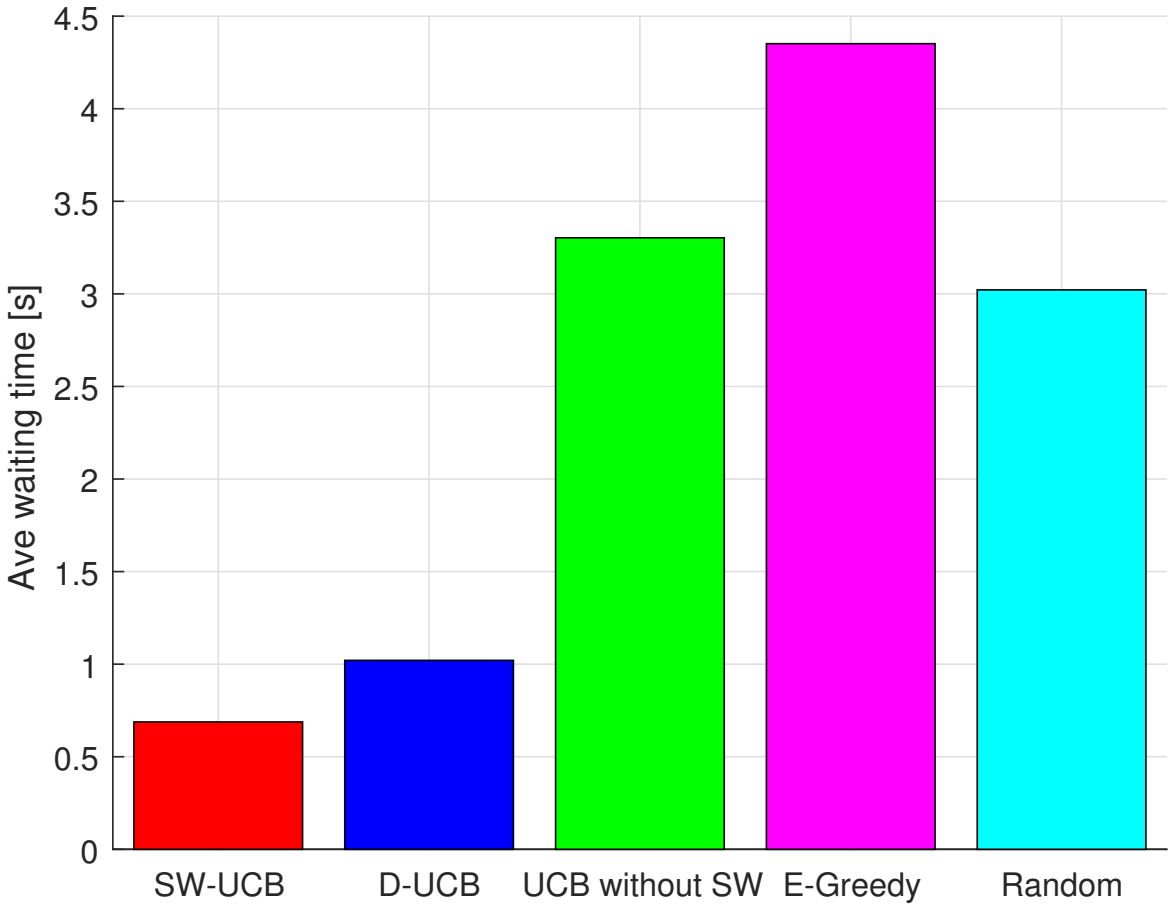


Fig. 4.16 Average packet waiting time

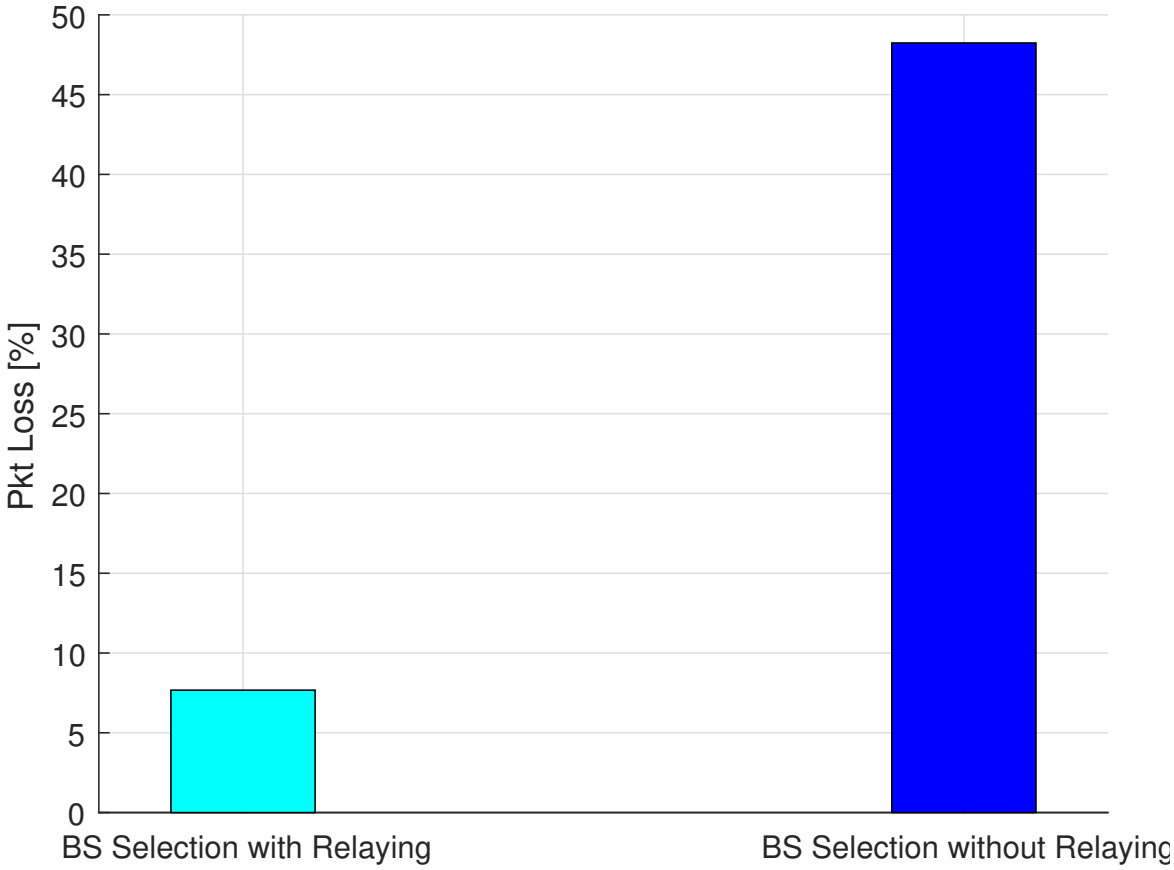


Fig. 4.17 number of lost packets

Chapter 5

Conclusion and future works

5.1 Final Conclusions

This dissertation is focused on one of the possible services in MEC and FC, which is computation offloading. In order to respond to the challenges of the future wireless networks in terms of energy consumption, latency and node/network lifetime, computation offloading at the network edge is a promising service for the end-user devices either in IoT scenarios or IoV.

In order to show my contributions in this area during my PhD, this dissertation has been designed in three main sections to propose solutions for computation offloading from different angles.

In the first part, the partial offloading in static scenarios have been analyzed. First a centralized and a distributed architecture have been proposed and the offloading portion for each generated task has been calculated considering energy consumption, latency and computational capability of the devices. Later the problem has been formulated as a CMOP considering both delay and energy and a NSGA-based method has been proposed. Extensive simulation results in both works compare the centralized and distributed architectures and the effect of genetic-based approach on reducing energy consumption, latency and increasing network lifetime. Both works can be considered as resource allocation algorithms in wireless networks.

The second part instead, is dedicated to harvesting solutions to see the impact of harvesting-enabled smart devices on the network life time. In one work, SWIPT technology has been exploited in FC to harvest energy, and the analysis is performed to see the impact of bandwidth and task size on the offloading decisions. In the other work instead, an IoT scenario is considered where the devices are able to harvest from the panels they are equipped with. This study analyzes the effect of battery capacity, number of nodes in

the network and different sunshine hour on network lifetime. We believe both works have well-contributed and can be considered in scenarios were energy management is an issue and the nodes do not have the possibility to be plugged to the electrical power.

The third part on the other hand, addresses the computation offloading problem in IoV or VEC. The first work considers two types of communications among devices: D2D and infrastructure-based communications. An information sharing approach is proposed to estimate the offloading portions. Later in the simulation, effect of server capacity and relaying on latency and outage is analyzed. In the second study, a learning-based method on the basis of bandit theory is proposed and a dynamic environment is designed to see how the vehicles are able to select the least congested network while moving in a road. The proposed ideas can both be used in IoV scenarios depending on the types of communications and availability of cellular networks and traffic patterns. In a real-world scenario were the arrival and departure of the vehicles are variant and the network traffic changes, selection of best network and BS is an important aspect in order to reduce the waiting time for task processing.

5.2 Directions for future works

There are many challenges to be addressed which have been out of the scope of this dissertation. In the following, the research directions for my future works will be briefly given.

In a computation offloading operation, there are several metrics to be considered from the users' side such meeting users delay constraint, or minimizing energy consumption, extending node/network lifetime, minimizing the packet loss, and so on. Most of the works in the literature concentrate on the user-centric objectives, however, when designing a resource allocation algorithm or making the offloading decisions, apart from the users utility, the servers and operators utility should also be considered. The operator utility can be the resources, such as spectrum, for which they have to sell the access to the BSs, and the servers utility can be the computational and storage capability such that for requesting a higher volume of processing higher costs should be paid from the users side.

To enable users to offload their tasks to BSs and to maximize their profit, the system operator can conduct an auction. The auction assigns the computing tasks from the users to an BS according to their bids, and calculates how much users pay in return for the computing services. An auction-based mechanism in one of our recent works in [6] has been proposed, however, we intend to extend this work and investigate more on this research direction.

Apart from auction-based solutions, the problems where the objectives are two-sided, can be solved with game theoretic approaches, where both users and servers' standpoints are taken into account. There are mainly two types of game theoretic approaches which are cooperative and non-cooperative depending on whether there is a competition among the individual players. We believe both auction-based and game theoretic-based approaches are promising for resources allocation problems in wireless networks.

The other point in computation offloading approaches is when number of objectives in a problem increases or when the problem is two-sided as explained before, the complexity of the network in large-scale rises sharply which might make it difficult to find optimal results in a close form solution. On the other hand, there exists a lot of randomicity in the network such as task generation, users and BS location, availability of the servers at different times, channel characteristics, velocity of the devices in mobile scenarios, queue state of the servers and so on. This dynamic property of networks make it difficult to design an optimal algorithm while considering all the possible situations. One promising methodology to address these complex networks with dynamic environment is machine learning solutions. Machine learning solutions among which I could mention Neural Network (NN), RL, Deep RL, and Bandit theory all share the same characteristics which is learning autonomously. These algorithms can be used for problems with randomicity such as wireless network where the agent which could be a user, vehicle, operator or any other element in the network learns from the history and makes the future decisions. We have already carried out two research activities in this area which are [6] and [5] proposing bandit and deep learning based methodologies to solve the resource allocation problems in network, however, I intend to study more in this interesting research direction.

The other issue in computation offloading is defining the requirements for the services that are being offloaded. Each offloading request which could be an image processing, calculating an operation or so on, needs some platform to be understood and performed, where a platform identifies a specific operating system or set of libraries able to execute a code specifically written for that specific service. MEC servers need to be implemented with the required platforms in order to perform the processing. To this aim, we have conducted a research activity on [12] where we study the placement of applications on the BSs in order to maximize the coverage area of the users. There is another ongoing work where we have been studying a heterogenous multi-service multi-platform environment with the aim of minimizing the system cost, where cost could be defined as the processing time or number of MEC servers to be implemented to cover the whole area.

Finally, in partial offloading scenarios, some portions might need to be executed only after the execution of specific other portions, which is known in the literature as task dependency

of the offloaded portions. This is indeed an interesting area that is not addressed in this dissertation and is planned to be investigated in the future works.

To wrap up, there exists many challenges to be addressed in wireless networks and more specifically on MEC and FC technologies. In this section best effort is made to briefly explain some of them to be carried out in the future.

References

[1] R. Mahmud, R. Kotagiri, and R. Buyya, "Fog computing: A taxonomy, survey and future directions," in *Internet of Everything: Algorithms, Methodologies, Technologies and Perspectives*, B. Di Martino, K.-C. Li, L. T. Yang, and A. Esposito, Eds. Singapore: Springer Singapore, 2018, pp. 103–130.

[2] G. Mois, T. Sanislav, and S. C. Folea, "A cyber-physical system for environmental monitoring," *IEEE Trans. Instrum. Meas.*, vol. 65, no. 6, pp. 1463–1471, Jun. 2016.

[3] Y. Meng, W. Zhang, H. Zhu, and X. S. Shen, "Securing consumer IoT in the smart home: Architecture, challenges, and countermeasures," *IEEE Wireless Commun.*, vol. 25, no. 6, pp. 53–59, Dec. 2018.

[4] O. Andrisano, I. Bartolini, P. Bellavista, A. Boeri, L. Bononi, A. Borghetti, A. Brath, G. E. Corazza, A. Corradi, S. de Miranda, F. Fava, L. Foschini, G. Leoni, D. Longo, M. Milano, F. Napolitano, C. A. Nucci, G. Pasolini, M. Patella, T. Salmon Cinotti, D. Tarchi, F. Ubertini, and D. Vigo, "The need of multidisciplinary approaches and engineering tools for the development and implementation of the smart city paradigm," *Proc. IEEE*, vol. 106, no. 4, pp. 738–760, Apr. 2018.

[5] F. Griffiths and M. Ooi, "The fourth industrial revolution - Industry 4.0 and IoT [Trends in Future I&M]," *IEEE Instrum. Meas. Mag.*, vol. 21, no. 6, pp. 29–43, Dec. 2018.

[6] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Energy-efficient offloading for mobile edge computing in 5g heterogeneous networks," *IEEE access*, vol. 4, pp. 5896 – 5907, Aug. 2016.

[7] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, Oct 2009.

[8] Y. Zhang, D. Niyato, and P. Wang, "Offloading in mobile cloudlet systems with intermittent connectivity," *IEEE Trans. Mobile Comput.*, vol. 14, no. 12, pp. 2516–2529, Dec. 2015.

[9] J. Gibson, R. Rondeau, D. Eveleigh, and Q. Tan, "Benefits and challenges of three cloud computing service models," in *2012 Fourth International Conference on Computational Aspects of Social Networks (CASoN)*, Nov 2012, pp. 198–205.

[10] B. Hay, K. Nance, and M. Bishop, "Storm clouds rising: Security challenges for iaas cloud computing," in *2011 44th Hawaii International Conference on System Sciences*, Jan 2011, pp. 1–7.

[11] C. Gong, J. Liu, Q. Zhang, H. Chen, and Z. Gong, "The characteristics of cloud computing," in *2010 39th International Conference on Parallel Processing Workshops*, Sep. 2010, pp. 275–279.

[12] J. Oueis, E. Calvanese Strinati, and S. Barbarossa, "Distributed mobile cloud computing: A multi-user clustering solution," in *2016 IEEE International Conference on Communications (ICC)*, Kuala Lumpur, Malaysia, May 2016.

[13] H. Yetgin, K. T. K. Cheung, M. El-Hajjar, and L. H. Hanzo, "A survey of network lifetime maximization techniques in wireless sensor networks," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 2, pp. 828–854, Second Quarter 2017.

[14] D. Mazza, D. Tarchi, and G. E. Corazza, "A partial offloading technique for wireless mobile cloud computing in smart cities," in *2014 European Conference on Networks and Communications (EuCNC)*, Bologna, Italy, Jun. 2014.

[15] B. Dai, "Prospect of 5g communication mode for energy internet," in *2018 2nd IEEE Conference on Energy Internet and Energy System Integration (EI2)*, Oct 2018, pp. 1–5.

[16] S. A. A. Shah, E. Ahmed, M. Imran, and S. Zeadally, "5g for vehicular communications," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 111–117, Jan 2018.

[17] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, "Heterogeneity in mobile cloud computing: Taxonomy and open challenges," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 369–392, First Quarter 2014.

[18] A. u. R. Khan, M. Othman, S. A. Madani, and S. U. Khan, "A survey of mobile cloud computing application models," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 393–413, First Quarter 2014.

[19] Y. Chao Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing - a key technology towards 5G," ETSI, White Paper 11, Sep. 2015. [Online]. Available: http://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp11_mec_a_key_technology_towards_5g.pdf

[20] M. Schneider, J. Rambach, and D. Stricker, "Augmented reality based on edge computing using the example of remote live support," in *2017 IEEE International Conference on Industrial Technology (ICIT)*, Mar. 2017, pp. 1277–1282.

[21] S. Kekki, W. Featherstone, Y. Fang, P. Kuure, A. Li, A. Ranjan, D. Purkayastha, F. Jiangping, D. Frydman, G. Verin, K.-W. Wen, K. Kim, R. Arora, A. Odgers, L. M. Contreras, and S. Scarpina, "MEC in 5G networks," ETSI, White Paper 28, Jun. 2018.

[22] Y. Cao, S. Chen, P. Hou, and D. Brown, "FAST: A fog computing assisted distributed analytics system to monitor fall for stroke mitigation," in *2015 IEEE International Conference on Networking, Architecture and Storage (NAS)*, Aug. 2015, pp. 2–11.

[23] V. Stantchev, A. Barnawi, S. Ghulam, J. Schubert, and G. Tamm, "Smart items, fog and cloud computing as enablers of servitization in healthcare," *Sensors & Transducers*, vol. 185, no. 2, pp. 121–128, Feb. 2015.

[24] "A new reality for oil & gas: Data management and analytics," Cisco, White Paper, Apr. 2015.

[25] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, Helsinki, Finland, Aug. 2012, pp. 13–16.

[26] D. Mazza, D. Tarchi, and G. E. Corazza, "A unified urban mobile cloud computing offloading mechanism for smart cities," *IEEE Commun. Mag.*, vol. 55, no. 3, pp. 30–37, Mar. 2017.

[27] L. Liu, Z. Chang, X. Guo, S. Mao, and T. Ristaniemi, "Multiobjective optimization for computation offloading in fog computing," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 283–294, Feb. 2018.

[28] *IEEE Standard for Adoption of OpenFog Reference Architecture for Fog Computing*, IEEE Std. 1934-2018, Jun. 2018.

[29] N. Lu, N. Cheng, N. Zhang, X. Shen, and J. W. Mark, "Connected vehicles: Solutions and challenges," *IEEE Internet of Things Journal*, vol. 1, no. 4, pp. 289–299, Aug 2014.

[30] N. Kumar, S. Zeadally, and J. J. P. C. Rodrigues, "Vehicular delay-tolerant networks for smart grid data management using mobile edge computing," *IEEE Communications Magazine*, vol. 54, no. 10, pp. 60–66, October 2016.

[31] D. Mazza, D. Tarchi, and G. E. Corazza, "A partial offloading technique for wireless mobile cloud computing in smart cities," in *2014 European Conference on Networks and Communications (EuCNC)*, June 2014, pp. 1–5.

[32] B. Han, P. Hui, V. S. A. Kumar, M. V. Marathe, J. Shao, and A. Srinivasan, "Mobile data offloading through opportunistic communications and social participation," *IEEE Transactions on Mobile Computing*, vol. 11, no. 5, pp. 821–834, May 2012.

[33] S. Kächele, C. Spann, F. J. Hauck, and J. Domaschka, "Beyond iaas and paas: An extended cloud taxonomy for computation, storage and networking," in *2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*, Dec 2013, pp. 75–82.

[34] X. Chen, J. Zhang, and S. Misra, "Socially-aware cooperative D2D and D4D communications toward fog networking," in *Fog for 5G and IoT*, M. Chiang, B. Balasubramanian, and F. Bonomi, Eds. Hoboken, NJ, USA: Wiley Telecom, 2017.

[35] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Signal Inf. Process. Netw.*, vol. 1, no. 2, pp. 89–103, Jun. 2015.

[36] M. Jo, T. Maksymyuk, B. Strykhalyuk, and C.-H. Cho, "Device to device based heterogeneous radio access network architecture for mobile cloud computing," *IEEE Trans. Wireless Commun.*, vol. 22, no. 3, pp. 50–58, Jun. 2015.

[37] M. Tao, E. Chen, H. Zhou, and W. Yu, "Content-centric sparse multicast beamforming for cache-enabled cloud RAN," *IEEE Trans. Wireless Commun.*, vol. 15, no. 9, pp. 6118–6131, Sep. 2016.

[38] M. Peng, S. Yan, K. Zhang, and C. Wang, "Fog-computing-based radio acees networks: issues and challenges," *IEEE Netw.*, vol. 30, pp. 46–53, July 2016.

[39] K. Intharawijitr, K. Lida, and H. Koga, "Analysis of fog model considering computing and communication latency in 5G cellular networks," in *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, Sydney, Australia, Mar. 2016.

[40] L. Yang, J. Cao, H. Cheng, and Y. Ji, "Multi-user computation partitioning for latency sensitive mobile cloud applications," *IEEE Trans. Comput.*, vol. 64, no. 8, pp. 2253–2266, Aug. 2014.

[41] M. Jia, J. Cao, and L. Yang, "Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing," in *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Toronto, ON, Canada, Apr. 2014, pp. 352–357.

[42] S. Chen, Y. Wang, and M. Pedram, "A semi-markovian decision process based control method for offloading tasks from mobile devices to the cloud," in *2013 IEEE Global Communications Conference (GLOBECOM)*, Atlanta, GA, USA, Dec. 2013, pp. 2885–2890.

[43] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *2016 IEEE International Symposium on Information Theory (ISIT)*, Barcelona, Spain, Jul. 2016, pp. 1451–1455.

[44] K. Kumar and Y. H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, Apr. 2010.

[45] B. Assila, A. Kobbane, A. Walid, and M. E. Koutbi, "Achieving low-energy consumption in fog computing environment: A matching game approach," in *2018 19th IEEE Mediterranean Electrotechnical Conference (MELECON)*, May 2018, pp. 213–218.

[46] Y. Yang, K. Wang, G. Zhang, X. Chen, X. Luo, and M. Zhou, "Maximal energy efficient task scheduling for homogeneous fog networks," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Apr. 2018, pp. 274–279.

[47] R. Venanzi, B. Kantarci, L. Foschini, and P. Bellavista, "MQTT-driven node discovery for integrated IoT-fog settings revisited: The impact of advertiser dynamicity," in *2018 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, Mar. 2018, pp. 31–39.

[48] W. Wang, Q. Wang, and K. Sohraby, "Multimedia sensing as a service (MSaaS): Exploring resource saving potentials of at cloud-edge IoT and fogs," *IEEE Internet Things J.*, vol. 4, no. 2, pp. 487–495, Apr. 2017.

[49] L. Chen, S. Zhou, and J. Xu, "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1619–1632, Aug. 2018.

[50] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013.

[51] C. You, K. Huang, and H. Chae, "Energy efficient mobile cloud computing powered by wireless energy transfer," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1757–1771, May 2016.

[52] W. Zhang, Y. Wen, and D. O. Wu, "Collaborative task execution in mobile cloud computing under a stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 14, no. 1, pp. 81–93, Jan. 2015.

[53] S. E. Mahmoodi, R. N. Uma, and K. P. Subbalakshmi, "Optimal joint scheduling and cloud offloading for mobile applications," *IEEE Trans. on Cloud Comput.*, 2018, early access. [Online]. Available: https://doi.org/10.1109/TCC.2016.2560808

[54] X. Cao, F. Wang, J. Xu, R. Zhang, and S. Cui, "Joint computation and communication cooperation for mobile edge computing," in *2018 16th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, Shanghai, China, May 2018.

[55] C. Xian, Y.-H. Lu, and Z. Li, "Adaptive computation offloading for energy conservation on battery-powered systems," in *2007 International Conference on Parallel and Distributed Systems*, vol. 2, Dec. 2007, pp. 1–8.

[56] K. Kumar and Y. H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, Apr. 2010.

[57] Z. Li, C. Wang, and R. Xu, "Computation offloading to save energy on handheld devices: A partition scheme," in *Proceedings of the 2001 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, ser. CASES '01. New York, NY, USA: ACM, 2001, pp. 238–246. [Online]. Available: http://doi.acm.org/10.1145/502217.502257

[58] J. Song, Y. Cui, M. Li, J. Qiu, and R. Buyya, "Energy-traffic tradeoff cooperative offloading for mobile cloud computing," in *2014 IEEE 22nd International Symposium of Quality of Service (IWQoS)*, Hong Kong, China, May 2014, pp. 284–289.

[59] W. Zhang, Y. Wen, and H. H. Chen, "Toward transcoding as a service: energy-efficient offloading policy for green mobile cloud," *IEEE Netw.*, vol. 28, no. 6, pp. 67–73, Nov 2014.

[60] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013.

[61] J. Cheng, Y. Shi, B. Bai, and W. Chen, "Computation offloading in cloud-RAN based mobile cloud computing system," in *2016 IEEE International Conference on Communications (ICC)*, Kuala Lumpur, Malaysia, May 2016.

[62] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Trans. Veh. Technol.*, vol. 66, no. 4, pp. 3435–3447, Apr. 2017.

[63] Y. D. Lin, E. T. H. Chu, Y. C. Lai, and T. J. Huang, "Time and energy aware computation offloading in handheld devices to coprocessors and clouds," *IEEE Syst. J.*, vol. 9, no. 2, pp. 393–405, Jun. 2015.

[64] Y. Kao, B. Krishnamachari, M. Ra, and F. Bai, "Hermes: Latency optimal task assignment for resource-constrained mobile computing," *IEEE Trans. Mobile Comput.*, vol. 16, no. 11, pp. 3056–3069, Nov. 2017.

[65] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.

[66] S. Hong and H. Kim, "QoE-aware computation offloading scheduling to capture energy-latency tradeoff in mobile clouds," in *2016 13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, London, UK, Jun. 2016.

[67] Z. Jiang and S. Mao, "Energy delay tradeoff in cloud offloading for multi-core mobile devices," *IEEE Access*, vol. 3, pp. 2306–2316, 2015.

[68] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3571–3584, Aug 2017.

[69] C. Wang, F. R. Yu, C. Liang, Q. Chen, and L. Tang, "Joint computation offloading and interference management in wireless cellular networks with mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 66, no. 8, pp. 7432–7445, Aug. 2017.

[70] J. Zhang, X. Hu, Z. Ning, E. C. . Ngai, L. Zhou, J. Wei, J. Cheng, and B. Hu, "Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2633–2645, Aug. 2018.

[71] J. Oueis, E. C. Strinati, and S. Barbarossa, "The fog balancing: Load distribution for small cell cloud computing," in *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, May 2015.

[72] S. Midya, A. Roy, K. Majumder, and S. Phadikar, "Multi-objective optimization technique for resource allocation and task scheduling in vehicular cloud architecture: A hybrid adaptive nature inspired approach," *Journal of Network and Computer Applications*, vol. 103, pp. 58 – 84, Feb. 2018.

[73] L. Cui, C. Xu, S. Yang, J. Z. Huang, J. Li, X. Wang, Z. Ming, and N. Lu, "Joint optimization of energy consumption and latency in mobile edge computing for internet of things," *IEEE Internet Things J.*, pp. 1–1, 2018.

[74] A. Bozorgchenani, D. Tarchi, and G. E. Corazza, "An energy-aware offloading clustering approach (EAOCA) in fog computing," in *2017 International Symposium on Wireless Communication Systems (ISWCS)*, Bologna, Italy, Aug. 2017, pp. 390–395.

[75] ——, "An energy and delay-efficient partial offloading technique for fog computing architectures," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Singapore, Dec. 2017.

[76] *Further advancements for E-UTRA physical layer aspects*, 3GPP TR 36.814, Rev. 9.0.0, Mar. 2010.

[77] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[78] J. Knowles and D. Corne, "The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation," in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99*, vol. 1, Washington, DC, USA, Jul. 1999, pp. 98–105.

[79] A. Bozorgchenani, D. Tarchi, and G. E. Corazza, "Centralized and distributed architectures for energy and delay efficient fog network based edge computing services," *IEEE Trans. Green Commun. Netw.*, vol. 3, no. 1, Mar. 2019.

[80] Y. Dodge, *The Oxford dictionary of statistical terms*. Oxford University Press on Demand, 2006.

[81] S. Cirani, L. Davoli, G. Ferrari, R. Léone, P. Medagliani, M. Picone, and L. Veltri, "A scalable and self-configuring architecture for service discovery in the internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 5, pp. 508–521, Oct 2014.

[82] K. Huang and V. K. N. Lau, "Enabling wireless power transfer in cellular networks: Architecture, modeling and deployment," *IEEE Trans. Wireless Commun.*, vol. 13, no. 2, pp. 902–912, Feb. 2014.

[83] P. Grover and A. Sahai, "Shannon meets Tesla: Wireless information and power transfer," in *2010 IEEE International Symposium on Information Theory*, Austin, TX, USA, Jun. 2010, pp. 2363–2367.

[84] X. Zhou, R. Zhang, and C. K. Ho, "Wireless information and power transfer: Architecture design and rate-energy tradeoff," *IEEE Trans. Commun.*, vol. 61, no. 11, pp. 4754–4767, Nov. 2013.

[85] R. Zhang and C. K. Ho, "MIMO broadcasting for simultaneous wireless information and power transfer," *IEEE Trans. Wireless Commun.*, vol. 12, no. 5, pp. 1989–2001, May 2013.

[86] B. Gurakan, O. Ozel, J. Yang, and S. Ulukus, "Energy cooperation in energy harvesting wireless communications," in *2012 IEEE International Symposium on Information Theory Proceedings*, Cambridge, MA, USA, Jul. 2012, pp. 965–969.

[87] K. Huang and E. Larsson, "Simultaneous information and power transfer for broadband wireless systems," *IEEE Trans. Signal Process.*, vol. 61, no. 23, pp. 5972–5986, Dec. 2013.

[88] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[89] J. Xu and S. Ren, "Online learning for offloading and autoscaling in renewable-powered mobile edge computing," in *2016 IEEE Global Communications Conference (GLOBECOM)*, Washington, DC, USA, Dec. 2016.

[90] C. You, K. Huang, and H. Chae, "Energy efficient mobile cloud computing powered by wireless energy transfer," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1757–1771, May 2016.

[91] J. Oueis, E. C. Strinati, S. Sardellitti, and S. Barbarossa, "Small cell clustering for efficient distributed fog computing: A multi-user case," in *2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall)*, Sep. 2015.

[92] M. Bouet and V. Conan, "Mobile edge computing resources optimization: A geo-clustering approach," *IEEE Transactions on Network and Service Management*, vol. 15, no. 2, pp. 787–796, June 2018.

[93] X. Shao, C. Yang, D. Chen, N. Zhao, and F. R. Yu, "Dynamic iot device clustering and energy management with hybrid noma systems," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4622–4630, Oct 2018.

[94] V. Raghunathan, A. Kansal, J. Hsu, J. Friedman, and M. Srivastava, "Design considerations for solar energy harvesting wireless embedded systems," in *Proceedings of the 4th international symposium on Information processing in sensor networks*. IEEE Press, 2005, p. 64.

[95] W. K. Seah, Z. A. Eu, and H.-P. Tan, "Wireless sensor networks powered by ambient energy harvesting (wsn-heap)-survey and challenges," in *2009 1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology*. Ieee, 2009, pp. 1–5.

[96] J. Xu, L. Chen, and S. Ren, "Online learning for offloading and autoscaling in energy harvesting mobile edge computing," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 3, pp. 361–373, 2017.

[97] C. Alippi, R. Camplani, C. Galperti, A. Marullo, and M. Roveri, "A high-frequency sampling monitoring system for environmental and structural applications," *ACM Transactions on Sensor Networks (TOSN)*, vol. 9, no. 4, p. 41, 2013.

[98] C. Alippi, R. Camplani, C. Galperti, and M. Roveri, "A robust, adaptive, solar-powered wsn framework for aquatic environmental monitoring," *IEEE Sensors Journal*, vol. 11, no. 1, pp. 45–55, 2011.

[99] W. Hu, N. Bulusu, C. T. Chou, S. Jha, A. Taylor, and V. N. Tran, "Design and evaluation of a hybrid sensor network for cane toad monitoring," *ACM Transactions on Sensor Networks (TOSN)*, vol. 5, no. 1, p. 4, 2009.

[100] R. Pinciroli, M. Gribaudo, M. Roveri, and G. Serazzi, "Capacity planning of fog computing infrastructures for smart monitoring," in *Workshop on New Frontiers in Quantitative Methods in Informatics*. Springer, 2017, pp. 72–81.

[101] Y. Nan, W. Li, W. Bao, F. C. Delicato, P. F. Pires, Y. Dou, and A. Y. Zomaya, "Adaptive energy-aware computation offloading for cloud of things systems," *IEEE Access*, vol. 5, pp. 23 947–23 957, 2017.

[102] M. Xia and S. Aissa, "On the efficiency of far-field wireless power transfer," *IEEE Trans. Signal Process.*, vol. 63, no. 11, pp. 2835–2847, Jun. 2015.

[103] L. R. Varshney, "Transporting information and energy simultaneously," in *2008 IEEE International Symposium on Information Theory*, Toronto, ON, Canada, Jul. 2008, pp. 1612–1616.

[104] R. Ishikawa and K. Honjo, "High-efficiency DC-to-RF/RF-to-DC interconversion switching module at C-band," in *2015 European Microwave Conference (EuMC)*, Paris, France, Sep. 2015, pp. 295–298.

[105] T. D. P. Perera, D. N. K. Jayakody, S. K. Sharma, S. Chatzinotas, and J. Li, "Simultaneous wireless information and power transfer (SWIPT): Recent advances and future challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 264–302, First Quarter 2018.

[106] F. K. Ojo and M. F. Mohd Salleh, "Throughput analysis of a hybridized power-time splitting based relaying protocol for wireless information and power transfer in cooperative networks," *IEEE Access*, vol. 6, pp. 24 137–24 147, 2018.

[107] *IEEE Standard for Safety Levels with Respect to Human Exposure to Radio Frequency Electromagnetic Fields, 3 kHz to 300 GHz*, IEEE Std. C95.1-2005, Apr. 2006.

[108] Atlante italiano della radiazione solare. ENEA. [Online]. Available: http://www.solaritaly.enea.it/CalcComune/Definizioni.php

[109] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2322–2358, Fourthquarter 2017.

[110] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1657–1681, thirdquarter 2017.

[111] N. Lu, N. Cheng, N. Zhang, X. Shen, and J. W. Mark, "Connected vehicles: Solutions and challenges," *IEEE Internet of Things Journal*, vol. 1, no. 4, pp. 289–299, Aug 2014.

[112] S. Bitam, A. Mellouk, and S. Zeadally, "Vanet-cloud: a generic cloud computing model for vehicular ad hoc networks," *IEEE Wireless Communications*, vol. 22, no. 1, pp. 96–102, February 2015.

[113] S. Abdelhamid, H. S. Hassanein, and G. Takahara, "Vehicle as a resource (vaar)," *IEEE Network*, vol. 29, no. 1, pp. 12–17, Jan 2015.

[114] W. Zhang, Y. Wen, and D. O. Wu, "Collaborative task execution in mobile cloud computing under a stochastic wireless channel," *IEEE Transactions on Wireless Communications*, vol. 14, no. 1, pp. 81–93, Jan 2015.

[115] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 4, pp. 3435–3447, April 2017.

[116] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 16, no. 8, pp. 4924–4938, Aug 2017.

[117] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 3860–3873, June 2016.

[118] R. Yu, Y. Zhang, S. Gjessing, W. Xia, and K. Yang, "Toward cloud-based vehicular networks with efficient resource management," *IEEE Netw.*, vol. 27, no. 5, pp. 48–55, Sep. 2013.

[119] Y. Shi, S. Chen, and X. Xu, "MAGA: A mobility-aware computation offloading decision for distributed mobile cloud computing," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 164–174, Feb 2018.

[120] F. Malandrino, C. Casetti, C. Chiasserini, and M. Fiore, "Optimal content downloading in vehicular networks," *IEEE Transactions on Mobile Computing*, vol. 12, no. 7, pp. 1377–1391, July 2013.

[121] Haibo Zhou, Bo Liu, T. H. Luan, Fen Hou, Lin Gui, Ying Li, and X. Shen, "Throughput evaluation for cooperative drive-thru internet using microscopic mobility model," in *2013 IEEE Global Communications Conference (GLOBECOM)*, Dec 2013, pp. 371–376.

[122] J. George and S. Sebastian, "Cooperative caching strategy for video streaming in mobile networks," in *2016 International Conference on Emerging Technological Trends (ICETT)*, Oct 2016, pp. 1–7.

[123] J. Thota, B. Bulut, A. Doufexi, S. Armour, and A. R. Nix, "Performance evaluation of multicast video distribution using lte-a in vehicular environments," in *2016 IEEE 84th Vehicular Technology Conference (VTC-Fall)*, Sep. 2016, pp. 1–5.

[124] Z. Lu, X. Sun, and T. La Porta, "Cooperative data offload in opportunistic networks: From mobile devices to infrastructure," *IEEE/ACM Transactions on Networking*, vol. 25, no. 6, pp. 3382–3395, Dec 2017.

[125] K. Ota, M. Dong, S. Chang, and H. Zhu, "Mmcd: Cooperative downloading for highway vanets," *IEEE Transactions on Emerging Topics in Computing*, vol. 3, no. 1, pp. 34–43, March 2015.

[126] Y. Sun, L. Xu, Y. Tang, and W. Zhuang, "Traffic offloading for online video service in vehicular networks: A cooperative approach," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 8, pp. 7630–7642, Aug 2018.

[127] J. Du, L. Zhao, J. Feng, and X. Chu, "Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee," *IEEE Transactions on Communications*, vol. 66, no. 4, pp. 1594–1608, April 2018.

[128] J. Du, L. Zhao, X. Chu, F. R. Yu, J. Feng, and C. I, "Enabling low-latency applications in lte-a based mixed fog/cloud computing systems," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1757–1771, Feb 2019.

[129] J. Wang, J. Peng, Y. Wei, D. Liu, and J. Fu, "Adaptive application offloading decision and transmission scheduling for mobile cloud computing," in *2016 IEEE International Conference on Communications (ICC)*, May 2016, pp. 1–7.

[130] D. Wang, Z. Liu, X. Wang, and Y. Lan, "Mobility-aware task offloading and migration schemes in fog computing networks," *IEEE Access*, pp. 1–1, 2019.

[131] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. ZHANG, "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading," *IEEE Vehicular Technology Magazine*, vol. 12, no. 2, pp. 36–44, June 2017.

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

[133] J. Du, F. R. Yu, X. Chu, J. Feng, and G. Lu, "Computation offloading and resource allocation in vehicular networks based on dual-side cost minimization," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1079–1092, Feb 2019.

[134] H. Wang, X. Li, H. Ji, and H. Zhang, "Federated offloading scheme to minimize latency in mec-enabled vehicular networks," in *2018 IEEE Globecom Workshops (GC Wkshps)*, Dec 2018, pp. 1–6.

[135] K. Zheng, F. Liu, Q. Zheng, W. Xiang, and W. Wang, "A graph-based cooperative scheduling scheme for vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 4, pp. 1450–1458, May 2013.

[136] L. Quan, Z. Wang, and F. Ren, "A novel two-layered reinforcement learning for task offloading with tradeoff between physical machine utilization rate and delay," *Future Internet*, vol. 10, no. 7, 2018.

[137] M. L, D. NN, and P. M., "Real-time task assignment approach leveraging reinforcement learning with evolution strategies for long-term latency minimization in fog computing," *Sensors*, vol. 18, Aug 2018.

[138] Y. Sun, X. Guo, J. Song, S. Zhou, Z. Jiang, X. Liu, and Z. Niu, "Adaptive learning-based task offloading for vehicular edge computing systems," *IEEE Transactions on Vehicular Technology*, 2019.

[139] H. Wu, X. Guo, and X. Liu, "Adaptive exploration-exploitation tradeoff for opportunistic bandits," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmässan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 5306–5314. [Online]. Available: http://proceedings.mlr.press/v80/wu18b.html

[140] Y. Zhou, C. Shen, and M. van der Schaar, "A non-stationary online learning approach to mobility management," *IEEE Transactions on Wireless Communications*, vol. 18, no. 2, pp. 1434–1446, Feb 2019.

[141] S. Abolfazli, Z. Sanaei, E. Ahmed, A. Gani, and R. Buyya, "Cloud-based augmentation for mobile devices: Motivation, taxonomies, and open challenges," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 337–368, First Quarter 2014.

[142] S. M. Oteafy and H. S. Hassanein, "IoT in the fog: A roadmap for data-centric IoT development," *IEEE Commun. Mag.*, vol. 56, no. 3, pp. 157–163, mar 2018.

[143] W. Zhang, Z. Zhang, and H. C. Chao, "Cooperative fog computing for dealing with big data in the internet of vehicles: Architecture and hierarchical resource management," *IEEE Commun. Mag.*, vol. 55, no. 12, pp. 60–67, Dec. 2017.

[144] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.

[145] S. Yan, M. Peng, and W. Wang, "User access mode selection in fog computing based radio access networks," in *IEEE ICC 2016*, May 2016.

[146] A. Bozorgchenani, D. Tarchi, and G. E. Corazza, "A control and data plane split approach for partial offloading in mobile fog networks," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, Barcelona, Spain, Apr. 2018.

[147] D. Huang and H. Wu, *Mobile Cloud Computing: Foundations and Service Models*. Cambridge, MA, USA: Morgan Kaufman - Elsevier, 2018.

[148] *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Base Station (BS) radio transmission and reception*, 3GPP TS 36.104.

[149] O. Boxma and B. Zwart, "Tails in scheduling," *SIGMETRICS Perform. Eval. Rev.*, vol. 34, no. 4, pp. 13–20, Mar. 2007.

[150] M. Boban, T. T. V. Vinhoza, M. Ferreira, J. Barros, and O. K. Tonguz, "Impact of vehicles as obstacles in vehicular ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 1, pp. 15–28, January 2011.

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

[152] X. Peng, B. Bai, G. Zhang, Y. Lan, H. Qi, and D. Towsley, "Bit-level power-law queueing theory with applications in lte networks," in *2018 IEEE Global Communications Conference (GLOBECOM)*, Dec 2018, pp. 1–6.

[153] S. Maghsudi and D. Niyato, "On power-efficient planning in dynamic small cell networks," *IEEE Wireless Communications Letters*, vol. 7, no. 3, pp. 304–307, June 2018.

[154] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2, pp. 235–256, May 2002.

[155] C. Hartland, S. Gelly, N. Baskiotis, O. Teytaud, and M. Sebag, "Multi-armed Bandit, Dynamic Environments and Meta-Bandits," Nov. 2006, working paper or preprint. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00113668

[156] A. Garivier and E. Moulines, "On upper-confidence bound policies for switching bandit problems," in *Algorithmic Learning Theory*, J. Kivinen, C. Szepesvári, E. Ukkonen, and T. Zeugmann, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 174–188.

[157] A. Bozorgchenani, D. Tarchi, and G. E. Corazza, "Mobile edge computing partial offloading techniques for mobile urban scenarios," in *2018 IEEE Global Communications Conference (GLOBECOM)*, Dec 2018, pp. 1–6.