UPC - Universitat Politècnica de Catalunya

Advanced Network Architectures Lab (CRAAX)

Department of computer architecture

PhD Thesis

presented in fulfilment of the requirements for the degree of

Doctor

of the Universitat Politècnica de Catalunya

PhD Student

Zeineb REJIBA

Mobility-aware mechanisms for fog node discovery and selection

---

Advisor: Xavier Masip-Bruin

Co-advisor: Eva Marin-Tordera

Acknowledgments

First of all, I would like to address my gratitude to Prof. Xavi Masip for offering me this PhD opportunity and accepting me within his research group. I would also like to thank him for having supervised my work during this PhD. His continuous help and guidance are greatly appreciated. I would also like to thank my co-advisor Prof. Eva Marín for her precious advice, feedback and support along the realization of my thesis.

Special thanks to my colleagues at CRAAX with whom I shared my PhD experience and with whom I had fruitful discussions during the last years.

I am also grateful to my friends, who have provided me with ongoing support, inspiration and encouragements throughout this journey.

Finally, it goes without saying that I express my deepest gratitude to my family; my parents, my brother and my grandfather for encouraging me and unconditionally supporting my endeavors.

Abstract

The recent development of delay-sensitive applications has led to the emergence of the *fog computing* paradigm. Within this paradigm, computation nodes present at the edge of the network can act as "fog nodes"(FNs) capable of processing users' tasks, thus resulting in latency reductions compared to the existing cloud-based execution model.

In order to realize the full potential of fog computing, new research questions have arised, mainly due to the dynamic and heterogeneous fog computing context. This thesis focuses on the following questions in particular: How can a user detect the presence of a nearby FN? How should a user on the move adapt its FN discovery strategy, according to its changing context? How should an FN be selected , in the case of user mobility and FN mobility? These questions will be addressed throughout the different contributions of this thesis.

The first contribution consists in proposing a discovery solution allowing a user to become aware of the existence of a nearby FN. Using our solution, the FN advertizes its presence using custom Wi-Fi beacons, which will be detected by the user via a scan process. An implementation of this approach has been developed and its evaluation results have shown that it results in a non-negligible energy consumption given its use of Wi-Fi.

This has led to our second contribution, which aims at improving the Wi-Fi scan performed in our discovery approach, especially in the case of user mobility. At a first stage, this improvement consisted in embedding information about the topology of the FNs in the beacons the user receives from previous FNs. We have shown that by adapting the scan behavior based on this information, considerable energy savings can be achieved, while guaranteeing a high discovery rate. However, as this approach is associated with a restrictive FN topology structure, we proposed a different alternative, at a second stage. This alternative leverages the history of cellular context information as an indicator allowing the user to infer whether an FN may be present in its current location. If so, the scan will be enabled. Otherwise, it is disabled. The simulation results comparing different classification algorithms have shown that a sequence-based model, such as a hidden-Markov model is able to effectively predict the FN presence in the current user location.

While the previous approaches have focused on a sparse FN deployment, our third contribution considers a high density of FNs. Consequently, as there are multiple nearby FNs that can process the user's tasks, it is important to derive a suitable FN selection strategy. This strategy should consider the time-varying set of FNs caused by the user's mobility. Besides, it should minimize the number of switches from one FN to another, in order to maintain a good quality of service. With these considerations in mind, we have shown that an adaptive greedy approach, that selects an FN having a good-enough delay estimate, achieves the best results.

Finally, unlike the previous contribution, where the focus has been on FN selection

when the user is mobile, our final contribution deals with mobile vehicular FNs (VFNs). Given the mobility of such VFNs, it is important to make the most of their resources, since they are only available for a short time at a given area. So, we propose that, in order to select an appropriate VFN for a given task, a reference roadside unit (RSU) responsible for task assignment can use advice from a neighbor RSU. This advice consists in the VFN that will result in the lowest delay for the current task, based on the experience of the neighbor RSU. The results have shown that, using the provided advice, the reference RSU can observe significant delay reductions.

All in all, the proposed contributions have addressed various problems that may arise in a fog computing context and the obtained results can be used to guide the development of the building blocks of future fog computing solutions.

Contents

List of Acronyms

AGFS Adaptive Greedy FN Selection

ANDSF Access Network Discovery and Selection Function

AP Access Point

API Application Programming Interface

BAS Block Allocation Scheme

BDSS Beacon-assisted Direction-aware Scan Scheme

BFS Block-based FN selection

BS Base Station

BSSID Basic Service Set Identifier

DHT Distributed Hash Table

DNS Domain Name System

DT Decision Tree

EN Edge node

FN Fog node

GA

GSM Global System for Mobile Communications

HMM Hidden Markov Model

HTTP Hypertext Transfer Protocol

IT Information Technology

IoT Internet of Things

KNN K-Nearest Neighbors

LAN Local Area Network

LimExp Limited Exploration

LoRa Long Range

MAB Multi-Armed Bandits

MANET Mobile Adhoc Network

MASN Mobile Adhoc Social Network

MDC Micro Datacenter

MDP Markov Decision Process

MEC Multi-access Edge Computing

ML Machine Learning

MS Mobile Station

NSD Network Service Discovery

OFRA OpenFog Reference Architecture

OS Operating System

OSI Open Systems Interconnection

OUI Organizationally-Unique Identifier

POI Point of Interest

RFC Request for comment

RL Reinforcement Learning

RSU Roadside Unit

RU Resource Unit

SARSA State-Action-Reward-State-Action

SD Service Discovery

SDK Software Development Kit

SLA Service Level Agreement

SMDP Semi-Markov Decision Process

SSID Service Set Identifier

SeV Serving Vehicle

TLV Type Length Value

TaV Task Vehicle

UCB Upper Confidence Bound

UE User Equipment

UFC User-Fog Contact

UI User Interface

VANET Vehicular Adhoc Networks

VC Vehicular Cloud

VFN Vehicular Fog Node

VM Virtual Machine

VMM Virtual Machine Migration

VSIE Vendor-Specific Information Element

VUCB Volatile Upper Confidence Bound

WLAN Wireless Local Area Network

WSN Wireless Sensor Network

ZeroConf Zero Configuratio Networking

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context and motivation

The past few years have witnessed the emergence of the Internet of Things (IoT) paradigm, where different kinds of sensor-enabled devices are enhanced with networking capabilities based on different wireless technologies such as Wi-Fi, LoRa, Bluetooth, etc... This not only allows them to communicate with each other but also gives them access to Internet. Consequently, a wide set of applications can be enabled, ranging from smart healthcare services to smart cities and smart manufacturing. These applications are based on the processing and analysis of continuous streams of data, thus requiring considerable computing resources. However, given the inherent nature of IoT devices, which are generally lightweight and resource-constrained, they cannot handle such compute-intensive tasks. This has led to the emergence of efforts to offload those tasks to the cloud [1] to leverage an almost unlimited pool of available IT resources.

Although such a solution seemed promising at first, it was later found that it raises several issues related to the specific requirements of IoT applications. In fact, several IoT use cases require real-time response times. But, given the exponential rate at which the IoT app data is generated, the centralized cloud approach fails to deal with such a velocity in real-time [2], since requests to the remote cloud have to go through a long path from the access network all the way up to the specific cloud server where the application is hosted. This incurs considerable delays, exceeding hundreds of milliseconds, which cannot be accepted in applications such as virtual reality, gaming and robotics, which require latencies in the order of milliseconds [3]. Furthermore, the transmission of this sheer volume of data to the cloud generates a high volume of network traffic and therefore puts a lot of burden on the network infrastructure. Data locality is also another important issue. In fact, typical IoT-related requests are needed in the surroundings of the device that generated them [2] and thus processing them in a remote cloud would be wasteful and inefficient.

In order to overcome these limitations, the Fog Computing [4] paradigm has emerged with the aim of bringing resources closer to the edge of the network where end users and data sources are located. It aims to meet the latency requirements of time-sensitive applications in addition to ensuring an efficient use of the underlying network. Fog computing

is defined as a highly-virtualized, distributed environment, which is expected to work in conjunction with the cloud. It relies on the use of a heterogeneous set of computing devices, generally referred to as "fog nodes" (FNs)[1], including edge routers, access points, set-top boxes [5], road-side units, vehicles [6], or even devices contributed by end-users [7]. Fog nodes could also be envisioned as logical entities comprised of a heterogeneous set of physical devices allowing services to be executed in a distributed manner [8]. These nodes are usually organized in a hierarchical topology, where the spatial and temporal scopes of the processed data grow as we move from the edge towards the cloud. The resulting architecture is shown in Fig. 1.1.

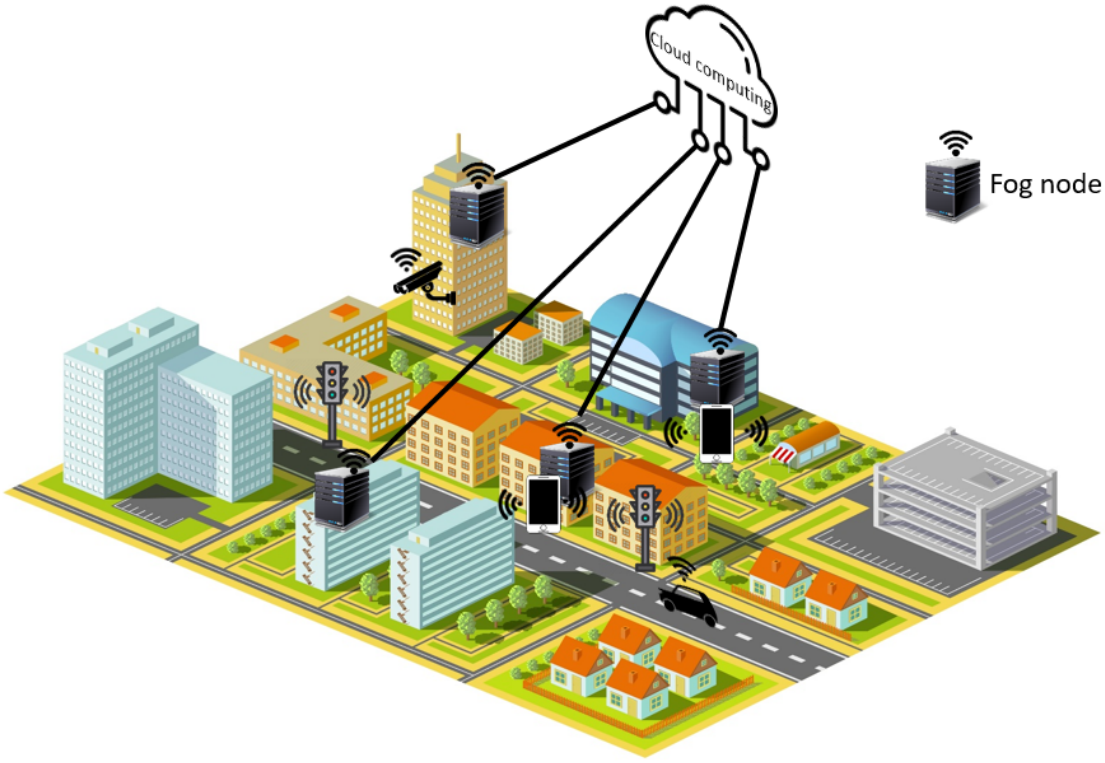


Figure 1.1: Example fog computing architecture in a smart city

Given the potential seen in such a novel computing paradigm, different research and industry initiatives have started to combine efforts in order to develop a unified fog computing architecture. The first reference architecture emerged from the OpenFog consortium (later merged with the Industrial Internet Consortium). This architecture called OpenFog Reference Architecture (OFRA) was the foundation for the IEEE Standard for Adoption of OpenFog Reference Architecture for Fog Computing[2] in 2018.

[1] Not to be confused with fogNodes™ by Nebbiolo Technologies.
[2] https://standards.ieee.org/standard/1934-2018.html

Besides fog computing, other edge-based computing paradigms can be found in the literature. Cloudlets [9], for instance, constitute one of the early efforts in this direction. A cloudlet is defined as "a trusted, resource-rich computer or cluster of computers that is well-connected to the Internet and is available for use by nearby mobile devices." It can be combined with a Wi-Fi access point into one entity that can act as a host for users' Virtual Machines (VMs). Cloudlets may also be referred to as "micro data centers", which is a term coined by Microsoft in [10]. Another initiative in this context is the Mobile Edge Computing [11] which emerged in order to provide cloud services at the edge of the cellular network. Later on, the scope of this initiative was extended to cover different radio access technologies, thus the name changed to multi-access edge computing (MEC). Other terms such as (mobile) edge clouds and mobile micro-clouds [12] have been also used to refer to MEC. A set of MEC servers can be used to deliver applications such as connected cars, augmented reality and edge video analytics.

1.2 Scope and objectives

A common challenge in these edge-based computing paradigms is the need for appropriately *discovering* and *selecting* fog nodes (alternatively cloudlets or MEC servers depending on the used terminology) so that their resources can be fully leveraged to process users' tasks. Addressing these two aspects while taking mobility into account constitute the main focus of this thesis.

More in detail, as a first step, we are interested in providing a discovery solution that allows a user to become aware of the existence of a nearby fog node, taking into account the inherent characteristics of the fog computing environment. For instance, existing discovery solutions based on querying a directory server at the cloud side would not be suitable in the fog, since they would still contribute to the overall core network traffic. Besides, solutions where the user and the FN are assumed to be pre-connected to the same wireless local area network (WLAN) do not necessarily fit in the fog context, where the user and the FN are not initially part of the same network domain.

In addition, as mobility is a characterizing feature of fog computing, a user on the move, potentially having power usage constraints, does not need to trigger discovery in areas with no fog coverage, in order to achieve power savings. On the other hand, discovery should indeed be performed in covered areas in order not to miss potential fog nodes and maximize the benefit brought up by the fog paradigm. Therefore, a user will need to acquire information that will assist it in the discovery decision-making process. Within this context, we will investigate ways to improve our discovery approach based either on previous knowledge about the FN topology or on the user's surrounding context.

Once the discovery process is done, the next step would be to select an FN from the list of the discovered FNs. The selected FN will execute the user's tasks, with the objective

of meeting some performance metric. A common metric in the fog computing context is to select the fog node that will result in a minimal latency [13]. However, the observed latency would highly depend on heterogeneous FN characteristics, potentially time-varying and initially-unknown at the user side. The FN selection task becomes more challenging when considering user mobility, which results in a dynamically-changing set of FNs, in addition to causing switching costs when the selected FN changes. These aspects will be carefully studied in this thesis.

Furthermore, since modern vehicles may be endowed with additional computation resources, such resources could be used to execute tasks on behalf of on-board users or even nearby vehicles, thus creating the concept of vehicular fog computing [6], where the vehicles could be considered as vehicular fog nodes (VFNs). In this context, the VFNs' mobility results in a highly-dynamic environment, which makes the selection of an appropriate VFN to process a task very challenging. As a result, a cooperative model involving collaboration among nearby roadside units (RSUs) for the purposes of optimizing the computation task assignment in such a dynamic environment will be investigated.

In order to address the aforementioned challenges, we define the following objectives to be achieved in this thesis:

- **O1:** Design a fog node discovery solution
  - **O1.1 :** Propose a fog node discovery solution taking into account the characteristics of the fog computing environment
  - **O1.2 :** Optimize the discovery process taking users' mobility into account
- **O2:** Propose a suitable FN selection strategy considering static fog nodes and a mobile user and taking the switching costs into account.
- **O3:** Define a vehicular fog node selection strategy considering a static RSU for performing the task assignment.

1.3 Organization of the thesis

In the following, we summarize how each of the thesis chapters contributes to the aforementioned objectives:

In Chapter 2, we present our first contribution, which consists in a Wi-Fi-based fog node discovery solution. The proposed solution relies on advertizing the FN information in a 802.11-based beacon frame to be broadcast by the FN. This allows neighboring users[3]

[3]Throughout this thesis, when the phrase "the user performs a given action" (either a scan or a selection), this means that a software component present within the user device is performing this action and not the user himself/herself.)

to perform a Wi-Fi scan specifically looking for these beacons. The example user shown at the beginning of the green area in Fig. 1.2 will be then able to detect the presence of the neighboring FN via this scan process.

In Chapter 3, we further improve this discovery solution, by putting a specific focus on the impact of user mobility on the effectiveness of the underlying scan process. In fact, instead of keeping the scan process always on, even when the user is moving in areas with no FN coverage (see the middle zone in the green area in Fig. 1.2), we propose to only enable scanning when the user is about to reach an area where an FN is present. We propose two potential options allowing the user to acquire knowledge regarding nearby FN contact opportunities. The first approach relies on embedding this information in the beacons sent by the the previously-encountered FNs, while the second approach makes use of the cellular footprint of a given location to infer the presence/absence of FNs, given historical information.

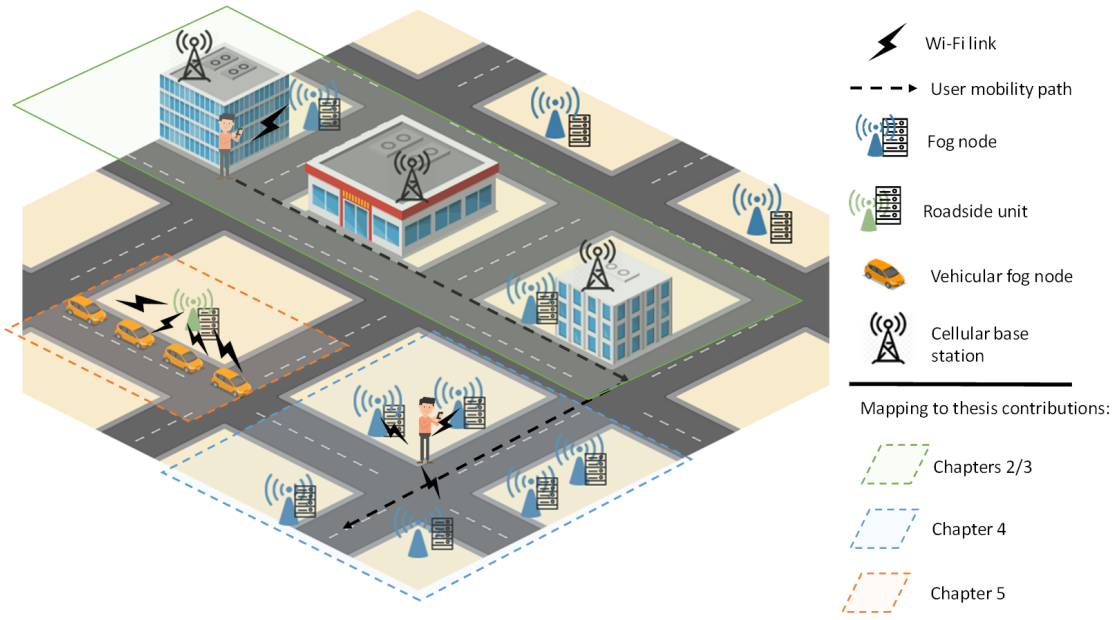


Figure 1.2: Considered scenario and its mapping to the thesis contributions

After the discovery step has been performed, Chapter 4 addresses the problem of the appropriate selection of the best FN to process a user's tasks, with a particular focus on a fog computing deployment with a high density of FNs, as shown in the blue area in Fig. 1.2. Since the user is moving in a high-density FN environment, this may trigger frequent switches from one FN to the other, therefore accumulating costs that will negatively impact the task execution performance. This aspect, along with the presence of a frequently

changing set of FNs will be carefully examined in our proposed schemes.

After considering user mobility, Chapter 5 focuses on FN mobility and how we can make a proper use of the capacities offered by mobile vehicular FNs. To this end, we present an RSU-based task assignment strategy that selects the appropriate VFN for task execution, as shown in the orange area in Fig. 1.2. This strategy will put a specific focus on the volatility characterizing the vehicular environment.

In Chapter 6, we provide an overall view on how the aforementioned contributions can be integrated.

Finally, a summary of the different thesis contributions is presented in Chapter 7, along with potential directions for further research.

Chapter 2

Wi-Fi-based fog node discovery

The discovery of resources and services is an essential building block of every computing and networking system. The emerging fog computing systems also need this component in order to make users aware of the existence of nearby fog nodes, so that they can make use of the resources they offer. This chapter addresses this specific problem. We start by providing a review of the related works addressing the discovery problem in the fog and other edge computing paradigms. Then, the motivation and the specific details of our proposed Wi-Fi-based discovery approach are presented. Finally, implementation details are shown in addition to evaluation results comparing the proposed approach to two widely-used discovery approaches.

2.1 Literature review

Resource discovery and service discovery have been thoroughly studied in a wide variety of fields including Mobile Ad Hoc Networks (MANETs)[14], Vehicular Ad Hoc Networks (VANETs)[15], Wireless Sensor Networks (WSNs)[16], the Internet of Things (IoT)[17], desktop grids[18] and inter-cloud environments[19]. Discovery approaches that have been proposed in the fog and other edge-based computing paradigms inherit several characteristics from the approaches that have been proposed in the aforementioned fields. These characteristics, summarized in Fig. 2.1, include:

- the scope of the discovery approach: it may be valid within a *local area*, or *beyond a local area*, i.e. a wider scope, either geographically or at the network-level, is considered. The latter approaches usually rely on the presence of a specific server, which may also be referred to as a *broker*, a *mediator* or a *repository*, hosting a list of the available fog nodes.

- the underlying architecture: The discovery mechanism may be based on a *client-server* architecture, where a central server hosts all the discovered resource information. In contrast, in a *peer-to-peer* setup, the discovery logic can be implemented in each of the peers forming the considered topology. A *hybrid* solution combining both is also feasible.

- the discovery message propagation method: it could be based on *unicast* from a client to a single server, *multicast*, i.e. the message is sent to a group of nodes in the network or a *broadcast* mechanism where all nodes in the network receive the discovery message.
- the OSI layer used to carry the discovery message. The approaches surveyed next either use the *link* layer, the *network* layer or the *application* layer.

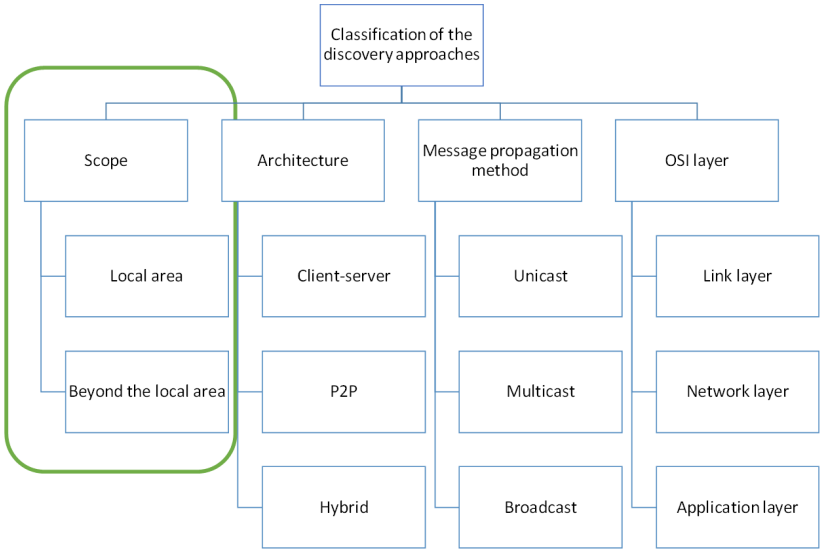


Figure 2.1: Classification of the discovery approaches

Since the surveyed discovery approaches may belong to different categories at the same time, in the following, we provide a classification based on the *scope* criterion. The description accompanying each approach also contains details about the other criteria. In addition, while the terms resource discovery and service discovery have been used interchangeably in many of the related works in the aforementioned fields, in this thesis, we will use the term *discovery* to refer to the process allowing a user, initially unaware of the presence of nearby fog nodes to become aware of such a presence. The exact amount of resources or services offered by the fog node may or may not be included as part of the FN discovery information.

2.1.1 Discovery in the local area

Among the widely-used discovery approaches that work at a local network level, we can cite approaches based on Zero Configuration Networking (Zeroconf), which has been used in [20] for cloudlet discovery. Zeroconf combines the use of DNS-Service Discovery (DNS-SD, RFC 6763) and Multicast DNS (RFC 6762), therefore allowing a client to discover the

IP address and the port number used by the cloudlet, without having an initial knowledge about this information. In [21], the IoT Hub, a Fog Node managing multiple smart objects makes itself discoverable by means of the same Multicast DNS method, using the _coap._udp.local. service type. Although this approach is supported in multiple devices through implementations such as Bonjour[1] and Avahi[2], one of its associated issues is that not every router supports IP Multicast packet routing, which would lead to missing candidate cloudlets. To address this issue, authors in [22] combine the multicast approach with a traditional cloud-based directory server hosting information about all available cloudlets. Authors in [23] highlight that Zeroconf-based mechanisms are limited in the type and size of the information returned, especially that cloudlet-ready apps require specific cloudlet metadata for a more informed decision making regarding computation offloading. As a result, when the IP address and the port of the cloudlet are retrieved via DNS-SD, an additional step is added, consisting in sending a HTTP GET request to the cloudlet to get its metadata.

Relying on a broadcast process instead, authors in [24] propose that a mobile device broadcasts a *Discovery* message in order to discover cloudlets in its wireless LAN. As a result, the available cloudlets return a *Discovery Reply* message. However, no details were provided regarding the implementation of the broadcast mechanism.

As it can be noted, the previous approaches assume the client and the cloudlet to be part of the same network, which may not always be the case in the fog computing context.

From a standardization perspective, the OpenFog Reference Architecture also envisioned a broadcast mechanism allowing a new member in a fog system to broadcast its information in order to allow other members in the fog cluster to become aware of its presence. However, no specific details were provided regarding possible options for implementing the broadcast mechanism.

Finally, in [25], the authors propose the exploitation of the built-in Bluetooth or Wi-Fi Direct interfaces present in users' devices to allow them to associate with one another into a Mobile Ad hoc Social Network (MASN). Then, in order to discover a neighboring FN, each user can ask its peers in the MASN to share information about nearby FNs.

2.1.2 Discovery beyond the local area

As introduced earlier, in this approach, a user can query a directory server for the list of FNs present within a wide scope. This process assumes that the different entities owning or managing these FNs register them at a previous step in the directory. Different works have considered different placement options for such a directory server. In fact, it may either be placed in the remote cloud, at the edge or in each level of the hierarchy.

[1] https://developer.apple.com/bonjour/
[2] https://www.avahi.org/

Among the works in the first category, we cite [26], where the authors present a framework for mobile computation offloading, called Pytos. In this framework, the list of available cloudlets in each location is stored in a repository hosted in a cloud server. As a result, whenever a client wishes to use the resources of a cloudlet, it has to query the cloud for the list of cloudlets present in its current location. In [27], the authors present a three-layered architecture, comprised of the cloud, the fog and the edge. More specifically, the resource discovery platform is implemented at the cloud layer and it contains the set of resources that can be found in the city. Such resources are reported to the cloud by a fog node managing the so-called "edge neighborhoods", which in turn are comprised of nearby edge devices. However, no details have been provided on how fog nodes are discovered. A DNS-based edge server discovery solution, termed *eDisco* has been proposed in [28]. This solution relies on the addition of a new *edge DNS SRV record* by edge server managers, therefore facilitating the lookup of the closest edge servers in the client's network path. Similarly, no implementation nor evaluation results have been provided.

As for works falling in the second category, we cite [29], where the authors propose the use of home routers as brokers to help clients discover surrogates that can perform computation tasks on their behalf. To support client and surrogate mobility, different brokers are used and interconnected into a Distributed Hash Table (DHT) structure. This enables an efficient dissemination of surrogate information updates to interested clients. Authors in [30] propose the Edge-as-a-service (EaaS) platform, which relies on the use of a master node that acts as a controller for the set of edge nodes connected to it. It is assumed that edge nodes know the IP address and port number of the master node. As a result, the controller uses this specific port to listen to service advertisements made by edge nodes and adds them to the list of available edge nodes (ENs). Kinaara is introduced in [31] as a framework allowing distributed discovery and allocation of resources at the edge. In Kinaara, edge devices are organized into "proximal clusters" managed by a mediator entity. An infrastructure-less D2D service discovery protocol such as Flashlinq[32] is envisioned to be used to allow edge devices to discover the different clusters that may be encountered during their mobility.

Finally, in the third category, we cite Foglets [33], which relies on a partitioned name server to maintain the list of fog nodes present at the different levels of the fog-cloud hierarchy. Similar to other approaches, no details have been given about how such an approach would be implemented in the real world.

2.1.3 Summary

When observing the reviewed works, the following facts can be noted:

- In most of the approaches of Section 2.1.1, users and cloudlets are assumed to be pre-associated to the same wireless LAN, which may be acceptable in the cloudlets

field, given the definition of cloudlets. However, such approaches cannot be applied in the fog computing context, since the pre-association assumption does not necessarily hold.

- In some approaches, discovery information is retrieved from a remote repository, which does not solve the latency and the network traffic reduction issues that fog computing is expected to address.
- Many proposals lack implementation details, which could limit their applicability in a real-world environment.

Therefore, with these considerations in mind, we present in the following a discovery solution that attempts to address these concerns.

2.2 Proposed Wi-Fi-based FN discovery solution

In this section, we first provide a review of the beacon stuffing concept. Then, we describe our proposed solution in detail.

2.2.1 Beacon stuffing review

Beacon stuffing [34] refers to the process of embedding customized information into 802.11 beacon frames. This custom information can be either included within the Service Set Identifier (SSID), the Basic Service Set Identifier (BSSID) or the Vendor-Specific Information Element (VSIE) fields of the 802.11 beacon frame, as shown in Fig. 2.2. Applications of this technique range from using the SSID field to broadcast distress signals in emergency situations [35], to using the VSIE field to advertise local opportunities such as the imminent arrival of a bus [36] or even to advertise privacy-awareness information such as whether the user is being captured by a surveillance camera [37].

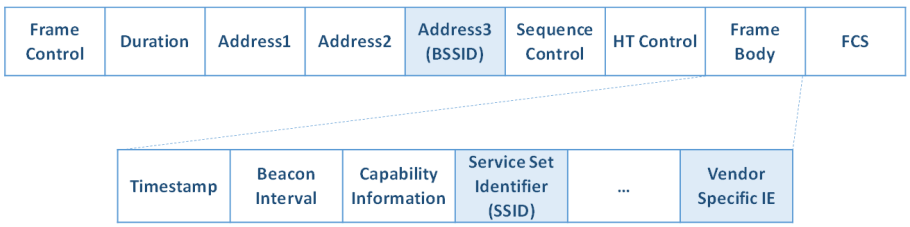


Figure 2.2: 802.11 Beacon structure

In our case, we propose the use of the VSIE field for FN discovery purposes. The use of this specific field instead of the two others is motivated by the following observations.

First, if the SSID field was used, the resulting beacons could be easily faked by users with little technical background. In addition, even if these beacons are not fake, they will be automatically detected by Wi-Fi client applications, which would mislead users who only want to connect to a Wi-Fi network and not to a fog node. As for the BSSID field, its use for purposes other than address transportation is not recommended, as it is not compliant with the 802.11 standard[36]. That is why, we use the VSIE field to carry the FN discovery information, since it is fully aligned with the intended usage of this field.

2.2.2 Wi-Fi-based discovery - Design details

As introduced earlier, our proposed discovery solution relies on embedding FN-related information into the VSIE field of the beacon frame. The FN then broadcasts this beacon via its Wi-Fi interface, thus advertizing its presence to users in its vicinity.

Fig. 2.3 shows the resulting VSIE structure.

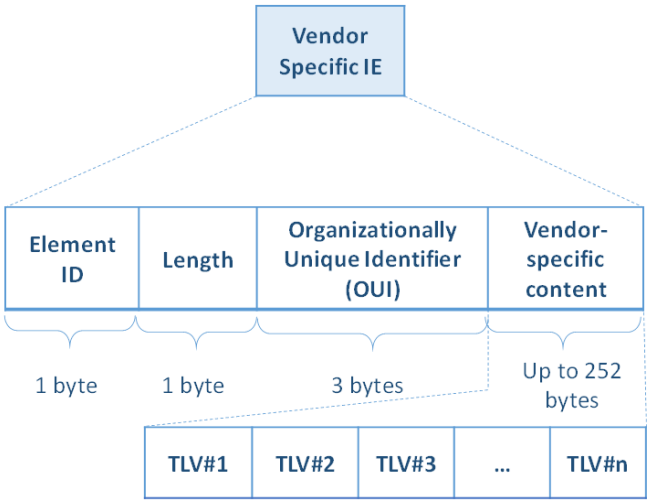


Figure 2.3: VSIE structure

The first three elements are common to all VSIEs and they include:

- the *VSIE identifier*, which indicates that this field is a VSIE. According to the standard, its value is 221 (DD in hexadecimal).
- the *length* of the VSIE content.
- the *Organizationally-Unique Identifier* (OUI), which could correspond to the OUI of the vendor of the FN device. In our case, we used *FF:22:CC* as a OUI, since it is not being used by any organization.

These fields are followed by the actual vendor-specific content, i.e. the custom information to be advertized within the beacon, which could take up to 252 bytes. In our case, this corresponds to some attributes that characterize the FN. We chose the Type-Length-Value (TLV) format to encode these attributes, as it is common in data communication protocols. More specifically, the TLV of each one of these attributes has 3 sub-fields, which are *type*, *length* and *value*, where the type field encodes the attribute type and it is represented by one byte, the length field contains the length of the attribute's value, which is provided last.

To allow FN discovery, we proposed the use of three potential types of TLVs. The first one contains the FN identifier generated by the identification scheme used by the fog computing system, the second one contains the types of tasks that can be managed by the FN, and the third one represents the urgency level, i.e. which task priorities can be handled by the FN depending on its load level.

The corresponding hexadecimal encoding for these attributes is shown in Table 2.1. This list could be eventually extended if needed and if the maximum vendor content length has not been exceeded.

Table 2.1: Used TLV types

TLV type	Hexadecimal value
FN ID	0x01
Service type	0x02
Urgency level	0x03

Examples for values that can be taken by the task type and urgency level TLVs are provided in Table 2.2 and Table 2.3.

Table 2.2: Task type codes

Task type	Hexadecimal code
High storage	0x01
Medium storage	0x02
Low storage	0x03
High bandwidth	0x04
Medium bandwidth	0x05
Low bandwidth	0x06
High processing	0x07
Medium processing	0x08
Low processing	0x09

Fig. 2.4 shows a hexadecimal encoding for an example VSIE. It indicates that it carries

Table 2.3: Urgency levels codes

Urgency level	Hexadecimal code
High	0x01
Medium	0x02
Low	0x03

a FN ID equal to a1:b2:c3:d4:a1:b2:c3:dd, that the FN prefers tasks that consume medium bandwidth and having a low urgency level.

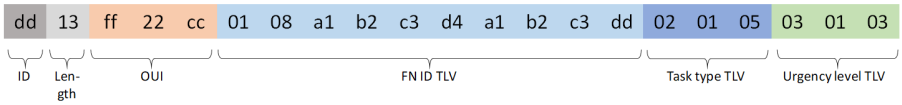


Figure 2.4: Example VSIE encoding

After encoding its information into the beacon as described above, the FN starts the broadcasting process. The user, on the other hand, starts scanning for such custom beacons using its Wi-Fi interface, without the need to be initially connected to the same wireless LAN as the FN. When the user detects a beacon that contains the FF:22:CC OUI, this means an FN is present in the vicinity. The user then decodes the advertized information and retrieves the FN attributes. Finally, it can connect to the FN if such attributes match its preferences.

2.3 Evaluation results

In this section, we outline the implementation details of our proposed solution. We also report the results obtained in our experiments and provide a discussion about several aspects related to the applicability of our proposal.

2.3.1 Implementation and evaluation setup

In order to implement the proposed solution, we used the Linux-based *hostapd* utility[3] on the FN side, which allows a Wi-Fi interface to act in access point mode.

For the purposes of our work, we specifically use it to create a customized broadcasting behaviour. A Python script has been used to perform the proper encoding of the VSIE information into the format expected by the *hostapd.conf* file. It takes as an input the broadcast parameters (i.e. the broadcast frequency and the name of the Wi-Fi interface)

[3]https://w1.fi/hostapd/

along with the FN information to be included in the VSIE as detailed in Section 2.2.2. The Python script is also used to trigger the broadcasting via the hostapd service.

On the user side, a mobile application based on the Android operating system has been developed as a discovery client. Since custom VSIEs are not included by default in the scan results provided by the Android Wi-Fi API, we used the *nlscanner* binary[4] developed in [36] to retrieve the content of the VSIE containing the FN information. This binary allows the VSIEs to be passed from the kernel space to the user space, without requiring rooting the android device.

The overall details of both user-side and FN-side implementations are depicted in Fig. 2.5.

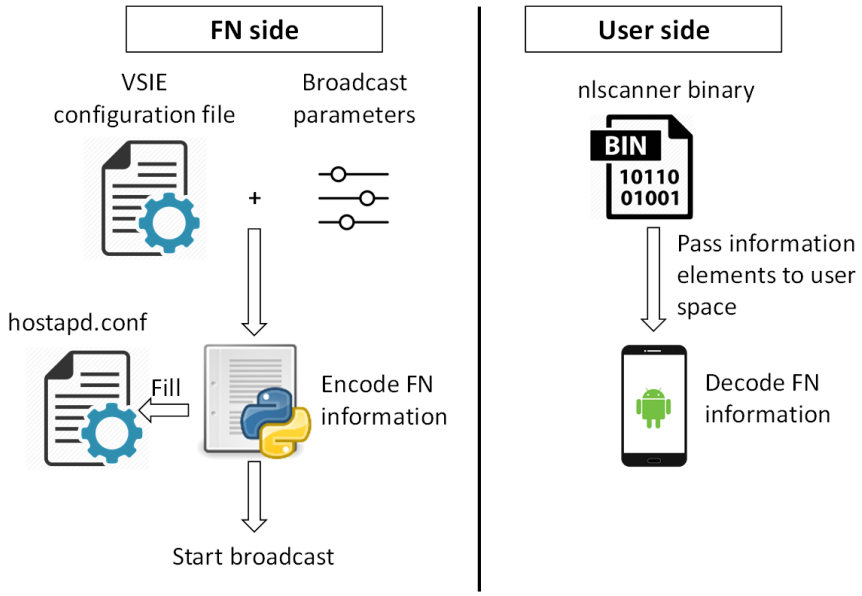


Figure 2.5: Implementation

We have also developed a Linux-based implementation of the user side discovery client using the *iw* utility and a Python script to decode the content of FN VSIE. This implementation[5] has been tested and validated in two projects, namely GUAU and mF2C[38]. However, in the following, we only focus on the mobile case to show the impact of our solution on devices with power consumption constraints.

We compare our proposed solution to two widely-used discovery approaches, which have been reported in Section 2.1. The first one relies on the use of Zero Configuration Networking [20] whereas the second one is based on a directory server, hosted in the cloud and that contains information about the available FNs (e.g. [26]).

[4]https://github.com/lows/lows
[5]https://github.com/mF2C/ResourceManagement/tree/master/Discovery

We used the open source RxDNSSD[6] library for developing the ZeroConf-based Android app, since Android's Network Service Discovery (NSD) API has resulted in unexpected crushes during our tests. On the FN side, we used the library *python-zeroconf*[7] to advertize a service called "_fn.tcp" in the same wireless network that the Android device is connected to.

As for the cloud-based discovery solution, we implemented a cloud-based web server using Heroku[8]. A HTTP GET request is triggered every 1 minute from the mobile device to the cloud server to retrieve the list of available FNs. The mobile device is connected to Internet via the same Wi-Fi network used in the ZeroConf solution.

In our proposed solution, the scan interval is also set to 1 minute, whereas the beacon broadcast is performed at the default $100ms$ interval.

All three solutions were tested on a smartphone running Android 4.1.2 as a user, while a laptop running Ubuntu 16.04 has been used as an FN.

2.3.2 Discovery times

The first metric that we evaluate is the discovery time. We first note that in the Zeroconf-based solution and the cloud-based solution, some necessary network-related operations are performed prior to the discovery process itself, therefore impacting the total time that will be experienced by the user.

The first one is the discovery of the Wi-Fi network. This step is a prerequisite for the DNS-SD FN service discovery in the Zeroconf solution, and to connect to Internet in the case of the cloud-based solution. More specifically, it corresponds to the time needed to scan the Wi-Fi channels and it generally takes $1.560\,ms$ in an Android device (See [39]). The second operation is the network association and it generally lasts $100\,ms$ according to [40].

The impact of these operations on the overall discovery process is illustrated in Fig. 2.6 for the 3 approaches. In our proposed solution, the time consumed in the "FN discovery" step corresponds to the time taken since the user's device starts listening for the FN's beacons until the time when the content of the beacon is successfully decoded. Our proposed solution combines WLAN discovery and FN discovery into a single step through the use of beacons, therefore resulting in time-savings. The user-FN association is performed at a later step, so it does not contribute to the discovery time. On the other hand, we note that the Zeroconf solution may require as many rounds of WLAN discovery, association and FN discovery as there are WLANs in the user's location, which may delay the FN discovery process. Finally, in the cloud-based solution, the discovery time refers to the

[6] https://github.com/andriydruk/RxDNSSD
[7] https://github.com/jstasiak/python-zeroconf
[8] https://www.heroku.com/

time between issuing the HTTP GET request and obtaining the response, excluding the time needed for the WLAN connection establishment.

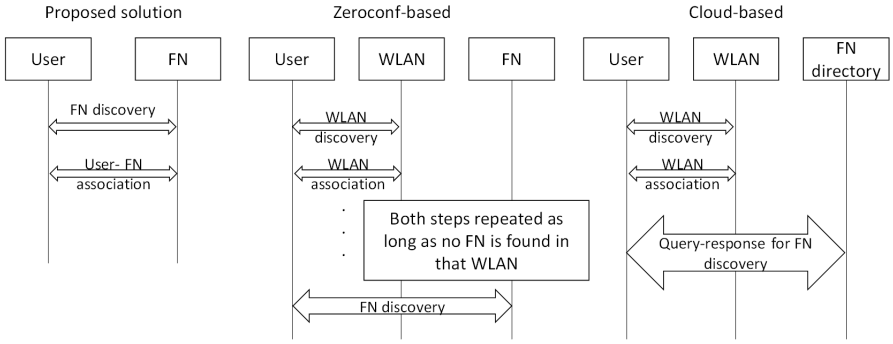


Figure 2.6: Comparison of the operations of the three considered discovery approaches

The results for the discovery times, excluding network-related operations, are shown in Table 2.4, where the values obtained for 10 different trials are provided.

Table 2.4: Discovery times

Trial #	Proposed approach	Zeroconf-based	Cloud-based
1	854 ms	141 ms	717 ms
2	832 ms	119 ms	158 ms
3	866 ms	162 ms	888 ms
4	830 ms	109 ms	259 ms
5	837 ms	575 ms	117 ms
6	863 ms	122 ms	673 ms
7	873 ms	217 ms	97 ms
8	858 ms	214 ms	187 ms
9	916 ms	234 ms	254 ms
10	835 ms	251 ms	333 ms
Average	**856.4 ms**	**214.4 ms**	**368.3 ms**
Standard deviation	24.69 ms	129.86 ms	269.18 ms

The discovery time obtained by our proposed approach was 856.4 ms on average, which falls within the standard range for a 802.11 scan duration. In fact, according to [39], the time spent by an Android device for dwelling on each Wi-Fi channel is 120 ms. Therefore, taking into account the 13 channels of the 2.4 GHz spectrum, the resulting time would be 1.560 ms. We note that the use of Wi-Fi results in many factors influencing the discovery time, such as the vendor of the Wi-Fi chip, the used OS, and the delay taken to process and decode the beacon information.

An average discovery time of 214.4 ms has been obtained by the Zeroconf solution.

However, as pointed out previously, this process assumes that the user and the FN are pre-associated to the same Wi-Fi network, which may not be the case in a fog computing context. In addition, the user has to go through all detected Wi-Fi networks, since it does not have prior knowledge on which network has an FN connected to it, which is not efficient. Since our proposal advertizes the FN information in the beacon, associations to networks where no FNs are present will be avoided.

Finally, as it can be seen, the centralized directory-based discovery solution achieves a discovery time of $368.3\,ms$ on average. It is worth-noting that the measurements show a high variance, where longer delays have been obtained. In addition, while it may be observed that our solution causes a higher delay, the fact that it is edge-centric results in reducing the load on the core network compared to the cloud-based solution, which is one of the objectives of fog computing.

2.3.3 Battery power consumption

The Trepn[9] profiler has been used to measure the energy consumed by each one of the discovery approaches. Five runs were performed for each experiment, each lasting 15 minutes and starting with the Android device having a full charge. In order to ensure that UI usage is not included in the measured energy consumption, each of the considered approaches was implemented as a background service.

For both our proposed solution and the Zeroconf-based solution, we evaluate two scenarios. The first one, called Advertizements enabled, is when there is indeed an FN in the user vicinity. As a result, advertizements are made by this FN (either using beacons or service advertizements depending on the approach) and when the user starts the discovery process it will discover it. The second scenario, called Advertizements disabled, describes the case where advertizements are absent. So, performing the discovery process on the user side will lead to wasted energy resources. As the cloud-based approach does not use advertizements, this distinction into these two scenarios is not applicable.

Table 2.5 depicts the obtained results. It shows that similar performances were obtained, especially for our proposed approach and the cloud-based approach. On the other hand, a slightly higher energy consumption is obtained when using the Zeroconf approach. This is due to the use of the default behaviour of the RxDNSSD library, resulting in DNS-SD queries being resolved more often, compared to the 1-minute discovery interval used in our approach and the cloud-based approach.

In addition, the obtained results can be attributed to the same factors as in Section 2.3.2. In fact, if we take into account the power consumption resulting from WLAN-related operations in the results of the Zeroconf and the cloud-based approaches, an increase in their overall energy consumption will be observed.

[9] https://play.google.com/store/apps/details?id=com.quicinc.trepn

Table 2.5: Battery power consumption

Proposed		Zeroconf-based		Cloud-based
Advertizements enabled	Advertizements disabled	Advertizements enabled	Advertizements disabled	
1474.86 mW	1484.19 mW	1619.55 mW	1372.57 mW	1459.39 mW

We also note that our discovery approach has resulted in an average power consumption of 1474.86 mW in the advertizements enabled scenario, which is in line with common values for the Wi-Fi scans that constitute the underlying component for our user-side discovery process. In fact, the study in [41] shows that an energy consumption between 60 mW to 150 mW is typical for a Wi-Fi scan. Since in our case 15 discovery calls were performed during the experiment, the obtained value of 98.32 mW per scan lies in the standard range.

Finally, Table 2.5 shows that the energy consumption of our approach in the "advertizements disabled" scenario is very close the one obtained when advertizements are enabled. This is due to the fact that the underlying scan is still made, even though there is no FN information to retrieve. This suggests that the process of retrieving information from the beacon and decoding it does not have a significant impact on the overall energy consumption. For the Zeroconf-based approach, a reduction in the energy consumption is observed in the "advertizements disabled" scenario, since the client app does not find advertizements to resolve. It is worth noting that for both approaches, the discovery process was wasteful for the device, since no FN was present in the area, which suggests the need for further optimizations of the discovery process. We present two schemes in this direction in Chapter 3.

2.4 Discussion

In this section, we provide a discussion regarding the potential advantages, disadvantages as well as some challenges that may be associated with the adoption of our proposed solution.

We start by summarizing the advantages of our approach. First, one of its key characteristics is that it allows the discovery of FNs without the need for the user to be pre-associated to the same WLAN as the FN. As a result, the user can retrieve the FN information prior to establishing a connection and does not have to perform unnecessary associations to WLANs even when they may not contain an FN, which results in a time- and energy-efficient process. Besides, our solution is specifically tailored to the discovery of fog nodes and not all devices present within the user's radio range. In addition, it leverages proximity to end users, which is a key factor in fog computing. Finally, since our solution is not broker-based, it avoids bottleneck problems that can occur when a high number of FNs connect to a single broker. In addition, it avoids the problem of broker discovery, which is usually configured in a static manner in the related literature.

As for the limitations that may be associated with our solution, we first cite the need to perform modifications at the kernel space level to ensure all operating systems and wireless chips are supported. To address this, the IEEE 802.11 standard and the different actors involved in fog computing can join efforts to facilitate the use of custom VSIEs across different types of devices. An additional worth-noting aspect is that the Wi-Fi card of the FN should support the Wi-Fi master mode in order to be able to broadcast beacons. As an aside, it cannot operate as a client for another Wi-Fi network and as a broadcaster simultaneously, unless such an option is available in the FN's wireless driver. Nevertheless, if the FN contains more than one Wi-Fi interface, then this problem can be eliminated. Finally, even though our solution leverages the Wi-Fi technology, which is envisioned to be widely used in fog computing contexts, it is only valid within a local scope. Therefore, for a wider-scale adoption, it can be enhanced with the use of a more generic discovery protocol with support for other communication technologies.

There are also some other challenges that would be worth-investigating, as we detail next:

- Optimizing the selection of the discovery parameters on the user and the FN sides: As it can be seen from the experiments in Section 2.3, we used a scan interval of 1 minute and a beacon interval of 100ms (the default value in conventional Wi-Fi). Both adaptive beaconing and scanning behaviors can be adopted instead, to ensure a more efficient discovery process.

- Another challenge would be how the FN sends the user the connection information after the discovery process. This should be done in a secure manner in order to prevent unauthorized users from associating with the FN. A federated authentication service similar to Eduroam could be envisioned to guarantee a secure access to the FN.

- The problem of detecting fake beacons, i.e. beacons not generated by a real fog node, should be also dealt with. To do so, techniques based on advances made in fake access point detection, such as the ones presented in [42], can be used.

2.5 Conclusion

In this chapter, we presented our Wi-Fi-based fog node discovery solution, along with its implementation details. A set of real-world experiments have been performed in order to evaluate this approach. The results have shown that our solution is comparable to some widely-used discovery solutions in terms of discovery times and energy consumption. It also addresses fog-specific characteristics since it removes the requirement of pre-association between the user and the fog node in addition to alleviating the core network traffic. As a

next step, we intend to optimize the behaviour of the wireless scan, by triggering discovery according to the FN deployment context, in order to achieve more energy savings.

Chapter 3

Scan optimization

Since mobility is an inherent characteristic in fog computing systems, special attention should be paid to situations where the user is moving in areas with no FN coverage while keeping the discovery process enabled. This is due to the fact that our discovery approach relies on the use of Wi-Fi scans, which will cause an unnecessary energy consumption in non-covered areas. Consequently, the underlying scan mechanism should ideally be disabled in areas without potential FN discovery opportunities and enabled in areas where an FN may be present. This creates a tradeoff between the need to achieve a high FN discovery rate and minimizing the energy consumption. To deal with this tradeoff, we propose two approaches. In the first one, the user receives information about the remaining distance until future FN encounters through the discovery beacon received from the current FN. While this approach does contribute to energy savings, it requires the FN manager to manually configure the topology information at deployment time. It also requires the user to monitor its direction of movement, using sensor readings which may not be perfectly accurate. As a result, in our second approach, we propose the use of the cellular footprint of a given location to predict the presence of an FN. This is shown to be a reliable alternative, especially for a sequence-based prediction mechanism, like a Hidden Markov Model.

3.1 Beacon-assisted direction-aware scanning scheme (BDSS)

We start this section by giving an overview about the common approaches allowing to optimize the scan process in 802.11 networks, since they work in a similar fashion as our proposed 802.11-based discovery approach. We then provide the details of our beacon-assisted direction-aware scan scheme and report its evaluation results.

3.1.1 Scan optimization in 802.11 networks

One common approach to optimize the scan process in 802.11 networks relies on the use of a dedicated server to obtain information that would *assist* the mobile device in the scan process.

[43] is an example study where this approach has been used. More specifically, the User Equipment (UE) can query the 3GPP Automated Network Discovery and Service Function (ANDSF) server for information about the networks present in its current location. This information includes the ID of the nearby access point (AP), its frequency and the channel it is operating on. This allows the UE to efficiently spend its scanning energy only on that specific channel. The authors also assume that GPS is already used by other services in the UE. As a result, when the user sends its current location to the ANDSF server, no additional energy usage is considered. The accelerometer sensor is also used in [43] to detect situations where the user is not moving in order to further reduce unnecessary scans. Another server-assisted approach can be found in [44], where the Mobile Station (MS) requests AP topology information from a designated server. The server then returns a list with the APs to be encountered next by the MS.

Instead of relying on a server to receive scan assistance information, authors in [45] adopt a direction-aware approach, where the geomagnetic sensor is used to determine the user's movement direction. Since the MS has a predefined list of neighboring APs in the different directions, the scan will be limited to the APs to be found along the user's movement direction, instead of the whole AP list.

Starting from the observation that a mobile user might be in the range of an AP for a few seconds only, authors in [46] consider that this duration would not be sufficient to perform a useful operation (e.g. open a web browser) within the wireless network. Therefore, network detection should only be run when there is a realistic prospect of a useful connection to be established with the AP. This is determined based on the average AP range, the user speed, as well as the scan delay, the association delay and the web page loading delay. Another approach is used in [47], where the authors propose a score-based scanning schedule for opportunistic networks. The surrounding environment of a certain device could be divided into 3 categories based on the current context: (i) event, requiring a high discovery frequency for not missing contact opportunities; (ii) occasional contact and (iii) quiet environment, where the scan should not be performed frequently. This adaptive approach achieves a very low scanning frequency in a quiet environment without sacrificing contact opportunities during events.

As it can be noted, the aforementioned approaches are specifically tailored to 802.11 networks. As a result, some aspects require careful attention if these approaches were to be applied in the optimization of the FN scan process, as we detail next:

- Using a dedicated server for scan assistance introduces an additional level of complexity in the fog system management operations. This would also require the mobile device to use its wireless interface for querying the server, therefore adding another source of energy consumption.

- Using the device's GPS location to assist it in the scanning process may be accept-

able when the device has a sufficient energy supply, however for battery-constrained devices, an alternative location provisioning mechanism would be preferable.

- Existing 802.11 scan optimization approaches are based on deployments where AP coverage areas are overlapping. This makes approaches based on neighboring AP lists well-suited to 802.11 network contexts, but not in the fog computing context, especially in its early deployment phases, where the FN density is likely to be low.

Starting from these observations, we introduce in the next section our proposed Beacon-assisted Direction-aware Scanning Scheme (BDSS) for 802.11-based FN discovery. Similar to the previous approaches, BDSS is an *assisted* approach, i.e. the device receives information that will *assist* it in the discovery process. However, unlike those approaches, the assistance mechanism is provided by means of beacons, instead of dedicated servers. Similar to [45], our approach leverages the geomagnetic sensor to determine the direction of movement. However, instead of monitoring neighboring APs in the followed direction, in our case we monitor the distance travelled until reaching FNs further away in the user's path, since we consider a low FN density.

3.1.2 BDSS Description

We first start by outlining the set of assumptions that we make regarding our system model. Then, we dig into the specific details of the proposed BDSS scheme.

3.1.2.1 Assumptions

The following assumptions will be considered throughout Section 3.1:

- We adopt a grid-based representation of the city where the FNs are deployed, since such a representation is similar to the structure of the roads and intersections in a city. As shown in Fig. 3.1, columns and rows are assumed to be aligned with the North-South and East-West axis, respectively. In addition, the user is assumed to move along these four cardinal directions (See the user path in blue).

- FNs are present at fixed grid locations, as a result, FN mobility is not considered here. Each grid cell is also assumed to contain only one FN, deployed at its center.

- The user's device is equipped with built-in sensors such as the accelerometer and the geomagnetic sensor that allow it to monitor its movement and its heading direction.

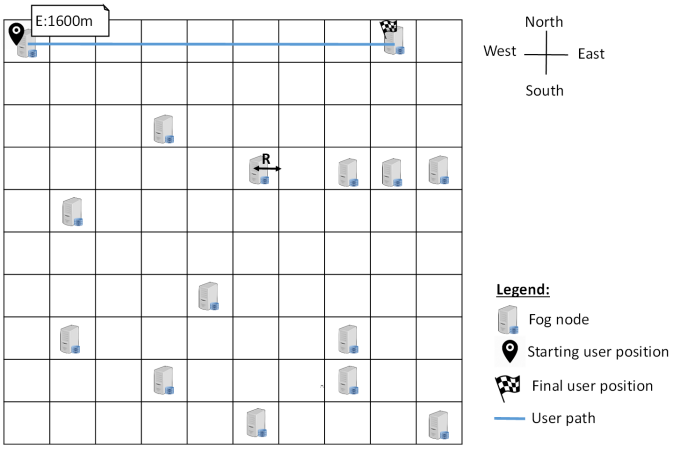


Figure 3.1: Example distribution of fog nodes in a grid

3.1.2.2 Beacon assistance - Details

In this section, we provide specific details about our proposed beacon-assisted approach. We adopt the term "beacon-assisted" since the beacon received while discovering the current FN contains information that will *assist* the user's device in optimizing the scan process for the FNs that it will encounter later along its path. More specifically, and in line with our approach of using the VSIE field of the beacon to carry custom information (See Chapter 2, Section 2.2.2), we propose the addition of the following TLVs:

- TLV_E (TLV_W, TLV_N and TLV_S respectively): Each of these TLVs has a value that is an indicator of the approximate distance between the FN sending the current beacon and the FN to be encountered if the user moves along the East, West, North, South direction, respectively. The content of the TLV value will be multiplied by 200 to obtain the exact distance in meters. This allows us to include information about FNs located up to 51.2 Km away, while using only one byte for the value field. An example of how these TLVs will be used is shown in Fig. 3.1. The beacon sent by the FN at the top-left grid cell indicates that another FN is located 1600m to the East. So, if the user was going to move straight along the East direction, the provided information allows it to disable the scan until reaching the indicated FN. This results in energy-savings and an increased discovery rate, since the topology information is reliably configured by the infrastructure manager.

- $TLV_{Channel}$: Similar to related works that include the Wi-Fi channel as part of the scan assistance information, we add this TLV to indicate the Wi-Fi channel to be used by the next FN in each of the four cardinal directions. This would allow avoiding scans on unused channels, therefore resulting in reduced scan durations. The channel information is organized into 4 4-bit sub-blocks, as follows: from left to right, the

first 4 bits represent the channel in use by the next FN located at the East (West, North, South, respectively) direction (if any, otherwise it would be 0). For instance, in Fig. 3.2, the 4 Least Significant Bits indicate that the FN located at 600m to the East of the current FN will be broadcasting beacons on Channel 6.

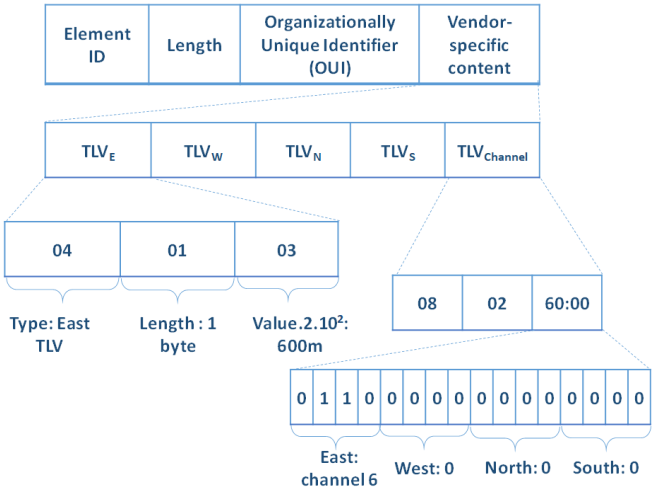


Figure 3.2: Hexadecimal encoding of the 4-directional FN location and channel information within the 802.11 VSIE

3.1.2.3 Process description

When all FNs are configured to broadcast scan assistance information as described in the previous section, the user can leverage this information to optimize its scan process, as described below:

1. ***The "no prior topology knowledge case"***: This situation occurs when the user has not discovered an FN yet, since the beginning of its mobility. As a result, it will not have information about future FN encounters.

   In this case, if the user keeps the scan disabled, it will miss the opportunity of discovering the FNs that are present along its path. To illustrate, if the user adopts this behaviour, it will miss all FNs along its path, starting from the FN present in the yellow-starred grid location in Fig. 3.3. Alternatively, if the scan was always enabled, e.g. triggered every 30s, regardless of the surrounding topology, it may be the case that the user has not moved out of its current location yet. This makes subsequent scans useless, since the scan results will be the same. For example, the scans will be useless in the part of the path marked with green arrows.

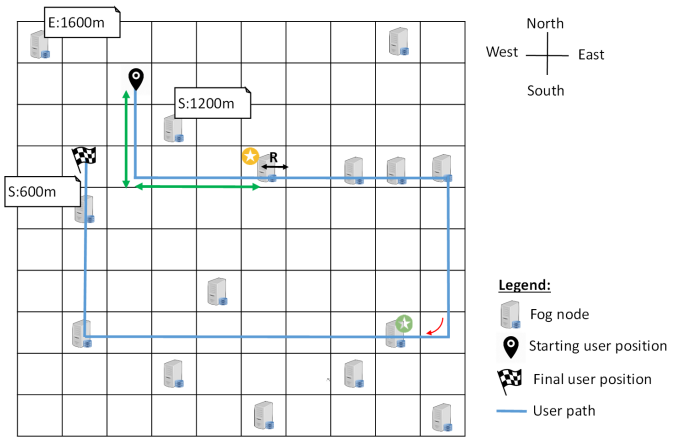


Figure 3.3: Example distribution of fog nodes in a grid - Particular situations

So, to balance the effects of these two approaches, we propose to enable the scan whenever a certain distance is traveled, thus referring to this part of our scan scheme as the *spatially-periodic scan*. This ensures that the scan frequency is reduced, especially when the user remains in the same location, thus avoiding unnecessary scans and saving energy. At the same time, it guarantees that the scan is performed at each newly-visited location, which will increase the discovery rate. We adopt the distance between two adjacent FN areas as the distance over which the scan is re-enabled (2R in Fig. 3.3), since it allows a suitable representation of all FN areas. We note that the spatially-periodic scan approach will be used in the simulations as one of the solutions we compare our full BDSS algorithm to.

2. ***Overall BDSS scan process description***: As shown in Algorithm 1, the first step consists in monitoring the user movement using the accelerometer sensor. If some movement is detected, subsequent steps will depend on the movement speed. In fact, if the user is moving at a high speed, the scan will be disabled (L10-11), since the user will not have enough time to connect with the FN and send it its tasks. On the contrary, if the user is moving at a lower speed, then it has a chance to connect to an FN for a long enough duration. In this case, if prior topology information has been acquired from a previous FN, direction monitoring through the geomagnetic sensor will be triggered (L13-14). If a direction change with respect to the previous FN is observed (see red arrow in Fig. 3.3), the scan is enabled according to the spatially-periodic scan approach defined previously, otherwise, the FNs present in the new direction will be missed (e.g. the green-starred FN). If no change of direction was observed, the user keeps track of the traveled distance with respect to the previous FN, using the accelerometer. When the FN indicated in the beacon of the previous FN is reached, the scan is enabled (L19). Otherwise, the scan will be disabled to achieve energy savings (L21).

As described previously, if the user has not acquired topology information yet, the spatially-periodic scan will be enabled (L22-23).

Finally, if no movement has been detected at the beginning, the scan will be enabled once to discover FNs in the current location and then it will be disabled (L25-26).

We note that the ProcessScanResults function (L1-6) is called whenever a scan is enabled. Its role is to decode the beacon assistance information, i.e. the FN topology information in the 4 directions and the used Wi-Fi channels, if any. It also determines the user's direction and retrieves the remaining distance to reach the FN to be found in this direction along with the channel to be used to scan for it.

Algorithm 1: BDSS Algorithm

```
1  Function ProcessScanResults(scanResults)
2      if scanResults list not empty then
3          topologyView ← decode(scanResults)
4          directionToRemember ← getHeadingDirection()
5          remainingDistance ←
            getRemainingDistance(topologyView,directionToRemember)
6          channelToUse ← getChannelNumber(topologyView,directionToRemember)
7  Algorithm Direction-aware scan decision making
8      Start movement monitoring
9      if Movement detected then
10         if High speed then
11             Disable scan
12         else
13             if Previous topology information available then
14                 Monitor movement direction
15                 if Change of direction compared to last FN then
16                     Enable spatially-periodic scan (on all channels)
17                 else
18                     if Distance threshold reached then
19                         Enable scan on channelToUse
20                     else
21                         Disable scan
22             else
23                 Enable spatially-periodic scan (on all channels)
24     else
25         Enable scan
26         Disable scan
```

3.1.3 Evaluation results

In this section, we provide a description of the evaluation setup and report the obtained results.

3.1.3.1 Evaluation setup

In order to evaluate the proposed BDSS scheme, we used the *mininet-wifi* [48] network emulator, since it already contains the *hostapd* and *iw* utilities. Similar to our initial implementation in Chapter 2 (Section 2.3.1), we used hostapd to embed the FN topology and channel information into the vendor elements field in the *hostapd.conf* file. We also used iw to retrieve the scan results, which were in turn decoded using a Python script.

User mobility is generated according to random movements along the four directions in the grid, where the starting and final user positions are also generated randomly.

The evaluation parameters are shown in Table 3.1. More specifically, Table 3.1 shows three different FN density scenarios that have been used in our evaluation. The first one corresponds to a sparse deployment, where 25% of the user's path is covered with FNs, while the two others correspond to a higher density where 50% and 100% of the path is covered with FNs.

Table 3.1: Evaluation parameters

Parameter	Value
Broadcast range, R	100 m
Emulated area	3000 m x 3000 m
Number of FNs	10 (25%) 20 (50%) 41 (100%), generated in random positions
Number of runs per scenario	10
Energy consumption per passive Wi-Fi scan per channel	11 mW[43]
Time needed to scan a single Wi-Fi channel	120 ms[39]

We compare our proposed BDSS scheme to the following schemes:

- BDSS without channel information, i.e. Alg. 1 with Line 6 excluded, and the scan is enabled on all channels in Line 19. This is to evaluate the effect of adding the channel information in the assistance information.
- The spatially-periodic scan scheme presented in the first part of Section 3.1.2.3, where two values are compared for the distance at which the scan is re-enabled, which are 2R (more frequent scan) and 4R (less frequent scan).

3.1.3.2 Obtained results

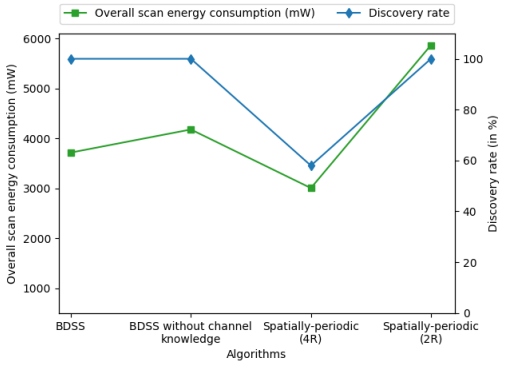
Fig. 3.4 shows a comparison between the four considered scanning schemes in terms of the overall scan energy consumption and the corresponding discovery rate. The latter represents the ratio of the number of FNs that were detected by the scan scheme to the number of FNs that are present in the user's path.

Fig. 3.4a shows the results for the sparse deployment case where 25% of the user's path is covered with FNs. In this case, the spatially-periodic scheme with a 4R distance interval obtains the lowest energy consumption since it has a reduced scan frequency. This comes at a cost of a low discovery rate, since multiple FN discovery opportunities have been missed when the scan was disabled. Our scheme, on the other hand, is able to reduce the energy consumption compared to the 2R spatially-periodic scheme, while obtaining a 100% discovery rate. Even though a few FNs were found during mobility, the channel information they provide further contributes to reducing the energy consumption compared to BDSS without channel knowledge.

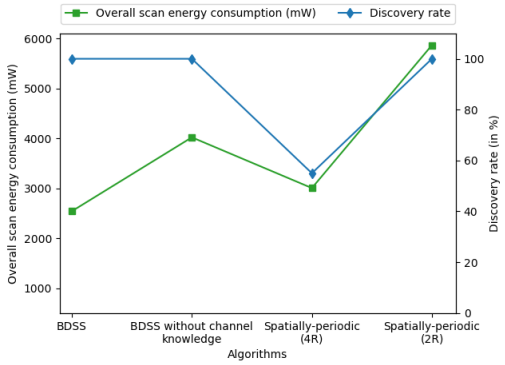
A similar trend is observed in Fig. 3.4b and Fig. 3.4c, for the 50% and the 100% coverage cases, respectively. We note that the more FNs are encountered, the more information about the topology is acquired, which contributes to a more energy-efficient scan process. When the user's path is fully covered with FNs, significant savings can be made thanks to the prior knowledge on the specific channel to use for scanning. As this information is not considered in BDSS without channel knowledge, all channels will be considered in each FN discovery, therefore resulting in an increased energy consumption.

In Fig. 3.5, we plot the total duration that each scheme spends on scanning. It is determined based on the number of performed scans per channel and the time needed to scan a single channel (shown in Table 3.1). As it can be seen, for BDSS, the scan duration is considerably reduced when the FN coverage increases, since more FNs were encountered and therefore provided information about the channels used by future FNs. We also note that the topology information is useless without the channel information, especially for the 100% coverage case, as shown by the values obtained by BDSS without channel knowledge. The two spatially-periodic approaches depend only on the value of the distance interval, therefore they result in a constant total scan duration, regardless of the FN coverage. The 4R scheme has a lower overall scan duration, since it performs scans less frequently than the 2R scheme.

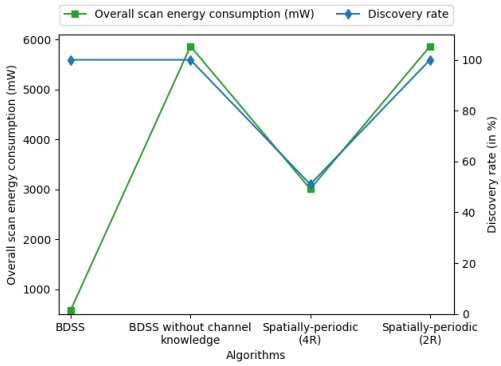
The number of turns in the user path also has an impact on the performance of BDSS. This impact is shown in Fig. 3.6. In the 25% coverage scenario, the total energy consumption is considerably increased when there are 25 turns in the user path. This is because with each turn, i.e. a direction change, a scan is done using all channels in order to avoid missing potential FNs in the new direction. However, as the deployment is sparse, FN discovery opportunities are rare, so no channel information will be acquired. However,



(a) 25%



(b) 50%



(c) 100%

Figure 3.4: Overall scan energy consumption (mW) Vs. Discovery rates

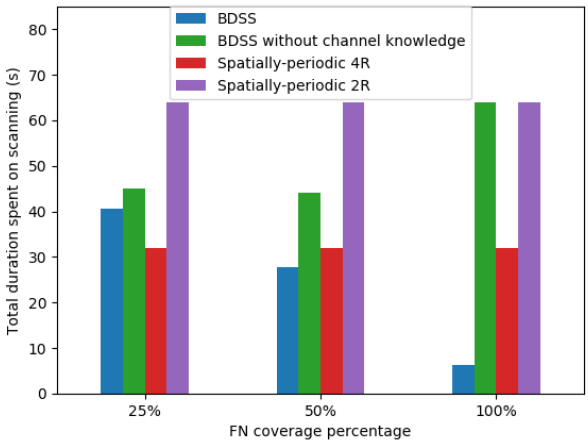


Figure 3.5: Total duration spent on scanning

the energy consumption is considerably reduced when the path has a few turns only, since the scan will be kept disabled as long as the user is moving straight, with respect to the previous FN. In the 100% coverage case, the number of turns does not have an impact on the energy consumption, since it is already minimized due to channel knowledge acquired from each encountered FN in the path.

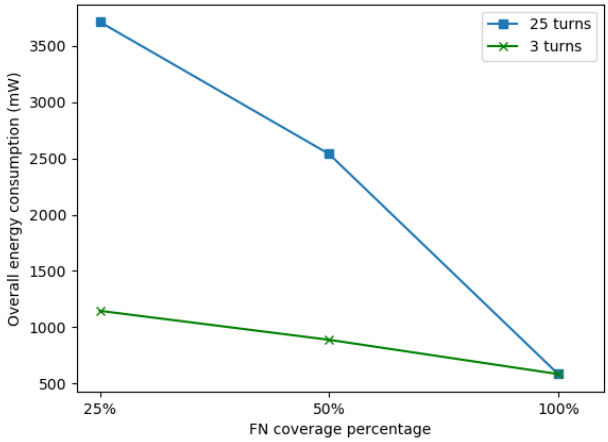


Figure 3.6: Impact of the number of turns in the user path on the overall scan energy consumption

Fig. 3.7 shows the relationship between direction monitoring and the overall scan energy consumption for different random runs of the 25% coverage case. As it can be noted, when there are more steps in the user path where direction monitoring is performed, the consumed energy is reduced. Even though the energy consumption shown here does not take into account the energy consumption generated from the geomagnetic sensor, existing studies[45] show that the geomagnetic sensor is a low-power sensor. Therefore, using it to improve the efficiency of the scan process will not add a large energy overhead.

3.1.3.3 Limitations

Even though the proposed BDSS scheme contributes to reducing the device's energy consumption while maintaining a high discovery rate, it can be associated with the following limitations:

- It assumes a perfect user mobility model, where the user is restricted to move along the four cardinal directions. This does not necessarily occur in a real-world user mobility pattern.

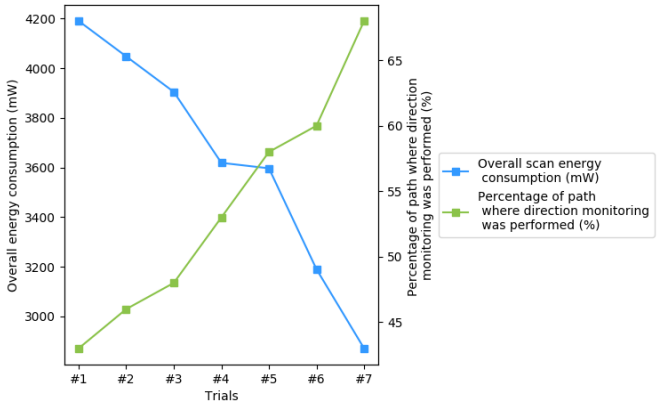


Figure 3.7: Overall scan energy consumption Vs. Direction monitoring

- The FN deployment is also assumed to be along the North-South and the East-West axes, which adds another restriction level for the fog infrastructure manager.
- The direction of the user mobility is assumed to be determined based on an algorithm that uses the geomagnetic sensor data as an input. However, this process does not necessarily have a 100% accuracy and is prone to errors, which may lead to FNs being missed. In addition, when the algorithm falsely indicates that the user has made a direction change, then scanning will be enabled on all channels instead of only one (if there is a FN in the user location) or none (if there are no FNs). This results in increased scan durations and the corresponding energy consumption.

This leads us to consider an alternative way to optimize the scan process, namely an approach that could infer whether a scan is necessary or not based on the user's context.

3.2 Context-aware scan for FN discovery

In this section, we outline the different context-awareness mechanisms used in the literature along with their advantages and limitations. We then explain the proposed use of the cellular footprint of a given location as a context-indicator that could help optimize the scan process. Evaluation results using different machine learning (ML) classification approaches are presented, as well.

3.2.1 Context-awareness mechanisms in related literature

Multiple context-awareness mechanisms have been proposed in the literature. Using GPS coordinates as an indicator for a user's context would be one straightforward approach.

Despite its high precision, GPS is associated with excessive energy costs. In addition, it is not suitable for indoor environments[49]. A different approach to achieve context-awareness relies on the use of the Wi-Fi footprint of a given location. More specifically, such approaches leverage information such as the list of nearby APs, their characteristics and the received signal strength levels in order to detect places of interest (POIs). Although this approach is able to detect POIs with high accuracy[50], it would not be suitable to optimize the energy-efficiency of the scan process as it uses the Wi-Fi interface to infer the surrounding context.

A more energy-efficient approach would then be to use cellular footprints, which consist in the set of neighboring cellular tower IDs and their signal strength levels. Such an approach has been widely used for localization[51], efficient AP discovery [52] and discovery of POIs[53]. It not only accurately represents the context of a given location, but it also does not incur energy consumption overheads, since the cellular information is always received by a mobile device. The study in [52] further confirms this. In fact, the authors examine the impact of historical information such as the cellular tower ID, bluetooth IDs of nearby devices and the user speed on the estimation of the time remaining until a user will encounter the next AP. The most promising results were obtained using the cell ID approach. The use of cellular footprints as an alternative to GPS localization has been studied in [51]. The authors have shown that, using information from the serving cell and two neighbor cells, only a small localization error has been incurred (20m). However, most modern mobile phones, Android-based for instance, do not report neighboring cell information. This is because this feature is not supported by their chip, even though it is supported by the Android SDK. Only information of the serving cell can therefore be retrieved. Such a limitation has led the authors in [54] to use this limited information in their proposed localization scheme. More specifically, a Hidden Markov Model is used to achieve this. It allows to obtain accurate localization results, despite the use of a single cell information in nearby areas, where cellular footprints can be the same. Therefore, motivated by the work in [54], our work also uses the cellular footprint from the serving cell only.

3.2.2 Description

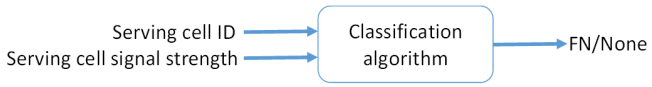


Figure 3.8: Illustration of the prediction of the presence/absence of an FN based on the cellular footprint

The objective of our context-aware scan scheme is to use historical cellular information to predict whether the currently-observed cellular information is representative of a loca-

tion where an FN is present or not. This can be made possible thanks to the extensive presence of cellular towers in urban environments. More specifically, the required cellular information, referred to as "cellular footprint", is the ***(serving cell id, signal strength)*** pair. Two phases need to be performed in order to achieve our objective.

The first phase is a data collection phase that is conducted with the scan always enabled. Within this phase, each observed cellular footprint is logged along with its true label, i.e. whether it is associated with the presence of an FN or not.

Then, the collected data will be used by a learning algorithm to predict whether a fog node is present or not, given the currently-observed cellular footprint, as shown in Fig. 3.8. In case the algorithm predicts the presence of an FN, the scan should be triggered to increase the discovery rate. Otherwise, the scan should be disabled to save energy.

Since the considered dataset is labeled, we use supervised learning algorithms and more specifically classification algorithms to achieve our goal. We consider two simple learning algorithms, K-nearest neighbors (KNN) and Decision Tree (DT) and a more complex, sequence-based learning algorithm, a Hidden Markov Model (HMM) , for comparison, as we describe next.

- **KNN**: KNN is based on calculating the distance between a newly-observed data entry and each of the historical data entries. The K entries which are closest to the new entry are then retrieved and the majority class found within these K entries will be the output value predicted by the KNN algorithm. In our case, we customize the original algorithm as follows:

  1) Since calculating the distance to each entry in the historical data whenever a scan decision has to be made can be time-consuming, we only consider distance calculations to a smaller subset of the data where only the data entries having the same cell ID as the one in the current cellular footprint are considered. Within this subset of data, distances are calculated based on the relative difference between the values of the signal strength levels, as follows:

  $$distance = \frac{|ss_1 - ss_2|}{max(|ss_1|, |ss_2|)} \tag{3.1}$$

  where ss_1 and ss_2 are the two signal strength levels to be compared.

  2) Since there may be situations where the current cell ID has not been observed before (especially when not enough historical data has been collected), in this case, no predictions are performed and the scan is enabled in order not to miss FN discovery opportunities. This may come at a cost of an increased energy consumption, but we can use the newly-collected cell ID information to improve the performance of the learning algorithm and obtain savings in future executions of the discovery service.

- **DT**: A decision tree can be used to model a classification problem using a tree structure. In our case, we first split the data according to the cell ID input variable.

Then, for each resulting subtree, we further split the data entries at specific split points, based on the signal strength levels. The Gini index[55], defined in Eq. 3.2, is used to determine how good a split is:

$$I_G = 1 - \sum_{j=1}^{c} p_j^2 \tag{3.2}$$

where p_j is the proportion of data entries that belong to class j for a specific tree node and c is the number of classes. Ideally, a perfect split results in a Gini index equal to 0, to indicate that the data entries belonging to each group created by the split have the same class.

To reduce the risks of overfitting, the splitting process stops under two conditions, i.e. when a maximum tree depth is reached or when a split contains a pre-defined minimum number of data entries. The class having the highest number of data entries in the considered tree node will be the predicted class. Similar to what we did in the KNN approach, the scan is triggered when the current cell ID is not present in the historical data.

- **HMM**: A Hidden Markov Model is a statistical model that can predict the most likely sequence of hidden states $(s_1, s_2, \ldots, s_N)$, given a sequence of input observations $(o_1, o_2, \ldots, o_N)$, where N is the length of the sequence.

  Mapping this model to our case, the observations would be represented by the set of cellular footprints, while the hidden states that will be predicted consist in the the presence or the absence of an FN.

  In order to make predictions, the following HMM characteristics will be used:

  - Emission probabilities referring to the probability of a given observation (cellular footprint) given a state (FN ID / none)
  - Transition probabilities corresponding to the probability of transition from one state at time t to another state at (t+1)
  - Prior probabilities corresponding to the general probability of occurrence of a certain state, when no prior observation is available.

  We use the Viterbi algorithm[56] to determine the most likely sequence of the hidden states based on the aforementioned probabilities. Then, the output of the HMM will be the last item of the most likely state sequence, i.e. s_N.

3.2.3 Performance evaluation

Fig. 3.9 illustrates the hypothetical FN deployment considered in our evaluation. It shows that three FNs were deployed at three locations of the city, which are our lab, a building of the engineering school belonging to our university and the city hall.

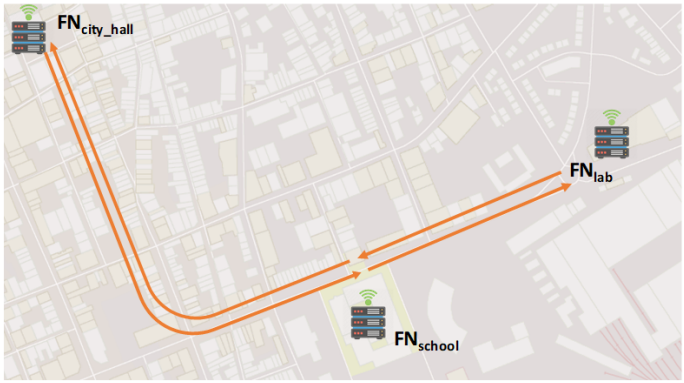


Figure 3.9: Considered data collection scenario

A GSM signal monitoring application[1] was used to collect the cellular context information of the FN deployment area. The application logs the ID and the signal strength of the serving cell every minute or when a handover to a different cell occurs. The monitoring application is enabled during the user's mobility along the path shown in orange arrows in Fig. 3.9. 10-minute stops are performed in each FN location, as this duration corresponds to the time usually spent by a user in a location of interest[53]. This also allows us to collect a sufficient amount of data entries for each FN location. Five different data collection experiments were carried out for two different network operators.

Since the city does not have an existing FN deployment in the considered locations, labels associated with the presence or the absence of an FN at a given location were manually added to the collected data. For example, all data entries corresponding to the time interval from when the user enters the school until leaving it after 10 minutes are tagged as FN_{school}. Each row of the obtained dataset is as follows:

$$\text{(cell id, signal strength, FN id)},$$

where FN id corresponds to the FN identifier (FN_{lab}, FN_{school}, FN_{city_hall}), if any; otherwise, it is set to "none".

Python was used to implement the different learning approaches presented in Section 3.2.2 and five-fold cross validation was used to evaluate the prediction performance. The parameters used for the different approaches were selected after various experimentation trials and are shown in Table 3.2.

In the following, we show the obtained results in terms of the discovery ratio, the saving ratio as well as the user-FN contact duration.

[1] https://play.google.com/store/apps/details?id=com.signalmonitoring.gsmsignalmonitoring&hl=en

Table 3.2: Paremeters of the used algorithms

Algorithm	Parameter
KNN	$k = 5$
DT	Maximum depth = 3 Minimum number of entries = 3
HMM	Emission probabilities, transition probabilities and prior probabilities: Determined based on their corresponding fractions in the training data Sequence length = 5

3.2.3.1 Obtained results

- Discovery ratio: Usually referred to as the true positive rate in classification problems, it corresponds to the ratio of the number of times in which the algorithm successfully classified a cellular footprint as corresponding to a specific FN, out of the total number of times where the user was located within the range of that specific FN. It is defined as follows:

$$Discovery\ ratio = \frac{Nb.\ of\ successful\ discoveries}{Nb.\ of\ total\ discovery\ opportunities}$$

  Fig. 3.10 shows the obtained discovery rates (in %). As it can be seen, both KNN and DT obtain similar performances, where FNs were successfully discovered more than 85% (resp. 76%) of the time for the first (resp. second) operator. The difference between the results of the two operators are likely due to different base station deployments in the considered locations. Both algorithms were not able to obtain higher discovery rates, due to the fact that the considered dataset has a small size, in addition to the presence of the FNs in nearby locations, which makes the prediction task challenging, especially that only the serving cell information can be used as an input. Prediction errors may also be attributed to the uncertainty caused by the fluctuations in the received signal.

  HMM, on the other hand, is able to successfully discover FNs 98% of the time, despite the imperfect data that was fed to it. Relying on a sequence of state transitions instead of a single one is the main reason for achieving such a high discovery rate.

- Saving ratio: The second considered performance metric is the saving ratio (more generally, the true negative rate). It allows us to assess the amount of energy savings that can be made by reducing the number of unnecessary scans in areas with no FN coverage. It is defined as follows:

$$Saving\ ratio = \frac{Nb.\ of\ savings\ achieved}{Nb.\ of\ actual\ saving\ opportunities}$$

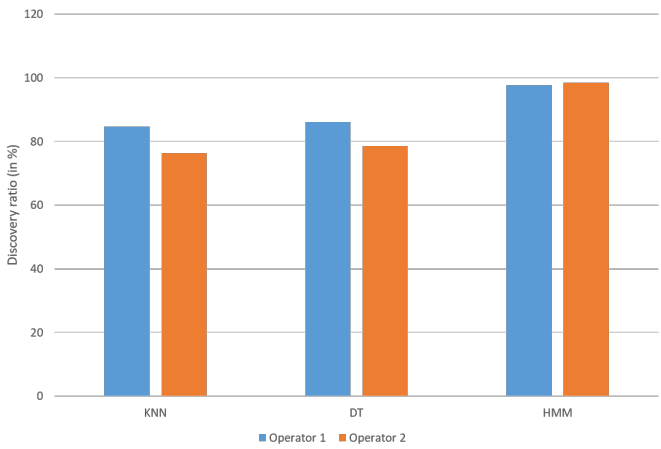


Figure 3.10: Discovery ratio

where the numerator indicates the number of times when the output of the algorithm is "none" in an area with no FN coverage (the scan can then be disabled in this area and a saving can be made) and the denominator corresponds to the number of times the user has been in an area where a saving can be made because no FN was present. As it can be seen in Fig. 3.11, KNN and DT obtain similar saving ratios up to 62%, in a pattern that resembles the results of the discovery ratios. We can then conclude that this behavior is due to the same factors impacting the discovery ratio.

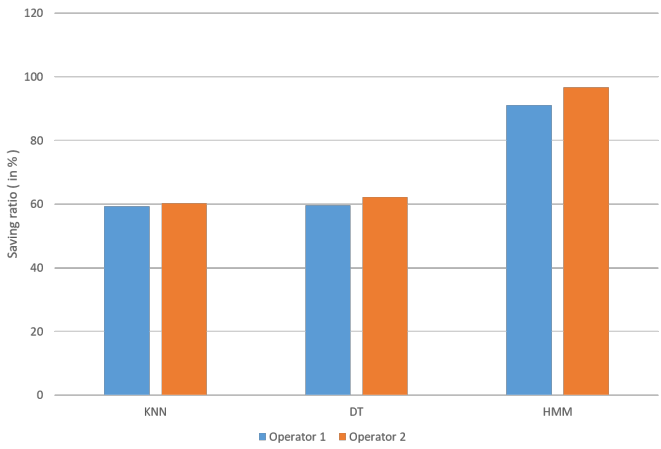


Figure 3.11: Saving ratio

As previously, more savings, up to 96%, can be achieved by leveraging the sequence-based nature of HMM.

- User-FN contact (UFC) ratio: It corresponds to the number of time steps during

which the user was connected to the FN after discovering it, divided by the number of time steps in which the user was actually located in the FN area. Ideally, a UFC ratio as close to 1 as possible would be preferred, for a better usage of the FN's resources. To better illustrate the UFC concept, let us consider the example shown in Fig. 3.12, where two incorrect predictions occur during the first two time steps. This means that the algorithm predicts "none" and the scan is kept disabled, while there is actually an FN that should have been detected. Although the FN is successfully detected in the third time step, the detection delay would result in a shorter user-FN contact duration, which would have a negative impact on the quality of service.

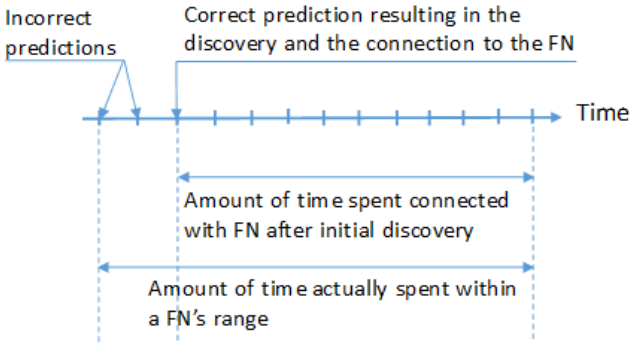


Figure 3.12: Illustration of the User-FN contact ratio

The UFC ratios of the three considered FNs are shown in Fig. 3.13 and Fig. 3.14 for the first and the second operator, respectively. A minimum UFC ratio of 80% per FN can be achieved by all algorithms. On average, a minimum UFC ratio of 92% can be guaranteed.

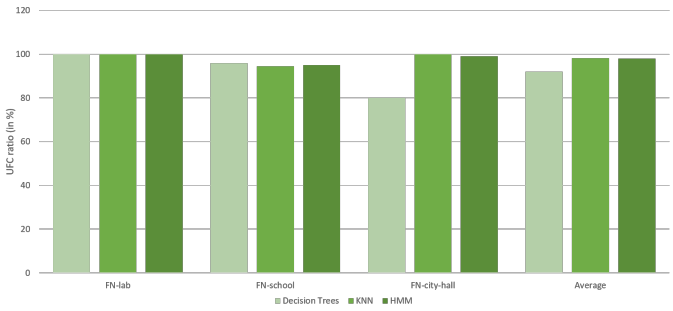


Figure 3.13: User-Fog contact ratio (Operator 1)

It is also interesting to note that despite the simplicity of KNN, it can obtain 100% UFC ratio for certain FNs (such as FN_{lab} in Fig. 3.14), especially when their corresponding measurements are relatively stable. When measurements fluctuate instead

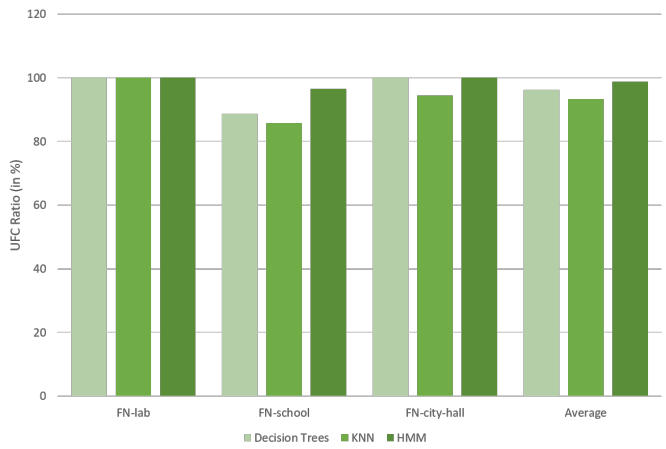


Figure 3.14: User-Fog contact ratio (Operator 2)

(like in the case of FN_{school} for the second operator), the obtained ratios are lower. Such fluctuations may be because the geographical placement of the considered BS of the second operator is associated with some environmental factors that lead to a bad signal.

3.3 Discussion

In this section, we present further considerations that could be taken into account to improve our context-aware scan scheme:

- Including channel knowledge in the context-aware scan: It has been shown in Section 3.1.3 that when the user has prior knowledge about the specific channel to be used by the FN to broadcast its beacons, considerable energy savings could be made. This feature could then be added to our context-aware scan scheme to improve the energy savings it could achieve. In fact, in the data collection process, information about the wireless channel used by a given FN can be stored. Then, when an FN is predicted to be present at a given location, the scan will be performed on this specific channel, instead of all the Wi-Fi channels.

- Dealing with the small amount of training data per user: As it was pointed out in the description of the evaluation setup, the presented results have been obtained from a small scale data collection experiment. This means that the three considered algorithms were trained using a small amount of data. Even though this has led to promising results, such results could be improved by considering a *federated learning* approach, where different nearby users contribute their learnt model to a central

server. Then, the latter will improve the shared learning model and send it back to users.

- Impact of the mobile FNs on the accuracy of the predictions: This situation occurs when the user correlates a certain cellular footprint with a given FN but that FN is a mobile FN. Thus, it is likely that the user will not detect it again in the future at the same location. In order to prevent this from causing incorrect predictions in the future, such mobile FNs should be tagged as *mobile* and not be included in the prediction process.

3.4 Conclusion

In this chapter, we explored two approaches for optimizing the scan process performed in our 802.11-based FN discovery solution. In the first approach, the user receives information about future FN encounters via the beacon that is already sent by the current FN. This allows to maintain a discovery ratio of 100%, while considerably reducing the scan energy consumption. However, this approach is associated with some restrictions on the user mobility pattern and the FN deployment. That is why, we propose an alternative, context-aware mechanism, that is able to infer the contexts in which it is suitable to avoid unnecessary scans, based on the cellular footprint of the user's location. Evaluation results show that a sequence-based learning approach such as a HMM outperforms simpler approaches such as KNNs and decision trees in terms of discovery ratios and energy savings. After the discovery process is performed, the user would have to select the most appropriate FN to send it its tasks. This problem will be addressed next, with a specific focus on the implications of the user mobility on the FN selection process.

Chapter 4

Fog node selection

In the previous chapter, we proposed approaches aiming to optimize the fog node discovery process, in a way that minimizes the underlying scanning in areas with no FN coverage. Such a use case may arise during early deployment phases of fog computing or it could occur in areas with very low demand for fog computing services. When the considered area is instead covered with a high number of fog nodes, which could occur in densely-populated urban environments, the focus should instead shift towards the proper selection of fog nodes that will process a user's computation-intensive tasks. This constitutes the problem to be addressed in this chapter. More specifically, since the mobility of a user in an area with a high density of FNs could trigger excessive switchings from one FN to the other, we propose different mechanisms allowing to ensure that the user connects to an FN having a good quality (in terms of delay), while minimizing the amount of the accumulated switching costs. We show that limiting the number of the considered FNs does not lead to a notable improvement in the performance. On the hand, we observe that selecting the same FN during a whole block of decision slots can bring significant improvements, whereas a greedy selection of an FN having a good-enough quality results in the best performance in such a dynamic environment.

4.1 Related works

The problem of mobility-induced FN selection has attracted a wide interest in the fog/edge computing research community. The efforts towards addressing this problem can be roughly divided into two main categories:

- Global control of the FN selection: In this case, given the user mobility and a global view of the different cloudlets/FNs/edge servers that may be encountered during mobility, the question is "which FN should host the user's service or the associated VM in a way that optimizes a certain performance objective". This question is usually associated with the topic of service migration in the literature.
- User-side control of the FN selection: In this case, a software component or a software agent present within the user's device is responsible for the task of FN selection, based on local information from the device's surroundings. This approach is in line with an

increasing trend towards making end user devices more intelligent[57]. In fact, the approaches proposed within this category usually rely on reinforcement learning (RL) techniques, where the software agent interacts with the nearby FNs in a sequential manner, in order to learn the best one among them to process the user's task. This can be done by using the framework of multi-armed bandits (MAB), where each FN is associated with an unknown reward distribution that should be learnt in a trial and error fashion. Alternatively, a more complex formulation, based on Markov Decision Processes (MDPs) could be used, where the state of the surrounding environment is also considered in the FN selection decision. RL algorithms such as Q-learning, either used in a tabular format or using a neural network are often adopted in the relevant literature.

In the following, we provide an overview of related works in both aforementioned categories.

4.1.1 Global control

Among the works in this first category, we cite [58], where the authors propose a mobility- and load-aware virtual machine migration (VMM) scheme based on a genetic algorithm (GA). The objective is to select the optimal cloudlet to host a user's VM such that the number of VM migrations is minimized. The authors in [59] consider the problem of cloudlet selection with the aim of minimizing the energy consumption costs, while ensuring that the user's SLA is met. To this end, they formulate the problem as a Mixed Integer Linear Program and find the suboptimal solution using the branch-and-cut algorithm. In [60], the authors propose a multi-attribute decision making approach to determine a strategy for deciding when to migrate a given service and to which cloudlet. Decision criteria include information about the available bandwidth, the computing capabilities of the cloudlets, costs associated to migration and communication as well as the resulting energy consumption. In [61], the authors propose a "mobility-based services migration prediction (MSMP)" scheme. Using MSMP, information about the user's mobility pattern and the expected load of the different micro data centers (MDCs) is used to determine an optimal plan that associates each portion of the user's service with a specific MDC. Finally, authors in [62] consider a partitioned network of FNs, therefore creating so-called FN "communities". Within this context, they use information such as the user's location and the amount of available resources at the different FNs to select an appropriate FN in a way that follows the mobility of the user, which results in a reduction of the service delay.

As it can be noted, the aforementioned approaches rely on the presence of prior knowledge of the global fog computing system information, which may sometimes be unavailable a-priori or be costly to acquire. That is why, other approaches have emerged to alleviate this requirement, but considering local user-side information in the decision making

process.

4.1.2 User-side control

One of the most relevant related works in this category is [63], where the authors consider a dense edge computing environment where multiple Base Stations (BSs) can process users' tasks. Within this context, they proposed a MAB-based approach to select the optimal BS. Even though the authors consider that the number of available BSs may vary because they may be turned off to save energy, they do not effectively address the problem of the switching costs that are incurred when different BSs are selected in two consecutive decision slots. The authors in [64] instead consider a fog computing environment, where FNs can be characterized with stationary or non-stationary delay distributions. They also consider the case of unpredictable FNs' arrivals and departures. However, their proposed MAB approach does not deal with switching costs. Similarly, the work in [65] considers a fog computing context, where the set of helper nodes have non-stationary delay distributions. They also propose a bandit approach to select the most suitable helper node. In [66], the authors focus on the specific case where FNs have different preferences towards the tasks that users may offload to them. Therefore, upon receiving a task from a user, each FN returns a binary feedback indicating its level of satisfaction with the received task type. As a result, over time, the user can learn which FN is most likely to prefer its tasks.

The authors in [67] consider that the edge servers have different reward and cost distributions. Therefore, they propose a bandit approach with the aim of maximizing the accumulated rewards (in terms of the number of tasks completed within a specific deadline) while keeping the incurred energy costs below a pre-determined budget. Instead of a simple bandit approach, authors in [68] use a contextual bandit approach, where the context including the computation complexity of the considered service, the real-time user location as well as the specific edge server that hosted the previous service is taken into account in the decision making process. The objective is to minimize a cost function taking into account the computation delay, the communication delay and the switching cost. However, since the considered context contains information about specific edge servers, it is not appropriate for the scenario with a dynamically-changing server set.

As opposed to MAB-based approaches, authors in [69] rely on a Q-learning algorithm to find the best BS to execute a user's task. They take into account information about the BS currently serving the user as well as the channel state and consider the minimization of the task execution speed as the learning objective. A more sophisticated double deep Q-network is used in [70] to select whether a task should be executed locally on the user's device or should be offloaded to one of the nearby BSs.

As will be explained later, since our proposed FN selection schemes also follow the user side control approach, we provide a summary of these related approaches in Table 4.1.

This will allow us to classify these approaches based on two criteria of interest to us, which are whether they consider a varying number of FNs and whether they take into account switching costs. The observation of Table 4.1 indicates that works addressing these two aspects are missing in the literature. This is why, in the following sections, we propose approaches that aim to address these issues in a joint manner.

Table 4.1: Comparison of related works on user-side control

Reference	Approach	Variable number of FNs /MEC servers	Switching cost considered
[63]	MAB	✓	✓(but not controlled)
[64]	MAB	✓	✗
[65]	MAB	✗	✓
[67]	MAB	✗	✗
[66]	MAB	✗	✗
[68]	Contextual MAB	✗	✓
[69]	MDP + Q-learning	✗	✓
[70]	MDP + Double deep Q-network	✗	✓

4.2 System model

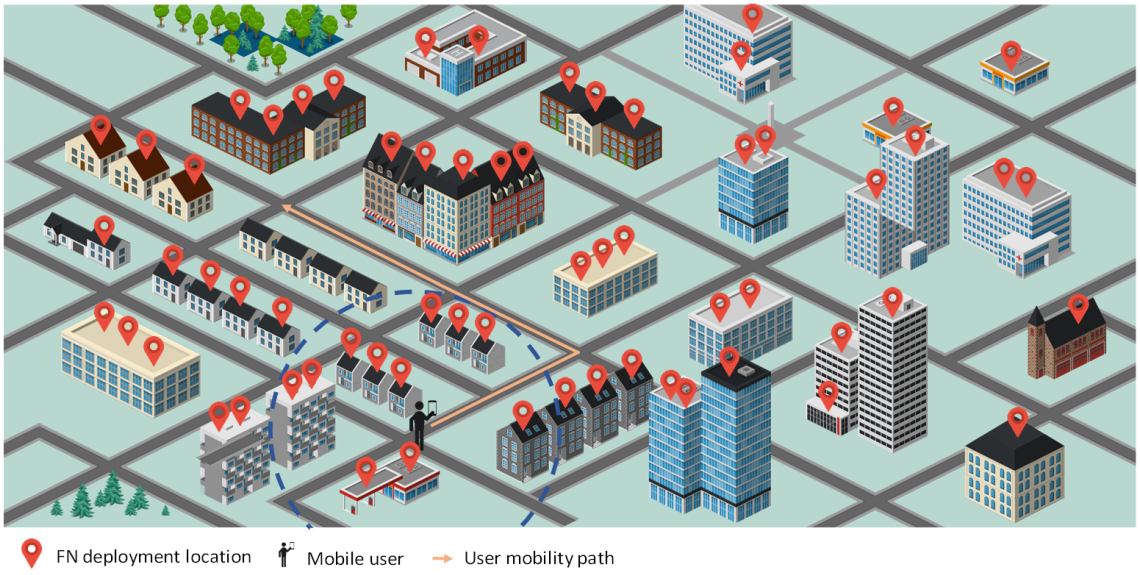


Figure 4.1: FN deployment scenario

Fig. 4.1 illustrates the scenario considered in this chapter, which consists in a high-density deployment of fog nodes in an urban environment. A representative user moves in this environment while executing an application comprised of a sequence of independent computation-intensive tasks $\{T_k\}_{k=1}^{K}$. Those tasks will be sent to one of the nearby fog

nodes for processing. We let $\mathcal{A}_k$ be the set of FNs available at the time at which task T_k was generated. We denote an FN present in $\mathcal{A}_k$ by a_{i_k}, where i_k is its index in the set. Clearly, the user mobility causes the set of detected FNs to change periodically, e.g. every Δ seconds.

To model a task T_k, we adopt a commonly-used model[71] where each task is characterized by a tuple (α_k, ω_k, β_k). Specifically, α_k corresponds to the input data size, ω_k denotes the task's computation intensity in terms of the number of CPU cycles required per input bit and β_k is the output data size.

The delay experienced by the user after sending a task to an FN is comprised of the following parts:

- Transmission delay d_{tx}^k: It represents how long it will take the user to transmit the task input data to the FN. It can be determined as follows:

$$d_{tx}^k = \frac{\alpha_k}{R_{i_k}} \tag{4.1}$$

  where R_{i_k} is the transmission rate of the wireless channel used by the user to communicate with the selected FN a_{i_k}. It is derived as follows:

$$R_{i_k} = W \log\left(1 + \frac{P_{tx} L}{\sigma^2 + I}\right) \tag{4.2}$$

  where W is the bandwidth of the wireless channel, P_{tx} is the transmit power of the user's device, σ^2 is the noise power, I is the interference power caused by other users connected to the same FN, and L is the path loss. We adopt the IEEE TGn task group channel model [72] to model the path loss, as follows:

$$L_{dB}(d_{i_k}) = \begin{cases} L_{FS}(d_{i_k}) \, , d_{i_k} \leq d_0 \\ L_{FS}(d_0) + 35 \log_{10}(\frac{d_{i_k}}{d_0}) \, , d_{i_k} > d_0 \end{cases} \tag{4.3}$$

  where d_{i_k} is the user-FN distance, L_{FS} is the free space path loss and d_0 is the reference distance. As we are considering pedestrian mobility in an urban environment, we specifically adopt model F in [72] where $d_0 = 30m$.

- Waiting delay d_w^k: It represents the time spent by the task T_k in the FN's task queue. This delay has a stochastic nature, since it may be affected by multiple factors, such as the rate at which different tasks arrive at the FN, the computation intensity of such tasks in addition to the FN's computation capacity. That is why, we model it as a random variable with parameters which are not initially known at the user side[1].

[1] In our simulations in Section 4.5, we use a normal distribution to model this waiting delay.

- Processing delay d_p^k: It is the time needed by the FN to process the task. It depends on the FN's CPU clock speed f_{i_k} as well as the task's intensity and its input size, as follows:

$$d_p^k = \frac{\alpha_k \omega_k}{f_{i_k}} \tag{4.4}$$

- Result transmission delay d_r^k: It is the time needed to return the computation results to the user. This delay is often omitted in related literature[63, 69], since the size of the result is usually considered negligible when compared to the input size.

- Switching delay d_{sw}^k: When the FN selected to execute a task T_k is different from the one selected to execute the previous task T_{k-1}, the user will incur a switching delay. Concretely, this delay would correspond to the delay needed to re-associate with the new FN, and the time needed by the latter to create a new virtual environment to host the user's tasks (e.g. a container).

$$d_{sw}^k = \begin{cases} C & a_k \neq a_{k-1} \\ 0 & a_k = a_{k-1} \end{cases} \tag{4.5}$$

The total delay associated with the task T_k can be then expressed as the sum of the aforementioned delay components, as follows:

$$d^k = d_{tx}^k + d_w^k + d_p^k + d_{sw}^k \tag{4.6}$$

Therefore, our goal is to find an FN selection strategy that minimizes the cumulative delay for the sequence of tasks $\{T_k\}_{k=1}^K$:

$$\min_{a_{i_k} \in \mathcal{A}_k} \sum_{k=1}^{K} d^k \tag{4.7}$$

Since our considered scenario is characterized with uncertainty, namely some parts of the delay are stochastic and are not available at the user side a-priori, we can model the FN selection problem using the framework of sequential decision-making under uncertainty. In particular, we will use (4.7) as an objective for a multi-armed bandits problem, as we describe in the following sections.

4.3 A review of multi-armed bandits

In their standard form, MAB problems are characterized with a fixed set of actions $\mathcal{A}$, where each action a has a reward distribution with an unknown mean μ_a. A decision-making agent sequentially selects actions, and upon each action selection, the agent receives

a reward (i.e. a sample from the reward distribution of the selected action). The question that arises is then the following: "how can a decision-making agent select actions in a way that maximizes its accumulated reward, without prior knowledge about the reward distributions?"

To answer this question, the agent has to go through the so-called exploration-exploitation dilemma. In fact, the agent has to explore some actions to create better estimates of their mean rewards. This may come at a cost of some selections resulting in receiving low reward values. On the other hand, the agent has to exploit the knowledge that it already acquired, and selects the action with the maximum mean reward estimate so far, in order to increase its accumulated reward.

A good MAB algorithm should then be able to appropriately balance exploration and exploitation. An efficient way to do this consists in the use of the Upper Confidence Bound (UCB)[73] algorithm. In UCB, each action a is characterized with a value as shown below:

$$i_a = \hat{\mu}(a) + \sqrt{\frac{\log(2t)}{N_t(a)}} \tag{4.8}$$

It is comprised of the current mean reward estimate $\hat{\mu}(a)$ and a confidence bound term (also referred to as the *exploration bonus*), where $N_t(a)$ is the number of times in which action a was selected up to time t. The latter ensures that the more an action is selected, the more certainty the agent gains regarding its reward, which will result in a more refined bound for that action. At the same time, when an action is not selected often, its exploration bonus will increase such that the agent will be forced to select it.

The performance of bandit algorithms is usually measured in terms of regret[2]. Intuitively, the regret measures the difference between the rewards accumulated by an oracle having exact prior knowledge about the rewards of each action and the rewards accumulated by the agent's action selection strategy. The regret accumulated until timeslot n can then be defined as follows:

$$\mathbb{E}[R_n] = \mathbb{E}[\sum_{i=1}^{n}(\mu^* - \mu_i)] \tag{4.9}$$

where μ^* is the *true* mean reward of the action selected by the oracle and μ_i is the mean reward of the action selected at time i.

[2]Also referred to as sampling regret.

4.3.1 Differences of the considered scenario with the standard MAB setting

Based on the aforementioned description, our FN selection problem can be formulated as a MAB problem with the following components:

- Agent: It is a software agent present in the user's device and managing the task of FN selection.
- Action: The fog node to send the task to.
- Rewards: Since the quality of the different fog nodes is determined based on their delay, we use a delay minimization formulation, instead of the standard reward maximization.

However, standard MAB algorithms such as UCB need to be tailored to our FN selection problem, since it has multiple differences with the standard MAB problem, as explained in the following points:

- **Varying action set**: While in the standard MAB setting, the set of actions is fixed, in our scenario the set of FNs dynamically changes as the user moves from one area to another. As a result, there is no single best action. Instead, in each ***epoch*** where the set of FNs remains the same, there may be a different optimal action. In the bandit literature, this problem is usually referred to as sleeping bandits[74], mortal bandits[75] or volatile bandits[76].
- **Short action lifespan along with a high number of actions**: Usually in bandit problems, the agent has a long-enough decision-making horizon to learn the best action. However, in our case, in each epoch the agent may be faced with a high number of FNs, given that we consider a high-density FN deployment scenario. At the same time, FNs have a short lifespan, i.e. the user remains within their range for a short amount of time. As a result, it would be challenging for the user to make the most of this short time in order to find the optimal FN to process its tasks.
- **Presence of switching costs**: As introduced in our delay model in Section 4.2, a switching cost is incurred whenever a different FN is selected in two consecutive decision slots. Ideally, the accumulated switching costs should be minimized, otherwise, they will have a negative impact on the overall performance. At the same time, if no switching is performed, the user may be stuck with a suboptimal FN. The accumulation of switching costs results in the so-called switching regret[77], defined as:

$$R_{sw}(n) = C \sum_{i=2}^{n} \mathbb{1}\{a_i \neq a_{i-1}\} \tag{4.10}$$

where C is a given switching cost.

In the following, we present different approaches that attempt to address the aforementioned concerns.

4.4 Proposed approaches

4.4.1 Limited Exploration for FN selection (LimExp)

Our first contribution mainly focuses on the fact that we have a high number of FNs, each having a limited lifetime.

Since this number of FNs is high, it is very likely that many of them will have similar delay performances. Therefore, omitting some of them from the selection process is not likely to cause a considerable loss in the overall performance. As a result, a natural solution that we adopt in this case is to limit the exploration process to a smaller subset of FNs of size N. This has an additional advantage of reducing the frequency of switches from one FN to the other during initial exploration.

This idea of using a subset of actions instead of the whole action set can be found in the bandit variant called infinitely many-armed bandits[78], where the decision maker is faced with a high number of action choices that can not even be explored once within the decision making horizon. Then, similar to [78], we use this subset approach in a variant of UCB, which we call LimExp. The resulting pseudocode is depicted in Alg. 2.

More in detail, the list of FNs detected in the vicinity of the user is retrieved in Line 3, if the current timeslot k corresponds to the beginning of an epoch. $\mathcal{B}$ denotes the set of the unexplored FNs within this set, i.e. the FNs which have never been selected before (L4). If this set is not empty, we check whether its size exceeds the considered subset size (L6-7). If not, all elements of the set are considered as exploration candidates (L8). Otherwise, a subset of size N is sampled from $\mathcal{B}$ (L10). Then, an explicit random exploration phase will occur, if there are FNs to be explored (L12-14). Otherwise, the FN will be chosen according to UCB (L16)[3].

Whenever an FN is selected to process the user's task, the total delay will be observed (L18). Then, statistics such as the sum of the delays observed so far for the selected FN and the number of times in which this FN has been selected are updated (L19-20). Note that when updating the delay sum $z(a_k)$, the switching cost is subtracted. This is

[3]Note the $-$ sign in the UCB calculation in Line 16, as opposed to the $+$ found in the definition in Eq. 4.8. This is because our problem targets delay minimization instead of the standard reward maximization setup.

because this cost is caused by the switching process and is not a characteristic of the FN's delay distribution. In addition, the resulting delay is normalized between 0 and 1, to be compliant with the ranges expected by UCB.

Algorithm 2: LimExp

input: N: The size of the considered subset of FNs

for $k \leftarrow 1$ **to** K **do**
  if *k is a timeslot in which the FN set changes* **then**
    retrieve set of visible FNs $\mathcal{A}_k$
    $\mathcal{B} \leftarrow get\_all\_unexplored(\mathcal{A}_k)$
    $exploration\_candidates \leftarrow \{\}$
    if $|\mathcal{B}| > 0$ **then**
      if $|\mathcal{B}| < N$ **then**
        $exploration\_candidates \leftarrow \mathcal{B}$
      else
        $exploration\_candidates \leftarrow sample(N, \mathcal{B})$
      end
  if $|exploration\_candidates| > 0$ **then**
    $a_k = random(exploration\_candidates)$
    $exploration\_candidates \leftarrow exploration\_candidates \setminus \{a_k\}$
  else
    $a_k = \arg\min_{a \in A_k}(\frac{z(a)}{n(a)} - \sqrt{\frac{log(2k)}{n(a)}})$
  end
  Observe delay $d(a_k)$
  $z(a_k) \leftarrow z(a_k) + normalize(d(a_k) - d_{sw}(a_k))$
  $n(a_k) \leftarrow n(a_k) + 1$
end

4.4.2 Block-based FN Selection (BFS)

Our second contribution is specifically tailored towards the problem of the presence of switching costs. More specifically, in order to avoid excessive switchings, an intuitive solution would be to maintain the same FN selection during an entire block of decision slots. This approach has been first proposed in [77] and was called the "Block Allocation Scheme" (BAS).

Therefore, based on this idea, we develop a Block-based FN Selection (BFS) scheme, where a combination of BAS and UCB are used. The pseudocode of BFS is summarized in Alg. 3.

Algorithm 3: BFS

1 $b \leftarrow 1$
2 remaining slots to use the current block size $r \leftarrow 0$
3 **for** $k \leftarrow 1$ **to** K **do**
4 **if** *k is a timeslot in which the FN set changes* **then**
5 retrieve set of visible FNs $\mathcal{A}_k$
6 calculate L according to Eq. 4.11
7 **if** $r = 0$ **then**
8 $a_{min} \leftarrow \arg\min_{a_i \in \mathcal{A}_k} \frac{z(a_i)}{n(a_i)} - \sqrt{\frac{\log(2k)}{n(a_i)}}$
9 $nb_{used} \leftarrow nb_{used} + 1$
10 **if** $nb_{used} > L$ **then**
11 $b \leftarrow b + 1$
12 $nb_{used} \leftarrow 0$
13 $r \leftarrow b$
14 $a_k \leftarrow a_{min}$
15 Observe delay $d(a_k)$
16 $z(a_k) \leftarrow z(a_k) + normalize(d(a_k) - d_{sw})$
17 $n(a_k) \leftarrow n(a_k) + 1$
18 $r \leftarrow r - 1$
19 **end**

The block size is initially set to $b = 1$ (L1). Then, in each decision slot k, if k is a decision slot in which the set of FNs changes (i.e. it is the beginning of an epoch[4]), the new set of FNs is retrieved (L5). Following that, in Line 6, the variable L representing the number of times in which the current block size should be kept the same is calculated as suggested in [77]:

$$L = \left\lceil \frac{2^{b^2} - 2^{(b-1)^2}}{b} \right\rceil . |\mathcal{A}_k| \tag{4.11}$$

where $|\mathcal{A}_k|$ is the cardinality of the current set of FNs. This allows the block size b to increase gradually, instead of being set in a static manner.

The next step consists in checking the value of r, which determines the number of the remaining time slots to use the current block size. If $r = 0$, the FN to be selected during the next block of decision slots, denoted as a_{min}, is determined based on UCB (L8). Additionally, the variable nb_{used} counting the number of times in which the current block

[4]Recall that an epoch is the set of consecutive decision slots where the set of FNs remains the same. As a result, the set of FNs changes every epoch.

size has been used is incremented (L9). If its value exceeds the value of L determined previously, then a new block size should be used, by incrementing the old value b by one (L10-11). r is also set to the new value of b (L13).

Finally, the following steps occur: the FN a_{min} is selected in the k^{th} decision slot, its delay is observed, the statistics $z(a_k)$ and $n(a_k)$ are updated and the value of r is decremented by one (L14-18).

4.4.3 Adaptive Greedy FN Selection (AGFS)

As it can be noted, both previous approaches are based on two variations of the UCB algorithm presented in Section 4.3. In fact, while LimExp can reduce the number of initial explorations by limiting the number of the considered FNs, subsequent explorations due to the exploration bonus in UCB are still maintained. The result is that an increased number of switchings are likely to occur within the short duration in which a given set of FNs is available to the user. Consequently, the user will not have enough time to remain connected to the best FN. On the other hand, in BFS, when an FN is selected because its exploration bonus is high while in reality it is a suboptimal FN, the block-based structure of BFS forces the user to remain connected to it for the entire block. This could result in a performance degradation for the user.

Therefore, with these considerations in mind, we also investigate a non UCB-based FN selection strategy. The resulting solution is based on the adaptive greedy approach proposed in [75] to deal with the specific case of mortal multi-armed bandits. In this context, since each action has a stochastic lifetime after which it is no longer available, the authors in [75] state that it suffices to select a reasonably good action, instead of selecting all actions indefinitely, as done in UCB. When applied to our scenario, this approach brings an additional advantage that was not envisioned in the original paper, which is the reduction of the accumulated switching costs.

Alg. 4 depicts our proposed Adaptive Greedy FN Selection (AGFS) scheme. In each decision slot k, the set of visible FNs is retrieved, if k corresponds to the beginning of an epoch (L2-3). Then, the FN having the minimal average delay estimate, denoted as a_{min}, is determined (L4). Let p_{min} denote its average delay, which is normalized in the range [0,1]. Then, with probability $min(1, c.(1-p_{min}))$, the FN to be selected in the k^{th} decision slot will be selected in a greedy manner (L7-8). Otherwise, an explicit exploration step will be performed by selecting an FN uniformly at random from the set $\mathcal{A}_k$ (L10).

More in detail, in the greedy selection step, if $c.(1-p_{min}) \geq 1$, then the delay performance of the FN will be considered good enough and it will be selected with probability 1. Otherwise, it will be selected with a smaller probability $c.(1-p_{min})$. Here, the constant c controls the threshold for considering whether an FN is sufficiently good or not based on its normalized mean delay estimate. When c is large, FNs with high delay estimates are

Algorithm 4: AGFS

1 **for** $k \leftarrow 1$ **to** K **do**
2 **if** *k is a timeslot in which the FN set changes* **then**
3 retrieve set of visible FNs $\mathcal{A}_k$
4 $a_{min} \leftarrow \arg\min_{a_i \in \mathcal{A}_k} \frac{z(a_i)}{n(a_i)}$
5 $p_{min} \leftarrow \frac{z(a_{min})}{n(a_{min})}$
6 draw a random number n_r
7 **if** $n_r \leq \min(1, c.(1 - p_{min}))$ **then**
8 $a_k \leftarrow a_{min}$
9 **else**
10 $a_k \leftarrow uniform(\mathcal{A}_k)$
11 **end**
12 Observe delay $d(a_k)$
13 $z(a_k) \leftarrow z(a_k) + normalize(d(a_k) - d_{sw})$
14 $n(a_k) \leftarrow n(a_k) + 1$
15 **end**

considered as good enough and are exploited with probability 1. This will be disadvantageous in situations where an FN with a mean delay estimate ≈ 0 exists. This situation does not occur in our scenario, where a value of $c = 2$ was adopted and led to the best results.

Similar to our previous approaches, after each FN selection, the delay is observed and the statistics for the chosen FN are updated (L12-14).

4.5 Evaluation results

In this section, we present the evaluation results corresponding to the three aforementioned algorithms. We start by explaining the mobility data collection approach. Then, the simulation setup and the obtained results are described.

4.5.1 Mobility data collection

In order to generate realistic FN availabilities that change as a result of the user mobility, we collected a dataset containing the set of Wi-Fi access points (APs) detected by a smartphone, in the city of Vilanova i La Geltrù. The APs would then represent the FNs in our scenario, according to different densities, as we explain later.

The data collection was performed at a walking speed and it lasted for a duration of $\sim$

30 minutes. This duration is typical for an application that would require fog computing capabilities, such as Pokémon Go[79]. The Wi-Fi scans allowing to detect the APs were performed every 30s, as recommended in the Android documentation[5].

Using the obtained mobility trace, two FN density scenarios were created, based on two different subsets of the data:

- High density: In this case, we take all APs having an SSID indicating a *public* city facility offering a free Wi-Fi network. In addition, we consider 50% of the APs having an SSID indicating a *private* Wi-Fi network (such as a home or an office network). These SSIDs usually start with an internet service provider name, followed by an ID. This case could correspond to the scenario where all public city facilities are offering FN capabilities, whereas only 50% of private APs have been upgraded by their owners as FNs. The resulting FN distribution is depicted in Fig. 4.2.

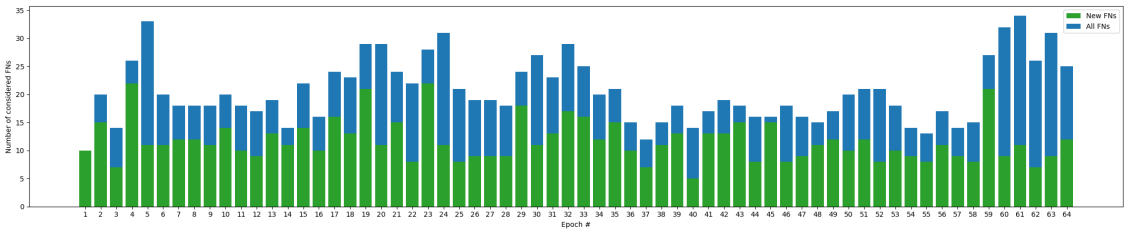


Figure 4.2: High density

- Ultra high density: This scenario differs from the previous case in that 75% of private Wi-Fi APs are considered as FNs. Fig. 4.3 shows the resulting distribution in the considered mobility duration.

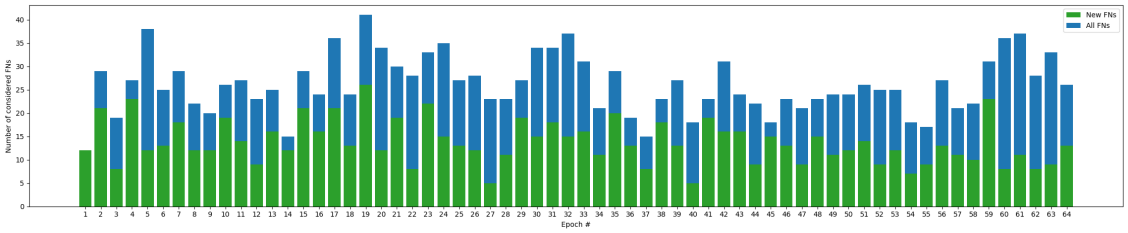


Figure 4.3: Ultra high density

[5]https://developer.android.com/guide/topics/connectivity/wifi-scan

4.5.2 Simulation setup

The simulations were conducted in Python using the parameters defined in Table 4.2 and the delay model introduced in Section 4.2. In particular, the waiting delay d_w is drawn from a different normal distribution for each FN. The mean μ of this normal distribution is chosen uniformly at random within the range $[0, 1]s$ and its standard deviation σ is chosen from the set $\{0.1, 0.2, 0.3, 0.4\}$. The switching cost C is $50\,ms$ unless otherwise specified. For the LimExp algorithm, the subset size, determined after comparing different values, was set to $N = 4$. Finally, the total number of considered timeslots is $K = 3840$, comprised of 64 epochs each having 60 timeslots.

Table 4.2: Simulation parameters

	Parameter	Value
Task characteristics	Input data size α_k	$1\,Mbit$
	Computation intensity ω_k	2640 cycles per input bit
Transmission delay calculation	Channel bandwidth W	$20\,Mhz$
	User - FN distance d_{i_k}	Uniformly at random in $\{10, 15, 20, 25, 30, 35, 40\}m$
	Transmit power of the user's device P_{tx}	$0.5\,W$
	Noise power σ^2	$2.10^{-13}\,W$
FN characteristics	CPU clock speed f_{i_k}	Uniformly at random in $\{2, 3, 4.5\}\,GHz$
	Waiting delay d_w (in s)	$\mathcal{N}(\mu, \sigma^2)$, where $\mu = \mathcal{U}(0, 1)$ and $\sigma \in \{0.1, 0.2, 0.3, 0.4\}$
Other parameters	Switching cost C	$\{50, 100, 200\}\,ms$
	c (in Alg. 4)	2
	Number of epochs where the set of FNs changes	64
	Number of timeslots in each epoch	60
	Total number of timeslots K	$64 \times 60 = 3840$

We compare the proposed algorithms LimExp, BFS and AGFS to the following algorithms:

- Auer: It is a UCB-based algorithm that has been proposed in the context of sleeping bandits[74]. In a first step, it explores each newly-appeared FN, then it selects FNs according to UCB. In this case, both initial explorations and the ones resulting from UCB will be harmful, since we consider a high number of FNs.

- VUCB: This algorithm has been proposed in the context of volatile multi-armed bandits[76]. It also selects actions based on UCB. However, it addresses action volatility by including the appearance time of each action in the confidence bound. We note that VUCB has been used in the related work[63].

- Oracle: This algorithm refers to a hypothetical scenario where prior knowledge about the full system information (i.e. information about the delay distributions of the FNs in our case) is available to the decision maker. Therefore, the oracle will select the FN that will result in the lowest delay in each decision slot. A switch from one FN to a better one will only occur between two different epochs.

The results presented next are the average of 50 different runs. Unless otherwise specified, the following results refer to the high FN density scenario.

4.5.3 Obtained results

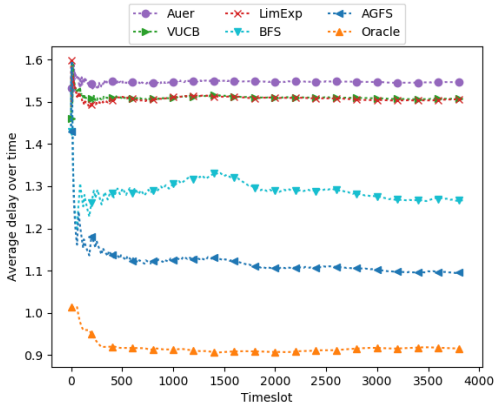
4.5.3.1 Overall evaluation

Fig. 4.4a plots the average cumulative delay (including the delays accumulated due to switchings). As it can be seen, at the beginning, LimExp brings a slight improvement over VUCB, however, in the remaining decisions, the performances of both algorithms are nearly the same. When we consider BFS, it can be seen that it allows to reduce the average delay compared to Auer, VUCB and LimExp. This is due to its block-based structure that reduces the frequency of switchings from one FN to the other. AGFS reduces the average cumulative delay even further, given that it settles on a sufficiently-good FN and avoids unnecessary explorations.

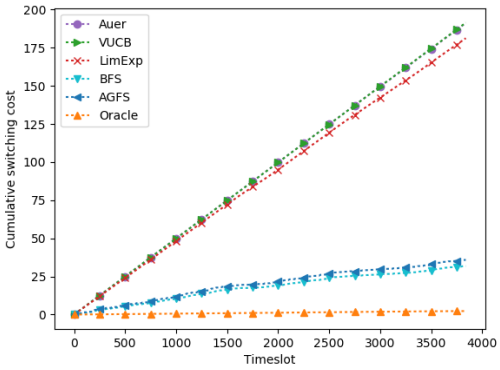
Fig. 4.4b shows the evolution of the cumulative switching costs over time. As it can be seen, Auer, VUCB and LimExp accumulate the highest costs. This is because they are based on UCB variants that do not appropriately deal with the presence of switching costs. In contrast, BFS and AGFS are able to significantly reduce these costs. More specifically, BFS outperforms AGFS, even though it is based on UCB. This is because it forces the same FN to be selected for multiple consecutive timeslots, thus preventing excessive switchings. We also note that the oracle approach accumulates negligible switching costs, since we do not have a unique optimal FN. Instead, there may be a different optimal FN in each epoch, which causes occasional switches to occur between two consecutive epochs.

Fig. 4.4c shows that the results obtained by each approach in terms of total regret are consistent with the trend observed for the average cumulative delay in Fig. 4.4a. More specifically, compared to the oracle, AGFS achieves the lowest regret, which is followed by BFS, while Auer, VUCB and Limexp achieve a high total regret.

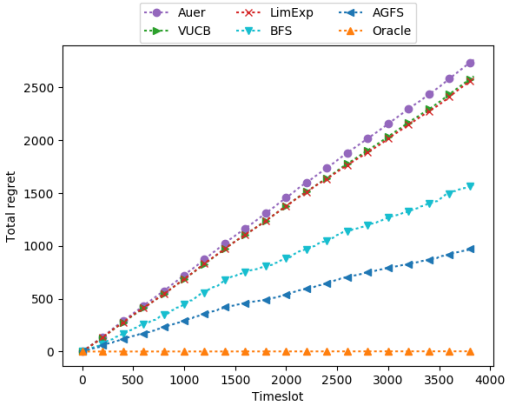
Fig. 4.4d illustrates the percentage of the time that the user spent connected to the optimal FN. As it can be noted, LimExp spends roughly the same amount of time connected to the optimal FN as VUCB. It is outperformed by BFS though, since its block-based structure often results in less time spent on explorations and more time spent connected to an optimal FN during a whole block of timeslots. However, in some situations, when using BFS, the user may be connected to a suboptimal FN for the whole block, due to insufficient explorations. This results in a lower percentage compared to AGFS, which obtains the highest percentage, given its inherent greedy behaviour.



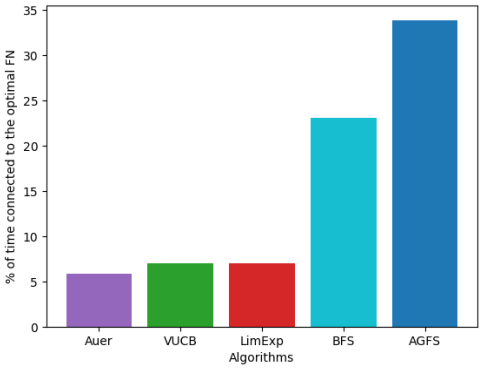
(a) Average cumulative delay



(b) Cumulative switching cost



(c) Total regret



(d) Ratio of simulation time where the user has selected the optimal FN

Figure 4.4: Obtained results

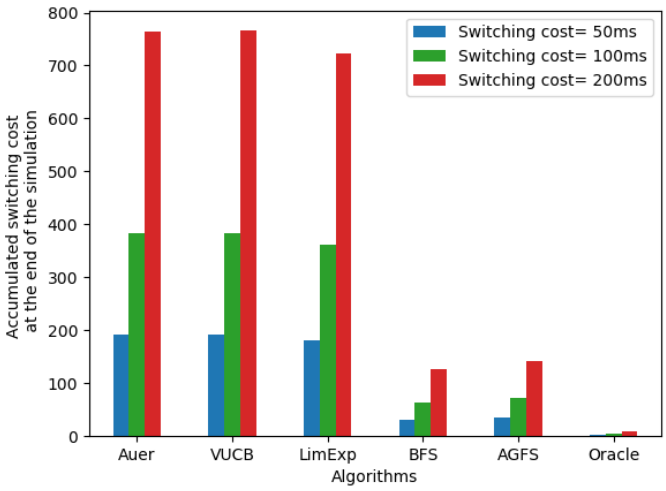


Figure 4.5: Accumulated switching cost

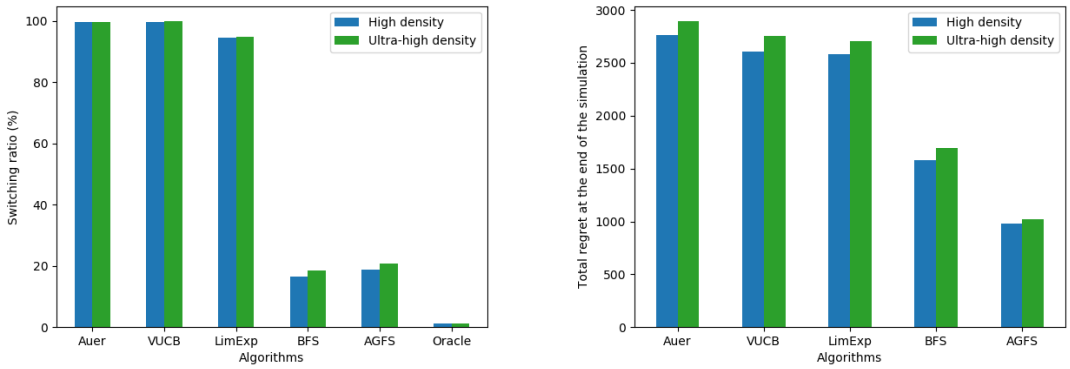
4.5.3.2 Impact of the switching cost

In Fig. 4.5, we evaluate the effect of using different values of the switching cost on the amount of switching costs accumulated by the considered approaches at the end of the simulation.

We first note that LimExp is able to obtain a slight reduction in the accumulated switching costs compared to Auer and VUCB, especially for high values of switching costs. But this slight reduction suggests that limiting the number of the considered FNs does not have a considerable advantage, as long as subsequent UCB-related explorations are maintained. In contrast, BFS and AGFS significantly reduce the amount of accumulated switching costs, with BFS having a slight advantage over AGFS. Clearly, such reduced switching costs will result in a better quality of service for the user.

4.5.3.3 Impact of the FN density

In Fig. 4.6a, we evaluate the different algorithms in terms of their switching ratios for the two considered FN densities. This switching ratio denotes the number of timeslots where a switch has been performed, divided by the total number of timeslots. We note that for both Auer and VUCB, a switch occurs in almost all timeslots, regardless of the FN density. LimExp is able to slightly reduce this ratio by a considering a smaller subset of FNs. However, the ratio it obtains is still high. This explains why there is almost no difference in the switching ratios of the aforementioned algorithms for the two FN density scenarios. However, with BFS and AGFS, we see that an increased FN density results in a



(a) Switching ratio

(b) Total regret

Figure 4.6: Comparison of the two FN deployment densities

slightly higher switching ratio. In terms of total regret, as shown in Fig. 4.6b, all algorithms obtain a higher regret for the ultra-high density scenario. This is due to the fact that an increased FN density, results in a higher number of FNs to explore. Specifically, some of those FNs may have a bad delay performance, which contributes to a higher accumulated regret.

4.6 Discussion

In this section, we provide additional notes about the applicability of our FN selection approaches:

- Adding a security cost: Although in the presented system model, the switching costs did not specifically account for security-related costs (e.g. (re)authentication), our model could be extended to include these costs. In fact, this is likely to lead to an increase in the total cost, which further motivates the need for controlling the switching behaviour.

- The case of a static environment: We note that our proposed AGFS approach, which obtained the best results in the simulations, is specifically tailored to dynamic environments where the FN availabilities change frequently. When the considered fog computing environment is instead static (when the user is stationary for instance), it would be recommended to adopt a UCB-based FN selection strategy like BFS, as it will allow to balance the exploration and exploitation processes efficiently while minimizing the switching costs.

- Multi-user case: Our analysis in this chapter has focused on the performance perceived from the perspective of a single representative user. The impact of collective selections from multiple users on the quality of service experienced individually by each one of them could also be seen as a possible extension.

4.7 Conclusion

In this chapter, we addressed the problem of FN selection, when a user is moving in an area having a dense deployment of FNs. This problem comes with an important challenge, which is to select an FN that will result in the lowest possible delay for the user's tasks, while minimizing the costs associated with switching from one FN to the other. To this end, we first proposed LimExp, an algorithm that limits the set of FNs considered for selection to a smaller subset. In fact, learning the best FN in the smaller subset will be faster, which would be beneficial since the user only remains a limited amount of time within the range of the encountered FNs. We also proposed BFS, a block-based FN selection approach that focuses on the minimization of the accumulated switching costs by maintaining the same FN selection during a block of consecutive decision slots. Finally, we presented a greedy alternative, AGFS, which selects the FN that was found to yield a sufficiently-good delay performance so far. The simulation results show that the latter approach brings significant improvements in terms of the average cumulative delay, compared to the first two.

It is worth-noting that all aforementioned approaches are specifically tailored towards a mobile user selecting static FNs. However, the problem of the appropriate selection of mobile FNs would also be worth-investigating, since vehicles having spare computational capacity can also act as fog nodes. With this in mind, the next chapter focuses on the vehicular FN selection problem, while addressing the challenges associated with the considered dynamic vehicular environment.

Chapter 5

Vehicular Fog node selection

In the previous chapter, the mobility of users and its impact on the fog node selection decisions has been discussed. However, in fog computing systems, mobility may not only be associated with end users, but it could also be a characteristic of fog nodes. In particular, vehicles having extra computing resources can offer those resources to process tasks assigned to them by a neraby roadside unit (RSU). In this case, such vehicles may be referred to as vehicular fog nodes (VFNs). An interesting question worth-investigating would then be "To which VFN each task should be assigned such that the task execution delays are minimized?". This question is mainly driven by the fact that VFNs are heterogeneous in terms of computation capacity and the tasks themselves may have different computation intensities. Additionally, finding the right VFN to select for a given task is not straightforward, since the vehicles' mobility creates a dynamic and an uncertain environment that leaves the RSU with a short time to learn the VFNs' delay performances. Therefore, to solve this problem, we propose a cooperation mechanism between two neighboring RSUs, where the RSU having sufficient knowledge about the performances of certain vehicles can transfer this knowledge to its neighbor, thus accelerating the task of finding the best assignment strategy for the latter. We show through simulations, that this approach is able to achieve considerable reductions in the cumulative task execution delay.

5.1 Related works

Recently, many works have emerged in the area of vehicle selection for the purposes of task assignment or virtual machine migration. Our overview of the related literature specifically focuses on the works where vehicles are on the move and can process tasks, as well. This is in contrast to works where the vehicles capable of processing tasks are static, e.g. in a parking lot. We make this distinction because the vehicles' mobility makes task assignment more challenging, compared to the static scenario. We provide an overview of these works in the following.

For VM migration, the relevant works can be found in the literature dealing with vehicular cloud computing[80]. For instance, the work in [81] proposes VM management schemes that take into account the vehicle's remaining time in the vehicular cloud (VC) in addition to its current workload. Based on this information, the next vehicle that

should host the VM is determined. The authors in [82] present a deep learning-based vehicular resource management scheme. The proposed scheme takes into account the vehicle's position, its speed, the number of vehicles ahead of it in the road segment and their average speed, and it returns the expected remaining time in the vehicular cloud. When a certain percentage of this remaining time elapses, a VM migration occurs from the vehicle which is about to leave to a vehicle which is expected to remain in the VC longer.

As for works dealing with vehicle selection for the purposes of task assignment, we distinguish works where the selection is performed in an individual manner by the task requestor itself, or where it is done by a different entity, usually a BS or an RSU. Among the relevant works in the first category, we cite [83], where a Task Vehicle (TaV), i.e. a vehicle having a task request, uses the MAB framework to select the serving vehicle (SeV) with the best delay performance. An enhancement to this approach is made in [84], where a combinatorial bandit formulation is used instead. This means that the TaV can select multiple SeVs simultaneously in order to increase the reliability of the task execution. Authors in [85] use a mortal bandit formulation, since the set of vehicular edge nodes for a given TaV is time-varying. More specifically, they use the computation capacities of the edge nodes as a contextual information in order to reduce the exploration space.

We now present a number of works where the selection is performed either at a BS deployed at the road or at a roadside unit. For example, the work in [86] uses the SARSA algorithm[87] for this purpose, taking into account information about the current task, the vehicle which executed the previous task in addition to the locations of the different vehicles in the road segment. A contextual bandit approach is instead used in [88], where the authors consider the task deadline and the TaV-SeV distance as context indicators that could guide the RSU in the SeV selection decisions.

Other approaches may not fit under the above-mentioned categories. For instance, the authors in [89] consider buses acting as cloudlets that can process requestors' tasks. In order to relay interdependent tasks from one cloudlet to another, the mobility information of both the requestor and the bus-based cloudlets is taken into account. A similar problem is addressed in [90] where the authors use a semi-Markov decision process formulation to describe the task assignment problem, where the information required for decision making consists in the location where the previous task was executed and the relative motion state between the requestor and the task execution cloudlet.

Finally, another research direction worth-mentioning relies on the use of a Semi-Markov Decision Process (SMDP) to derive resource allocation policies in a vehicular cloud environment[91, 92, 93, 94, 95]. These works consider a system where tasks with different priorities arrive to the vehicular cloud. Upon task arrival, the decision-making agent has to decide the number of resource units (RUs) to assign to it from the total RU pool in the vehicular cloud. However, this approach is different from our model since it

considers an abstract model of a resource pool and may allow assignment of tasks on RUs belonging to different vehicles, which may not be acceptable for certain task types.

To summarize, our work is most closely-related to the RSU-based task assignment approaches, where the RSU runs a learning algorithm allowing it to select the most suitable VFN for a given task, as described next.

5.2 System model

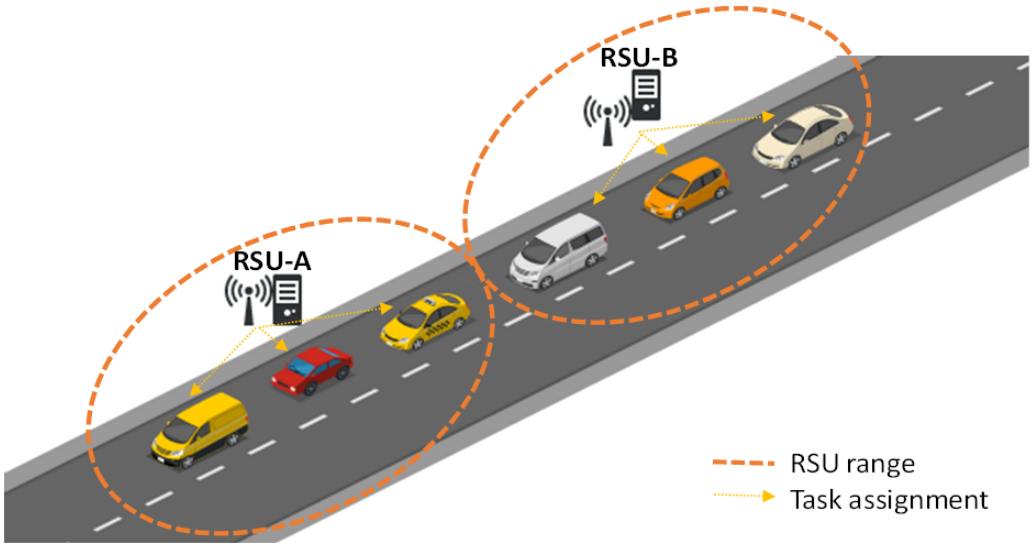


Figure 5.1: Vehicular fog computing scenario

Fig. 5.1 illustrates the considered vehicular fog computing scenario. As it can be seen, we consider a road segment where multiple RSUs may be deployed. At a given time slot k, each RSU has a set of vehicles $\mathcal{V}_k = \{v_{j_k}, j_k \in [1, |\mathcal{V}_k|]\}$ present within its range. $\mathcal{V}_k$ and as a result its cardinality $n_k = |\mathcal{V}_k|$ change as new vehicles enter the RSU's range and older ones leave it.

Each vehicle v_{j_k} is characterized with the amount of its available computation capacity f_{j_k} (in CPU cycles per second), which determines how fast it can execute tasks.

On the other hand, tasks are modeled in the same way as in Chapter 4, i.e. using a tuple $(\alpha_t, \omega_t, \beta_t)$ where α_t denotes the input data size of task t, ω_t is the computation intensity of the task (i.e. the number of required CPU cycles per input bit) and β_t is the output data size. We use the computation intensity of the tasks to distinguish them into three types $\mathcal{T} \in \{H, M, L\}$. ω_H denotes the computation intensity of a high-intensity task H. Similarly, ω_M and ω_L correspond to the intensities of task types M and L, respectively.

Taking into account the task's computation intensity and the vehicle's computation

capacity, the delay taken by a vehicle v_{j_k} to process a task t can be given by:

$$d_{exec}(v_{j_k}) = \frac{\alpha_t \omega_t}{f_{j_k}} \tag{5.1}$$

We focus on a representative RSU performing the assignment of computation tasks to the VFNs within its range. Its objective is to select VFNs in a way that results in a minimum cumulative delay for a given time horizon K:

$$\min \sum_{k=1}^{K} d_{exec}(v_{j_k}), s.t.\ v_{j_k} \in \mathcal{V}_k, \forall k \in \{1, \ldots, K\} \tag{5.2}$$

Since the RSU does not necessarily know the delay performances of the different VFNs prior to the task assignment step, we will adopt a multi-armed bandit approach to meet the objective defined in Eq. 5.2.

5.3 Proposed solution: Combining learning with advice

Instead of using the standard MAB setting as in Chapter 4, we use another variation called *contextual bandits*[96], where the agent observes some contextual information that can assist it in the decision-making process. So, unlike the standard bandit setting where the reward distributions depend only on actions, in the contextual bandit both actions and contexts determine the rewards that can be obtained by the agent.

More in detail, this framework can be used in our task assignment problem, by defining the following entities:

- **Agent**: The RSU.
- **Context**: It is the task type $\mathcal{T}$ (based on the task's computation intensity as described in Section 5.2).
- **Action**: The VFN to assign the task to.
- **Reward**: The VFN's task execution delay, which needs to be minimized.

According to these definitions, the RSU will then interact with the different VFNs by selecting them to execute tasks. The feedback it receives in the form of the task execution delay will allow it to assess the quality of each VFN. However, it is worth noting that the VFNs' mobility leaves the RSU with a very short amount of time to learn their performances under different contexts (i.e. task types).

To deal with this problem, the RSU could ask its neighbor for advice. To illustrate, in Fig. 5.1, since RSU-B precedes RSU-A (in terms of the order in which vehicles visit them

according to their movement direction), this allows it to learn the performances of some vehicles before RSU-A. As a result, it can share this knowledge with RSU-A to allow it to accelerate its learning process.

This idea can be found in the literature under different terms. For instance, we can cite the *teacher-student* paradigm [97] where an experienced learning agent can advise a student agent during early phases of its learning process. A similar concept can be found in the field of wireless networks in [98], where the authors present *docitive networks* where cognitive devices can learn faster, using advice from more knowledgeable devices in the network. We also cite [99], where the authors use apprenticeship learning to allow neighboring nodes in wireless mesh networks to share their spectrum decision policy with newly-joined nodes.

Since this advice-based approach has been found beneficial in the aforementioned domains, we adopt it in our contextual bandit-based task assignment strategy.

More in detail, we adopt a greedy algorithm as a basis for our proposed algorithm, i.e. it selects the VFN having the current best delay estimate for the considered task type. We then add the advising process to this algorithm, while also setting a limit to the budget that could be spent on getting advice, similar to the approach presented in [100]. This budget could be interpreted as a limit on the bandwidth consumption associated to the process of receiving advice. In our algorithm, it is implemented as the number of decision slots in which an RSU is allowed to receive advice. In what follows, we let T-RSU denote the teaching RSU, i.e. the one providing advice, while S-RSU refers to the student RSU, i.e. the RSU receiving advice.

Then, from the S-RSU's perspective, Alg. 5 will be executed to select the VFN to process a given task.

The first step in the algorithm is to retrieve the task type and the set of VFNs $\mathcal{V}_k$ which are present within the range of the S-RSU at decision slot k (L2-3). Then, S-RSU checks for the existence of unexplored VFNs for the considered task type $\mathcal{T}$ (L4). If there are indeed unexplored VFNs, then it can ask T-RSU for advice regarding the performances of those VFNs (L5). This step is feasible only when S-RSU has sufficient remaining budget for acquiring advice (L6). If this is not the case, S-RSU has to perform an exploration step on its own (L15).

If advice is available, i.e. the T-RSU returns the VFN having the minimum average delay for task type $\mathcal{T}$, the S-RSU will decrement its budget and the current task will be assigned to this returned VFN (L8-10). On the other hand, if the T-RSU could not provide advice, then S-RSU has to perform its exploration on its own (L12).

When the S-RSU does not have VFNs to explore in the current decision slot, it will perform a greedy selection of the VFN with the minimal average delay estimate for the current task type (L18).

Upon each selection of a VFN, the delay it took for executing the task will be observed (L20). This information will be used to update the sum of the delays observed for this VFN for this task type, in addition to the number of times it has been selected for this task type (L21-22). This will be useful to guide the RSU in its future decisions.

Algorithm 5: Algorithm

```
1  for k ← 1 to K do
2      T ← get_task_type(t)
3      V_k ← The set of V-FNs in range
4      U ← get_all_unexplored(V_k, T)
5      if |U| > 0 then
6          if budget > 0 then
7              advice ← check_for_advice(V_k)
8              budget ← budget − 1
9              if advice available then
10                 a_k ← advice
11             else
12                 a_k ← random(U)
13             end
14         else
15             a_k ← random(U)
16         end
17     else
18         a_k = arg min_{v∈V_k} (z_v(T) / n_v(T))
19     end
20     Observe delay d_exec(a_k)
21     z_{a_k}(T) ← z_{a_k}(T) + d_exec(a_k)
22     n_{a_k}(T) ← n_{a_k}(T) + 1
23 end
```

5.4 Results

In this section, we explain the setup used for our simulations, then we present the results obtained by our proposed approach combining the learning process with the advice mechanism.

5.4.1 Simulation setup

To simulate realistic mobility in the considered vehicular fog computing scenario, we used the vehicular mobility trace[101] corresponding to the Europarc roundabout in Creteil, France, shown in Fig. 5.2. This trace has been derived based on real traffic data observed in this roundabout during rush hours. Each row of data contains information about the moving vehicle, its position, its speed, the used lane and the timestamp. This information is updated periodically, every 1 second.

For our simulations, we filtered the original dataset both temporally and spatially. More specifically, we only used the data corresponding to the time interval 8AM - 8:10AM and the ones associated with vehicles' positions within the most frequently visited lane. We assume this lane contains one S-RSU and one T-RSU. The statistics regarding the number of vehicles observed within S-RSU for the considered duration are reported in Table 5.1.

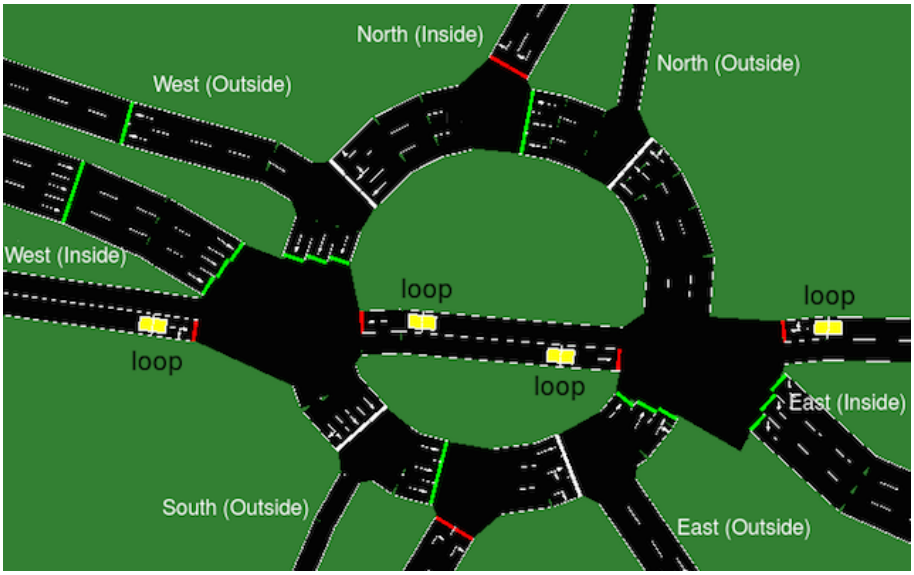


Figure 5.2: Europarc roundabout where the trace was collected

Table 5.1: Statistics of the considered data for S-RSU

Statistics	Value
Minimum number of vehicles at a given time	1
Average number of vehicles at a given time	≈ 8
Maximum number of vehicles at a given time	18

Simulations have been carried out in Python using the parameters shown in Table 5.2 and evaluations include comparisons of the following approaches:

Table 5.2: Simulation parameters

Parameter	Value
RSU range	150m
Task input size α_t	1Mb
Task computation intensity ω_t	$\omega_L = 250; \omega_M = 2500; \omega_H = 10000$ cycles/bit [102]
Available CPU frequency of V-FNs f_j	Uniformly at random in [1, 25] GHz
Number of assignment decisions per second	1 (i.e. $\approx$ 600 decisions in the considered 8AM-8:10AM duration)

- **Independent:** In this case, the RSU has to learn the VFNs' performances independently, without relying on the advice mechanism. This corresponds to Alg. 5 without the Lines 6 to 13.
- **Unlimited:** In this case, the S-RSU can use the advice from the T-RSU without budget limits.
- **Limited:** This corresponds to the proposed approach, i.e. the S-RSU can use advice from the T-RSU, up to a given budget limit. We specifically compare two values for this budget $b = \{100, 200\}$, i.e. the budget is available for the first 100, resp. 200 decision slots.
- **Oracle:** This corresponds to a hypothetical reference scenario where the RSU is provided with prior knowledge about the VFNs' performances. As a result, it will always select the one that will result in the lowest delay for the current task type.

The following results correspond to an average of 20 different random seeds.

5.4.2 Obtained results

Fig. 5.3 shows the accumulated delay over time for the different approaches. Since the S-RSU is learning from scratch in the independent approach, it results in the highest cumulative delay. This is caused by high delays observed in situations where the RSU explores unknown VFNs having bad delay performances. As it can be seen, this can be avoided by making use of the T-RSU's advice, thus resulting in lowest cumulative delays. In fact, the acquired advice will allow the S-RSU to focus its selections on the most profitable VFNs and avoid unnecessary explorations of unknown VFNs. Obviously, the higher the budget for receiving advice, the higher the gains that can be achieved in terms of delay reductions.

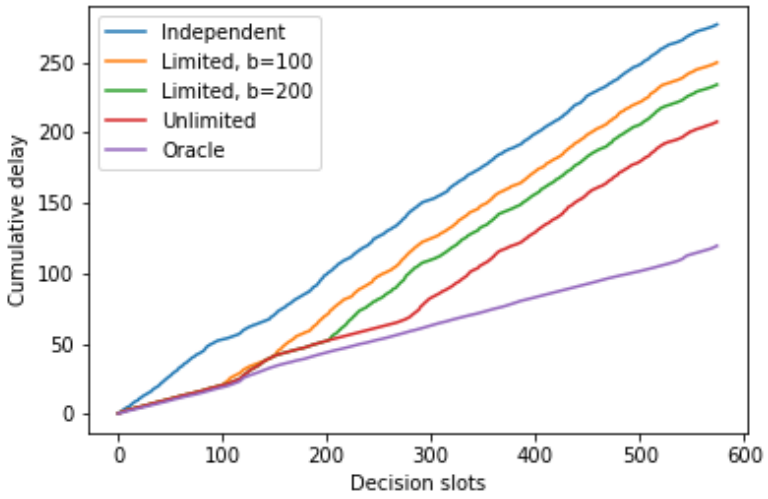


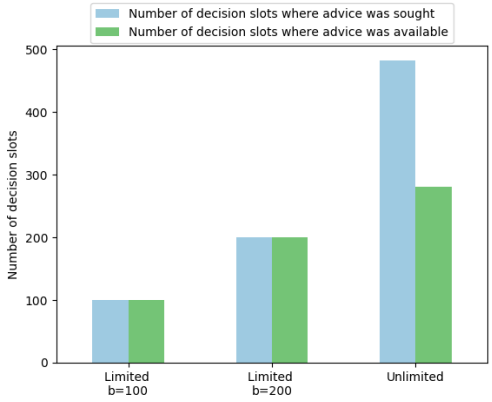
Figure 5.3: Cumulative delay

In order to assess the frequency at which the S-RSU requests advice, we plot in Fig. 5.4a the number of decision slots where advice was requested and when it was available. The figure shows that for both *limited* approaches, this number of decision slots corresponds to the value of the budget. This means that once the budget is entirely used, an S-RSU using the limited approach has to make its decisions on its own, even if this means that it has to try vehicles with unknown performances. On the contrary, when the S-RSU has an unlimited budget, it relies on the advice mechanism for $\approx 80\%$ of the time. This means that in $\approx 80\%$, it is in presence with new VFNs with unknown performances, which is an expected characteristic of the considered dynamic vehicular environment.

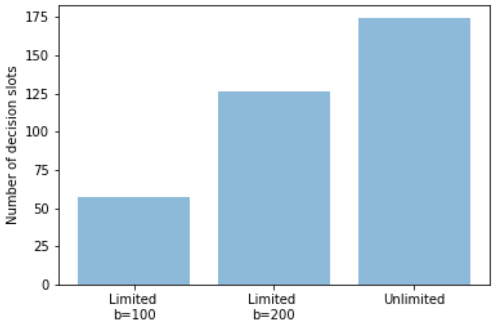
As for the number of decision slots where advice was available, Fig. 5.4a shows that for the *unlimited* approach, the S-RSU was only able to acquire advice for ≈ 280 decision slots, i.e. $\approx 58\%$ of the times in which it asked for it. It is worth noting that the unavailability of the advice occurs during the last decision slots. When we look at the characteristics of the vehicular data within that time frame, we observe that the S-RSU and the T-RSU have different vehicle dynamics. Since the two budget-limited approaches stop checking for the availability of the advice at the 100^{th} and 200^{th} decision slots, respectively, they were not affected by these different dynamics.

Next, we evaluate the efficiency of getting advice from the T-RSU. To this end, we plot in Fig. 5.4b the number of decision slots in which the VFN recommended by the T-RSU matches the one that would have been selected by the oracle. A comparison between Fig. 5.4b and Fig. 5.4a indicates that this is not always the case, i.e. the best VFN according to T-RSU is not the same as the one that S-RSU should have ideally selected. Such a

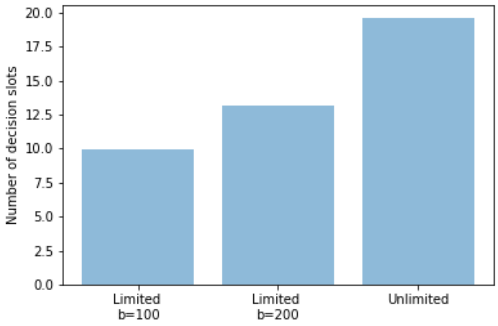
situation occurs because T-RSU is also continually learning the performances of the VFNs within its range. As a result, it is possible that it still did not learn the performances of certain VFNs for the considered task type before they leave its range and enter the range of S-RSU. The number of times in which this situation occurs is depicted in Fig. 5.4c. In this case, the T-RSU's advice will consist in the best VFN among the ones that it has already tried for the considered task type. By following its neighbor's advice in this case, the S-RSU misses the opportunity to find the best VFN independently. Therefore, a tradeoff emerges between always relying on the advice provided by the T-RSU to enable a faster learning process and learning in an independent manner which could result in situations where VFNs with bad delay performances are selected.



(a) Number of decision slots where advice was sought/available



(b) Number of decision slots where the advice matches the V-FN selected by the oracle



(c) Number of decision slots where T-RSU did not try the best V-FN for the current task

Figure 5.4: Advice characteristics

5.5 Discussion

In the previous section, we have shown how combining advice with the learning process can improve the learning performance compared to the case where learning is done independently. However, there are still further considerations that could be taken into account to achieve greater improvements. These considerations are described next:

- As it can be noted, in our proposed algorithm, advice is sought whenever the RSU is in presence of unexplored VFNs. Since the presence of unexplored VFNs will occur frequently as long as they enter the RSU's range, this means that the RSU will be relying on the T-RSU during most of the decision slots. As a result, it may miss the opportunity to discover high performing VFNs which are present only within its coverage area. Therefore, the RSU should adequately alternate the processes of learning independently and acquiring advice in order to achieve an optimal performance.
- Another aspect worth-considering is that in our proposal, the S-RSU is getting advice with regards to a specific task type (i.e. context). This means that both the S-RSU and T-RSU are aware of each other's contexts and have the same context space. If this is not the case, a suitable function allowing the translation of the teacher's context to the student's context (and vice-versa) should be designed, as suggested in [100].
- The main motivation to use external advice in this paper was that the exploration of each unknown VFN for the different task types could be wasteful since there is a limited number of interactions between the RSU and the vehicles because of mobility. Another direction for improvement could then consist in leveraging the order that exists in the context space (i.e. $L < M < H$, see Section 5.2). So, if a VFN has shown a poor performance for task type L, then its exploration for task types M and H could be avoided, since it would yield a poor performance, too.

5.6 Conclusion

In this chapter, we studied the problem of vehicular fog node selection for purposes of task assignment. We considered an RSU-based task assignment process, where the RSU has to interact with the different VFNs in order to learn their delay performances for different computation task types. However, since vehicles are constantly moving, the RSU does not have enough time to learn their delay performances from scratch. So, to deal with this problem, we proposed to combine learning with advising. This means that an RSU who has already acquired some information regarding the performances of certain VFNs can advise its neighbor VFN regarding the best VFN to select, therefore accelerating its learning process. We have demonstrated through simulations that this allows reductions

in the cumulative delay, compared to the case where advising is not used. Additionally, we have shown that the combined use of learning and advising may result in a tradeoff, as to whether the RSU should constantly request advice when in presence of unknown VFNs or whether it should fall back to learning independently especially when the teaching RSU is not able to provide useful advice. Ideally, the S-RSU should decide between both processes based on the T-RSU's confidence in the expected performance of the recommended VFN.

Chapter 6

Overall operation workflow

In the previous chapters, we described in detail different steps that may take place in a fog computing context, starting from the initial FN discovery process to the subsequent FN selection and also considering user mobility and vehicular FN mobility. In this chapter, we provide a holistic view on how the proposed contributions can be used in realistic fog computing operations, by presenting high-level execution workflows both at the user and the FN sides.

6.1 User-side operations

As shown in Fig 6.1, operations executed at the user's side would differ depending on whether its surrounding environment is characterized with a high or a low density of FNs. A potential way to determine the type of the surrounding environment would be to correlate the observed cellular context information with either type of environment.

Then, if the user is located in an area with a high FN density, then it makes sense to keep the scan enabled to retrieve the updated list of nearby FNs. Consequently, this list will be fed as an input to the AGFS algorithm presented in Chapter 4 to allow the user to select the appropriate FN, while minimizing the delay and the switching costs.

On the other hand, if the user is at a location where the FN deployment is sparse, i.e. there are locations with no FN coverage, then, as indicated in Chapter 3, the user will retrieve its current cellular footprint and provide it as an input to the Hidden Markov Model, in addition to the sequence of the preceding footprints. Based on this, the HMM will predict whether the user is present at a fog node location, in which case the scan will be enabled and the detected FN will be selected. However, if the prediction is that no FN is expected to be present, then the user will maintain the scan disabled to achieve energy savings.

6.2 FN-side operations

The FN-side operations depend on whether the FN is also an RSU or not, as shown in Fig. 6.2. The first step would be to start broadcasting the custom 802.11 beacons, as

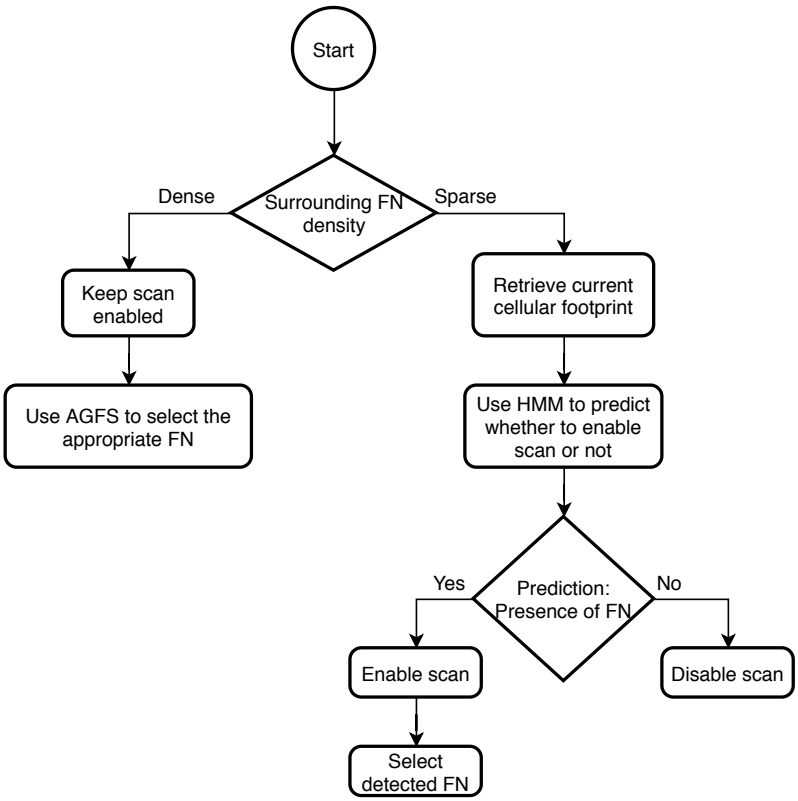


Figure 6.1: Execution workflow at the user side

described in Chapter 2. In the FN case, this will allow users to detect its presence, so that

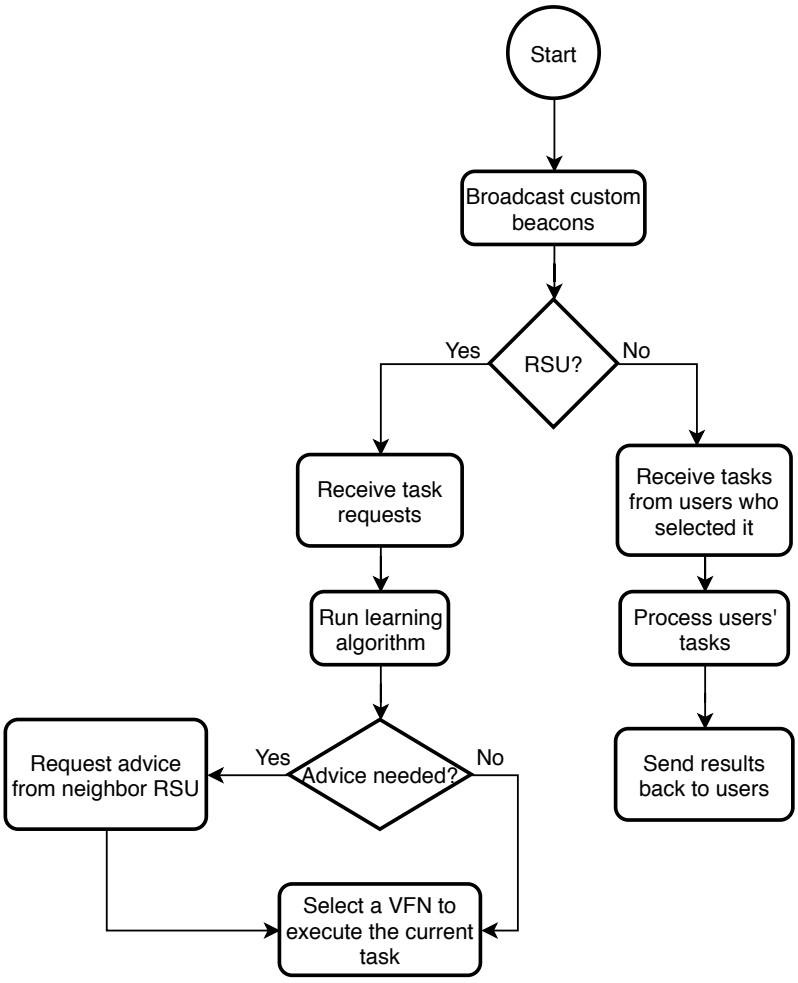


Figure 6.2: Execution workflow at the FN/RSU side

they can send it their tasks. On the other hand, when the FN is also an RSU, this will allow vehicles to know that it is involved in the vehicular fog computing system and that they can participate as candidates for the task assignment process. We envision the use of an additional 802.11 information element to indicate whether the FN is an RSU or not, to help its intended users in the discovery process. We note that since our discovery process uses the 802.11 protocol, it is compatible with the 802.11p standard[103] used in vehicular environments, which would facilitate the vehicle-RSU communications.

Then, the (non-RSU) FN will start receiving tasks from the users who selected it. As described in Chapter 4, such tasks might experience a waiting delay before being processed. Finally, after the processing step, the computation results will be sent back to

their respective users.

If the FN also has an RSU role, then it will receive task requests from users on the road or vehicles not having enough computation capacity. Then, based on the learning algorithm presented in Chapter 5, it will determine whether it needs to obtain additional advice to guide its VFN selection process. If that is the case, it will request this advice from its neighbor RSU. Depending on the availability of the advice, it may either end up choosing the VFN recommended by the neighbor or select a VFN in an independent manner.

6.3 Conclusion

This chapter has provided a description of the different steps that may be executed by a user or by a fog node. Although these steps have been presented at a high-level, additional steps may be added to the workflows to include more advanced features. Such features could include the addition of access control mechanisms to ensure that fog nodes are only accessed by authorized users. In addition, pricing schemes could be incorporated so that FNs, and particularly vehicular FNs, can be rewarded for their offered resources.

Chapter 7

Conclusions and future research directions

The fog computing paradigm has been attracting increasing attention in the past few years. This is mainly due to its ability to reduce latencies by leveraging computation nodes at the edge of the network, i.e. fog nodes. Inherent characteristics of fog computing include the mobility of users and resources, the dynamically-changing resource availabilities as well as the heterogeneity of resources, especially that they may be offered by multiple resource providers. This gives rise to the following question: "How to enable a proper use of the resources offered by the fog nodes given such characteristics of the fog computing environment?". The goal of this thesis is indeed to address this question by breaking it into two main questions. The first one is "How can a user become aware of the existence of nearby fog nodes in the first place?". Second, "How to appropriately select FNs to execute computation-intensive tasks, both when a user is on the move and when the FN is on the move?".

In the following, we summarize the main thesis contributions that have been developed in line with these questions. Then, we present possible areas to conduct additional research related to the topics of this thesis.

7.1 Contributions

Our first contribution is related to the question of the discovery of fog nodes. To answer this question, we proposed a Wi-Fi-based discovery solution, where a fog node can advertize its presence by means of customized 802.11 beacons. This allows user devices in the vicinity to scan for these custom beacons in order to become aware of the existence of the FN. The major advantage of this approach is that it is edge-based, i.e. it naturally alleviates the load of the core network. In addition, it does not require the user and the FN to be pre-associated to the same wireless network. It has also been validated in two projects, namely GUAU and mF2C[38] and it has been presented in the following two publications:

- Z. Rejiba, X. Masip-Bruin, A. Jurnet, E. Marin-Tordera, and G.-J. Ren, "F2C-Aware: Enabling Discovery in Wi-Fi-Powered Fog-to-Cloud (F2C)Systems," in 2018

6th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud), pp. 113–116, IEEE, 2018.

- Z. Rejiba, X. Masip-Bruin, and E. Marín-Tordera, "Analyzing the deployment challenges of beacon stuffing as a discovery enabler in fog-to-cloud systems," in 2018 European Conference on Networks and Communications (EuCNC), pp. 1–276, IEEE, 2018.

Our second contribution, is also related to the discovery process, however with a specific focus on the underlying scan performed on the user device's side. In fact, this scan would be unnecessary when the user is moving in areas with no fog node coverage. Therefore, disabling the scan in these situations will result in energy savings, especially since it is based on the use of wireless resources. On the other hand, the scan should be enabled in locations where FNs are present, in order not to miss the opportunity to use their offered resources. This creates a tradeoff of whether to scan or not to scan. To address this tradeoff, we first proposed an approach where the user can receive scan assistance information from previously-encountered FNs. This assistance information includes the distance remaining until another FN can be found and the Wi-Fi channel it uses to broadcast its beacons. This allows the user to only enable the scan when the new FN is reached, therefore reducing scan-related energy consumption. Although this first approach is promising, it is associated with certain restrictions in the user movement direction as well as the structure of the FN topology. This has led us to propose a different approach to overcome this limitation. More specifically, we proposed the use of the cellular footprint of a given location as an indicator of the presence or absence of an FN in that location. This is done by collecting historical cellular footprint data and feeding it into a classification algorithm that will predict whether an FN is present or not. We have shown that in this case, leveraging the sequence of the few past cellular observations, leads to better predictions.

The aforementioned approaches have been published in the two following papers:

- Z. Rejiba, X. Masip-Bruin, and E. Marín-Tordera, "A Beacon-assisted direction-aware scanning scheme for 802.11-based discovery in Fog-to-Cloud systems," in 2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), pp. 1–6, IEEE, 2018.

- Z. Rejiba, X. Masip-Bruin, and E. Marín-Tordera, "Towards a context-aware Wi-Fi-based Fog Node discovery scheme using cellular footprints," in 2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp. 1–6, IEEE, 2018.

Our third contribution rather focuses on a fog computing scenario with a high density of FNs, which could be the case in densely-populated areas of the city, with a high demand for fog computing resources. A natural question that arises in such a context is which FN

to select to process a user's task, such that the overall delay is minimal. In fact, since we consider a mobile user, the set of visible FNs dynamically changes, thus resulting in frequent changes of the selected FN. However, this change comes at a cost of a switching delay, which should also be kept minimal. We proposed three approaches to address this issue. The first one consists in limiting the set of the considered FNs to a smaller one, the second approach keeps the selected FN fixed for a block of consecutive decision slots, and the third approach greedily selects an FN with a good enough average delay and sticks with it as long as it is available. It is indeed this last approach that was found to yield the best results in terms of average task delay.

Publications related to this research problem are indicated below:

- Z. Rejiba, X. Masip-Bruin, and E. Marín-Tordera, "A survey on mobility-induced service migration in the fog, edge, and related computing paradigms,"ACM Computing Surveys (CSUR), vol. 52, no. 5, pp. 1–33, 2019.
- Z. Rejiba, X. Masip-Bruin, and E. Marín-Tordera, "A user-centric mobility management scheme for high-density fog computing deployments," in 2019 28th International Conference on Computer Communication and Networks (ICCCN), pp. 1–8, IEEE, 2019.
- Z. Rejiba, X. Masip-Bruin, and E. Marín-Tordera, "Towards user-centric,switching cost-aware fog node selection strategies," Future Generation Computer Systems. Submitted.

Our final contribution focuses on the problem of how to appropriately select an FN to execute a given task when this FN is on the move. Such mobile FNs are usually present within vehicles offering part of their computational resources and are therefore called vehicular FNs. Within this context, an RSU is usually responsible for collecting task requests from nearby vehicles or users and assigning them to the VFNs within its range. Given the uncertainty and the dynamic nature of the considered vehicular environment, approaches relying on learning in an online manner have started to emerge. Inspired by this, we also adopt a similar approach. However, instead of learning the delay performances of the VFNs from scratch, we propose to allow the RSU to leverage advice received from a neighbor RSU who has already observed the performances of these vehicles for different task types. This allows it to learn the best VFN for a given task faster, especially that it is dealing with a dynamic environment, where it can only have access to a specific set of vehicles for a limited amount of time. We have shown that when this advice is available, this can lead to benefits in terms of cumulative task execution delay.

The following publication is the result of this contribution:

- Z. Rejiba, X. Masip-Bruin, and E. Marín-Tordera, "Computation task assignment in vehicular fog computing: A learning approach via neighbor advice," in 18th IEEE

International Symposium on Network Computing and Applications (NCA 2019), pp. 1–5, IEEE, 2019.

7.2 Potential areas for future research

Overall, this thesis has contributed solutions to the problems of the discovery and selection of fog nodes, which are among the main aspects that need to be addressed in the context of fog computing. Building upon this, several possibilities exist to extend the scope of the studied research problems, as described next:

- Discovery in intermediate fog layers: While the Wi-Fi-based discovery solution proposed in this thesis is edge-centric and is based on physical proximity of users and fog nodes, it is specifically-tailored towards the lowest layers of the fog computing architecture. This brings the need to complement it with additional discovery mechanisms that could span higher fog layers, thus allowing discovery at a larger scale. This could be done for example by addressing the concerns associated with the related works dealing with discovery beyond the local area (Section 2.1.2).

- From single user FN selections to multi-user FN selections: Chapter 4 has focused on finding an appropriate FN selection strategy for a single representative user. Therefore, it would be interesting to investigate the consequences emerging from the situation where multiple nearby users adopt the same FN selection strategy. As this may trigger a tradeoff between maximizing individual utilities versus collective utilities, game theoretical approaches could be envisioned to study the multi-user scenario.

- Inter-dependent tasks: Both Chapter 4 and Chapter 5 have modeled computation tasks in an independent manner, i.e. no inter-dependencies between tasks were considered. This could be extended to a more complex scenario where such inter-dependencies exist, thus placing additional constraints on the selections of FNs or VFNs, which would even be more challenging given the inherent mobility characterizing users and VFNs, respectively.

- Caching and pre-instantiation of containers to achieve time savings: The FN selection problem could be studied in conjunction with the caching problem on the FN side. In fact, if the FN adopts an efficient caching strategy that stores copies of the computation results of the commonly-requested tasks, the processing time can be avoided and the user can observe reduced delays. Similarly, if the FN pre-instantiates a set of containers that are usually required to execute users' tasks, the instantiation time can be saved, which could be highly-beneficial when the user switches from one FN to another.

Bibliography

[1] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?," *Computer*, vol. 43, no. 4, pp. 51–56, 2010.

[2] A. V. Dastjerdi, H. Gupta, R. N. Calheiros, S. K. Ghosh, and R. Buyya, "Fog computing: Principles, architectures, and applications," in *Internet of things*, pp. 61–75, Elsevier, 2016.

[3] C. C. Byers, "Architectural imperatives for fog computing: Use cases, requirements, and architectural techniques for fog-enabled iot networks," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 14–20, 2017.

[4] 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*, pp. 13–16, 2012.

[5] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog computing: A platform for internet of things and analytics," in *Big data and internet of things: A roadmap for smart environments*, pp. 169–186, Springer, 2014.

[6] 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, 2016.

[7] X. Masip-Bruin, E. Marín-Tordera, G. Tashakor, A. Jukan, and G.-J. Ren, "Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud computing systems," *IEEE Wireless Communications*, vol. 23, no. 5, pp. 120–128, 2016.

[8] E. Marín-Tordera, X. Masip-Bruin, J. García-Almiñana, A. Jukan, G.-J. Ren, and J. Zhu, "Do we all really know what a fog node is? current trends towards an open definition," *Computer Communications*, vol. 109, pp. 117–130, 2017.

[9] 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, 2009.

[10] "emergence of micro datacenter (cloudlets/edges) for mobile computing." https://www.microsoft.com/en-us/research/wp-content/uploads/2016/11/Micro-Data-Centers-mDCs-for-Mobile-Computing-1.pdf. Accessed: 2020-04-22.

[11] "Multi-access edge computing (mec)." https://www.etsi.org/technologies-clusters/technologies/multi-access-edge-computing. Accessed: 2020-04-22.

[12] S. Wang, R. Urgaonkar, T. He, M. Zafer, K. Chan, and K. K. Leung, "Mobility-induced service migration in mobile micro-clouds," in *2014 IEEE military communications conference*, pp. 835–840, IEEE, 2014.

[13] M. Mukherjee, L. Shu, and D. Wang, "Survey of fog computing: Fundamental, network applications, and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 1826–1857, 2018.

[14] C. N. Ververidis and G. C. Polyzos, "Service discovery for mobile ad hoc networks: a survey of issues and techniques," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 3, pp. 30–45, 2008.

[15] B. K. Mohandas, A. Nayak, K. Naik, and N. Goel, "Absrp-a service discovery approach for vehicular ad hoc networks," in *2008 IEEE Asia-Pacific Services Computing Conference*, pp. 1590–1594, IEEE, 2008.

[16] W. Sun, Z. Yang, X. Zhang, and Y. Liu, "Energy-efficient neighbor discovery in mobile ad hoc and wireless sensor networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1448–1459, 2014.

[17] S. K. Datta, R. P. F. Da Costa, and C. Bonnet, "Resource discovery in internet of things: Current trends and future standardization aspects," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pp. 542–547, IEEE, 2015.

[18] P. Trunfio, D. Talia, H. Papadakis, P. Fragopoulou, M. Mordacchini, M. Pennanen, K. Popov, V. Vlassov, and S. Haridi, "Peer-to-peer resource discovery in grids: Models and systems," *Future Generation Computer Systems*, vol. 23, no. 7, pp. 864–878, 2007.

[19] L. Kapoor, S. Bawa, and A. Gupta, "Hierarchical chord-based resource discovery in intercloud environment," in *2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*, pp. 464–469, IEEE, 2013.

[20] G. Lewis, S. Echeverría, S. Simanta, B. Bradshaw, and J. Root, "Tactical cloudlets: Moving cloud computing to the edge," in *2014 IEEE Military Communications Conference*, pp. 1440–1446, IEEE, 2014.

[21] S. Cirani, G. Ferrari, N. Iotti, and M. Picone, "The iot hub: A fog node for seamless management of heterogeneous connected smart objects," in *2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking-Workshops (SECON Workshops)*, pp. 1–6, IEEE, 2015.

[22] K. Ha, *System infrastructure for mobile-cloud convergence*. PhD thesis, Carnegie Mellon University, 2016.

[23] S. Echeverría, G. A. Lewis, J. Root, and B. Bradshaw, "Cyber-foraging for improving survivability of mobile systems," in *MILCOM 2015-2015 IEEE Military Communications Conference*, pp. 1421–1426, IEEE, 2015.

[24] S. Chilukuri, S. Bollapragada, S. Kommineni, and K. Chakravarthy, "Raincloud-cloudlet selection for effective cyber foraging," in *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, IEEE, 2017.

[25] S. Soo, C. Chang, and S. N. Srirama, "Proactive service discovery in fog computing using mobile ad hoc social network in proximity," in *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 561–566, IEEE, 2016.

[26] E. A. S. Mendoza, A. F. da Conceição, A. H. M. Aliaga, and D. Vieira, "Pytos: A framework for mobile computation offloading in python," in *2015 11th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, pp. 262–269, IEEE, 2015.

[27] I. Murturi, C. Avasalcai, C. Tsigkanos, and S. Dustdar, "Edge-to-edge resource discovery using metadata replication," in *2019 IEEE 3rd International Conference on Fog and Edge Computing (ICFEC)*, pp. 1–6, IEEE, 2019.

[28] A. Zavodovski, N. Mohan, and J. Kangasharju, "edisco: Discovering edge nodes along the path," *arXiv preprint arXiv:1805.01725*, 2018.

[29] J. Gedeon, C. Meurisch, D. Bhat, M. Stein, L. Wang, and M. Mühlhäuser, "Router-based brokering for surrogate discovery in edge computing," in *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pp. 145–150, IEEE, 2017.

[30] B. Varghese, N. Wang, J. Li, and D. S. Nikolopoulos, "Edge-as-a-service: Towards distributed cloud architectures," *arXiv preprint arXiv:1710.10090*, 2017.

[31] A. Salem, T. Salonidis, N. Desai, and T. Nadeem, "Kinaara: Distributed discovery and allocation of mobile edge resources," in *2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pp. 153–161, IEEE, 2017.

[32] F. Baccelli, N. Khude, R. Laroia, J. Li, T. Richardson, S. Shakkottai, S. Tavildar, and X. Wu, "On the design of device-to-device autonomous discovery," in *2012 Fourth international conference on communication systems and networks (COMSNETS 2012)*, pp. 1–9, IEEE, 2012.

[33] E. Saurez, K. Hong, D. Lillethun, U. Ramachandran, and B. Ottenwälder, "Incremental deployment and migration of geo-distributed situation awareness applications in the fog," in *Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems*, pp. 258–269, 2016.

[34] R. Chandra, J. Padhye, L. Ravindranath, and A. Wolman, "Beacon-stuffing: Wi-fi without associations," in *Eighth IEEE Workshop on Mobile Computing Systems and Applications*, pp. 53–57, IEEE, 2007.

[35] A. Al-Akkad, L. Ramirez, A. Boden, D. Randall, and A. Zimmermann, "Help beacons: Design and evaluation of an ad-hoc lightweight sos system for smartphones," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1485–1494, 2014.

[36] S. Zehl, N. Karowski, A. Zubow, and A. Wolisz, "Lows: A complete open source solution for wi-fi beacon stuffing based location-based services," in *2016 9th IFIP Wireless and Mobile Networking Conference (WMNC)*, pp. 25–32, IEEE, 2016.

[37] B. Könings, F. Schaub, and M. Weber, "Prifi beacons: piggybacking privacy implications on wifi beacons," in *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*, pp. 83–86, 2013.

[38] "mf2c, towards an open, secure, decentralized and coordinated fog-to-cloud management ecosystem." www.mf2c-project.eu. Accessed: 2020-04-22.

[39] K. A. Nguyen, Z. Luo, and C. Watkins, "On the feasibility of using two mobile phones and wlan signal to detect co-location of two users for epidemic prediction," in *Progress in Location-Based Services 2014*, pp. 63–78, Springer, 2015.

[40] C. Xu, W. Jin, Y. Han, G. Zhao, and H. Tianfield, "Mp-sdwn: A novel multipath-supported software defined wireless network architecture," in *International Conference on Communicatins and Networking in China*, pp. 119–128, Springer, 2016.

[41] N. Brouwers, M. Zuniga, and K. Langendoen, "Incremental wi-fi scanning for energy-efficient localization," in *2014 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 156–162, IEEE, 2014.

[42] S. Jana and S. K. Kasera, "On fast and accurate detection of unauthorized wireless access points using clock skews," *IEEE transactions on Mobile Computing*, vol. 9, no. 3, pp. 449–462, 2009.

[43] K. Doppler, C. B. Ribeiro, and J. Kneckt, "On efficient discovery of next generation local area networks," in *2011 IEEE Wireless Communications and Networking Conference*, pp. 269–274, IEEE, 2011.

[44] C.-C. Tseng, K.-H. Chi, M.-D. Hsieh, and H.-H. Chang, "Location-based fast handoff for 802.11 networks," *IEEE Communications letters*, vol. 9, no. 4, pp. 304–306, 2005.

[45] S. Han, M. Kim, B. Lee, and S. Kang, "Directional handoff using geomagnetic sensor in indoor wlans," in *2012 IEEE International Conference on Pervasive Computing and Communications*, pp. 128–134, IEEE, 2012.

[46] T. Casey and G.-M. Muntean, "Scan-or-not-to-scan-balancing network selection accuracy and energy consumption," in *2015 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 1371–1376, IEEE, 2015.

[47] S. Lu, S. Shere, Y. Liu, and Y. Liu, "Device discovery and connection establishment approach using ad-hoc wi-fi for opportunistic networks," in *2011 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pp. 461–466, IEEE, 2011.

[48] R. R. Fontes, S. Afzal, S. H. Brito, M. A. Santos, and C. E. Rothenberg, "Mininet-wifi: Emulating software-defined wireless networks," in *2015 11th International Conference on Network and Service Management (CNSM)*, pp. 384–389, IEEE, 2015.

[49] E. Zeljkovic, R. Riggio, and S. Latré, "Exploiting distance information for transparent access point driven wi-fi handovers," in *2017 IEEE 18th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pp. 1–6, ieee, 2017.

[50] C.-S. Lin and S.-H. Hsu, "Efficient self-adjustment places of interest finding by eliminating dynamic access point beacons in public places," in *International Symposium on Wireless and pervasive Computing (ISWPC)*, pp. 1–6, IEEE, 2013.

[51] G. Aloi, G. Caliciuri, V. Loscrí, and P. Pace, "Accurate and energy-efficient localization system for smartphones: A feasible implementation," in *2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 3477–3481, IEEE, 2013.

[52] J. Jeong, J. Lee, Y. Kim, J. W. Lee, and S. Chong, "Impact of surrounding information on wi-fi sensing efficiency," in *2013 International Conference on ICT Convergence (ICTC)*, pp. 410–415, IEEE, 2013.

[53] K. Yadav, V. Naik, A. Kumar, and P. Jassal, "Placemap: Discovering human places of interest using low-energy location interfaces on mobile phones," in *Proceedings of the Fifth ACM Symposium on Computing for Development*, pp. 93–102, 2014.

[54] M. Ibrahim and M. Youssef, "A hidden markov model for localization using low-end gsm cell phones," in *2011 IEEE International Conference on Communications (ICC)*, pp. 1–5, IEEE, 2011.

[55] L. Breiman, *Classification and regression trees*. Routledge, 2017.

[56] G. D. Forney, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[57] F. Boccardi, R. W. Heath, A. Lozano, T. L. Marzetta, and P. Popovski, "Five disruptive technology directions for 5g," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 74–80, 2014.

[58] M. Islam, A. Razzaque, and J. Islam, "A genetic algorithm for virtual machine migration in heterogeneous mobile cloud computing," in *2016 International Conference on Networking Systems and Security (NSysS)*, pp. 1–6, IEEE, 2016.

[59] Q. Fan, N. Ansari, and X. Sun, "Energy driven avatar migration in green cloudlet networks," *IEEE Communications Letters*, vol. 21, no. 7, pp. 1601–1604, 2017.

[60] D. Zhao, T. Yang, Y. Jin, and Y. Xu, "A service migration strategy based on multiple attribute decision in mobile edge computing," in *2017 IEEE 17th International Conference on Communication Technology (ICCT)*, pp. 986–990, IEEE, 2017.

[61] A. Nadembega, A. S. Hafid, and R. Brisebois, "Mobility prediction model-based service migration procedure for follow me cloud to support qos and qoe," in *2016 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2016.

[62] S. Filiposka, A. Mishev, and K. Gilly, "Community-based allocation and migration strategies for fog computing," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, IEEE, 2018.

[63] Y. Sun, S. Zhou, and J. Xu, "Emm: Energy-aware mobility management for mobile edge computing in ultra dense networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2637–2646, 2017.

[64] Y. Tan, K. Wang, Y. Yang, and M.-T. Zhou, "Delay-optimal task offloading for dynamic fog networks," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2019.

[65] Z. Zhu, T. Liu, Y. Yang, and X. Luo, "Blot: Bandit learning-based offloading of tasks in fog-enabled networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 12, pp. 2636–2649, 2019.

[66] S. Zhao, Z. Zhu, F. Yang, and X. Luo, "Online optimal task offloading with one-bit feedback," in *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 683–687, IEEE, 2018.

[67] S. Ghoorchian and S. Maghsudi, "Multi-armed bandit for energy-efficient and delay-sensitive edge computing in dynamic networks with uncertainty," *arXiv preprint arXiv:1904.06258*, 2019.

[68] T. Ouyang, R. Li, X. Chen, Z. Zhou, and X. Tang, "Adaptive user-managed service placement for mobile edge computing: An online learning approach," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 1468–1476, IEEE, 2019.

[69] J. Wang, K. Liu, M. Ni, and J. Pan, "Learning based mobility management under uncertainties for mobile edge computing," in *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 2018.

[70] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4005–4018, 2018.

[71] 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, 2017.

[72] V. Erceg, L. Schumacher, P. Kyritsi, *et al.*, "Ieee 802.11-03/940r4 tgn channel models," *IEEE P802*, vol. 11, 2004.

[73] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[74] R. Kleinberg, A. Niculescu-Mizil, and Y. Sharma, "Regret bounds for sleeping experts and bandits," *Machine learning*, vol. 80, no. 2-3, pp. 245–272, 2010.

[75] D. Chakrabarti, R. Kumar, F. Radlinski, and E. Upfal, "Mortal multi-armed bandits," in *Advances in neural information processing systems*, pp. 273–280, 2009.

[76] Z. Bnaya, R. Puzis, R. Stern, and A. Felner, "Social network search as a volatile multi-armed bandit problem," *HUMAN*, vol. 2, no. 2, p. 84, 2013.

[77] R. Agrawal, M. Hedge, and D. Teneketzis, "Asymptotically efficient adaptive allocation rules for the multiarmed bandit problem with switching cost," *IEEE Transactions on Automatic Control*, vol. 33, no. 10, pp. 899–906, 1988.

[78] Y. Wang, J.-Y. Audibert, and R. Munos, "Algorithms for infinitely many-armed bandits," in *Advances in Neural Information Processing Systems*, pp. 1729–1736, 2009.

[79] A. B. O. de Gortari, "Empirical study on game transfer phenomena in a location-based augmented reality game," *Telematics and Informatics*, vol. 35, no. 2, pp. 382–396, 2018.

[80] M. Whaiduzzaman, M. Sookhak, A. Gani, and R. Buyya, "A survey on vehicular cloud computing," *Journal of Network and Computer applications*, vol. 40, pp. 325–344, 2014.

[81] T. K. Refaat, B. Kantarci, and H. T. Mouftah, "Virtual machine migration and management for vehicular clouds," *Vehicular Communications*, vol. 4, pp. 47–56, 2016.

[82] A. M. Mustafa, O. M. Abubakr, O. Ahmadien, A. Ahmedin, and B. Mokhtar, "Mobility prediction for efficient resources management in vehicular cloud computing," in *2017 5th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, pp. 53–59, IEEE, 2017.

[83] 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*, vol. 68, no. 4, pp. 3061–3074, 2019.

[84] Y. Sun, J. Song, S. Zhou, X. Guo, and Z. Niu, "Task replication for vehicular edge computing: A combinatorial multi-armed bandit based approach," in *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, IEEE, 2018.

[85] R. Zhang, P. Cheng, Z. Chen, S. Liu, Y. Li, and B. Vucetic, "Online learning enabled task offloading for vehicular edge computing," *IEEE Wireless Communications Letters*, 2020.

[86] F. Sun, N. Cheng, S. Zhang, H. Zhou, L. Gui, and X. Shen, "Reinforcement learning based computation migration for vehicular cloud computing," in *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 2018.

[87] R. S. Sutton, "Generalization in reinforcement learning: Successful examples using sparse coarse coding," in *Advances in neural information processing systems*, pp. 1038–1044, 1996.

[88] L. Chen and J. Xu, "Task replication for vehicular cloud: Contextual combinatorial bandit with delayed feedback," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 748–756, IEEE, 2019.

[89] Z. Wang, Z. Zhong, D. Zhao, and M. Ni, "Vehicle-based cloudlet relaying for mobile computation offloading," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 11, pp. 11181–11191, 2018.

[90] Z. Wang, Z. Zhong, and M. Ni, "A semi-markov decision process-based computation offloading strategy in vehicular networks," in *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 1–6, IEEE, 2017.

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

[92] C.-C. Lin, D.-J. Deng, and C.-C. Yao, "Resource allocation in vehicular cloud computing systems with heterogeneous vehicles and roadside units," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3692–3700, 2017.

[93] Q. Wu, H. Liu, R. Wang, P. Fan, Q. Fan, and Z. Li, "Delay sensitive task offloading in the 802.11 p based vehicular fog computing systems," *IEEE Internet of Things Journal*, 2019.

[94] H. Liang, X. Zhang, J. Zhang, Q. Li, S. Zhou, and L. Zhao, "A novel adaptive resource allocation model based on smdp and reinforcement learning algorithm in vehicular cloud system," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 10, pp. 10018–10029, 2019.

[95] Q. Wu, H. Ge, H. Liu, Q. Fan, Z. Li, and Z. Wang, "A task offloading scheme in vehicular fog and cloud computing system," *IEEE Access*, 2019.

[96] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *Proceedings of the 19th international conference on World wide web*, pp. 661–670, ACM, 2010.

[97] M. E. Taylor, N. Carboni, A. Fachantidis, I. Vlahavas, and L. Torrey, "Reinforcement learning agents providing advice in complex video games," *Connection Science*, vol. 26, no. 1, pp. 45–63, 2014.

[98] L. Giupponi, A. Galindo-Serrano, P. Blasco, and M. Dohler, "Docitive networks: an emerging paradigm for dynamic spectrum management," *IEEE Wireless Communications*, vol. 17, no. 4, p. 47, 2010.

[99] Y. Wu, F. Hu, S. Kumar, J. D. Matyjas, Q. Sun, and Y. Zhu, "Apprenticeship learning based spectrum decision in multi-channel wireless mesh networks with multi-beam antennas," *IEEE Transactions on Mobile Computing*, vol. 16, no. 2, pp. 314–325, 2016.

[100] F. L. Da Silva, R. Glatt, and A. H. R. Costa, "Simultaneously learning and advising in multiagent reinforcement learning," in *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pp. 1100–1108, International Foundation for Autonomous Agents and Multiagent Systems, 2017.

[101] M.-A. Lèbre, F. Le Mouël, and E. Ménard, "On the importance of real data for microscopic urban vehicular mobility trace," in *2015 14th International Conference on ITS Telecommunications (ITST)*, pp. 22–26, IEEE, 2015.

[102] J. Kwak, Y. Kim, J. Lee, and S. Chong, "Dream: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 12, pp. 2510–2523, 2015.

[103] D. Jiang and L. Delgrossi, "Ieee 802.11 p: Towards an international standard for wireless access in vehicular environments," in *VTC Spring 2008-IEEE Vehicular Technology Conference*, pp. 2036–2040, IEEE, 2008.