Mechanisms for service-oriented resource allocation in IoT

Universitat Politècnica de Catalunya

Departament d' Arquitectura de Computadors

Thesis presented in fulfilment of the requirements for the degree of Doctor for the Universitat Politècnica de Catalunya

Research Group: CRAAX

PhD Student: Vitor Barbosa Souza

Advisor: Xavier Masip-Bruin

Co-Advisor: Eva Marín-Tordera

January, 2018

Acknowledgements

Firstly, I want to thank my wife, Suzana, for being always by my side on moments of happiness and difficulties, always manifesting love and patience. I want also to extend my acknowledgement to all the other members of my family, who were not physically present but have always supported me on my decision of studying abroad, especially my parents, Sebastião and Edirlene, my brothers Vinícius and Vladimir, and my grandmother Maria Geralda.

In addition, I thank all the friends I met in Barcelona, for the relaxing moments and for being like a family to me during the last four years.

I thank my advisor Xavi and co-advisor Eva for the valuable tutorship and high availability for both meeting and reviewing my papers. This is further extended to Wilson, who I have worked in collaboration since my first year in CRAAX, contributing for the publication of several ideas. I must also thank all CRAAX students. The good working environment has enabled me to go through this process as smoothly as possible.

I must not forget to acknowledge the Capes Foundation and the Informatics Department of the Federal University of Viçosa for fomenting and enabling me to give this important step on my academic education.

Barcelona, January 2018

Vitor Barbosa Souza

Abstract

Albeit several IoT applications have been recently deployed in several fields, including environment and industry monitoring, Smart Home, Smart Hospital and Smart Agriculture, current deployments are mostly host-oriented, which is undoubtedly limiting the attained benefits brought up by IoT. Indeed, future IoT applications shall benefit from service-oriented communications, where the communication establishment between end-points is not dependent on prior knowledge of the host devices in charge of providing the service execution. Rather, an end-user service execution request is mapped into the most suitable resources able to provide the requested service. Furthermore, this model is a key enabler for the design of future services in Smart Cities, e-Health, Intelligent Transportation Systems, among other smart scenarios.

Recognized the benefits of this model in future applications, considerable research effort must be devoted for addressing several challenges yet unsolved, such as the ones brought up by the high dynamicity and heterogeneity inherent to these scenarios. In fact, service-oriented communication requires an updated view of available resources, mapping service requests into the most suitable resources taking several constraints and requirements into account, resilience provisioning, QoS-aware service allocation, just to name a few.

This thesis aims at proposing and evaluating mechanisms for efficient resource allocation in service-oriented IoT scenarios through the employment of two distinct baseline technologies. In the first approach, the so-called Path Computation Element (PCE), designed to decouple the host-oriented routing function from GMPLS switches in a centralized element, is extended to the service-oriented PCE (S-PCE) architecture, where a service identifier (SID) is used to identify the service required by an end-user. In this approach, the service request is mapped to one or a set of resources by a 2-steps mapping scheme that enables both selection of suitable resources according to request and resources characteristics, and avoidance of service disruption due to possible changes on resources' location.

In the meantime, the inception of fog computing, as an extension of the cloud computing concept, leveraging idle computing resources at the edge of the network through their organization as highly virtualized micro data centers (MDC) enabled the reduction on the network latency observed by services launched at edge devices, further reducing the traffic at the core network and the energy consumption by network and cloud data center equipment, besides other benefits. Envisioning the benefits of the distributed and

coordinated employment of both fog and cloud resources, the Fog-to-Cloud (F2C) architecture has been recently proposed, further empowering the distributed allocation of services into the most suitable resources, be it in cloud, fog or both.

Since future IoT applications shall present strict demands that may be satisfied through a combined fog-cloud solution, aligned to the F2C architecture, the second approach for the service-oriented resource allocation, considered in this thesis, aims at providing QoS-aware resource allocation through the deployment of a hierarchical F2C topology, where resource are logically distributed into layers providing distinct characteristics in terms of network latency, disruption probability, IT power, etc. Therefore, distinct strategies for service distribution in F2C architectures, taking into consideration features such as service transmission delay, energy consumption and network load. Concerning the need for failure recovery mechanisms, distinct demands of heterogeneous services are considered in order to assess distinct strategies for allocation of protection resources in the F2C hierarchy. In addition, the impact of the layered control topology on the efficient allocation of resources in F2C is further evaluated. Finally, avenues for future work are presented.

Table of Contents

List of Figures

List of Tables

List of Acronyms

ABR	Area Border Router
ASON	Automatically Switched Optical Network
AP	Access Point
AS	Autonomous System
CaaS	Control as a Service
CI	Critical Infrastructures
DC	Data Center
DCD	Device Context Database
DHT	Distributed Hash Table
DLP	Data Level Parallelism
DM	Decision Module
DNS	Domain Name System
DSE	Dynamic Service Execution
DSM	Distributed Shared Memory
DTLS	Datagram Transport Layer Security
ECC	Elliptic Curves Cryptography
F2C	Fog-to-Cloud
GMPLS	Generalized Multi-Protocol Label Switching
HID	Host Identifier
IaaS	Infrastructure as a Service
ILP	Integer Linear Programming
ILSA	ID/LOC Split Architecture
IoT	Internet of Things
IPv4	Internet Protocol version 4
ISP	Internet Service Provider
ITS	Intelligent Transportation System
LAN	Local Area Network
LSP	Label Switched Path
LSPDB	Label Switched Path Database
M2M	Machine-to-Machine
MDC	Micro Data Center
MKP	Multidimensional Knapsack Problem
MPLS	Multi-Protocol Label Switching
NE	Network Element
NSI	Network State Information
OFRA	OpenFog Reference Architecture
OSPF	Open Shortest Path First
PaaS	Platform as a Service
PCE	Path Computation Element
PCEP	Path Computation Element Communication Protocol

PCEPM	PCEP Module
PCM	Path Computation Module
PCRep	Path Computation Reply
PCReq	Path Computation Request
PCC	Path Computation Client
PHA	Personal Health Assistant
QoS	Quality of Service
RFID	Radio-Frequency Identification
RSVP	Resource Reservation Protocol
RTT	Round Trip Time
SaaS	Software as a Service
SDN	Software Defined Network
SID	Service Identifier
SLA	Service Level Agreements
SOA	Service-Oriented Architectures
SOM	Service Orchestrator Module
SP	Service Provider
SPCE	Service-Oriented Path Computation Element
S-PCEP	Service-Oriented PCE Communication Protocol
SRT	Service Response Time
TE	Traffic Engineering
TED	Traffic Engineering Database
TTL	Time to live
VWSN	Video-based Wireless Sensor Networks
WAN	Wide Area Network
WLAN	Wireless Local Area Network
WSN	Wireless Sensor Networks

1. Introduction and problem statement

1.1.Future network applications

A plethora of new heterogeneous connected objects are expected to be deployed in the next few years demanding broad connectivity anywhere, anytime and anyhow, and providing ability of collecting and sharing environment data, interacting with humans and other devices, performing predefined actions, among others, through the employment of smart machine-to-machine (M2M) communication. Leveraging the large availability of those devices, traditional services, such as traffic control, healthcare, surveillance, environment temperature control and others, are experiencing a shift to a new category of service through seamless integration with popular, government and industry demands.

These factors are fueling not only the evolution on traditional services but also the deployment of novel categories of services, enabling Smart Transportation, Smart Monitoring, Smart Home, e-Health, Smart Cities and Smart Agriculture, just to name a few. To designate this novel paradigm for internet services, the name Internet of Things (IoT) was coined [1] [2].

In this scenario, “thing” can be any device endowed with network connectivity enabling it to either receive structured data or requests for specific task execution, or sending information regarding itself or any temporary data such the ones collected by the attached sensors. Hence, examples of things may include traffic lights (Smart Cities), cars (Smart Transportation), wearables (e-Health) or tracking sensors (Smart Monitoring). However, recognized the potential benefits provided by such smart scenarios, the unstoppable growth of connected devices, as shown in Figure 1.1 [3], and the consequent creation of a large myriad of data and network traffic, undoubtedly brings several challenges, such as the need for high processing and storage capabilities, mobility of devices and security issues. Simultaneously, along with the development of datacenter (DC) technologies, cloud computing has been positioned as a key enabler for IoT applications [4]. This is motivated by the on demand self-service, scalable, location independent, pay-as-you-go online computing model, offering huge storage capacity and processing resources of cloud commodities, properly matching several required IoT services demands, including for example highly demanding services, such as media delivery or DC remote backup solutions. Moreover, cloud computing brings virtualization capabilities and massive datacenters facilities to IoT, so that users can take advantage by offloading services execution.

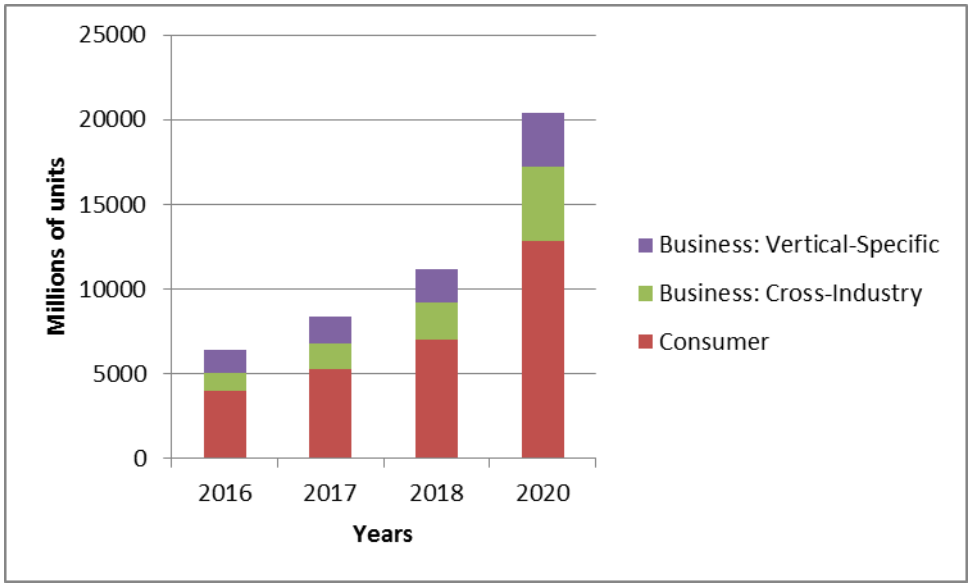


Figure 1.1: Evolution on the amount of "Things" in recent and upcoming years.

Additional challenges on the wide deployment of IoT services are related to the need for coping with such large amplitude of heterogeneous objects, in terms of hardware capabilities and architectures, communication enablers, energy and mobility profiles, functional attributes, software, user interface and cost [5]. Therefore, the IoT concept embraces all the required technologies and strategies to put together these heterogeneous devices –enabling their communication and, thus, deployment of innovative services–, including strategies for unique identification of resources, tasks allocation, network technologies integration, data aggregation, security, energy saving, etc. [6]. Indeed, albeit several IoT applications are successfully deployed nowadays, most out of them are limited to small areas such as in industrial or domestic applications. Future IoT applications shall leverage the enormous growth on amount and capacity of devices in order to deploy services to broad areas, such as in smart cities. Such services shall rely on a collaborative model, where mobile end-users will share not just collected data but also idle resources.

1.2. The problem rationale: The need for service oriented communication

As described in [7], despite the fact that the IoT term has been used for the first time in 1999, the concept of using Internet to connect appliances has been studied since 1989. During this period, a coffee pot, a toaster and a camera have been remotely controlled through Internet connection in distinct experiments. These first experiments have an important aspect in common with several IoT service implementations nowadays; the host-oriented communication model. Host-oriented communication is the IP-based model

employed by the traditional Internet architecture, where the destination host is known *a priori* when a client intends to execute a service. This is the case when the end-user request a webpage, an audio or video streaming, application download, or remotely adjust the temperature of his/her apartment when he/she is about to reach out home. Albeit the host-oriented communication fulfills the requirements for some current applications, several future IoT services can benefit from the so-called service-oriented communication –also referred to as information or content oriented [8].

The service-oriented communication is a model inspired on service-oriented architectures (SOA), coined as a paradigm for architecting the Internet, where implemented services are loosely coupled and end-user requests may be satisfied by executing, in a coordinated way, a flow of independent and reusable services. In SOA, services' implementations as well as their interactions are transparent to the end-user. Therefore, from the user's perspective, the requested service is viewed as a single atomic operation [9].

In a service-oriented communication model, the client side requests the execution of a service identified by an ID rather than the location LOC where the host providing the service is. The LOC may be an IPv4 address, as it is the case for current host-oriented communications. Therefore, an ID must be mapped to either one or a set of LOCs taking into account the parameters defined in the service layer, whilst the network layer is responsible for setting up the network resources required to provide the demanded service.

The decoupling of the service identification and resource location into ID and LOC enables independent service composition and routing process. However, whilst the service layer is responsible for the service requirements and the network layer is responsible for setting up the resources, the mapping of IDs into LOCs may be seen as an intermediate layer between service and network layers, decoupling the service requirements from the resources set up process. In order to clarify this idea, let's consider, on one hand, the host-oriented communication model, where any change related to a LOC –for instance, due to mobility, address migration, etc.– will substantially affect all services making use of that host, disrupting the established connections. On the other hand, the employment of IDs on the service-oriented model makes use of an intermediate management layer, which is responsible for keeping the state of the LOCs associated to each ID; hence, the service layer does not need to keep any information regarding the LOCs.

It is worth mentioning that the service-oriented communication stems in the fact that the initiator of a communication is solely interested in setting up a connection for the service provisioning. The relevant is the "What" (the service to be provided) rather than the "Who" or the "Where" (the LOC of the node who will provide the service). Furthermore, from a technological perspective, the decoupling of functionalities envisioned by the service-oriented communication paradigm enables long-term scalability, since the impact on new Internet applications is more noticeable at the service layer and less at the network layer, i.e., the service layer focuses on the composition of the service, whereas the network layer

solely focuses on providing enough bandwidth to set up the demanded services and network features, such as mobility and Traffic Engineering (TE) [10].

Therefore, it seems obvious that the deployment of the services envisioned for future IoT will immensely benefit from the service-oriented communication model. The expected dynamicity and volatility, which will be further increased by the envisioned collaboration model and the employment of resources shared by end-users on the move, hinders the connection establishment between client and the destination host. In such scenario, the user requesting the service does not have previous information regarding the resources available at the edge of the network, therefore, an agile coordination mechanism is required for discovering, selecting and allocating the most suitable resources according to request parameters, while providing resilience and QoS. A comparison of the main characteristics regarding host-oriented and service-oriented communication models is briefly presented in Table 1.1.

Table 1.1: Host-oriented vs Service-oriented communication models

	Host-oriented communication	**Service-oriented communication**
Destination host(s) selection	Source host	Third party entity, such as a broker
Requested service execution	Destination host	Flow of smaller distributed tasks
Knowledge about destination hosts	Mandatory	Optional
Disruption tolerance	No	Yes

1.3.Problem

In this section, we provide insights about the limitations of the current network architecture that are preventing the deployment of the new IoT scenario. Some of the limitations include routing, addressing schemes, as well as the need for innovative control plane strategies, including novel resource discovery and mapping features, just to name a few.

As presented in the previous section, the host-oriented communication model currently employed in the Internet is not suitable for an IoT scenario, demanding a shift to a service-

oriented communication model. However, the IPv4 addressing model, which is employed in the current Internet architecture, suffers from the double functionality problem [11]. This problem is caused by the employment of IP addresses both as a locator at the network layer and identifier at the upper layers. In order to exemplify the double functionality problem, consider that, when a connected object changes its aggregation point, the respective locator, which is an IP address, will be re-assigned. This will have a substantial impact on all its established communications. This address re-assignation yields several negative effects on the network, such as i) significant degradation of the communication quality; ii) eventual connection disruptions, increasing the complexity for the deployment of mobility features; and iii) occurrence of a non-negligible impact on resilience, mobility and TE features.

Furthermore, the IPv4 suffers from the well-known address depletion problem. This problem, which is already an issue in currently employed host-oriented model, is related to the large growth on the number of connected devices in the last years. As future IoT scenarios shall experience an even higher growth of connected objects on the next years, this problem will undoubtedly be aggravated. Indeed, end-users are using distinct, ever smaller and smarter connected objects requiring new connectivity demands. These connectivity demands make scalability to become a real problem considering that the IP-based addressing scheme deployed so far –mainly IPv4 with less than 2^{32} addresses– is not enough to support the huge addressing space envisioned for an IoT network scenario (see Table 1.2). Moreover, this increasing demand for Internet connectivity is highly affecting the overall routing performance, including the Domain Name System (DNS) performance as well as the deployment of new Internet applications [12].

Another limitation of current network architectures is related to the control plane architecture commonly used in today`s networks. Albeit distributed control schemes currently used, such as in Automatically Switched Optical Network (ASON) and Generalized Multi-Protocol Label Switching (GMPLS) provide acceptable performance in current network scenarios, their performance might be suboptimal in IoT scenarios [13]. This is mainly because routing strategies based on highly distributed control planes exhibit low performance under highly dynamic large scenarios, due to the signaling overhead imposed by distributed control schemes [14]. On the other hand, scalability and reliability on large scenarios are important concerns on centralized control plane architectures, such as in Software Defined Network (SDN) [15].

Furthermore, in traditional host-oriented networking, for instance in SDN, control plane does not have knowledge about the edge devices. Rather, controllers keep only information regarding forwarding devices topology and, among others, the control plane can define the switches (forwarding devices) that should be used in order to establish communication between distinct networks. On the other hand, next generation service-oriented IoT applications will require the edge resource selection according to the services offered by them. Moreover, the amount of information regarding the edge resources whose controllers should keep will increase according to the amount of services offered by them. Therefore,

the presented factors show that current control plane technologies cannot fulfill the service-oriented IoT requirements.

Table 1.2: Characteristics of future IoT

Devices demanding internet connection >>2^{32}
Smart connected objects with enhanced capabilities
Network features: TE, green networking
New users roles: consumers + producers = prosumers
New network scenarios: Virtualized Data Centers, Smart Cities, Smart Transportation
Dynamic Set up/tear down connections in short-term basis
Proactive network reconfiguration
Mobility without communication disruption (Full Mobility)

In addition, the deployment of a collaborative model shall arise several concerns regarding, for instance, to security features such as authentication, identity management, access control, integrity, privacy and availability, among others.

1.4.Thesis motivation

According to what has been stated so far, the future Internet of Things shall bring several benefits to distinct areas of society, including health, welfare, industry, government, agriculture, transportation and security. However, there is no doubt that current networks technologies must undeniable evolve to keep up the pace of the IoT applications.

Indeed, there is a wide of research studies available in the literature pushing for the demise of both conventional routing and communication architectures, since they would lead to suboptimal operation in an IoT scenario. However, the envisioned service-oriented IoT is

still on its infancy, requiring several issues to be addressed before its fully deployment. It is clear the need for developing new strategies and technologies intended to enable efficient storage and to retrieve updated information regarding resources at the edge of the network in a highly dynamic scenario with myriad of mobile and volatile resources. The execution of future Internet services will demand efficient selection, allocation and orchestration of the most suitable resources according to requested services. However, albeit efficient resource selection, service orchestration and protection mechanisms require an updated view of the available resources, update mechanisms must cope with high signaling overhead, commonly observed in distributed control plane architectures.

It is unquestionable the need for solutions able to address the enumerated issues. Therefore, promising technologies, such as the Path Computation Element (PCE) routing architecture [16] and the novel fog computing paradigm [17], must be assessed aiming at addressing issues that are hindering the fully deployment of the service-oriented IoT. This includes the provisioning of novel strategies for connectivity, as well as the assessment of enhanced architectures for service allocation and composition in scenarios with high dynamicity, including the evaluation of protection strategies, QoS provisioning and secure communication for edge devices.

1.5.Objectives

The main objective of this thesis is to assess novel mechanisms for allocation of resources demanded for execution of services in service-oriented IoT scenarios. More specifically, the main objective is split into two technical objectives as follows.

- Technical Objective 1 (TO1): Enhancement of the conventional host-oriented PCE routing architecture aiming at exploring potential implementations of service-oriented PCE in future IoT scenarios.
- Technical Objective 2 (TO2): Assessment of fog computing as an enabler for future IoT services deployment as well as the proposal of QoS-aware service allocation strategies for combined fog and cloud architectures.
- Technical Objective 3 (TO3): Assessment of the impact of control plane topology on QoS-aware resource selection.

In TO1, the evaluation of a service-oriented PCE (SPCE) as a centralized entity for selection of resources for service provisioning is performed. Therefore, whilst conventional PCE is devoted to the selection of the best path to the destination host, the SPCE assumes the responsibility for selecting possible destination hosts and selection of the most suitable one according to the invoked service demands. To cope with scalability demands, an enhanced 3-layered approach of the so-called ID/LOC Split Architecture (ILSA) is proposed.

In TO2, the so-called Fog-to-Cloud (F2C) architecture [18] is employed as the platform for the deployment and evaluation of proposed strategies regarding service allocation in IoT. The first set of experiments is devoted to the QoS-aware selection of resources aiming at the allocation itself. The goals of these strategies are both reducing the latency experienced by the services while guaranteeing the fulfillment of the service requirements and resource offerings, and optimal service allocation taking into consideration load balancing and energy consumption. TO2 also includes the assessment of strategies for the allocation of protection resources in order to avoid service disruptions due to failures on the initially allocated resources.

In TO3, the end-to-end communication setup is evaluated taking into consideration distinct control plane topologies, including strategies for the dynamic deployment of controllers. The rationale behind TO3 is the QoS provisioning for sensitive services requiring realtime resource allocation.

1.6.Thesis structure

In this section, the structure for the rest of the thesis is presented in details.

Chapter 2: Presents the first approach for the service composition in the assessed scenario. This approach is based on an extension of the PCE routing protocol to a novel architecture so-called service-oriented PCE (SPCE), which employs the decoupling of ID and LOC functions of traditional internet addressing scheme.

- Section 2.1 presents the background for this proposal, introducing the main concepts regarding both PCE architecture and ID/LOC splitting architectures.
- Section 2.2 shows details of the proposed SPCE architecture, including the enhanced ILSA scheme for service-oriented communication establishment making use of a 2-step mapping. Further, the scalability of the proposed scheme is also assessed.

Chapter 3: Enumerates the reasons for shifting the base technology employed on this thesis from SPCE to the novel F2C architecture.

Chapter 4: Introduces the second approach for the service composition for the future IoT, which is based on the combination of cloud computing and the novel fog computing architecture. The whole chapter is devoted to presenting the background regarding the technologies employed on the rest of the thesis.

- Section 4.1 revisits the state-of-the-art of cloud computing technology, with special focus on the employment of its resources by IoT applications, for instance, for service execution offloading.

- Section 4.2 presents the fog computing concept and its main characteristics, analyzing the importance on future internet scenarios, benefits and limitations of the envisioned collaborative model.

- Section 4.3 introduces the F2C architecture emphasizing its goals and the reasons for needing coordination on cloud and fog resources usage. Further, this section presents the challenges on implementing such architecture highlighting the challenges assessed in this thesis.

Chapter 5: Discusses the service allocation characteristics in F2C including the resource selection phase. For simplicity, the presented mathematical model is split into two distinct sections: 5.2 and 5.3.

- Section 5.1 describes the challenges for service allocation at resources at the edge of the network.

- Section 5.2 introduces a preliminary approach of the mathematical model where the service allocation problem is modeled in planning scenario using integer linear programming (ILP) considering one single type of resource.

- Section 5.3 extends the previously presented model for considering requests of distinct service types as well as distinct shared resources types. Moreover, the model includes a simple approach for energy consumption comparison.

- Section 5.4 presents and evaluates implemented heuristics for service allocation in online scenario. Results show the impact of mobility in disruption probability, latency, and power consumption.

Chapter 6: Is devoted to resilience aspects of the service allocation in IoT scenarios.

- Section 6.1 introduces the discussion regarding failure recovery mechanisms in F2C. Moreover, it describes challenges introduced by mobility, volatility and heterogeneity in IoT scenarios.

- Section 6.2 presents distinct allocation protection strategies, highlighting pros and cons. The mathematical model for each presented strategy is exposed, as well as respective results and discussion.

Chapter 7: Assesses the layered control architecture in F2C, taking into account its distributed nature and impact on QoS due control decisions latency.

- Section 7.1 presents the concerns related to the control plane topology in F2C architecture and the negative impact of the number of edge devices on QoS, especially for sensitive services requiring real-time service allocation.

- Section 7.2 positions the Control as a Service (CaaS) concept, describing the scenario, as well as its benefits in a real-time service execution environment.
- Section 7.3 describes the implementation of a testbed and show results regarding the impact of the number of resources managed by a control device on its processing delay.
- Section 7.4 describes the performed simulations and attained results regarding the F2C control topology assessment.

Chapter 8: Summarizes the proposed ideas of this Thesis and concludes.

Chapter 9: Suggests avenues for future work.

- Section 9.1 describes the service placement and execution considering the particularities of the F2C architecture, further highlighting challenges to be overcome.
- Section 9.2 introduces the multidimensional control plane concept, presenting main benefits through three distinct use-cases. Several challenges and opportunities for futures work are enumerated.

2. First approach for service composition: the Service-oriented PCE architecture

2.1.Background

In this section, the main concepts regarding the technologies behind the proposed Service-oriented PCE architecture are presented. Firstly, the Path Computation Element is introduced as the baseline technology and, later, the ID/LOC Split Architecture concept is presented as the key mechanism for enabling service-oriented service composition.

2.1.1. Path Computation Element (PCE)

The so-called Path Computation Element (PCE) [16] is a network paradigm that has proved in recent years its ability to address the main limitations presented by distributed source-based routing schemes currently used, such as GMPLS and ASON. In GMPLS, for instance, after receiving a connection request, a GMPLS controller computes a Label Switched Path (LSP) to reach the destination optical router based on local information stored on Traffic Engineering Database (TED) and Label Switched Path Database (LSPDB). Once the LSP in computed, a label is requested by means of a signaling Resource Reservation Protocol (RSVP) Path message, which travels through the computed path until the destination, triggering the transmission of a RSVP Resv message, which travels back from destination to source, being responsible for the wavelength (WSON) of spectrum (SSON) reservation and label assignment [19]. Whist the distributed source-based approach employed by GMPLS is suitable for current optical transport networks, future optical networks will embrace several interconnected administrative domains, where such routing schemes are limited [20].

Therefore, the Path Computing Element (PCE) architecture is based on the decoupling of path computation actions from underlying switches through a network application (the PCE itself) that can be placed within a network node (e.g., router) or on dedicated elements whose function is to compute one or a set of paths between source and destination network nodes. The path computation may be done using conventional routing algorithms, such as Open Shortest Path First (OSPF), extended with TE features along with a network graph built and stored in the TED.

The Path Computation Client (PCC) is any client requiring the path computation to a specific destination. Therefore, the PCC connects to the PCE and, by means of the Path

Computing Element Communication Protocol (PCEP) [21], requests a path from the source, i.e. the PCC itself, to the destination node.

PCEP is an extensible open protocol providing a set of messages that are exchanged between peers over TCP, ensuring reliable communication and simplifying implementation. Hence, once a TCP connection is established between peers, the PCC may send path computation request (PCReq) messages specifying path parameters, such as demanded bandwidth, priority, and points of failure that should be excluded from the computed path, among others. As a result, the PCE may send a path computation reply (PCRep) message containing either one or a set computed paths, or a negative reply, if a path could not be determined, which may specify the reasons for the negative reply.

In addition, PCEP provides communication among distinct PCEs. The PCE architecture, including distinct communication models, is shown in Figure 2.1. As the figure shows, the centralized PCE model may be employed through either one single PCE, receiving all path computations for a given domain, or more than one, where backup PCE(s) are deployed to avoid single point of failure issues. On the other hand, distributed path computation within a single domain may be performed by more than one PCE, where PCEs may either compute paths without collaboration or cooperate for the shared computation of a single path.

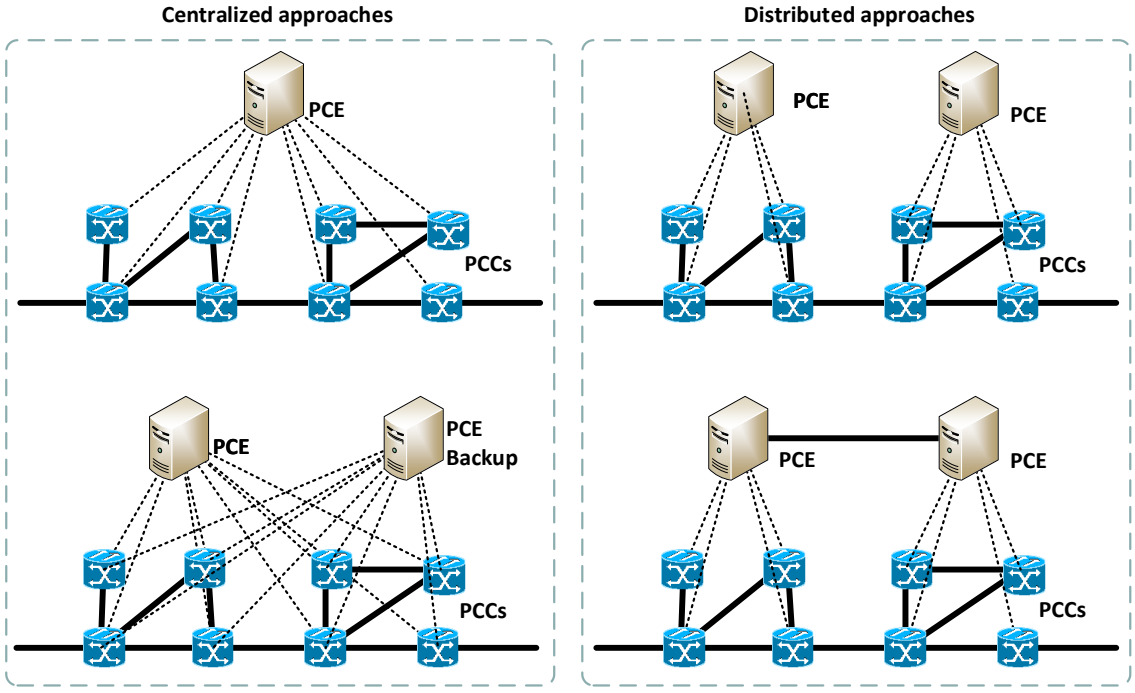


Figure 2.1: PCE communication models.

Multidomain path computation

In multidomain path computation, GMPLS controllers suffer from a limited view of network topology. Whilst each controller has a complete view of its respective autonomous system (AS) topology, only reachability information of neighbor ASs is viewed by controllers through aggregated representations, hindering optimal end-to-end multidomain path computation.

The implementation of PCE-based routing in multidomain scenarios leverages the horizontal interdomain communication among distinct PCEs, so that they can build the optimal end-to-end path in a collaborative fashion. The strategy employed is the Backward Recursive Path Computation (BRPC) [22] which achieves optimal path computation if the sequence of domains is known. In other words, it is assumed that the chain of domains traveled from source to destination is previously known. Hence, the path computation request is forwarded by each PCE to a PCE located in the next domain in the sequence, until it reaches the PCE in the destination domain. The multidomain path is, thus, computed by the BRPC procedure in a backwards fashion where the PCE located in the destination domain computes the shortest paths to the destination router from each Area Border Router (ABR) adjacent to the previous domain in the chain. Further, the cost (i.e., amount of hops) relative to the employment of each ABR used as ingress router is sent to the PCE located in the previous domain, which repeats the BRPC procedure to compute the shortest paths from source (or ingress ABRs, if it is not the source's domain) to the received ABRs, viewed as egress routers in this domain.

Since BRPC demands previous knowledge regarding the chain of domains in multidomain path computation, strategies such as hierarchical PCE may be employed. In this strategy, each domain is controlled by a respective child PCEs (c-PCE) whereas a parent PCE (p-PCE) is deployed as a higher-level responsible for the path computation among distinct c-PCEs. Therefore, the selection of used domains may be based on aggregated cost information [19].

Architectural approaches

The PCE concept can follow different architectural approaches regarding the knowledge of established LSPs as well as regarding the entities that may initiate a communication. In what concerns the LSPs, the PCE architecture may be either stateless or stateful. Therefore, in the conventional stateless PCE architecture, the path computation is based on information available in the TED, which means that the PCE can compute LSPs according to the known network state information (NSI), but it is not aware of the actual existing LSPs state. It may lead to suboptimal path computations and increased blocking probability since it is not based on an updated TED, that is, the TED may not be synchronized with the actual NSI. For instance, it is possible that the PCE computes two or more LSPs sharing the same wavelength if it receives those requests at the same time.

To minimize the impact of an outdated TED, some workarounds were proposed, introducing some sort of statefulness to the stateless PCE. In the first proposal, the PCE uses a cache mechanism to store recently computed LSPs and avoid the use of allocated resources [16]. In a similar strategy, by assuming that the LSP was successfully established, the stateless PCE acts proactively updating the TED, rather than waiting for the routing protocol to update it, thus, if errors are detected during the signaling phase, the PCE is notified by the source node and updates the TED again releasing the wavelength [23]. One distinct strategy consists in retaining the received requests for a certain time for later computing all the received requests concurrently [24]. Moreover, whilst a stateful PCE approach, computing paths based on actual network state and actual LSPs state, can be more efficient, the use of an extra database to store LSP state (i.e.: LSPDB), besides the TED, will introduce a high complexity to the path computation procedure, requiring more processing resources [19].

On the other hand, regarding the communication initializer, PCE may assume either passive or active behavior. Originally, PCE works passively, waiting for PCReq messages and, upon receiving, computing LSPs satisfying the request and sending a PCRep to the corresponding PCC. Thus, the PCE should never start a PCEP session [16]. To allow the PCE to start a PCEP session, a distinct PCE architecture makes use of an active PCE, which may allow it to suggest LSP updates to the clients or even the creation or deletion of LSPs as suggested by [25] and [26] though the introduction of new messages to the PCEP protocol.

2.1.2. ID/Locator Split Architectures (ILSA)

Distinct approaches to cope with the issues related to routing and addressing schemes currently used by Internet have been proposed, being classified into two groups. Clean-slate schemes are disassociated to the traditional layered-structure employed by the OSI model whilst non-disruptive schemes are coupled to the OSI structure. In both cases, the main objectives are the provisioning of service-oriented communication, decoupling of identification and location functions while also dealing with the depletion of addresses, as it happens in IPv4 addressing scheme. Despite the fact that a numerous research efforts have been devoted to both approaches in the recent years, network carriers seem reluctant to adopt clean-slate architectures mainly due to the migration task difficulty, and the potential disruption on provided services possibly promoted by this migration [27].

Non-disruptive approaches, such as ILSA [28], are increasingly being adopted in current network research trends, due to its capability to cope with the main weakness of IPv4 addressing schemes that hinders the deployment of applications requiring large mobility and presenting high volatility. Therefore, by decoupling IP functions and assigning an independent set of addresses for identification and location purposes, both the double functionality problem and the depletion of IPv4 addresses can be addressed. The service layer relies on the use of Identifiers (IDs) to support end-to-end connectivity, whereas in the network layer, routing functions are enabled by the use of Locators (LOCs).

In order to clearly illustrate the basic operation of an ILSA scheme, let's consider the scenario depicted in Figure 2.2, where an end-user using a mobile device is connected to a webserver by means of ISP 1. While connected to this ISP, the device LOC is the IP address "x.x.x.x" (provided by the ISP) and the ID "aaa" is assigned by the webserver in the service layer. Therefore, the service provisioning is done through mapping the device ID into the current LOC, which is performed by ILSA. Whenever the end-user moves to a different aggregation point, a new network address (LOC) is assigned to it. Since the device ID is not modified, the communication established between the Webserver and the ID is not affected by the reassignment of LOC "y.y.y.y". This is different from what occurs in host-oriented communications, where an IP address is used for both the service and network layer.

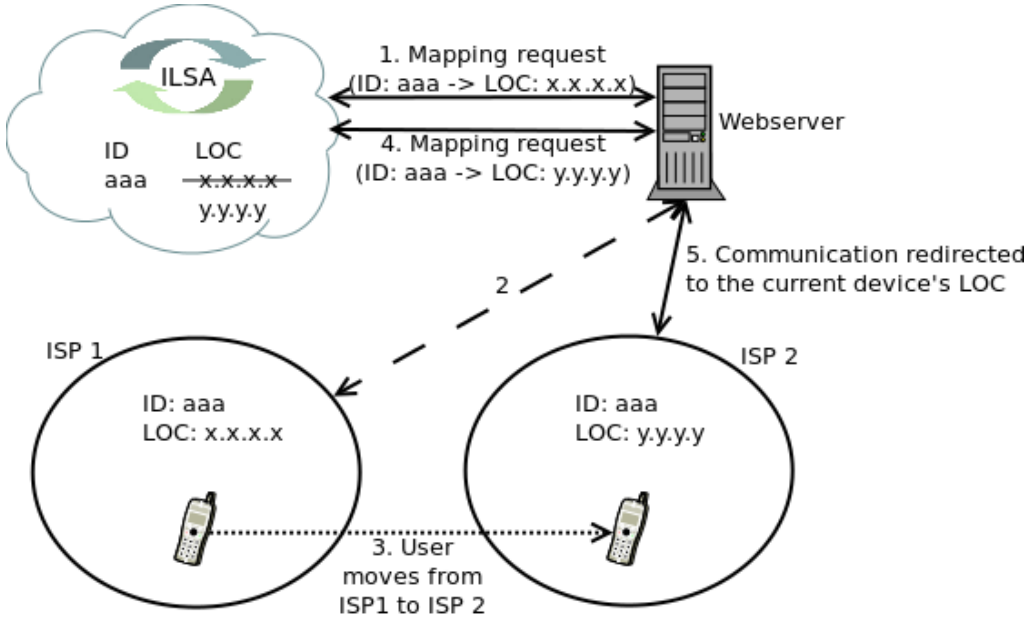


Figure 2.2: ILSA basic operation.

2.2.SPCE as a concept

The inherent deployment of thousands of network elements (NEs) in IoT scenarios, such as smart cities or vehicular networks, turns IoT routing as well as IoT service composition into large distributed problems. Moreover, the heterogeneity and dynamism perceived on IoT scenarios is adding even more complexity in the way NEs must be managed. Aligned to the current trend in decoupling control tasks to a centralized entity, we propose to extend PCE to accommodate the particular needs of IoT in the NEs selection (for service composition) and the routing.

The PCE is already established as an architectural solution to detach path computation actions from the network routers. However, extending PCE to support IoT constraints is not

a minor task. This complexity is mainly added by the fact that, while in traditional routing the source nodes only queries a path between two clearly defined endpoints (the hosts), in an IoT scenario, the service provider, when composing a new service, is not worried about the edge nodes but on different real-time information from the network. Hence, in practice, the service provider does not know what NEs are located in the destination domain area, nor the availability of the existing devices in terms of application/service layer resources (e.g., processing, storage and energy availability), and, even how to connect to each one of the mobile NEs (i.e. what is the best path). To cope with these issues, we propose using an extension of PCE enriched with ILSA, hereon referred to as a Service-oriented PCE (SPCE) [10]. This section starts by presenting a preliminary approach for enhancing PCE with ILSA schemes. Later, the main SPCE concepts, as well as its architecture and distinct use cases for SPCE are presented. The proposals presented in this chapter have been already validated by the work presented in contributions [10] and [29].

2.2.1. Preliminary approach for the PCE / ILSA collaboration

In order to illustrate the advantages of the collaboration between PCE and ILSA scheme, Figure 2.3 depicts a multi-domain optical network where PCC sends a path computation request to a service-aware PCE. As shown in step 1, the source node is attached to an ID rather than a LOC, contrary to conventional PCE schemes. Similarly, the host "server", which is providing the demanded service, is also attached to a distinct ID. In order to map the ID "server" to a LOC, the PCE makes use of an ILSA scheme, see step 2. Once the LOCs are obtained, the PCE computes the best path to domain 2 according to request requirements, which are service-dependent (e.g., bandwidth, delay), and send a Path Computation Reply (PCREp), as shown in step 3. Finally, the optical lightpath can be established.

By attaching the destination of a path computation to an ID, the network resources allocated to a path can change on the fly according to the LOC best mapping the service requirements. To put this into context, let's consider that, in case of a failure affecting optical node B, the service-aware PCE might compute a new path in an agile manner. This is because the destination of the computed path is reflected on the TED as it is attached to ID "server". This ID can be mapped into a new LOC, such as LOC C, i.e., the decoupling of ID/LOCs provides an abstraction layer. Unfortunately, in conventional PCE scenarios, the failure in optical node B will impact strongly on all paths with node B as a destination since the information stored in the TED would be inaccurate, i.e., reflecting that network resources are allocated to the failed optical node, which is no longer available.

In the scenario depicted in Figure 2.3, it is shown the advantages of the collaboration of ILSA and PCE schemes, where the ID is explicitly specified by the PCC. Nevertheless, we consider that this involves high synchronization between ILSA and the PCC. In addition,

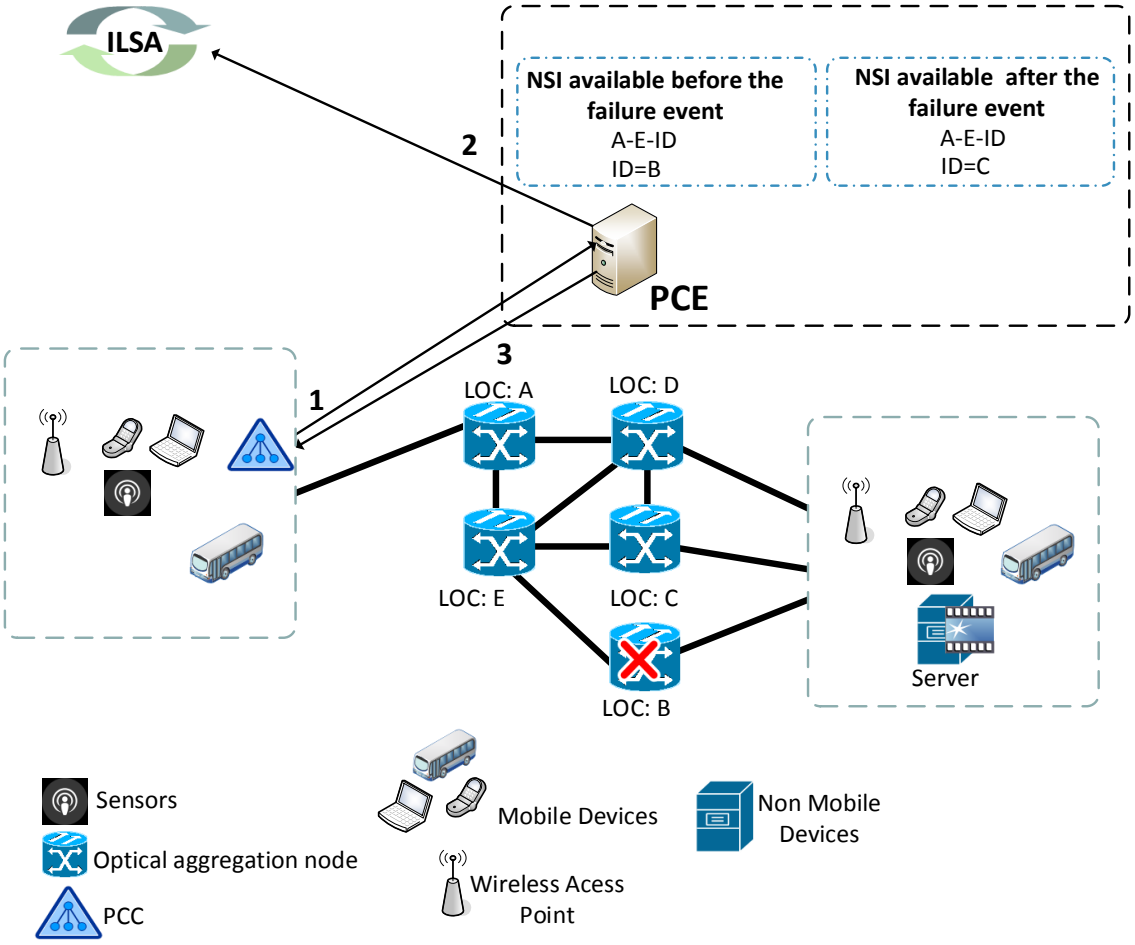


Figure 2.3: Collaboration between ILSA and PCE.

this hinders the deployment of network-aware service-composition, i.e., the PCC will select an ID based solely on the service layer requirements, without considering the network layer state such as bandwidth or amount of lightpaths available to establish connection to the selected ID. This has a strong impact on both the business models and the quality of the service provided. To cope with this issue, in the proposed service-oriented PCE it is envisioned that the PCC only requests the desired service, including a specific set of requirements. The SPCE is responsible for selecting the ID(s) and finally the LOC of the nodes which best fit the requested service.

2.2.2. Enhanced ILSA model

In the proposed enhanced ILSA scheme, the ID/LOC mapping is divided into a 2-step procedure where two distinct ID classes are employed in order to represent Service ID (SID) and Host ID (HID). Therefore, in the SPCE architecture, a PCReq can request as endpoint a specific type of service. For instance, a SID may be a host-server or a wireless sensor node. Furthermore, the PCReq may contain more than one SID, in case that the final service is the composition of different services; and of course, this request to the SPCE should also include some service layer parameters, such as energy availability, storage requirements, mobility profile or city area in the case of sensors. This is one of the main advantages of the proposed service oriented architecture: a request endpoint is a service (or services) instead of a specific host. For instance, a service request demanding high energy availability is mapped to one or more HIDs corresponding to nodes meeting this requirement.

After receiving the PCReq, the SPCE scheme maps the SIDs to HIDs of possible NEs providing these services. In the example depicted in Figure 2.4, the SID=Sound_Sensor is mapped into three different devices able to provide the requested resource, where two of them are mobile and the third is a fixed microphone installed in a building. The HIDs are represented by Mobile_device_1, Mobile_device_2 and Fixed_device_1. Finally, these HIDs are mapped into LOCs by using an ILSA scheme. The SPCE will select the most suitable LOCs (which correspond to the specific NEs required on providing the demanded service).

The SPCE architecture considers both, mobile and fixed devices. In the case of mobile devices, a single device (characterized by its HID) can be located at different positions due to its mobility. For instance, the device can frequently change its access point, which might result in a change of aggregation point at the optical layer. In this mobile scenario the SPCE architecture should be able to provide different LOCs according to different service layer parameters for these devices depending on the current location of the device and/or also according to other requirements. In the example of Figure 2.4, the Mobile_device_1 could be in n different locations ($LOC:IP_1$, … , $LOC:IP_n$) and the Mobile_device_2 could be in m different locations ($LOC:IP_x$, … , $LOC:IP_{x+m}$), whereas the Fixed_device_1 (which is a microphone in a city building) has associated only the $LOC:IP_y$. As it is mentioned, the SPCE will select for each device the more suitable LOC. For instance, if Mobile_device_1 is located in the IP address IP_2, the SPCE will provide the location $LOC:IP_2$.

2.2.3. SPCE architecture

In this section, we describe an overview of the SPCE architecture. We envision an IoT scenario where the access layer is based on mobile as well as fixed network technologies. Each access domain is formed by a wealth of heterogeneous network elements (NEs) that are enabling new utilization of their generated big data [30]. On the other hand, the network core is based on optical flexible technologies [31], where each optical node is the aggregation point of an access domain. We assume that connectivity of NEs to respective access domain is already established. Therefore, our goal solely consists to establish connectivity between the aggregation points of each access domain.

In the proposed architecture, the goal of the PCC is to deliver agile service orchestration. For this purpose, it acts as service provider and relies on the SPCE features in order to achieve: 1) the Identification of the nodes providing the requested service, and; 2) establish physical connectivity to the aggregation point of each wireless node.

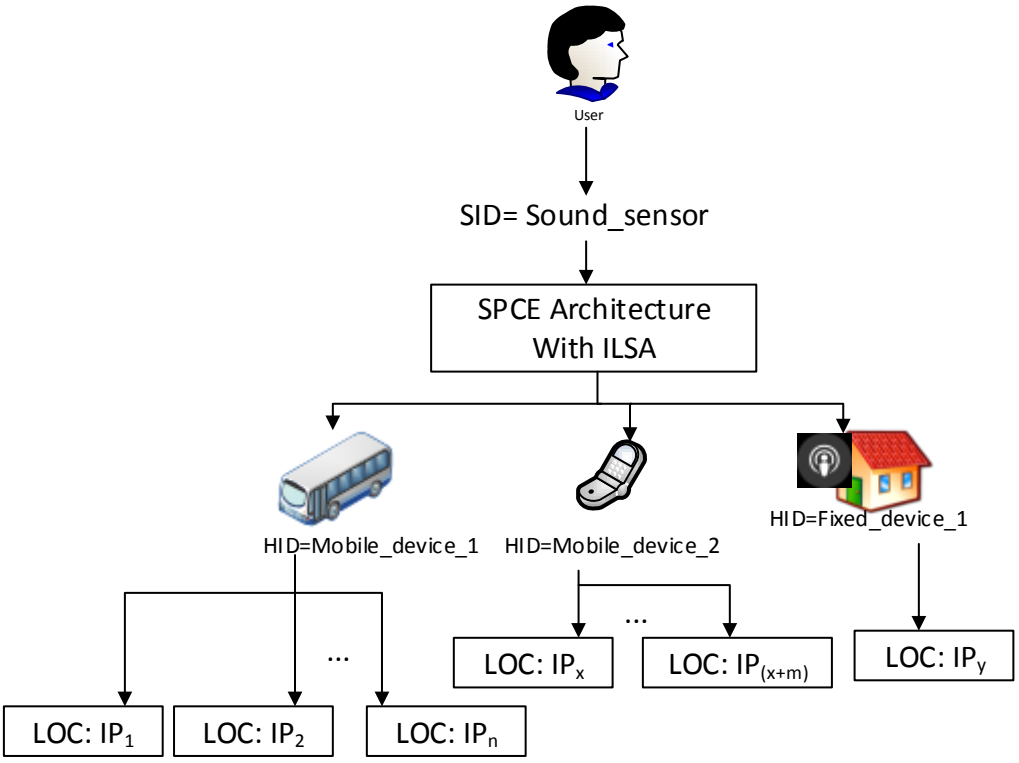


Figure 2.4: Enhanced ILSA 2-steps mapping.

On the other hand, the SPCE goal is twofold: 1) identifying the host IDs of the NEs offering the requested service with the minimum cost; and once the host identifiers are obtained and are mapped to their respective LOCs, 2) to compute path (based on the obtained LOCs) with the minimum blocking probability as well as low optical resources

consumption. The rationale behind this twofold goal is driven by the aim of enabling network-aware service composition. The SPCE makes use of the network state information (NSI) and an enhanced ILSA scheme to select both the best NEs and the best path between the PCC and each NE.

It is intuitive that this novel path computation strategy imposes changeling requirements to the conventional PCE architecture. As a network-aware service composition architecture, the SPCE must select the best NEs considering both network layer and service layer constraints. Therefore, the SPCE is responsible for enabling the service orchestration by the PCC.

An overview of the service request by the PCC in the SPCE architecture is illustrated by Figure 2.5. Therefore, in step 1, a PCReq message is delivered to the SPCE containing the main information regarding the request, for instance, request for a video server connection as well as specific connection demands. In step 2, SPCE communicates with ILSA server in order to obtain information about resources suiting the requirements. Special attention must be devoted to step 2, since a mobile service-oriented scenario requires a more daring paradigm. In order to cope with the NEs mobility, the SPCE relies on the enhanced ILSA scheme, presented on the previous section, to retrieve HIDs and respective current LOCs. Therefore, the PCC does not know beforehand what the specific NEs it will connect are, the PCReq message requests just for a service, identified by a SID along with a set of message parameters. Thus, based on NSI stored by the SPCE, it selects one or a set of HIDs able to satisfy the request. Finally, the SPCE, by means of an ILSA server, maps each selected HID to its respective LOC. After selecting HIDs and obtaining their respective LOCs, SPCE can compute the optimal path to destination and reply to PCC, as depicted by step 3 in Figure 2.5. Finally, the lightpath between client and server LOCs may be established.

The SPCE architectural building blocks, as well as the scenario where PCC is inserted is shown in Figure 2.6. The building blocks of the presented architecture are described as following lines and the employed symbols are described in Table 2.1.

- An extension of the PCE Communication Protocol referred to as (S-PCEP). This PCEP extension supports both ID based endpoints as well as service layer requirements –in addition to optical network layer requirements.

- **PCEP Module (PCEPM).** This module receives and sends PCReq and a PCRep respectively. It receives a PCReq in the form (HID_s, SID, $Req_{a,n}$), that is, the endpoint of the request is a required service (SID), whereas it sends a PCRep with one or a set of computed lightpaths. The PCReq receiving is illustrated in Figure 2.6 (step 1).

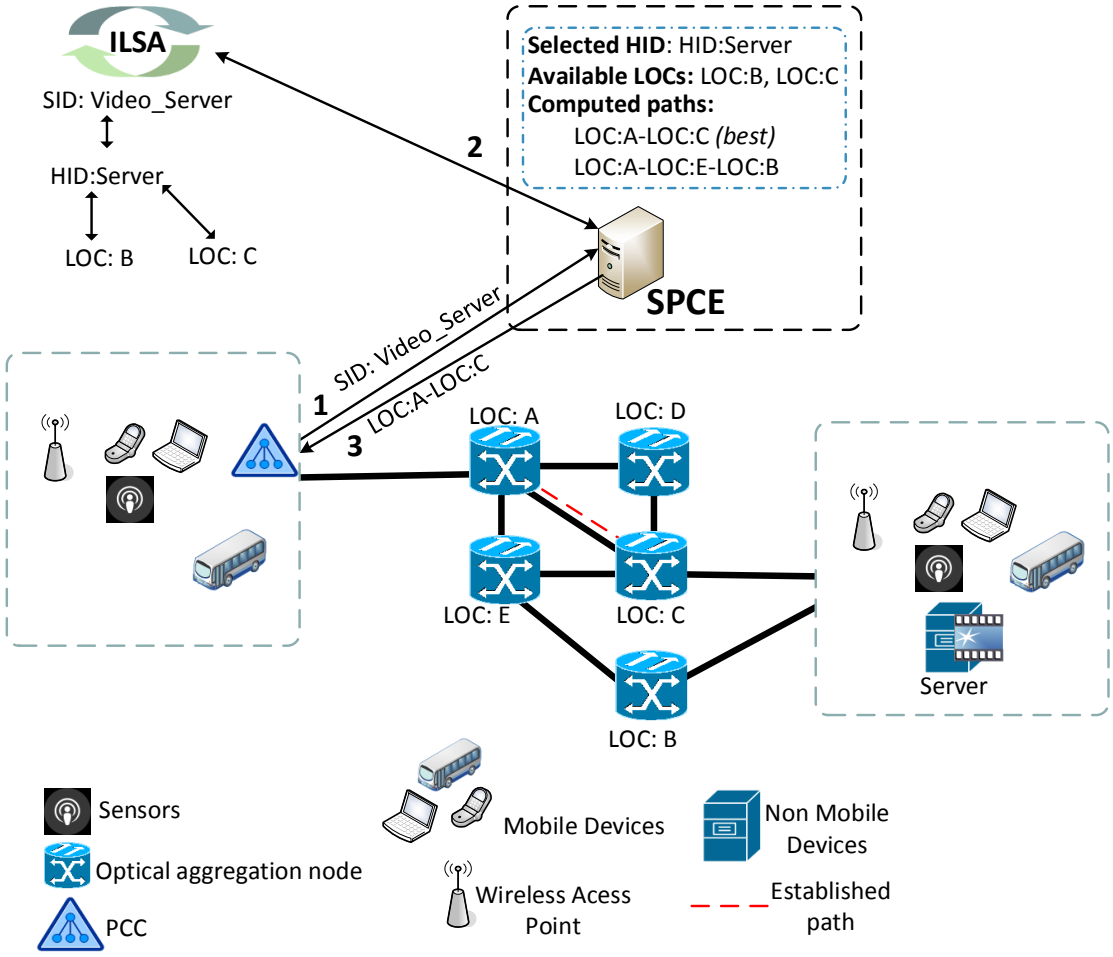


Figure 2.5: Service request in SPCE architecture.

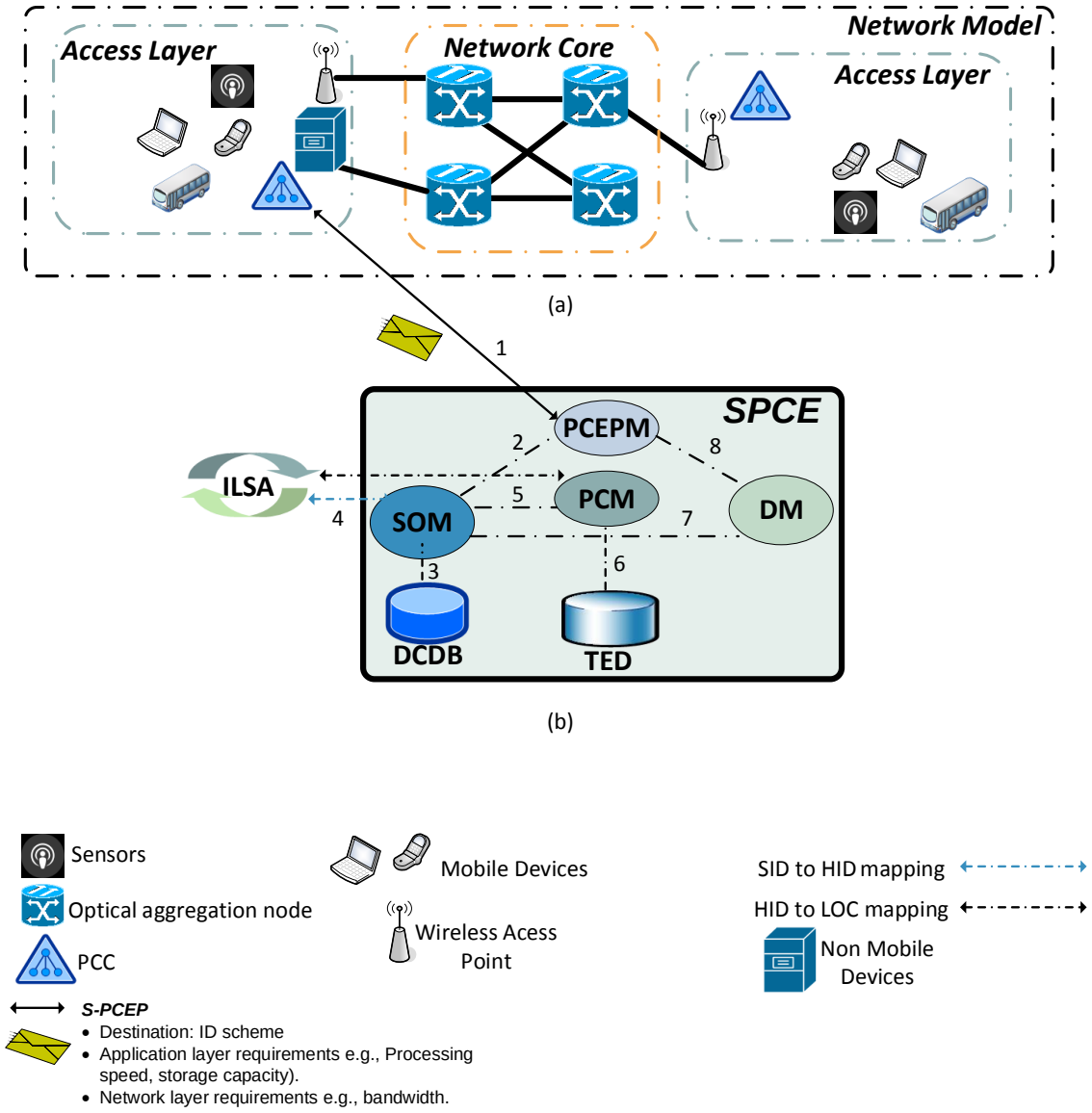


Figure 2.6: (a) SPCE network model; (b)SPCE architecture.

Table 2.1: Table of symbols

Symbol	Meaning
HID_n	Host Identifier, where n is a NE –source (s) or destination (d)– registered at the ILSA scheme.
SID	Service Identifier.
$req_{a,n}$	PCReq requirements where *a* and *n* are respectively application and network requirements
$l_{s,d}$	Optical lightpath where *s* and *d* are the locators of a source and destination NE respectively.
m_d	Service layer parameters offered by HID *d*.

- **Service Orchestrator Module (SOM).** The SOM is responsible for the destination HID lookup process, i.e., based on a given SID (step 2) it selects the HID of the NEs offering the requested service with the minimum cost (from the service layer perspective). To this end, the SOM uses state information available on the Device Context Database (DCDB) (step 3). Once the HID lookup process is done, the SOM communicates with an ILSA scheme in order to obtain the LOC associated to each HID selected (step4). In addition, the SOM receives updates related to service identifiers (SID) from the ILSA scheme. Based on these SID updates, it proactively performs HID lookup process.

- **Device Context Database (DCDB):** The DCDB stores state information that is not generated by the Traffic-Engineering Routing Protocols. This information is the one which will be used by the SOM for the HID lookup process according to service layer requirements such as energy availability, storage resources or mobility profile. To illustrate an example of the service layer requirements, consider the mobility profile metric. If a mobile NE *X* sharing specific sensors has been detected at the same place every workday between 1:00PM and 2:00PM for the last weeks, the SPCE can make some assumptions considering that the NE's sensors have a high probability of being available during that period on the next workday. Therefore, from the service layer perspective, the NE *X* is optimal for services that require real time environment information. Otherwise, constantly moving NEs only provide environment information during the (short) time period they are within a domain.

- **Path Computation Module (PCM):** The PCM performs path computation actions based on HID endpoints, specifically as the source node an HID specified in the S-PCEP message, as endpoint the HID computed by the SOM (step 5). The PCM communicates with the ILSA scheme to obtain the mapping of the HID into LOC in order to perform path computation actions. To this end, the PCM uses the NSI available in the TED (step 6). In addition, the PCM receives updates related to host identifiers (HID) from the ILSA scheme. Based on these ID/LOC updates, the PCM recomputes optical lightpaths.

- **TED**: Similar to the conventional PCE architecture, the TED is responsible for storing the NSI.

- **Decision Module (DM).** The DM receives as input from the SOM a set of tuples in the form ($\{HID_d, m_d\}$, $l_{s,d}$) (step 7). Each tuple is formed by a destination HID with its service layer parameters and an optical lightpath computed by the PCM. This optical lightpath has the Locator of HID_d as a destination and, as source, the locator of HID_s. Based on the given set of tuples the DM selects the one presenting the minimum cost considering both service and optical layer cost (step 8).

SPCE use cases

In the following lines, we illustrate through two use cases how the SPCE might fit in a service-oriented communication model in mobile scenarios. In the first use case we show how in a smart city scenario, the SPCE can proactively react to traffic conditions. For instance, let's consider the scenario shown in Figure 2.7, where each transportation vehicle may request Video Capture (VC) functions from other vehicles. These VC functions are processed by the cloud servers in order to get a holistic view of the entire road intersection. For instance, in step 1, the PCC at cloud requests a path computation with SID:ALSA_Buses. This SID selection is because the PCC is interested to get live VC from busses belonging to the company named ALSA. Once SPCE receives the request, a communication with ILSA server is established, as shown in step 2, in order to map the requested SID into suitable HIDs and further obtaining the respective LOCs (in this example, HID:ALSA1 and LOC:C). With the obtained LOC, the SPCE computes the optical lightpath B-E-C and sends it to PCC, as shown in step 3. However, since the providers of VC functions are not fixed, the vehicle identified with HID:ALSA1 changes its access domain to access domain 1. This will trigger the proactive provision of a new lightpath as it is shown in Figure 2.7(b). Notice that once the HID:ALSA1 is registered in Access Domain 1, in the ILSA scheme its LOC is changed to a new LOC (LOC:A) (step 1). The ILSA scheme informs the SPCE of this LOC change (step 2). Then, the SPCE recomputes the new lightpath B-E-A and sends it to the PCC (step 3). Therefore, the PCC is able to reach HID:ALSA1 through the employment of the new lightpath.

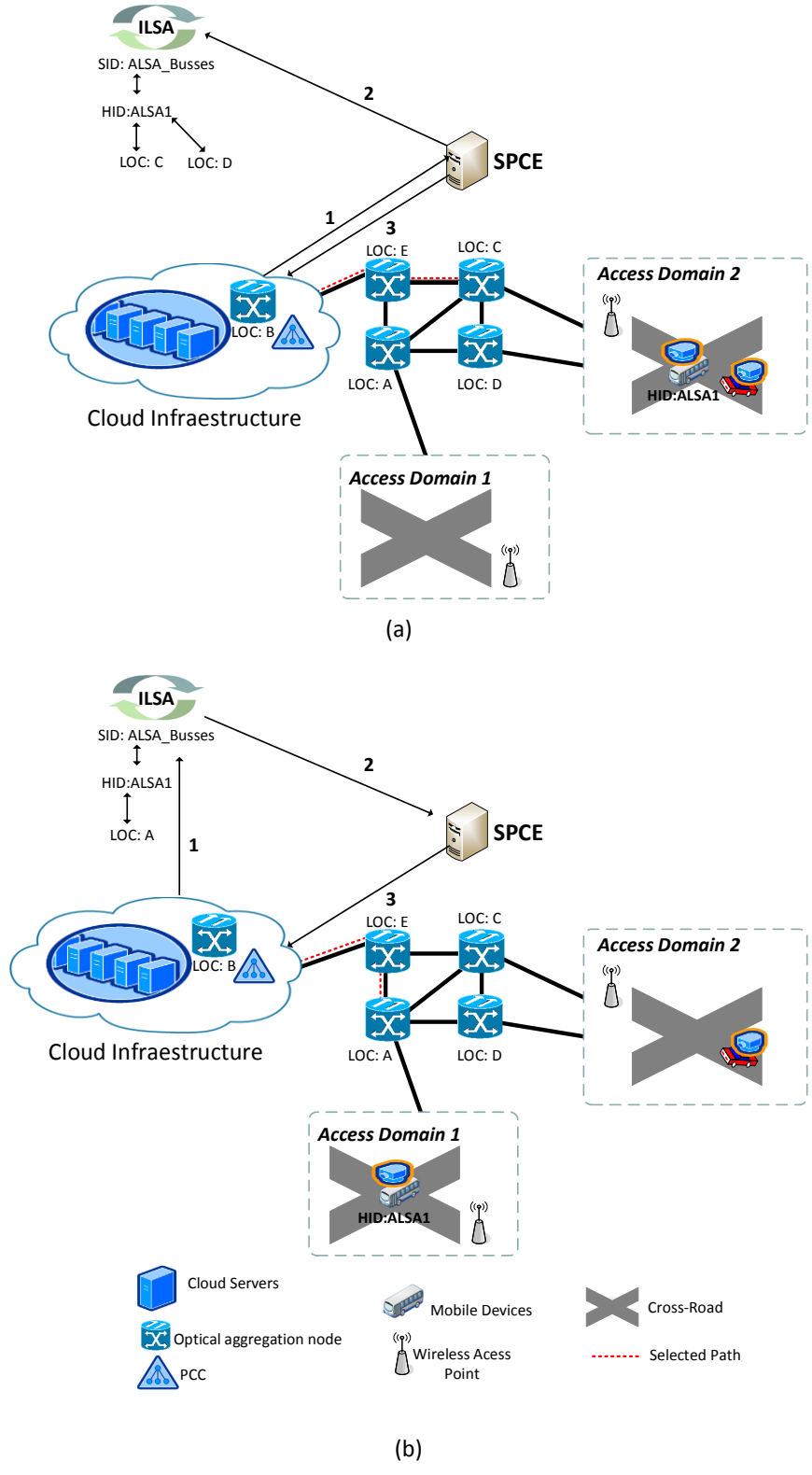


Figure 2.7: SPCE use case 1.

In general, future IoT service providers may demand communication to many distinct and heterogeneous devices, such as sensors and actuators, in order to enable the composition of distinct services. Moreover, the employment of mobile sensors can help on the coverage problem in WSN, especially in Video-based Wireless Sensor Networks (VWSN) [32], since the coverage will not rely solely on sensors deployed by the service providers, but also on NEs such as smartphones and vehicles.

Therefore, another use case of SPCE in IoT scenarios is related to the establishment of multicast connections requiring the setup of more than one lightpath. Let's consider the scenario where the goal of the PCC is to compose an image recognition service which makes use of cameras embedded on mobile NEs. This image recognition service is based on obtaining image content leveraging mobile resources in a city, and further processing in order to identify certain graphical content [33] [34]. This use case is illustrated by Figure 2.8, which shows more details about the architecture. In this example, a user located between two access points (AP1 and AP2) requests to a PCC (located in Access Domain 3 and, thus, connected to LOC:C) the provisioning of a service able to identify his/her lost dog, using cameras near his/her position. The PCC requests the establishment of paths to the more suitable resources for the service execution. The following steps describe the path computation process in this use case.

1. The PCC sends to the SPCE a PCReq with SID:CAMERA, as well as service layer requirements demanding the target resources to be placed close to the user positioning. His/her positioning is determined by the PCC through the received user-context information.

2. When a SPCE receives the request, it maps the SID:CAMERA to a set of HIDs (HID:CAR_13, HID:CAR_20 and HID:PHONE_5) that are capable of providing the requested service based on their context. Once the destination NEs are selected, the SPCE maps their location (LOC) by means of an ILSA scheme. Thus, HID:CAR_13, HID:CAR_20 and HID:PHONE_5 are respectively mapped to LOC:A, LOC:B and LOC:A. Then, the PCEP uses the NSI available in the TED to compute the path to each selected NE.

3. The PCRep containing the computed optical lightpath as well as the NEs addresses are sent to the PCC.

4. The PCC establishes the optical lightpaths to C-A and C-D-B, further connecting to the selected devices according to the routing information received from the SPCE.

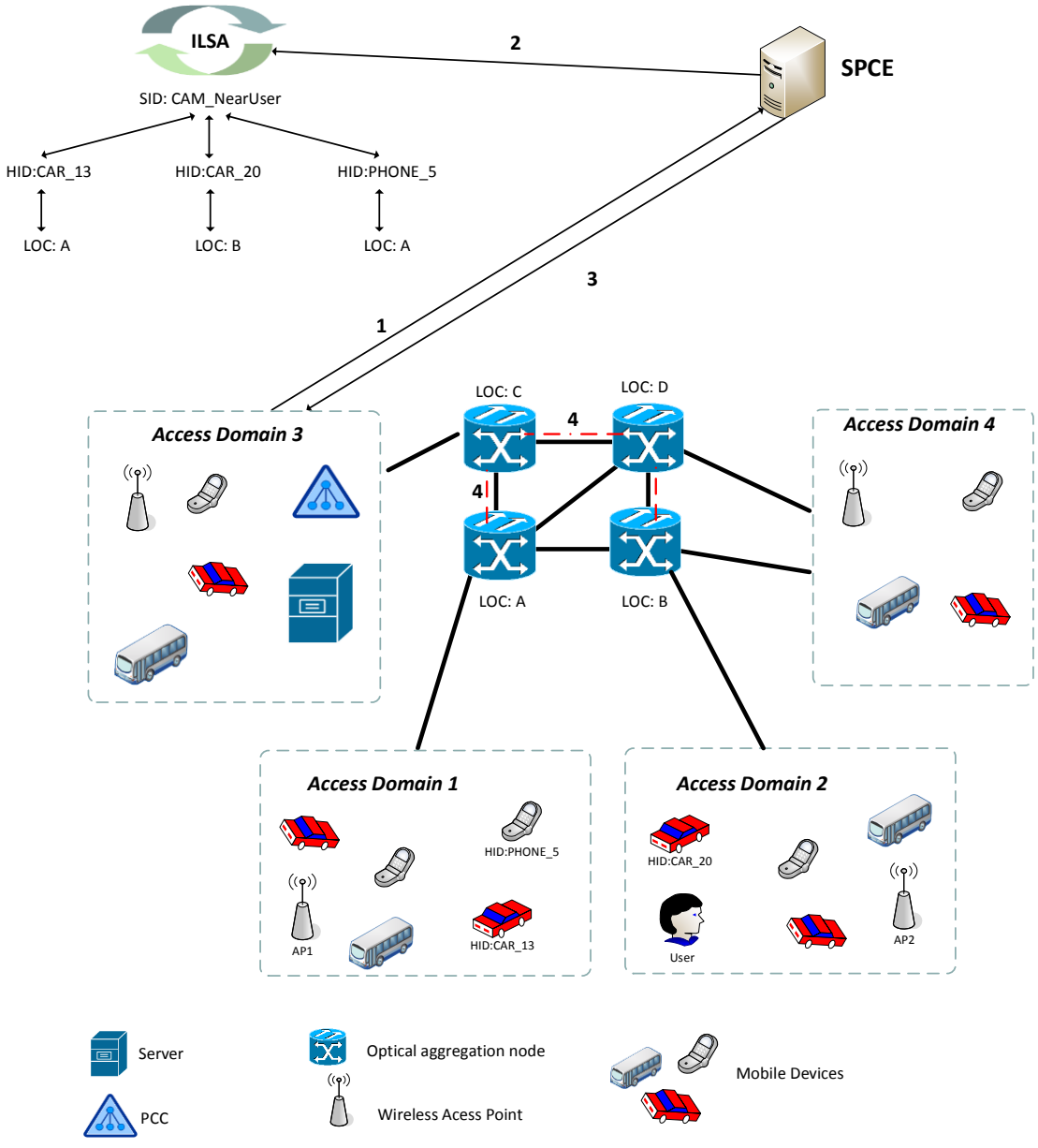


Figure 2.8: SPCE use case 2.

Routing approach for the SPCE

Conventional PCE schemes commonly follow a proactive routing approach. Under this approach, a PCE scheme uses Network State Information (NSI) stored in the Traffic Engineering Database (TED) in order to provision a service. The TED is based on NSI that is periodically updated by means of a traffic-engineering routing protocol used by the network nodes. An advantage of this routing approach is that it avoids the high signaling overhead required for the synchronization of the NSI stored in the TED. However, in highly dynamic scenarios, where lightpath are setup and torn-down in a short-term basis, the TED may be inaccurate. The negative effect of inaccurate NSI is the impact on the overall blocking probability.

On other hand, another routing approach referred to as reactive routing, assumes that NSI is collected on the fly at the time of a service request arrival. This reduces the inaccuracy related to NSI. However, a handicap of this approach is that the time required to provision a service is longer in comparison with a proactive routing approach because NSI needs to be gathered online, i.e., consider that the non-neglected delay of the network links could increase the time required to collect NSI.

It is intuitive that a routing scheme should have the benefits related to accuracy of NSI and the fast provisioning times of both proactive and reactive routing approach respectively. In light of this, we propose a SPCE that may follow a hybrid routing approach. That is, for service requests not requiring low provisioning times but demanding sensitive (susceptible to inaccuracy) NSI, e.g. because the destination nodes are moving at fast speed, SPCE might adopt a reactive routing approach. For service requests not requiring sensitive NSI, but demanding low provisioning times, e.g. emergency applications, SPCE might adopt a proactive routing approach.

2.2.4. Assessing SPCE scalability

Several works in the literature have highlighted the potential benefits on the employment of both traditionally connected mobile devices (such as smartphones, tablets, etc.) and those not initially designed to embed network functionality (such as vehicles, wearables, etc.) as mobile network elements (NEs) [35], [36], [37]. For instance, the large amount of existing mobile devices able to collect data distributed in a city can be employed by service providers and/or city administrators, diminishing the need for deploying new sensors in the city for data collection. In addition, new services may be built through data aggregation from distinct sources, creating a community effect. Moreover, the distributed nature of the NEs maintenance tasks can be highlighted, since each NE owner is responsible for maintaining its own device.

We must emphasize that, currently, mobility is globally affecting both network providers and service providers. As an example, let us consider the evolution observed in vehicular networks panorama during the last decade. The ability to connect distinct vehicles

has driven the so-called Intelligent Transportation System (ITS) paradigm, leading to the deployment of an unforeseen set of functionalities and innovative services, driving new models for smart city management, novel strategies for traffic control, as well as novel solutions for Smart-Health. We consider that ITS is a clear example on how the overall society may benefit from a proper combination of mobility and pervasive connectivity through the deployment of novel services with high added value. From the technical perspective, however, it must also be considered that ITS scenarios present inherent characteristics which lead to a contrast among others wireless mobile network scenarios, such as the displacement velocity of vehicles, which is in average much higher than the observed in NEs in other scenarios, hence, conducting to a high handover frequency on Wireless Local Area Networks (WLAN). On the other hand, it is worth highlighting that several contributions in the literature have already proposed solutions for some of the specificities for these ITS scenarios. For instance, mobility patterns assessed in [38], [39] can enable easier and faster handover prediction.

Unfortunately, mobility does not bring only advantages. While mobility is fueling new opportunities for all stakeholders, new challenges also arise. One of the aspects is related to currently available mapping schemes, such as DNS, requiring the assessment of its performance in highly dynamic scenarios. Another crucial aspect refers to the control overhead required to handle the deployment of an advanced addressing scheme such as ILSA. To deal with that issue, the scientific community is pushing for decoupling control and data plane functionalities, limiting the impact overhead they may have on the data plane.

Service provisioning specifications

In this section we present a path computation model with a clear focus on the service setup delay in order to compare the scalability of the proposed scheme. In order to illustrate the main differences, we propose a comparison between delays conducted by the SPCE setup and the host-oriented PCE computation model using DNS for mapping.

In order to calculate the delay in an IoT scenario, we consider a generic service demanding to communicate with a high number of distributed IDs. The service setup delay may be considered as the accumulative aggregation of individual delays as follows: (a) send request from PCC to SPCE; (b) mapping SID to HIDs; (c) mapping HIDs to LOCs; (d) compute paths to each LOC; (e) send the reply from SPCE to PCC; and (f) establish each path according to the reply received by the PCC. It is worth mentioning that despite the fact that (a) and (e) delays may vary slightly due to eventual PCEP modifications required by the new paradigm, we assume the difference as negligible. Delay (f) depends only on the peers establishing the connection and the use of PCE or SPCE is not relevant to the communication establishment although the use of an ILSA scheme may have a high impact on communications maintenance. However, the delays expressed in (d) are different between host-oriented PCE and SPCE due the amount of NEs used for service orchestration in SPCE, resulting in several path computation actions increasing the ILSA delay in this phase.

A significant difference in comparing delays generated by both systems falls into the mapping phase. Indeed, in the proposed SPCE architecture, the mapping phase, i.e., (b) and (c), is much different from the classical host-oriented PCE, which is concerned just on mapping one single specific ID to one specific address. Here, we particularly focus on delay in (c), i.e., HID to LOC mapping, in order to compare the mapping delay used by the proposed architecture against the current internet architecture based on DNS. The final objective is to show the scalability of the proposed model in terms of mapping latency.

Service provisioning with SPCE

In order to assess the delay resultant from the HID to LOC mapping employed by SPCE, further confronting with conventional host-oriented PCE, a model for the delay evaluation is generated. This model, as well as the obtained results, presented in the rest of this section is further described by the work available in [29].

As described in [11], some distinct approaches try to address the ID to LOC mapping in ILSA schemes, each one with pros and cons. Among them, schemes based on DHT (Distributed Hash Table) seem to be the most appealing for the IoT scenario we are considering, which needs a high scalability.

Thus, since one of the most used DHT-based schemes is Chord [40] –which provides logarithmic lookup time (O log(p), where p is the number of ID/LOC pairs in the overlay network) and requires a logarithmic amount of memory per node [41]–, the ILSA mapping scheme proposed for SPCE is Chord-based. In fact, despite not intended for an ILSA scheme, authors in [42] propose an approach for a DHT that can be easily adapted to any addressing scheme (not only to IPv4 or IPv6, which is usually the case in conventional DNS schemes). Moreover, DHT Mapping Systems have good performance in large-scale scenarios such as IoT. Figure 2.9 illustrates the interaction of the SPCE with the ILSA Chord-ring topology. The usage of the envisioned Chord-based mapping system results in the following expression to determine the delay:

$$D_{ILSA} = \text{Routing Path Length} * \text{Virtual Links Delay} \quad (2.1)$$

where D_{ILSA} is the delay for the NE selection based on an ILSA scheme; Routing Path Length is the number of hops in the Chord-ring required to reach the destination Chord-node (i.e., the Chord-node which contains the LOC of the HID being mapped); and the Virtual Links Delay is the average delay of each virtual link which composes the Chord-ring.

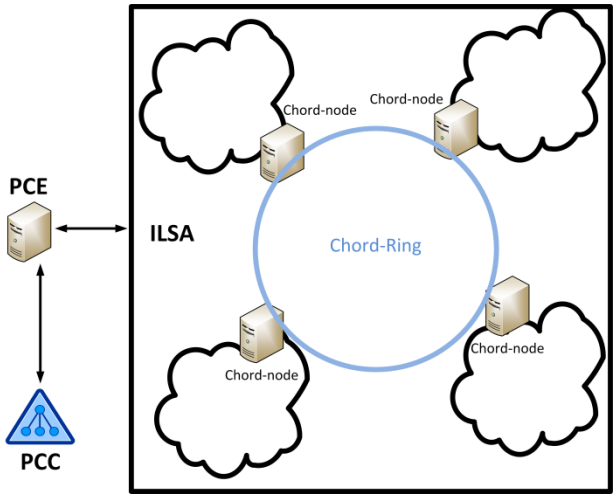


Figure 2.9: Chord-ring topology used by ILSA.

It is obvious that the average routing path length varies according to the number of Chord-nodes required to compose the Chord-ring. Thus, the Chord-ring size is a function which considers the average storage capacity of each Chord-node in bytes (SC), the total number of HIDs (N_HIDs), and the size of each entry of the mapping system used to map a HID to a LOC (SEntry), as shown in the following equation:

$$N_{chord\_nodes} \geq N\_HIDs * SEntry / SC \quad (2.2)$$

where N_{chord_nodes} is the minimum number of chord nodes needed to create the Chord-ring topology. In addition, according to [40], the average routing path length for a DHT-based mapping system, arranged as ring, such Chord, can be described as:

$$\text{Routing Path Length} = \tfrac{1}{2} \log_2 (N_{chord\_nodes}) \quad (2.3)$$

Host-oriented PCE

As mentioned, our focus in this section is to show the scalability of the proposed model in terms of latency, especially in relation with the mapping delay. As a host-oriented architecture, we assume the traditional PCE makes use of DNS servers in order to map the destination ID to an IP address.

Nevertheless, the DNS mapping time depends on many different configurations on the servers, including its location, number of supported entries, TTL of each DNS record type (e.g., A, NS, and PTX), server load, number of retransmissions, the algorithm used for caching and its efficiency on hits/miss, etc. Thus, because of the high number of variables, many works assessing DNS servers limit their analysis only to observations of real values contained in the log files of the evaluated servers (e.g., [43], [44]). Hence, instead of

computing the DNS delay through mathematical models, we used values provided by these state-of-the-art contributions.

Evaluation

In this section, we present preliminary results of service setup time according to the time model presented for the chord-ring topology employed by ILSA. We compare the results of the SPCE proposed architecture, making use of ILSA mapping strategy, and the host-oriented PCE, with DNS mapping.

To achieve the presented results, we adopted as parameter the value of 10ms for the average Delay of the Virtual Links. The minimum numbers of Chord-nodes to compose the Chord-ring according to the number of HIDs and the average Chords-nodes storage capacity were obtained by using Equation 2.2 and are presented in Table 2.2. All presented results were generated considering HIDs with 20 bytes and LOCs with 12 bytes, i.e., 32 bytes for each entry.

Table 2.2: Required number of chord-nodes

	Storage capacity of Chord-nodes			
Number of HIDs		**65536**	**262144**	**1048576**
	30000	25	7	2
	40000	30	8	2
	50000	35	9	3
	60000	40	10	3
	70000	44	11	3
	80000	49	13	4
	90000	54	14	4
	100000	59	15	4
	110000	64	16	4
	120000	69	18	5

Figure 2.10 shows a comparison of the delay obtained to map one single ID (HID in SPCE) into a LOC (address in DNS-based) for the time model presented for the SPCE vs the values observed in [43] and [44], which consider the use of DNS servers in order to request a path to only one destination node. We show in the x axis the overall amount of potential IDs and, as expected, the larger this number is, the larger the mapping time consuming will be.

On one hand, we are aware that future works must consider not only one NE in order to enable service orchestration, and the mapping of distinct IDs will introduce an extra delay. Also, the path computation in an IoT scenario may result in a substantial increase of the SPCE path computation time, in comparison with the host-oriented PCE with less NEs. On the other hand, the results shown in both Table 2.2 and Figure 2.10 illustrate a very small scenario with a low number of IDs in order to enable the comparison with presented DNS results. For instance, the Chord-nodes storage capacity of the results presented in Figure 2.10 is limited to only 65536 bytes.

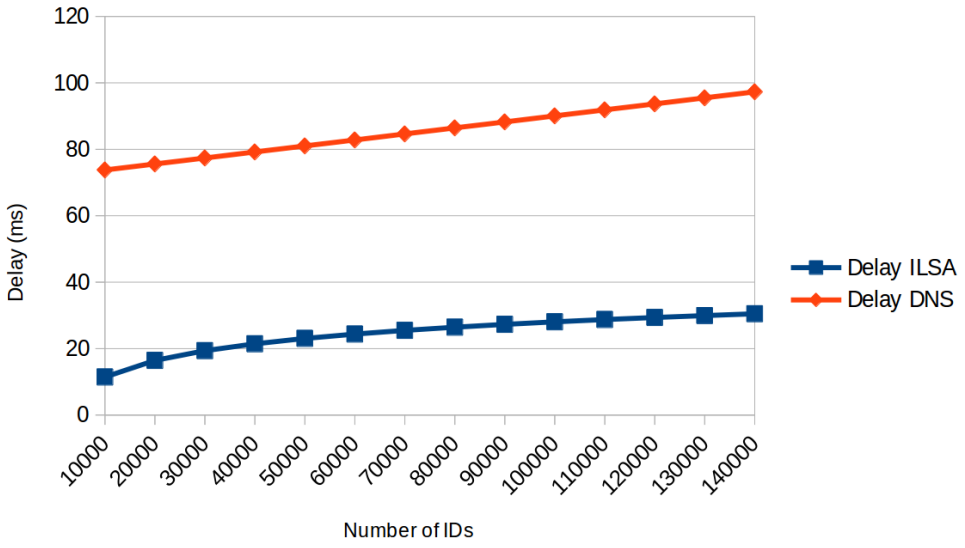


Figure 2.10: Delay variation in ILSA and DNS mapping schemes.

To compare these results of SPCE with the traditional host-oriented PCE using DNS, we need to take into account the described characteristics of IoT as dynamicity and the high number of nodes. Thus, because of these characteristics, we assume that to resolve a DNS lookup in this scenario the number of referrals would be increased. According to values observed in real DNS servers by [44], 40% of lookups with one referral are resolved in more than 100ms and, when using two or more referrals, more than 95% of lookups have latency longer than 100ms and 50% have latency longer than 1000ms.

The presented values show that the use of ILSA may be a good alternative to the DNS based schemes when developing networks with a high number of IDs, such as in IoT

scenarios. For instance, [44] states that NS records tend to have a TTL much larger than A records. It is also claimed by [45] that the use of TTL value 0 for A records have a limited impact on traffic load when using DNS on mobile networks, but in IoT, with a large number of mobile nodes and a high mobility rate, such as in vehicular networks, this may not be true.

3. Moving from SPCE to F2C

In this work, we have firstly envisioned the tailoring of the PCE network routing technology for enabling its employment as a service routing solution in a service-oriented scenario. However, some of the characteristics observed in the PCE architecture make this technology to be not the best option for the envisioned future IoT applications, as described in the following lines.

- Albeit ILSA mechanisms has shown a high scalability for resource mapping in a centralized scenario, new IoT services are expected to require real-time service placement. This includes the network transmission delay, which can yield a high impact, especially when employed mapping servers are located at DCs geographically located at large distances from end-users, such as it happens in web services deployed in cloud premises.

- Required signaling traffic generates a high load at the network core. This is valid both for the demanded updates to DCDB and updates to the mapping information placed on distributed ILSA servers. Furthermore, the need for frequent communications of SOM and PCM with ILSA servers yields an extra load on network links.

- Since end-user personal data needs to be transmitted in order to allow the deployment of certain services, security and privacy concerns must also be mentioned as a drawback to centralized mechanisms, such as SPCE, where sensitive user data must travel through larger distances, augmenting the risk of being captured by attacks.

- Since it is based on a routing technology, the SPCE aims at the provisioning of the best path to connect a service provider to the most suitable resource at the edge of the network in order to provide the required data. This collaboration model limits the amount of resources that may be shared by end-users to data collection resources.

However, in parallel to the conducted research, a new IoT paradigm comes up with the empowerment of novel network architectures, such as fog computing. This new scenario paves the way to reformulate the problem, looking for a solution much more suitable than a PCE based approach. Therefore, based on benefiting from cloud and fog computing, we

have envisioned an alternative that may enable further collaboration between core and edge resources. We believe that this collaboration is a trend on future network applications since several QoS parameters demanded by real-time applications, such as low network latency, may be provided by the employment of resources at the edge of the network. Moreover, besides reducing the perceived service latency, the employment of fog resources may also optimize network usage and increase private data security by diminishing the distance between the end-user and the IT premises running the real-time services.

Furthermore, different from the PCE based approach, the employment edge resources in fog computing is not limited to gathering of data for processing at service providers. Rather, the employment of idle edge resources, including processing and storage resources, enables the creation of highly virtualized micro data centers (MDC) geographically close to end-users.

In the next chapter, the background technologies employed for the rest of the conducted work are presented. These technologies include the cloud computing –with special focus on its application on IoT scenarios–, fog computing and combined Fog-to-Cloud model.

4. New IoT paradigm: Fog-to-Cloud (F2C) computing

4.1.Cloud computing in IoT scenarios

Advances on the mobile computing paradigm, including emerging wireless communications technologies, as well as the development of low-cost sensors and the growth of smart data processing in M2M communication, have recently facilitated the deployment of the so-called Internet of Things (IoT). In parallel, the evolution on DCs has also enabled the development of cloud computing, providing virtually unlimited processing and storage capacities in an on demand self-service, scalable, location independent, pay-as-you-go online computing model. Indeed, cloud servers are composed by a set of DCs offering distinct cloud service-on-demand models, such as Software as a Service (SaaS), Platform as a Service (PaaS) or Infrastructure as a Service (IaaS).

The IoT and cloud computing wedding has enabled the development of services requiring the soaring cloud servers' capabilities in conjunction with the distributed data-on-demand provisioning enabled by IoT devices deployed in smart environments [4]. This symbiotic mutualistic relationship enables IoT services leverage cloud capacity in order to store myriads of data generated by the huge amount of sensors deployed in a distributed fashion in distinct environments, such as cities or industries. Furthermore, the capacity for analysis of unprecedented complexity enables efficient data-driven decision making and prediction algorithms employment in IoT scenarios. On the other hand, cloud service providers leverage the distributed real-time data provisioning enabled by the distributed real world things [46].

The advent of this new paradigm, enabling ubiquitous and pervasive computing, has undoubtedly driven several IoT services, tailored to distinct scenarios, to pop up, as shown in Figure 4.1. These services include Personal Health Assistant (PHA), Intelligent Transportation Systems, Smart Home, Environment Monitoring, among others [1]. Healthcare services, for instance, may enable remote medical supervision to patients at their home through the employment of body sensors, enabling measurement of several body characteristics, such as pulse rate and blood pressure [47]. On the ITS domain, vehicular data clouds can store several dynamic information regarding vehicles, such as individual mobility profile, vehicles maintenance data, among others. Hence, service providers may act proactively managing traffic, reducing road congestion, increasing road safety, recommending car maintenance, etc. [48].

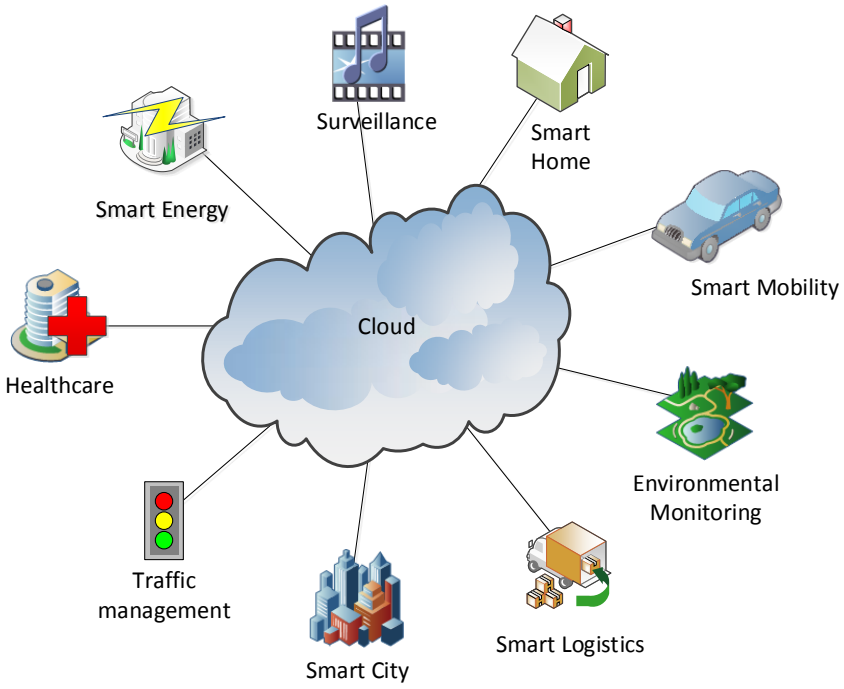


Figure 4.1: IoT applications enabled by Cloud.

This smart scenario is expected to grow with no foreseen limits, fueled by the unstoppable technological evolution which unquestionably promotes the continuous generation of new edge devices, with ever more powerful capacities, including the capacity to collect large amounts of data and thus fostering the development of new services with more demanding needs in terms of capacity, quality delivery, security or end-user latency. This situation of continuous evolution produces a growing in spiral with no end, so that the higher the edge devices capacities are, the more complex and exigent the requirements of end services will be.

A relevant number of works in the literature have focused in solutions for distinct issues regarding service allocation in cloud, such as QoS-aware scheduling mechanisms, VMs management and migration with dynamic load prediction, as well as QoS/green computing trade off [49] [50] [51]. In fact, albeit the effort expended by the community on green DC implementations, this topic persists as one of the main subjects in DC researches, turning out new challenges in the employment of cloud computing in IoT scenarios [49] [50]. However, in such scenario, green computing strategies are usually limited to the management of workload within cloud resources. Hence, strategies such as load prediction algorithms are employed in order to distribute the VMs efficiently, minimizing the number of used physical machines and allowing some of them to be switched off, while avoiding severe overload.

The deployment of an integrated control plane for geographically distributed DCs has also been subject of study recently. In [52], authors introduce an enhanced network control

plane architecture leveraging distinct strategies for network and IT resources aggregation in DCs. Furthermore, the paper also proposes an orchestration mechanism to enable the identification and selection of DC services, by extending PCE to an IT-aware architecture. To that end, an extension to PCEP is proposed –incorporating new parameters to PCEP notification message embracing IT and storage information–, facilitating an IT-aware path selection. Albeit that work presents some good proposals for service allocation in cloud DCs, it does not consider scenarios with high resource dynamicity, such as those envisioned in IoT. Hence, it lacks an efficient mechanism to keep an updated resource database.

Although several researches in cloud technology have been conducted, the intrinsic distance from cloud resources to end-user is hindering the deployment of IoT services with specific requirements. The latency introduced by the communication between end-points in a cloud centric IoT contrasts with the expected QoS on real-time services. Healthcare, for instance, may require the appropriate actions to be taken immediately when abnormal scenario is met. In a similar manner, autonomous vehicles must recognize environment modifications immediately. Other sensitive IoT scenarios include real-time monitoring in manufacturing plants or real-time navigation in traffic control systems [53].

4.2.Fog computing

The demand for extending the cloud computing model in order to diminish the communication latency between end-user and cloud resources, along with the smart scenario promoted by IoT and the increasing hardware capacities observed at the edge devices –notably mobile devices, such as phones or vehicles–, stimulated the advent of fog computing [17]. The main rationale behind fog computing is to bring the IT resources closer to the end-user aiming at the provisioning of reduced and predictable latency for IoT services offloading by making use of the computational resources of NEs located at the network edge in a highly virtualized and distributed fashion. With this approach, a service may avail itself from the geographically distributed NEs in scenarios such as Smart Cities or Smart Transportation Systems, among others, decreasing the demand for cloud resources.

Besides preventing high network delays on service transmission, the use of available IT resources located at the edge, combined as micro data centers, may further reduce the overall Service Response Time (SRT). Moreover, additional benefits may be attained from proximity between end-points, contrasting with the issues arising from the strict employment of cloud data centers in IoT scenarios, as follows.

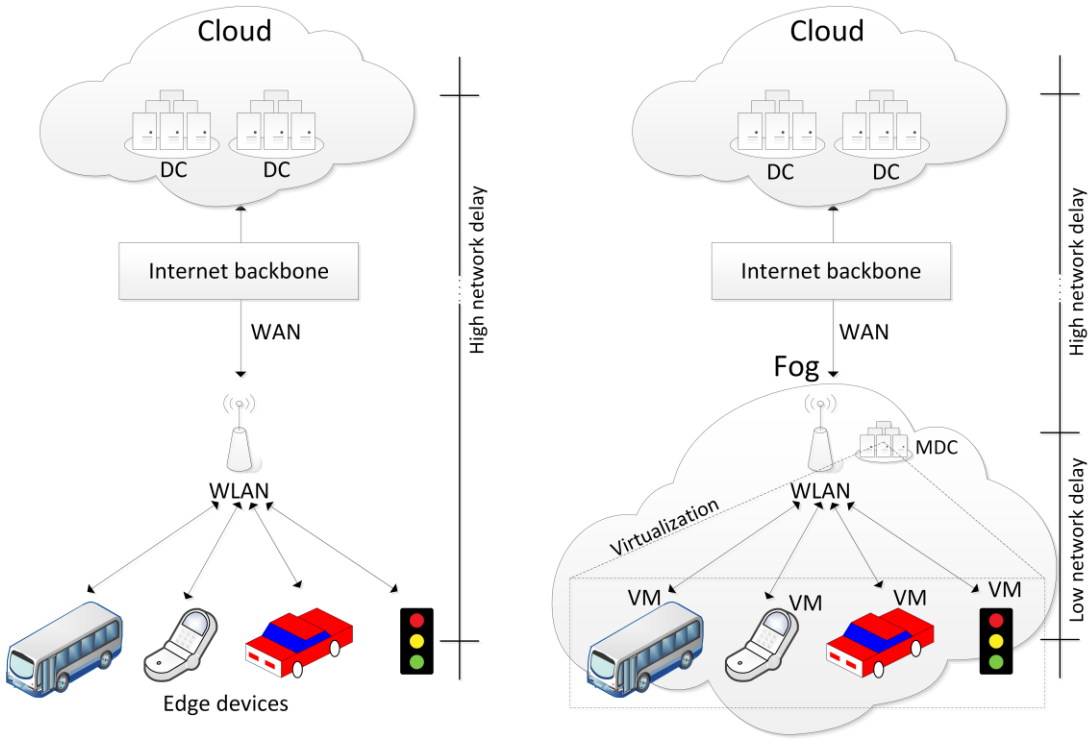
- Improved security and privacy for the end-user since, for specific applications, personal data can be processed at fog resources, rather than requiring it to be sent to cloud, increasing the risk of attackers to obtain sensitive user data.
- Reduced bandwidth consumption at internet backbone due to diminished traffic at the network core.

- Reduced energy consumption at cloud data centers as well as at internet backbone, also resultant from the diminished usage of resources at the network core.

- Efficient synchronization of distributed control planes since each fog is responsible for underlying resources in its respective area in a hierarchical view, not requiring the complete view of resources managed by distinct fog servers [53].

- Reduced internet connection costs for end-users since several services shall be deployed through WLAN, diminishing the amount of data required to be transferred to cloud servers through costly mobile cellular networks.

The comparison between cloud and fog architectures, in terms of latency, is further illustrated by Figure 4.2. Notice that, rather than establishing a high delay communication between cloud DCs and edge devices –such as sensors and end-user devices–, the fog provides a low delay MDC by leveraging idle resources shared by resources at the edge of the network.



(a) (b)

Figure 4.2: Distinct architectures applied to IoT: (a) Cloud computing (b) Fog computing.

It is worth mentioning that even the ISPs are likely to embrace the combined paradigm of fogs and clouds as opposed to the cloud only, due the concerns related to both power consumption and the amount of network bandwidth that must be dedicated to cloud computing connectivity. Recall that cloud infrastructures commonly demand high-speed Internet connectivity from ISPs in order to support the cloud services demanded by end-users all around the world. This network overloading is ever stressing telecom companies mainly because this network traffic increment does not turn into additional revenues, with the aggravating circumstance that network equipment should be continuously upgraded to meet the strong conditions posed by IoT services –data collection at the edge and data processing at cloud premises–, with no benefit for telecom companies.

Nevertheless, as a new technology, fog computing is also surrounded by major issues which, if unaddressed, may hinder its real deployment and exploitation as well as limit its applicability. Several constraints are imposed by the NEs in a fog scenario, mainly related to their reduced capacity in terms of processing and storage in comparison with the cloud, as well as reduced availability stemmed from their mobility, volatility and limited energy source causing undesired service disruptions. Thus, the objective of fog is not to replace cloud. Rather, they may be considered as complementary, since the most suitable resources should be allocated for distinct services execution, be it in fog, be it in cloud.

From an implementation perspective, in order to enable the creation of interoperable and secure fog systems, the OpenFog Reference Architecture (OFRA) [54], created by the OpenFog Consortium, defined the most important features characterizing fog computing, as well as the pillars of OFRA: security, scalability, openness, autonomy, programmability, RAS (reliability, availability and serviceability), agility and hierarchy. Moreover, the OFRA further highlights the collaboration of fog computing with cloud computing in a hierarchical topology.

Nevertheless, what is unquestionable is the need for a coordinated and efficient management of the diversity of resources distributed at fog and cloud premises. Therefore, a novel computing paradigm, designed to cope with these demands while taking into consideration the specificities of next generation Internet services and resources, is introduced in the next section in detail.

4.3.F2C architectural model

Fog-to-Cloud (F2C) has been recently proposed in [18] as a novel computing paradigm benefitting from cloud and fog premises availability, proposing a layered architecture, where heterogeneous resources are dynamically and hierarchically distributed according to key characteristics, including information related to available IT resources, energy and mobility profile, vicinity, among others [55]. By working in a collaborative model under a coordinated and orchestrated management, F2C aims at the provisioning of enough capacity

to execute services while getting the most of the resources according to the layer they are located. Therefore, client requests may be executed in lower service response time, reduced network load and better energy efficiency.

The combined F2C model envisions a hierarchical architecture leveraging both cloud and fog frameworks in a coordinated fashion, making it possible for new services to be created, but also improving the whole performance, considering for example execution time, parallel execution, edge processing, fog security, low resources utilization and high energy efficiency. To this end, a comprehensive control and management strategy must be defined, addressing coordination aspects as well as the different relationships to be set among the different cloud/fog components

The collaborative model adopted by F2C architectures paves the way to design new service execution strategies –enabling distributed execution of services over multiple devices– properly tailored to the dynamic, volatile and heterogeneous resources scenario defined in F2C. Therefore, the F2C coordination enables the distributed service execution not only in a horizontal fashion, within fogs presenting resources with similar characteristics, but also in a vertical fashion, where a service can be distributed within layers consisting of resources with distinct characteristics. The service distribution is performed through the mapping of sub-tasks of a service into the most suitable resources, according to sub-tasks demands, be it at fog, be it at cloud.

Therefore, distinct devices are distributed into distinct architectural layers according to their respective characteristics, including processing capacity, energy availability, mobility profile, and communication technologies, just to name a few. Unlike common hierarchical topologies, such as hierarchical cloud implementations, the layered topology presented in F2C is dynamic, in accordance to the dynamicity observed in IoT scenarios. Hence, resources' context plays a key role on the definition of their distribution within distinct architectural layers in a given moment. For instance, a static connected resource may be logically moved into a distinct layer if it is disconnected from power source or starts moving.

The 3-layered F2C topology depicted in Figure 4.3, based on the one proposed in [55], considers the mobility and proximity to end-user as the main attribute for layers definition. Hence, the lower layer (fog layer 1) is composed mainly of resources on the move, such as vehicles and smartphones, providing low network latency at the cost of a higher disruption probability. The upper fog layer (fog layer 2) is composed of permanently or temporarily static resources in a smart scenario whose network latency may be higher than the perceived in the first fog layer resources, albeit it is still considerably lower than the cloud communication delay. This layer may embrace both fixed resources in a building and resources provided by cars in a parking lot, for instance. The cloud layer is formed by reliable, dedicated and static servers with on-demand computing features such as processing and storage capabilities, accessible through the Internet backbone, providing nearly unlimited resources.

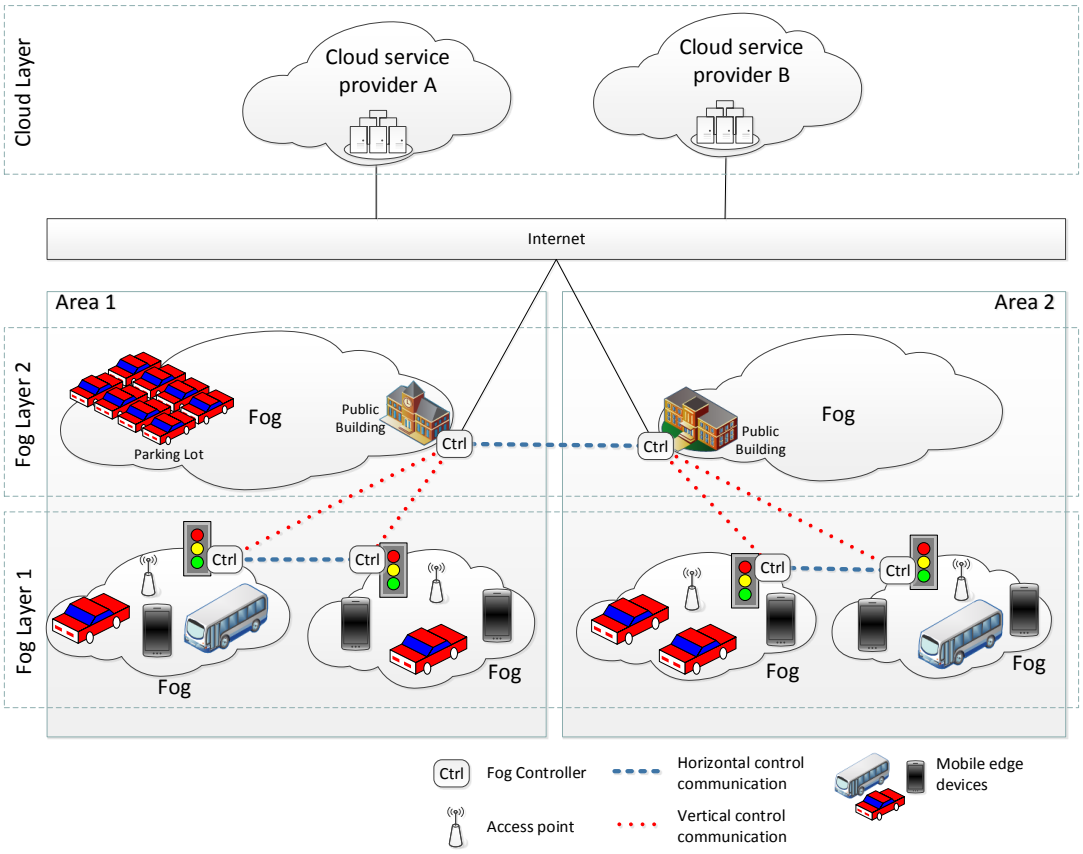


Figure 4.3: 3-layered topology example for F2C.

As previously mentioned, the high capacity is often provided in the cloud at the cost of high access latency, whilst the opposite is observed in the fog. Therefore, fog controllers at the lower layer are responsible for the discovery and management of a large amount of edge resources, possibility on the move, keeping individual context information. Thus, it can map services with low requirements in terms of processing and storage capacity, as well as overall execution time, into the most suitable edge resources.

On the other hand, fog layer 2 controllers keep information regarding more static resources, which, in comparison to resources at layer 1, may be easily clustered according to pre-established policies providing higher reliability and lower volatility. Moreover, controllers at this layer may keep aggregated information regarding available underlying resources, enabling proper forwarding of services demanding low requirements to the most suitable underlying fog.

The distributed control employed by F2C introduces a number of challenges which should be overcome in order to enable the deployment of such architecture, including semantic adaption, coordinated layer orchestration, cloud and fogs identification, as well as resources discovery and service allocation, just to name a few. Some of these issues are being tackled by an H2020 European project so-called mF2C [56], which is an ongoing work aiming at designing and implementing, for the first time, the F2C architecture in a real scenario. The research presented on the following chapters of this thesis is mostly focused on the assessment of some of the service allocation related issues.

The service allocation itself must cope with the specific aspects associated to cloud and fog resource models, requiring distinct strategies to properly map IoT services into the most suitable available resources, taking into account the type of shared resources, capacity of resources, energy consumption, network transmission delay and load balancing, among others. Moreover, albeit some proposals may be found in the recent literature discussing about the service allocation on Clouds and Fogs, many issues remain unsolved yet, since studies on F2C are still premature.

On the other hand, the benefits of F2C architectures may be diminished by failures affecting the computing commodities driven by the dynamicity and volatility imposed by edge devices. Indeed, these failures may be prohibitive for the achievement of the envisioned performance in F2C systems, posing extra challenges for the service allocation. In order to withstand possible failures, the design of novel protection strategies, specifically designed for distributed computing scenarios is required.

Moreover, the successful deployment of real-time services, demanding very low delay on the allocation of distributed resources, depends on the assessment of the impact of controlling decisions on QoS. The employed control topology, for instance, may highly affect control decisions latency, having a decisive role on the QoS-aware service allocation. The control topology definition in a layered management architecture, such as in F2C systems, must include the evaluation of distinct parameters, including individual controllers

capacity, overall amount of controllers, and the number of layers in the F2C control topology.

In addition to what have been stated so far, it is undeniable that a key aspect in the F2C design refers to security, since F2C arises security issues besides those yet unsolved in fog and cloud. Unfortunately, security strategies employed by cloud computing require computation power not supported by devices at the edge of the network, whereas security strategies in fog are yet on their infancy. Undoubtedly, there is a clear need for novel security strategies able to handle all components in the F2C architecture. In the next chapters, each one of these aspects are discussed in a comprehensive manner, through the presentation of strategies, mathematical models and attained results.

5. Service allocation in F2C

5.1.Challenges on service allocation at the edge of the network

Service allocation in fog computing is currently more subject to research than deployment. Indeed, cloud and fog, even though conceptually similar, present crucial differences. Unlike cloud computing, any fog computing system must deal with the intrinsic constraints brought by the edge devices, such as mobility, energy issues, reliability, and heterogeneity, among others. These characteristics are raising several challenges for the service allocation in F2C scenarios, which are introduced in this section, along with recent efforts on the service allocation at the edge of the network.

Real-time service execution requires the allocation of resources close to end-users, enabling low service transmission latency, while considering edge devices constraints, such as limited capacity in terms of both IT resources and available battery. Therefore, when a set of requests is received, each request must be mapped into the most suitable resources in an optimal fashion, considering the high heterogeneity perceived in the edge as well as distinct service requirements. The service allocation must consider service characteristics such as type and capacity of required resources, execution priority and estimated energy consumption, just to name a few.

Several studies have assessed scenarios where, such as F2C, fog commodities are placed at the edge of the network and the cloud infrastructure is closer to the conventional network backbone [57], both operating together. However, it is with no doubt that more research efforts are required to distill the challenges related to the actions supporting the allocation of IoT services in scenarios, especially in F2C.

Service allocation in cloud computing is well positioned in network research, in comparison to fog computing. Key studies describing the needs and the structure of data, management and control plane schemes for cloud can be found in [58] or [59]. Related studies focusing on control actions needed for service allocation can be found in [60] or [61]. In [62], authors target the allocation of resources for clouds in dynamic scenarios. Other studies, either present a methodology for service allocation in cloud scenarios [63], or consider service allocation in mobile cloud scenarios [64].

However, the research on the fog field is still immature. There are no standards nor a widely accepted position related to what defines the overall combined fog-cloud

architecture. On the fog computing side, several works assessed the use of IoT devices as source of data for cloud services. Only few works have explored the potential benefits, in terms of SRT, brought by using the processing capacity of devices at the edge of the network, as first proposed by fog computing. For instance, authors in [65] describe the so-called FUSION framework and its architectural aspects regarding services orchestration and selection at fog (here called "edge clouds"). The work aims at achieving low delay on service execution focusing exclusively on data acquisition delay, where fog devices are data sources of which the resources are not shared with other fog clients. This means that resources volatility or data quality are not considered. In the same way, albeit the Real-Time Thing Allocation algorithm (RTTA), proposed by [66], is demonstrated to be a run-time computation efficient heuristic, only data collection devices are considered and, besides that, the only evaluated constraint is the energy consumption, assuming that all other resources are unlimited.

On the other hand, the work in [67] presents a map-reduce strategy for splitting a complex service into small tasks enabling their parallel execution within edge devices, with further combination of results. However, the focus on the distributed execution of services demanding high processing capabilities on CPU constrained edge devices yielded results worse than the centralized execution. However, authors argue that their solution is useful for service providers with small/moderate budget running delay tolerant service. Albeit [68] assesses the service allocation and energy consumption in fog and cloud computing systems, the presented architecture have emphasis on static service planning scenarios; in other words, the set of services to be allocated is known in advance. Moreover, the presented architecture is not F2C-compliant.

The work presented in [69] proposes a strategy for mobile service offloading to a remote server when a static threshold is surpassed. However, the strategy does not allow service partitioning and, thus, parallel execution. Rather than employing language-based fine-grain solutions, such as JVM (Java Virtual Machine), for enabling service division, [70] proposes a coarse-grained and language-independent approach. It is stated by the authors that a service developer is able to quickly apply modifications to the code with little training, tailoring complex applications, obtaining few errors and high quality.

The work in [57] introduces a prediction strategy for the pre-allocation of resources in fog scenarios based on customer's loyalty, measured by service relinquish probabilities. This study is focused on scenarios where customers have intermittent connectivity, rather than on mobility.

In this chapter, we introduce and formulate the service allocation problem in F2C scenarios, whose main goal is to minimize the latency experienced by a service to reach out to the resources meeting the service requirements. Whilst sections 5.2 and 5.3 limit the problem formulation to planning, the service allocation in section 5.4 is extended to an online scenario. These ideas, presented in this chapter, have been previously described by the works published in [55] [71] and [72]. The formulated service allocation problems differ

from previous works in the literature in the following: (1) it does not limit the service allocation at the edge resources to data acquiring; (2) it contemplates distinct constraints inherent to edge devices such as processing and energy limitations; (3) it is compliant to F2C distributed and parallel service allocation.

5.2.Preliminary approach for service allocation in F2C

Scenario description

The envisioned F2C scenario embraces both fog and cloud resources, which are hierarchically distributed in a vertical manner into three distinct layers, as shown in Figure 5.1. In the considered model, the hierarchy of resources is determined by their capacity, as well as their vicinity and reachability to end-users. For instance, devices presenting high mobility, such as vehicles traveling along a road, have low reachability to end-users, since the constant movement, especially for vehicles in high velocity, increases the disruption probability of eventually allocated services due to the reduced time they will be on the scope of end-users. Figure 5.1 also incorporates the so-called Access Network, including the connectivity and devices on the user side to allow the users to access the required resources to run a service. In the following lines, more details regarding distinct F2C layers are provided.

- The access layer is composed by end-user devices connected by means of distinct access technologies, such as WiFi or 4G. These end-user devices may both request and offer resources to the F2C model.

- The first fog layer (fog layer 1) is the one closer to end-users, which are usually connected to this layer through a single-hop wireless connection. This layer is composed by fog servers with low capacity, which aggregates available resources on the access layer and guarantees a low-delay access to nearby users through a pool of virtualized resources. An example of the first fog layer is the concept of vehicular clouds, which are mobile scenarios formed by vehicular networks providing cloud computing features [73].

- The second fog layer (fog layer 2) is composed by fog servers embracing both nomadic and fixed devices. The resources aggregated by fog servers in this layer are in a neighborhood wide area, enabling a collaborative sharing with medium capacity and latency. For instance, fog premises in public buildings or vehicles in a parking lot may share their resources, enabling the creation of a micro datacenter able to offer medium access latency.

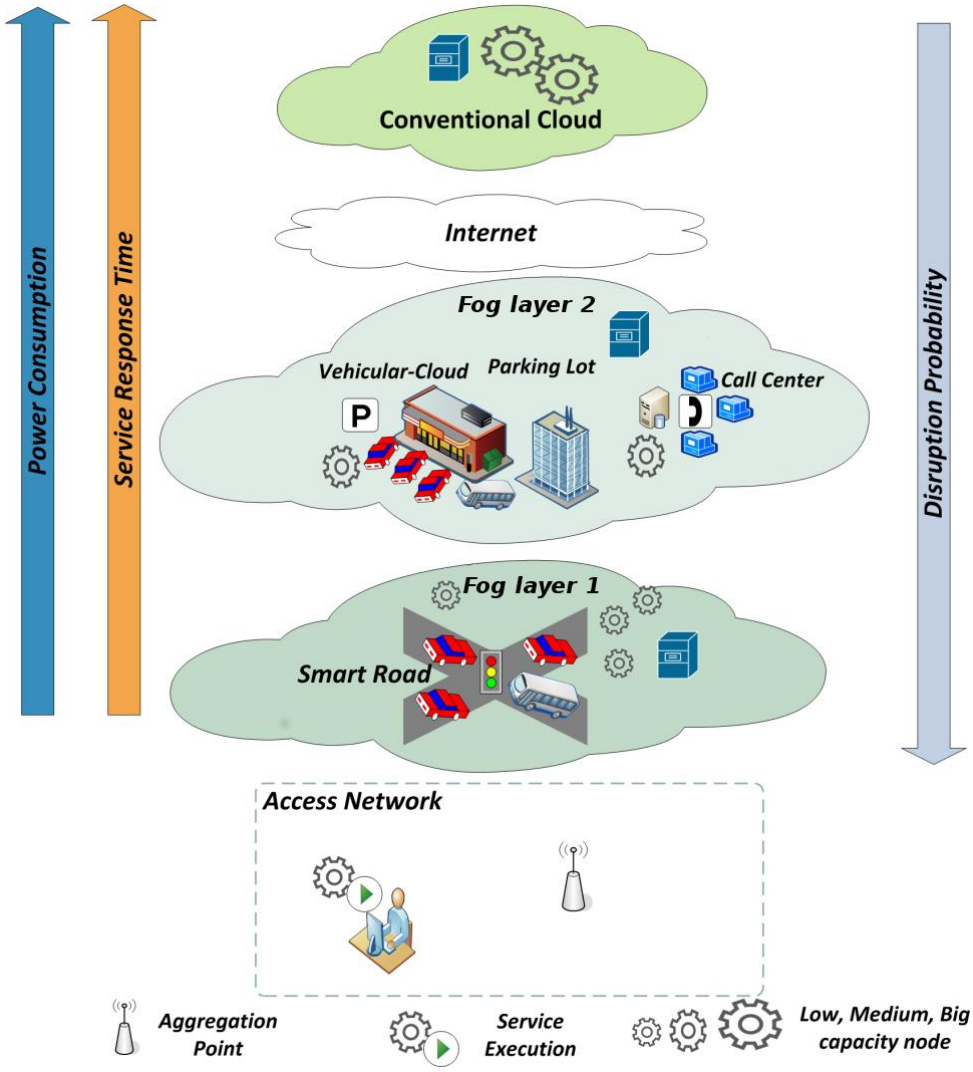


Figure 5.1: Combined F2C architecture layers.

- The higher layer is formed by conventional cloud servers, accessible through the Internet backbone, providing almost unlimited resources at the cost of a high latency.

System model

Since the deployment of architectures based on the IoT concept faces plenty of difficulties, this section is focused on the selection of resources for services allocation considering a simplified use case. The following assumptions are considered.

- All service demands are coming from end-users directly connected to fog layer 1 (see Figure 5.1);
- The described model is generic in terms of the employed resources. Hence, it considers one resource type that is demanded by all services, though may be allocated by one unique service in a given moment, e.g., CPU;
- Albeit services are demanding the same resource type, services are categorized into two distinct types in terms of the amount of required resources, as described later in this section;
- The total capacity of each fog, as well as the amount of resources required by services, are represented through the number of slots, i.e., a slot is the measurement unit used to represent the minimum resource allocation.

The NSI, and IT resources information employed for the service-resource mapping are represented by a static graph, containing exclusively the accessible resources in a given moment and the total slots available at each node, as shown in Figure 5.2 –we consider a F2C node to refer to a fog or cloud in F2C topology, as in [74]. We consider an IoT resource accessible when a connection to this resource may be set regardless of the number of available slots. It is worth mentioning that, as aggregation points, the fog layer 1 nodes capacity is precisely the sum of individual capacities of each accessible end-user device in the underlying layer, as showed in Figure 5.2. Moreover, the distributed allocation of a service may be accomplished by employing slots of distinct fog nodes (including nodes in distinct layers).

Leveraging this static representation of the resources topology, we propose to model the service allocation problem as an ILP problem. Therefore, we highlight two main goals: i) to obtain a low latency on mobile services transmission; ii) the need for provisioning the highly demanding requirements of both existing and future IoT services.

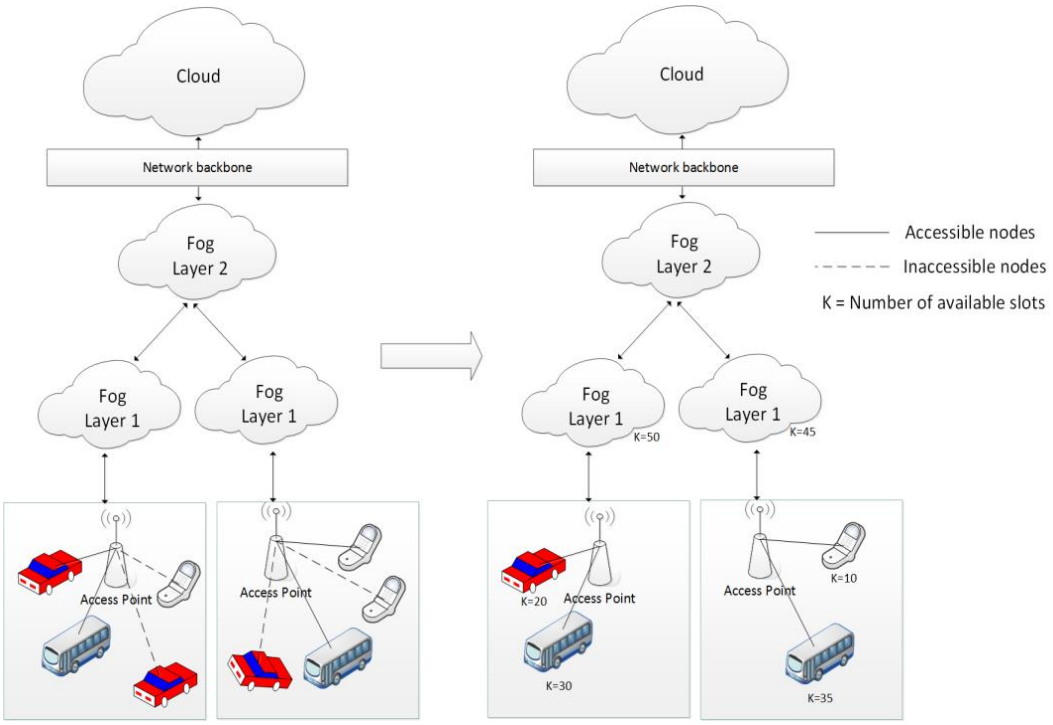


Figure 5.2: Used NSI in terms of reachability and capacity of IoT user-devices.

That said, in the modeled ILP problem, our main objective is to minimize the total delay for services requesting resources. In the model, the delay D observed by the set of services S is minimized according to Equation (5.1). For the sake of understanding, Table 5.1 defines the metric values used in this section.

$$\text{min: } \sum_{i=0}^{|S|} D_i \tag{5.1}$$

Equations (5.2) and (5.3) represent nodes and slots limitations. Equation (5.2) is the constraint responsible for avoiding the allocation of a number of slots higher than the F2C node capacity, whereas Equation (5.3) prevents the same slot of a resource to be used by more than one service simultaneously. It is worth mentioning that the model meets the described topology, where two sort of fog nodes with distinct capacities, i.e., distinct number of slots, are considered according to the layer they are located at, as well as one cloud node able to run all services in S, i.e., with unlimited capacity in the problem scope.

Table 5.1: Symbols definition for the service allocation model

Symbol	Definition
S	Set of services requiring IT resources in order to be executed
R	Set of accessible F2C nodes
K_r	Set of slots of a specific F2C node *r*, i.e, respective node capacity
U_i	Requirements of service *i* in terms of number of slots
N_r	Slot allocation delay for a resource *r*
D_i	Total delay until successful allocation of resources demanded by service *i*

$$\sum_{k=0}^{|K_r|} \sum_{i=0}^{|S|} Y_{i,r,k} \leq K_r, \forall r \in R \tag{5.2}$$

$$\sum_{i=0}^{|S|} Y_{i,r,k} \leq 1, \forall r \in R, \forall k \in K_r \tag{5.3}$$

where $Y_{i,r,k}$ is an integer linear variable defined as:

$$Y_{i,r,k} = \begin{cases} 1, & \text{if service } i \text{ is allocated in slot } k \text{ of resource } r \\ 0, & \text{otherwise} \end{cases}$$

Furthermore, two distinct types of services are considered based on the so-called mice and elephants effect. Therefore, services are distinct in terms of the required resource capacity, where the mice consists in a large amount of services with low requirements, i.e., a few number of slots, whist the elephants represent a small number of services requiring a large number of slots. Regardless the amount of required slots, the service allocation is accomplished only if all required slots can be successfully allocated. Thus, the objective function must comply with the constraint given by Equation (5.4).

$$\sum_{r=0}^{|R|} \sum_{k=0}^{|K_r|} Y_{i,r,k} = U_i, \forall i \in S \tag{5.4}$$

Finally, in order to calculate the delay for each service allocation, we compute the delay added by each slot allocation in distinct nodes, as shown in Equation (5.5). Hence, the delay added by a node *r* is denoted by N_r. In order to meet the described topology, the delay added by nodes in fog layer 1 must be configured to be lower than the delay added by nodes in the fog layer 2. Aligned to that, the delay added by the cloud must be higher than the delay added by the underlying fog nodes.

$$\sum_{r=0}^{|R|} \sum_{k=0}^{|K_r|} Y_{i,r,k} \times N_r = D_i, \forall i \in S \tag{5.5}$$

One may notice that the overall service allocation delay obtained by Equation (5.5) takes into account the delay for each allocated slot of a service. In order to compute the total delay for each service, two distinct approaches are implemented. In the first approach (serial), when a service requires two or more slots from the same resource, the delay of just one allocation for this resource is considered. For instance, let's consider that the delay of any node *r* on the first fog layer is N_r=1 time unit. In this sense, if a service requiring 3 slots is allocated on 2 distinct nodes in this layer, e.g., 2 slots of resource *r1* and 1 slot of resource *r2*, then the final delay is $N_{r1} + N_{r2} = 2$ time units.

On the other hand, the second approach (parallel) takes into account the possibility of parallel allocation observed on several IoT services. For instance, let's consider a service requiring 3 slots, distributed among fog layer 1, fog layer 2 and cloud, with delays 1, 2 and 10 time units respectively. In this parallel approach, the final delay is 10 time units if all slots are successfully allocated, whilst the previous serial version faces a delay of 13 time units. In the following section, the results attained by the described model are presented, as well as a comparison between both allocation approaches.

Results

The results achieved by the presented model were obtained through the employment of the well know optimization tools PuLP [75] and Gurobi Optimizer [76]. The presented values are attained through the average of 30 executions of the implemented model.

In the employed scenario, it was considered, for sake of simplicity, that all fog nodes located in the same layer have the same capacity, whilst the available cloud capacity is always enough for the execution of the set of services in the simulation trials. Moreover, individual slots in distinct resources and layers are identical, that is, distinct slots may offer the same sort of IT resources. As a consequence, services differ from each other only by the capacity (number of slots) required to fully allocate them. Hence, in the presented results we use the mice and elephants terminology, representing respectively, the large amount of services requiring few slots in opposite to the low amount of services requiring higher number of slots.

The simulation parameters employed for attaining the presented results are shown in Table 5.2. In all performed simulations, the total number of services ranged from 10 to 90, whereas the total capacity of the cloud node was the sum of all services requirements.

Table 5.2: Parameters employed in the service allocation model

Parameter	**Value**
Number of fog nodes in layer 1	4
Number of fog nodes in layer 2	2
Number of cloud nodes	1
Total capacity of each node in layer 1	20 slots
Total capacity of each node in layer 2	100 slots
Delay of fog layer 1 nodes	1 time unit
Delay of fog layer 2 nodes	2 time units
Delay of cloud node	10 time units
Percentage of mice	90% of number of services
Percentage of elephants	10% of number of services
Mice requirements	3 slots
Elephants requirements	30 slots

Figure 5.3 shows the services allocation average delay versus the number of requested services when we consider the overall time of a service as the sum of delays in a serial execution (first approach), and also when we consider the delay as the maximum delay in a parallel execution (second approach). Further, we show a comparison for both types of services, mice and elephants. The obtained results show that mice and elephants delays are very similar in a parallel allocation approach, i.e., when slots from distinct resources and layers are allocated simultaneously. On the other hand, in a serial allocation approach, one should first notice that, as expected, all delays are higher in comparison to the parallel delay independently of the service type.

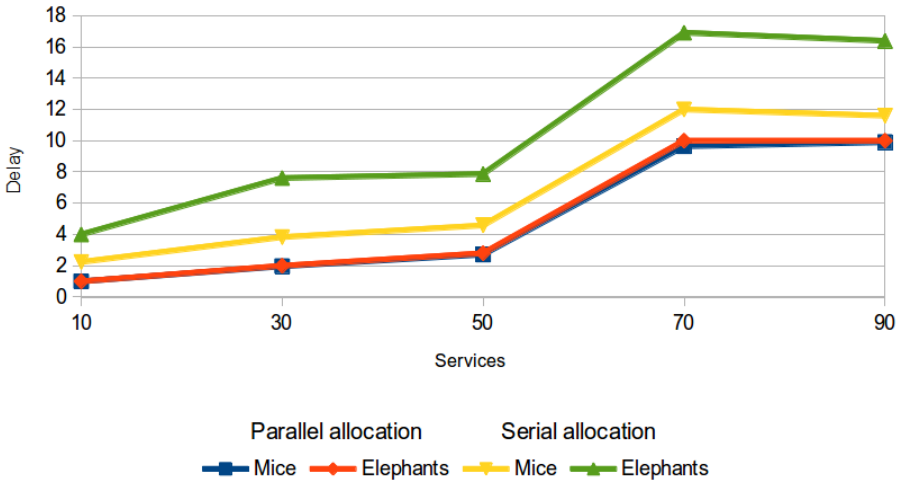
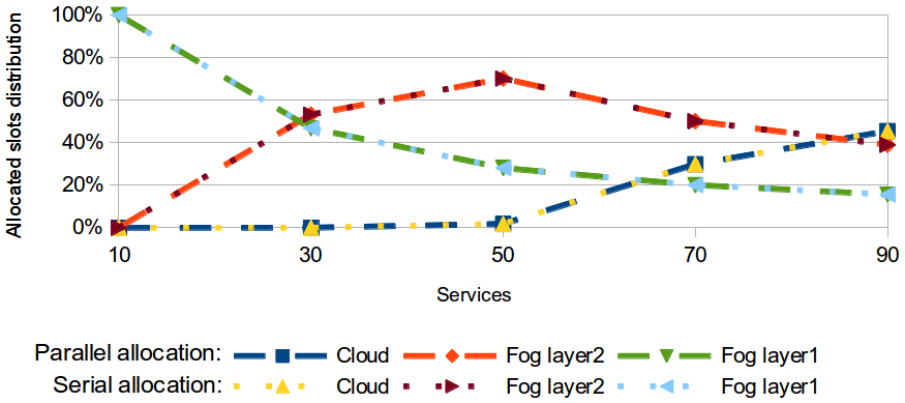


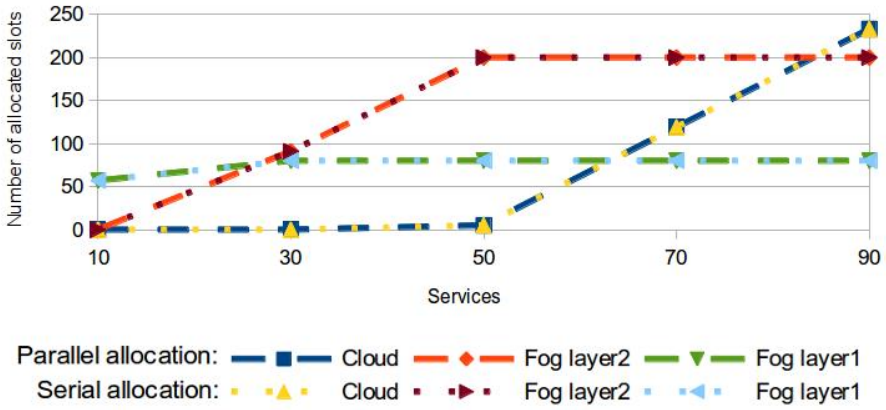
Figure 5.3: Delay versus total services.

Moreover, it is also worth to mentioning that, in contrast to the parallel allocation approach, the average delay for elephants are considerably higher than for mice in serial allocation. This may be explained by the fact that the distributed allocation of a service may increase the number of distinct resources used in the same layer of the model. Consequently, the distributed allocation of an elephant service has a higher probability of embracing a higher number of distinct resources. Nevertheless, this distribution among resources on the same layer does not affect significantly the final delay of services allocation in the parallel approach. On the other hand, in a serial allocation, even with the distributed allocation in fog nodes within the same layer, the allocation of elephant services may result in a higher delay, due to the higher probability of using a larger amount of distinct nodes.

A comparison of the slots allocation in distinct layers of the F2C topology can be seen in Figure 5.4(a). As it may be noticed, as the amount of services increase, the variation on the percentage of allocated slots at each layer shows the same behavior for both parallel and serial approaches. In other words, both approaches prioritize the allocation of resources at nodes located in the lowest layers, where lower delays can be obtained. Therefore, fog nodes in fog layer 1 are selected until the extinguishment of their available slots. This behavior is shown in Figure 5.4(b), where the amount of allocated slots at each layer is shown according to the amount of services to be allocated. As depicted, the growth on the number of slots in the fog layer is limited to 80 units, which is the total amount of slots available in this layer. As the number of services is increased, the new services must be offloaded to fog layer 2, which can afford the capacity demanded by 50 services, at maximum. This strategy would scale up for scenarios with more fog layers. In this simplified scenario, the allocation of resources in cloud would only occur when all the slots at fog layers have extinguished.



(a)



(b)

Figure 5.4: Slots allocation per F2C layer versus number of services.

When analyzed together, the irregular growth on the allocation delay, shown in Figure 5.3, is explained by the slots allocation on each layer, depicted in Figure 5.4. Notice that, when the number of services is increased from 50 to 70, the extinguishment of available resources in both fog layers 1 and 2, leads to the allocation of all new service requests at cloud node, driving to an increase in the average delay.

On the other hand, the increasing on the number of services to a value higher than 70 does not have a negative impact on the observed average delay. This is due to the fact that we consider only one cloud node, preventing the distributed service allocation in distinct cloud providers. Therefore, the delay for resource allocation is computed only once, regardless of the amount of resources required by the service.

5.3.Service allocation according to resource type

The F2C concept allocates available resources into distinct fog and cloud logical layers, enabling service decomposition and parallel execution in one or more layers. This is done by taking into account the suitability of the heterogeneous resources to fit the requirements for each subservice produced by the service decomposition. Therefore, the definition of each logical layer must consider resources capacity, reliability and volatility, just to name few parameters. Albeit the service distribution and parallel execution in the F2C architecture mirrors the benefits emerged from the high performance computing field, where parallel software execution was performed through homogeneous multicore platforms, a challenging coordination on the allocation of QoS-aware services is required in a F2C control plane, enabling decomposing, distributing, and aggregating services deployed in dynamic IoT scenarios. Indeed, besides the challenges introduced by fog computing, the service distribution must be performed taking into account the distinct requirements for each subservice in conjunction with the distinct service types each F2C resource can allocate.

In the preliminary approach for resource allocation –presented in the previous section–, all fog and cloud nodes are assumed to offer the same resource type, and, as a consequence, the service requests differ from each other only in terms of the amount of required resources. In this section, we identify distinct service types each F2C node can execute and extend the previously presented service allocation model by adding the necessary model constraints for mapping distinct services into the most suitable F2C node, taking into account both network and service constraints.

Service atomization

The distribution of a service for parallel execution in F2C leverages the service decomposing strategy so-called service atomization. The rationale behind the service atomization is to decompose a highly demanding service into service slices, tailored to enable their parallel execution. The obtained slices –hereinafter called atomic services– may present heterogeneous characteristics in terms of complexity and resources needs. Therefore,

a portion may consist of low-effort tasks, such as simple data reading or even low-complexity processing of raw data, and, thus, may be executed on constrained edge devices, whilst tasks with higher requirements may be allocated on robust resources and executed in parallel with smaller tasks.

The outcome of the service atomization process is an atomic service execution graph, shown in Figure 5.5(a), where atomic services are represented by nodes and atomic service dependencies are presented by edges. Therefore, atomic services not dependent on each other directly or indirectly, such as the green nodes in Figure 5.5(a), may be executed in parallel. Moreover, the type of service required by each atomic service is identified by a service identifier (SID), and must be allocated in a F2C node implementing the required service. As depicted in Figure 5.5(b), each F2C node shows a set of SIDs it can allocate at its underlying resources and the service allocation can be performed by mapping the requested SIDs into specific F2C nodes according to their suitability.

It is worth mentioning that we do not go into details on the atomization process itself. Earlier papers coping with service slicing can be found in the literature [77] [78] [79]. Rather, our focus is on the atomic service allocation for distributed and parallel execution in F2C systems. Therefore, we assume that a state-of-the-art atomizing strategy has been previously employed in order to create the execution graph for a given service.

Service allocation strategy

The employment of edge resources for service allocation as proposed by fog computing enabling complex processing and data collection on edge devices is a trend. Nevertheless, even admitting that several benefits may be attained by the employment of devices at the edge of the network, a service allocation strategy must also ponder the fact that these devices, besides being constrained both in IT and energy availability, should always keep enough free resources in order to run user processes. Therefore, whilst F2C may leverage idle resources, such as CPU, shared by one smartphone, the smartphone should limit (automatically or through its owner intervention) the total capacity shared as a fog resource. Indeed, sharing idle CPU with no limitations might quickly drain unplugged devices energy. Moreover, even if a device is plugged, stressing the CPU usually results on overheating, which might be harmful to the device[1]. Consequently, the collaborative model envisioned by fog computing requires the creation of new business models and policies in order to define the amount of idle resources that may be allocated at edge devices. Several aspects must be considered for the definition of these policies, including device type (e.g., smartphone, car, laptop, surveillance devices, etc.), its context (e.g., mobility, battery charge, IT resources load, etc.), as well as the maximum accepted resources load and the type of the resource to be shared by the device (e.g., processing, storage, real-time data, etc.) according to user's preferences.

[1] *Battery in portable devices may have reduced lifetime due overheating, as well as overclocked CPUs working continually at maximum capacity without adequate refrigeration*.

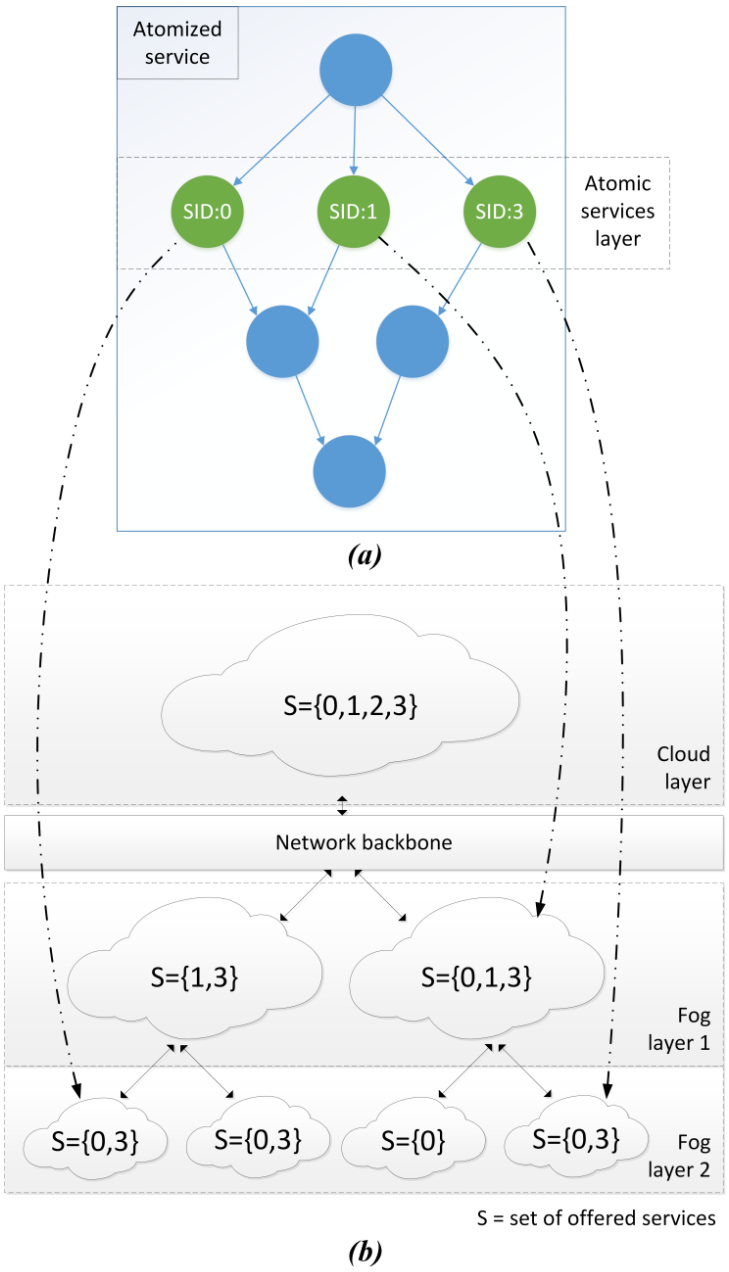


Figure 5.5: Service atomization and mapping.

In this work, it is considered that the resources shared by an edge device are specified in terms of type of services (SIDs) as well as the total amount of service instances it is willing to execute. In other words, a device offering services SID:1 and SID:2, and limiting the number of simultaneous service instances to 2 would allocate either two instances of SID:1, or two instances of SID:2, or one single instance of each service. The rationale behind this strategy is the assumption that the overall amount of service requests satisfied by a device is more relevant than the number of allocations for each SID. Moreover, if the same device of the previous example (offering SID:1 and SID:2) is limited to the allocation of only one service instance, for example due to reduced available memory, the device is not expected to define explicitly the SID to be offered to F2C (SID:1 or SID:2); rather, the F2C control plane may request the allocation of the service that may enable the optimal allocation of existing atomic services.

Analogous to the slot concept employed in the previous section, a slot is now extended to represent one instance of service offered by F2C resources. Hence, the slots are employed to represent the total capacity of F2C nodes in terms of both supported SIDs and number of requests it can allocate. As shown in Figure 5.6, each slot is linked to one of the service instances each underlying device is willing to satisfy. Indeed, since F2C nodes are abstractions of the available underlying resources, the employed strategy results in an architecture with service-agnostic allocation slots, where one slot may provide distinct SIDs, depending on the underlying resource capabilities.

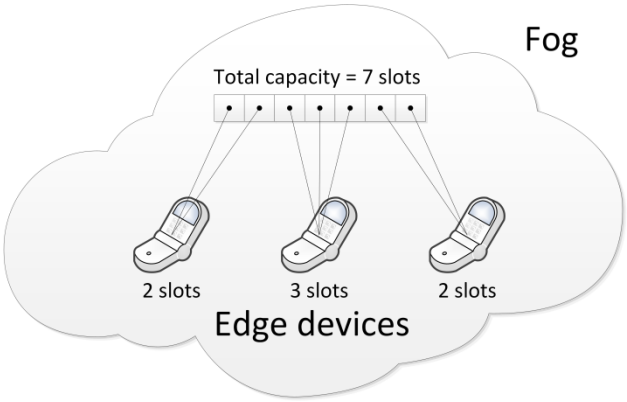


Figure 5.6: Shared resources and fog slots relationship.

In order to cope with the parallel allocation of atomic services –i.e., atomic services represented by nodes located at the same layer of the atomized service graph–, the service request must be specified in terms of the number of slots of each required SID that should be allocated on F2C nodes. Therefore, the F2C control plane is responsible for the distribution of the atomic services among the available slots, matching atomic service requirements and slot offerings.

Extended service allocation model

The allocation model presented in this section extends the one presented in section 5.2 in the sense that it aims at reducing the service allocation delay considering both the demanded SIDs and F2C nodes capabilities, whilst providing load balancing and energy consumption balancing among distinct fogs. Similar to the previous model, we assume the delay added by each slot allocation as a constant value assigned according to the layer the respective F2C resource is located at, respecting the described 3-layered F2C architecture, where resources located at lower fog layers present low service allocation delay, whilst higher delay is observed in the cloud layer.

The resources availability is statically represented by means of service-agnostic slots. Hence, in a given moment, each slot may provide the execution of at most one SID, according to the SIDs provided by the underlying devices. In addition, it is considered a coarse-grain approach for the service allocation and, as a consequence, we assume that all devices connected to the same fog are capable of executing the same SIDs.

In the energy availability side, a simple approach for energy consumption comparison among distinct layers is added to the model. In order to represent the consumed energy in the underlying devices, the concept referred to as energy cell is employed, which is analogous to the resource slot previously presented. Hence, each slot allocation consumes a number of energy cells and the total availability depends on the type of devices connected to the fog or cloud. For instance, a fog constituted by smartphones may present a lower number of energy cells than a fog constituted by smart vehicles. On the other hand, fogs and clouds constituted by plugged devices, such as plugged smartphones, plugged electrical vehicles, or datacenters, may be considered with unlimited energy.

The model presented in section 5.2 for the service allocation problem in F2C is extended, according to the described strategy, to a multidimensional knapsack problem (MKP) where the objective is threefold: (1) minimizing the overall delay for the allocation of all services taking into account the observed delay on distinct F2C layers; (2) minimizing the overload of individual fogs in terms of processing capacity; and (3) minimizing the excessive energy consumption on energy constrained fog resources. The list of symbols used in this section is summarized by Table 5.3, whilst the model objective, embracing the aforementioned goals, is given by the Equation (5.6).

$$\text{Minimize:} \quad \sum_{i\in S}\sum_{r\in R}\sum_{k\in K_r}\sum_{s\in O_r} Y_{i,r,k,s}T_r + \sum_{i\in S}\sum_{r\in R}\sum_{k\in K_r}\sum_{s\in O_r} Y_{i,r,k,s}W_{r,k} + \sum_{i\in S}\sum_{r\in R}\sum_{c\in C_r} Z_{i,r,c}V_{r,c} \tag{5.6}$$

where Y and Z are zero-one integer variables used to respectively indicate the slots allocation and the energy cells usage, so that:

$$Y_{i,r,k,s} = \begin{cases} 1, & \text{if service } i \text{ is being allocated in resource } r \\ & \quad \text{slot } k \text{ for SID } s \text{ execution} \\ 0, & \text{otherwise} \end{cases}$$

$$Z_{i,r,c} = \begin{cases} 1, & \text{if service } i \text{ being allocated in resource } r \\ & \quad \text{will consume energy cell } c \\ 0, & \text{otherwise} \end{cases}$$

subject to:

$$\sum_{i \in S} \sum_{k \in K_r} \sum_{s \in O_r} Y_{i,r,k,s} \leq |K_r| \,, \ \forall r \in R \tag{5.7}$$

$$\sum_{i \in S} \sum_{s \in O_r} Y_{i,r,k,s} \leq 1 \,, \ \forall r \in R \ \wedge \ \forall k \in K_{\mathrm{r}} \tag{5.8}$$

$$\sum_{r \in P_s} \sum_{k \in K_r} Y_{i,r,k,s} = N_{s,i} \,, \forall i \in S \ \wedge \ \forall s \in L_i \tag{5.9}$$

$$\sum_{i \in S} Z_{i,r,\mathrm{c}} \leq 1 \,, \ \forall r \in R \ \wedge \ \forall c \in C_{\mathrm{r}} \tag{5.10}$$

$$\sum_{r \in R} \sum_{c \in C_r} Z_{i,r,c} = \ U_i F_i \,, \ \forall i \in S \tag{5.11}$$

$$\sum_{c \in C_r} Z_{i,r,\mathrm{c}} = \sum_{k \in K_r} \sum_{s \in O_r} Y_{i,r,k,s} \, F_i \ , \ \forall r \in R \ \wedge \ \forall i \in S \tag{5.12}$$

In the proposed model, the objective function presented in (5.6) comprises the three presented goals, where each term of the equation matches one of the goals respectively, i.e., minimizing latency, processing overload, and energy consumption. The constraints previously defined by Equations (5.2), (5.3) and (5.4) are tailored to Equations (5.7), (5.8) and (5.9), with analogous roles to the ones previously presented. Therefore, Equation (5.7) is the capacity constraint, responsible for ensuring that the number of slots consumed by the services will never be higher than the resource capacity, which is specified by the total number of slots available on each resource. Moreover, Equation (5.8) ensures that two or more services will never be allocated at the same resource slot. Notice that, according to the presented service allocation strategy, each slot can offer distinct SIDs, however, when several slots are allocated, for instance for the provisioning of computing service, the allocation of slots for data collection shall be forbidden. This is accomplished by Equation (5.8).

Table 5.3: Symbols definition for the extended allocation model

Symbol	Definition
S	Set of services to be allocated
L_i	Set of SIDs required by service *i*, i.e., a list containing its requirements in terms of slot types
$N_{s,i}$	Number of instances of SID *s* required by *i*, i.e., its requirements in terms of amount of slots per SID
U_i	Total number of slots required by service *i*, i.e., $U_i = \sum N_{s,i}\ ,\ \forall s \in L_i$
R	Set of F2C nodes, i.e., clouds and fogs
K_r	Set of slots provided by node *r*
O_r	Set of SIDs offered by node *r*
P_s	Set of all nodes able to offer SID *s*
T_r	Delay added by the allocation of an atomic service in node *r*, according to its layer
$W_{r,k}$	Weight of the slot *k* in node *r*
C_r	Set of energy cells available in F2C node *r*
$V_{r,c}$	Weight of the energy cell *c* in F2C node *r*
F_i	Energy consumed by each slot of service *i*

Equation (5.9) is in charge of ensuring, for one requested service, the allocation of all atomic services expected to be executed in parallel, in addition to the mapping of the SID required by each atomic service to an available F2C slot offering the respective SID. It is worth mentioning that, in order to accomplish the matching between the service request and offer: i) we link each service type to a distinct SID, and ii) the amount of SIDs may vary according to the scenario being simulated as well as the services representation granularity. For instance, we may link processing, real-time data and storage services to SID:1, SID:2 and SID:3, respectively, and represent a fog node offerings as {1,2} if it is able to provide processing and real-time data services. Moreover, the set of atomic services to be allocated in parallel for a specific atomized service may be expressed by {1,2,2,2,3}, meaning that,

according to the atomization graph, it is possible the parallel execution of one instance of SID:1 and SID:3, and three instances of SID:2. Moreover, a fine-grained approach for the atomic services allocation in the proposed model would allow the specialization of the cited services, for instance, by distinguishing among real-time data types, as well as the creation of new atomic service types, such as data aggregation service.

On the other hand, Equations (5.10), (5.11) and (5.12) are in charge of modeling the allocation of energy cells ensuring coherency and consistency to the constraints defined for slots allocation. Therefore, Equation (5.10) imposes the constraint for cells allocation in an analogue fashion regarding the constraint imposed by (5.8) for slots allocation. Equation (5.11) assigns the exact number of energy cells consumed for each service, taking into account the number of atomic services that are supposed to be executed in parallel, whereas Equation (5.12) defines the relationship between slots allocation and cells consumption on each resource, assuring that the number of cells consumed by a service on one resource is directly linked to the amount of slots allocated on the respective resource.

In order to provide load balancing on F2C resources allocation, distinct weights are assigned to each node slot according to the model parameters $W_{r,k}$ and $V_{r,c}$. These values are related to F2C resources' load preferences. In other words, the set of slots with lower weight is totally allocated on each resource before the allocation of slots with higher weight takes place. This strategy, guaranteed by the second term of (5.6), achieves load balancing, since the allocation of lower weight slots is performed in other resources, if available and all constraints observed. In an analog way, the third term of (5.6) is responsible for balancing the energy consumption among distinct fog resources.

Results

In the next lines, the results obtained by the described model are presented, assuming distinct simulation scenarios. For all presented results, we adopt the same base F2C topology, consisting in one cloud layer constituted of cloud data centers, and two fog layers, where constrained devices, such as smartphones, are grouped in the lowest fog lower, within 4 distinct fogs, whilst vehicles in a parking lot are clustered on the second fog layer, formed by 2 fogs. Moreover, we consider three distinct service types that may be possibly offered by each one of the F2C resources: (1) short-term storage – SID:1; (2) long-term storage – SID:2; and (3) processing – SID:3.

For the first simulated scenario, the parameters described in Table 5.4 were employed. As shown by parameters, this scenario employs homogeneous F2C resources, where resources in the same layer offer not only the same set of SIDs, but also the same number of slots and cells. The impact caused by the amount of resources on the allocation distribution is assessed through a constant increment on the number of slots into both fog layers and comparing the distribution of the allocated slots. It is worth mentioning that this simulation aims at comparing the allocation of resources required for services not requiring real-time data from edge devices. The rationale behind this is to compare the behavior of service

allocation on each layer, as the amount of resources on the layers is modified from a traditional cloud environment towards a F2C scenario with enough resources on each layer. Therefore, we show the advantages, in terms of core resources load, of employing fog for processing and short-term storage tasks. The results presented in Figure 5.7 and Figure 5.8 show the reduction of slots allocation and energy consumption in the cloud respectively. However, the reader should notice that, in this scenario, even when the amount of slots are considerably increased, the employment of cloud resources, as well as energy consumption, never reaches zero since a portion of the requested atomic services cannot be provided by fog layers, as detailed in Table 5.4. Moreover, when the total number of slots is increased exclusively in the first fog layer, the amount of slots allocated at cloud presents the maximum reduction of 50%, since fog resources in this layer provide one single SID (see Table 5.4). Nevertheless, although this reduction may be considered a reasonable achievement, the increment on the total number of slots in the second fog layer proved to be efficient in this simulated scenario by diminishing even more the amount of allocated slots and, consequently, the energy consumption at cloud.

Table 5.4: Parameters employed in the extended allocation model

Parameter	**Value**
Delay to communicate with distinct F2C resources	Fogs in layer 1 = 1 time units Fogs in layer 2 = 2 time units Cloud = 10 time units
Requested services	40
Offered SIDs	Fogs in layer 1 = {3} Fogs in layer 2 = {1,3} Cloud = {1,2,3}
Required atomic services SIDs	90% of services= {1,3} 10% of services= {1,1,1,2,2,3,3,3,3,3}
Energy cells consumption	70% of atomic services consume 2 cells 30% of atomic services consume 3 cells
Slots weight ($W_{r,k}$)	0, if *k* between 0% and 60% of slots in *r* 1, if *k* between 60% and 80% of slots in *r* 3, if *k* between 80% and 100% of slots in *r*
Energy cells weight ($V_{r,c}$)	1, if *c* between 0% and 60% of cells in *r* 2, if *c* between 60% and 80% of cells in *r* 5, if *c* between 80% and 100% of cells in *r*

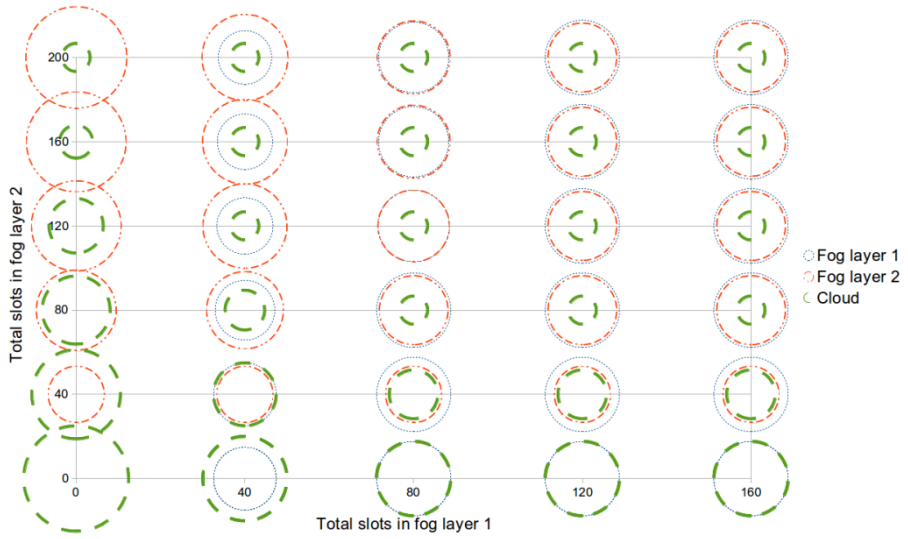


Figure 5.7: Comparison of amount of allocated slots in distinct F2C layers.

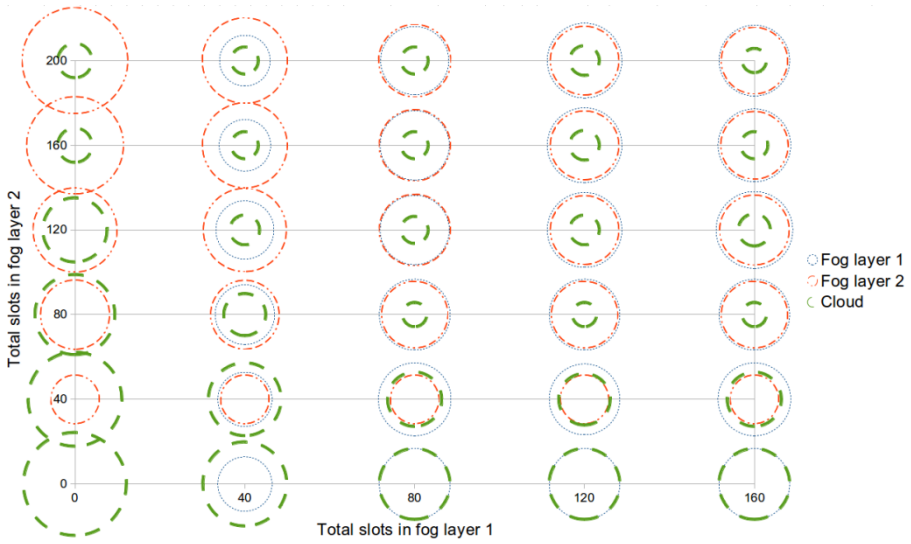


Figure 5.8: Comparison of energy consumption in distinct F2C layers.

Rather than presenting the distribution of the allocated slots into distinct layers, Figure 5.9 presents the allocation distribution through distinct fogs into the first fog layer, obtained on the same scenario. As shown by Figure 5.9, the deployment of weights to distinct slots and energy cells (see Table 5.4) guarantees load balancing as the total number of available slots on each fog node is modified. Indeed, if a fog has 60% of its slots allocated, the F2C control shall avoid the allocation of a new slot in this fog if a less loaded fog is able to satisfy the requested atomic service. Since the F2C layers are composed by homogeneous fogs in the described scenario, the amount of slots and energy cells are uniformly distributed among all nodes in fog layer 1. Consequently, the parameters $W_{r,k}$ and $V_{r,c}$ –showed in Table

5.4–, responsible for defining the distribution of weight values, are set with the same values for all fogs, resulting in the load distribution observed in Figure 5.9.

On the other hand, the employment of heterogeneous resources, introducing distinct characteristics for each fog in the first fog layer is considered in the second simulated scenario. This scenario might be observed, for instance, in a shopping center, where fog servers may be deployed into distinct sectors, such as cinema, hall, food court, supermarket, and parking, among others. In such scenarios, underlying devices may present a large heterogeneity in terms of mobility, system load, energy availability, as well as device type, impacting on the overall capacity. For the sake of comparison, we keep the same set of offered services employed in the first scenario and redistributed the number of available resources –i.e., number of slots– in the first fog layer decreasing resources for fog 1 and fog 2, while increasing resources for fog 3 and fog 4. Consequently, the weights assigned for the allocation of slots and energy cells in fog 1 and fog 2 were tailored in order to reduce the allocation rate of the resources constituting these fogs. Therefore, the employed weights for slots and energy cells allocation are those showed in Table 5.5. As depicted by Figure 5.10, a distinct distribution of the service allocation, in comparison with the previous simulations, may be observed, especially when the overall amount of slots in the first fog layer is reduced. However, since the weight is defined to the same value for all fog nodes when the allocation rate is below 50%, the increasing on resource availability leads to a similar usage rate.

The presented strategy contemplates the distinct characteristics inherent to each F2C resource in terms of services offered by the underlying devices, thereby performing the service offers and atomic service requirements matching. Moreover, by modelling the service allocation as a MKP, optimal solutions in terms of allocation delay are achieved while enabling an adaptive load and energy consumption balancing according to the resource and energy availability in distinct scenarios.

Table 5.5: Allocation weight for fog 1 and fog 2

Parameter	**Value**
Slots weight ($W_{r,k}$)	0, if *k* between 0% and 50% of slots in *r* 3, if *k* between 50% and 70% of slots in *r* 5, if *k* between 70% and 100% of slots in *r*
Energy cells weight ($V_{r,c}$)	1, if *c* between 0% and 50% of cells in *r* 4, if *c* between 50% and 70% of cells in *r* 7, if *c* between 70% and 100% of cells in *r*

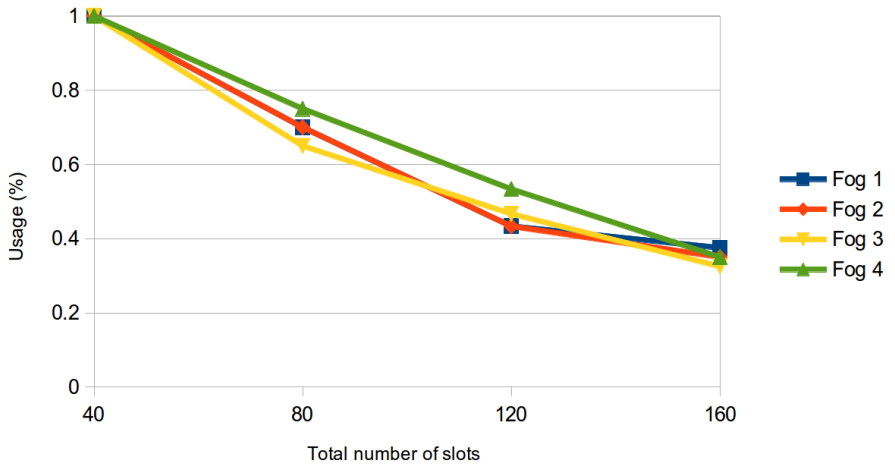


Figure 5.9: Fog load in the first fog layer with homogeneous distribution.

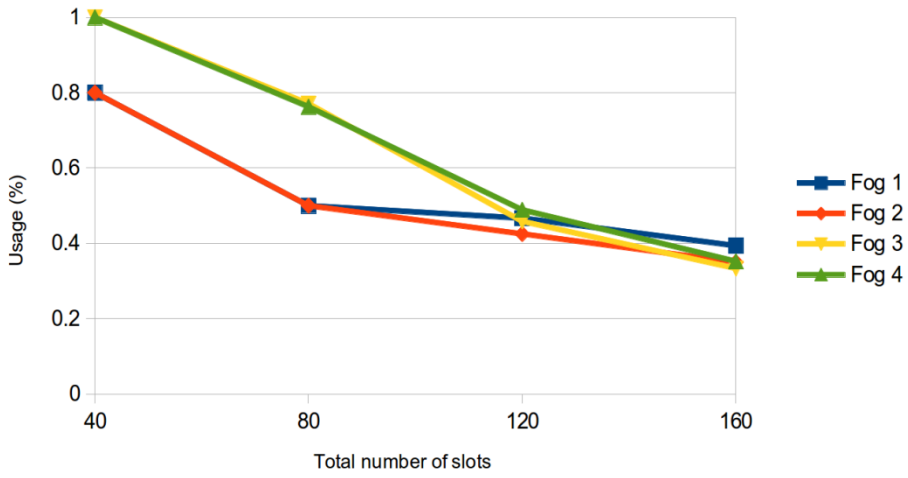


Figure 5.10: Fog load in the first fog layer with heterogeneous distribution.

5.4.Dealing with edge resources dynamicity

Dynamicity is a characteristic innate to the edge resources considered for the scenarios implemented in this chapter. In fact, edge resources mobility is one of the key features considered for F2C layers definition. However, the static models presented so far are not suitable for coping with issues, such as service disruption, arising from edge resources dynamicity. In the next section, we focus on service allocation solutions in combined F2C architectures in dynamic scenarios, taking into consideration their service response time, network bandwidth, energy consumption and disruption probability. The evaluated scenario considers fog devices mobility to illustrate the dynamicity and volatility associated to the overall fog capacity, showing the impact of rapid fog capacity changes on the services execution

Through the analysis of the F2C architecture showed by Figure 5.1, two key issues highly impacting on the resources volatility and the overall performance can be observed, namely mobility and power consumption. It seems rather logical to assume that mobility grows as moving down on the F2C architecture, therefore lower layers present higher volatility than higher layers. Let us consider a particular service, deployed in a smart city scenario, that is allocating resources within a fog, according to a specific resource allocation policy. As shown in Figure 5.11, the fog is constituted by different devices, including a static traffic light as well as distinct mobile resources –such as cars and buses. With this topology, the total fog capacity may dynamically change according to the amount of devices setting the fog at a certain time. Let's also assume the following: i) as in previous sections, a slot represents one unit of IT resources; ii) the deployed service requires 3 fog slots that must be allocated depending on real-time resource availability; iii) each individual device may only provide one resource slot; and iv) the traffic light is the device implementing control plane functionalities for management and allocation of available resources. Therefore, a successful execution of a service request relying on the fog capacity strongly depends on the real-time resources availability. Figure 5.11(a) depicts a scenario where 4 slots, provided by three different cars and one bus, are available for service execution. As it can be seen, however, only 2 slots are available in Figure 5.11(b), since 2 cars have, for instance, left the fog area, resulting in service disruption. Indeed, the latter may occur when the edge-device executing a job of a service leaves the fog (such as due to battery limitation, SLA policy, or similar). For simplicity, we assume in the global design that a job reallocation does not impact on the service execution exclusively when it occurs within the same fog layer; more generally, and as a subject of further studies, job reallocation can also be performed between different layers.

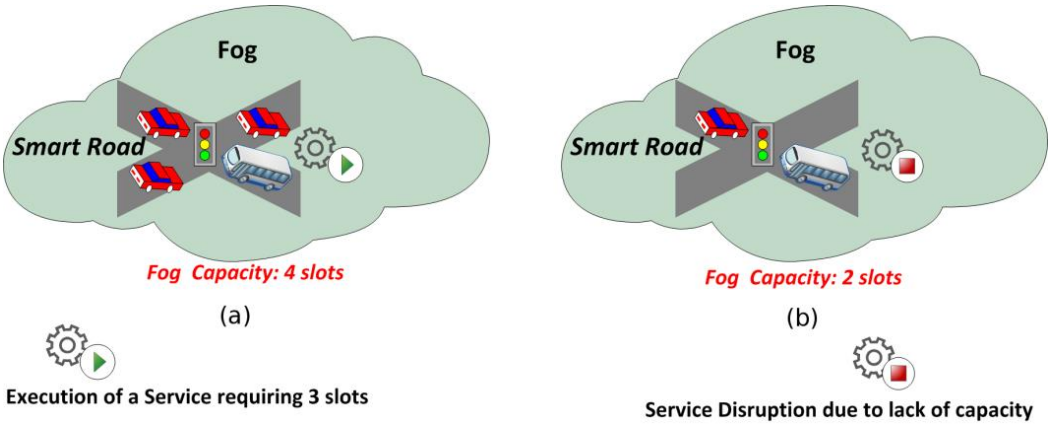


Figure 5.11: Service execution: (a) successful; (b) disruption.

As briefly introduced earlier, power consumption is another key issue in the envisioned collaborative F2C model. Indeed, the energy availability is of high importance since it can also lead to the disruption of a service execution, i.e., it is another cause of resources volatility. In the envisioned scenario, power consumption increases with the increasing layer hierarchy in the F2C architecture, as shown in Figure 5.1. From that perspective, hence, it would be beneficial to shift services execution to lower consumption layers –closer to the edge. However, energy saving policies applied to edge devices may also limit their availability, thus increasing devices volatility, which is higher at lower layers in the F2C architecture.

Performance assessment of dynamic service execution

In this section, in order to illustrate the benefits of the proposed architecture, we analyze the performance of the Dynamic Service Execution (DSE) problem in combined F2C scenarios in the context of service response time, power consumption, network bandwidth occupancy and disruption probability. In addition, we propose two algorithms for job allocation, where we show how basic allocation strategies may substantially impact on the overall performance for all the evaluated parameters. For easy understanding, Table 5.6 lists the set of symbols used in the model. The rationale behind the DSE problem is to successfully run independent service requests, randomly received, with specific demands regarding the amount of IT resources to be allocated in a combined F2C scenario. In this context, a service is defined as a computing job that can be split into tasks, which can be run in a parallel manner either at fog or cloud layers. Moreover, each task requires a minimum of one resource slot, denoted by 1 *u*, such as processing, storage and networking speed [80].

Table 5.6: Symbols definition for the dynamic allocation model

Symbols	Meaning
S	Set of services.
X_i	State of service *i*: 1 executed, 0 otherwise, $i \in S$.
K_i	Number of tasks of service *i* that can be run in parallel.
$W_{k,i,r}$	1 if the task *k* of service *i* is using a resource unit of layer *r*, 0 other-wise, where $r \in R$ and $k \in K$.
R	The set of layers.
L_i	Service Response Time.
b	Baseline power consumption.
a	Slope of load-dependent power consumption.

In this model, a service is assumed to be successfully executed ($X_i = 1$) only if all its computing tasks (i.e., in which the service can be split into) are allocated either at cloud or fog layers. Therefore, Equation (5.13) quantifies the ***success of service execution***, where $W_{k,i,r}$ defines if a task of service *i* is using 1 *u* of a server positioned in layer *r*, and K_i is the set of tasks constituting service *i*. It should be noted that tasks belonging to one single service do not need to be always allocated at the same layer, which is the salient feature of F2C systems.

$$X_i = \sum_{k=0}^{|K_i|} \sum_{r=0}^{|R|} \frac{W_{k,i,r}}{|K_i|} \tag{5.13}$$

Since all tasks of a service are executed in a parallel and independent manner, the ***Service Response Time* (*Li*)** is the maximum latency among all its allocated workloads, as described by Equation (5.14). Moreover, we define L_i as the sum of the transmission latency, which is the round trip time to obtain the information demanded for the service execution, and the processing latency.

$$L_i = \max (L_r \times W_{k,i,r}) \tag{5.14}$$

The power consumption parameter can be generally determined as the addition of the energy consumed in fog and cloud premises. However, in our approach, we only

consider cloud power consumption, due to the fact that it is a dominant factor compared to mobile end-devices, and also because the assessment of power consumption in the fog would require a slightly different approach. This is because end-devices may refuse the processing if their energy reserves are limited. Thus, the power consumption is shown by Equation (5.15), where *b* is the baseline power consumption (assumed to be 325W as in [81]), *a* is the slope of the ***load-dependent power consumption*** (assumed to be 30.5 Joules per Gigabit [81]), and *x* is the traffic rate, i.e.,

$$Power = b + ax \tag{5.15}$$

The ***network bandwidth occupancy*** is defined as the set of network resources required for transmitting the service data to the cloud. Thus, in our analysis, this parameter does not consider the traffic related to jobs executed at fog layer. Finally, the ***service disruption probability*** is defined as the rate of services disrupted during their execution due to the lack of resources. This parameter considers tasks that have been allocated to a fog layer, whose resources turn to be insufficient to support the job demands.

Based on these four parameters, we propose two heuristics for job allocation in order to analyze the performance of the model: *First-Fit* and *Random-Fit.* In *First-Fit,* workloads are initially allocated to available resources in the lower fog layer, until all resources are extinguished. This is performed in a Bottom-Up approach, in the sense that, when the bottom layer runs out of capacity, the next upper layer is considered, eventually allocating workload at cloud if all fog resources are unavailable. In *Random Assignment,* a layer is randomly selected for a job allocation. Strategies First-Fit and Random-Fit are respectively described by Algorithm 1 and Algorithm 2. It is worth noting that Algorithm 1 is considering a 3-layered F2C architecture, such as the one shown in Figure 5.1, with 2 fog layers and 1 cloud layer. However, it can be easily tailored to distinct F2C architectures.

Algorithm 1: Overall Procedure for First-Fit.

Input: (service *i*)
Output: (*Allocated/NoAllocated*)
1: **for** each task ***j*** in ***Ki*** **do**
2: **if** layer 1 has enough capacity **then**
3: allocate task ***j*** in layer 1
4: **else if** layer 1 does not have enough capacity, but layer 2 does **then**
5: allocate task ***j*** in layer 2
6: **else**
7: allocate task ***j*** in cloud

Algorithm 2: Overall Procedure for Random-Fit.

Input: (service *i*)
Output: (*Allocated/NoAllocated*)
1: **for** each task j in K_i **do**
2: Obtain *X*=Set of layers with capacity > 0
3: **while** task j is not allocated and $|X| > 0$ **do**
4: Randomly select a layer *r*
5: **if** layer *r* has enough capacity **then**
6: allocate task j in *r*

Based on the previous assumptions, we evaluate service response time, power consumption, network bandwidth occupancy and disruption probability for: i) a conventional cloud scenario; ii) a F2C architecture composed by the cloud and one single fog layer (fog layer 1), hereinafter referred to as F2C-1; and iii) a F2C architecture with cloud and two fog layers (fog layer 1 and 2), according to the one presented in Figure 5.1, hereinafter referred to as F2C-2. The simulation model used to validate and obtain the simulation results presented been built using the well-known network simulation tool Omnetpp [82]. For the simulation implementation, the following assumptions were considered.

- A workload requires one resource slot units, which is locked for other requests while the holding time is not over. The holding time is assumed to be random time standing for the time resources will be reserved for a particular service. A resource unit consumes 10 Mbits of network bandwidth.

- 90% of services requests are mice services. Mice are defined as services that consume a minimum amount of slots, with low holding time, and low arrival time (interval between service request arrivals); web search is a common example of mice services. The other 10% of the services are the so-called elephant services. Elephant services require a high amount of slots (ten times more than mice), and their holding time is also larger in comparison with mice services; video on demand, backup file transfer are common examples of elephant services [83].

- Mice service request arrivals are modeled as a Poisson process with exponentially distributed inter-arrival time, and with a mean of 10 time slots. The holding time of each mice service is modelled as a random variable of 10 time slots on average, independent of other holding times and also independent of the service request arrival time. Mice services require two slot units of resources.

- Elephant service request arrivals are modeled as a Poisson process with exponentially distributed inter-arrival time, and with a mean of 100 time slots. The holding time of each elephant service is modelled as a random variable of

500 time slots on average, independent of other holding times and also independent of the service request arrival time. Elephant services require 10 slot units of resources.

- Fog layer 1 layer consists of 10 mobile devices. Each device in this layer has 2 units of capacity. The policy for a mobile device to get in/get out of this layer is according to a Poisson process with exponentially distributed inter-arrival times with an average time of 50 time slots.

- Fog layer 2 layer consists of 2 mobile devices. Each device in this layer has 20 units of capacity. The policy for a mobile device to get in/get out of this layer is according to a Poisson process with exponentially distributed inter-arrival times with an average time of 500 time slots.

- The average transmission latency (L_r) values per slot for cloud, fog layer 2 and 1 are 10, 2 and 1 ms respectively.

Service response time evaluation

Figure 5.12(a) shows the transmission latency where fog nodes are static for the following architectures: Cloud (C), F2C-1 (fog layer 1 and Cloud) and F2C-2 (fog layer 1, fog layer 2 and Cloud). As expected, the conventional cloud exhibits the highest transmission latency due to the usually long geographic distance between cloud and end-users, impacting negatively on the time required to acquire the information processed by the cloud layers. In a cloud scenario, the transmission latency is not affected by the heuristic used for jobs allocation –i.e., Random-Fit or First-Fit– since the workloads are always offloaded to the same F2C layer.

The simulation results related to the F2C-1 architecture show a significant reduction on the transmission latency in comparison with the conventional cloud, motivated by the geographical proximity between fog servers and end-users. Nevertheless, because of the limited capacity of fog layer 1, the cloud must be used for the allocation of some workloads in order to avoid service blocking, producing a negative impact on the transmission latency.

However, we also show in Figure 5.12(a) that the simulation results for F2C-2 present a reduced transmission latency in comparison to both F2C-1 and the Conventional Cloud. Indeed, this reduction is motivated by the use of an additional fog layer, which, albeit presenting higher transmission latency than the lower fog layer, it is considerably still lower than the one perceived for the conventional cloud.

Notice that for both F2C-1 and F2C-2 architectures, the employed heuristic for atomic service allocation has a considerable impact on the transmission latency. For instance, First-Fit has lower transmission latency in comparison with Random-Fit since it attempts to allocate all tasks according to the end-users geographical proximity. First-Fit considers first the lowest layer (fog layer 1). In case of lack of capacity in this layer, the fog layer 2 is

considered, and finally the cloud. However, Random-Fit selects in a random manner either cloud or fog layer 2 or fog layer 1 for each task of a service, resulting in a negative impact on the transmission latency.

Load-dependent power consumption

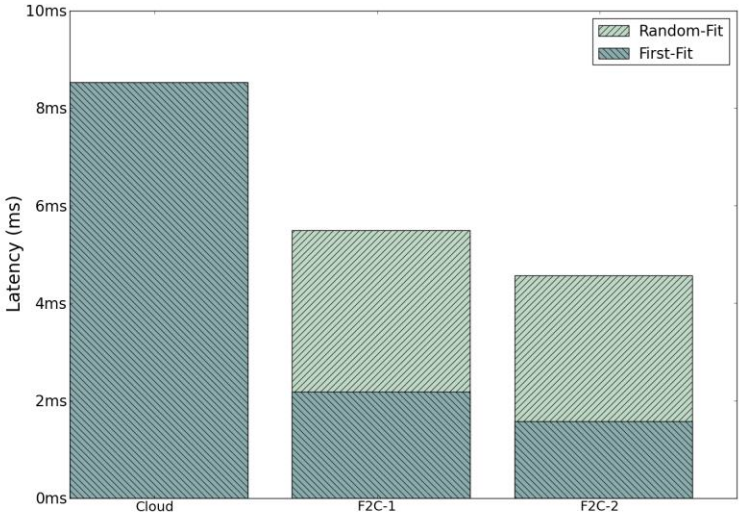
Figure 5.12(b) shows the total energy consumption without considering mobility features by the cloud using both Random-Fit and First-Fit for Cloud, F2C-1 and F2C-2 architectures. It should be noted that we only consider the power consumed by the cloud, since the power consumed in fog layers (mainly formed by mobile devices) can be neglected in comparison with the cloud. As it can be observed, the conventional cloud exhibits the highest total energy consumption. Notice that in a cloud scenario, the energy consumption is not affected by the heuristic used for service allocation.

Simulation results for F2C-1 show a significant reduction of the total power in comparison with the conventional Cloud architecture. However, similar to the transmission latency evaluation, due to the limited capacity of the lower layer, the cloud must be used for successful service allocation, which increases total consumed power. Nevertheless, the total power consumed can be reduced by the use of an F2C architecture. Notice that, by means of First-Fit, it is possible to achieve lower power consumption in comparison with Random-Fit.

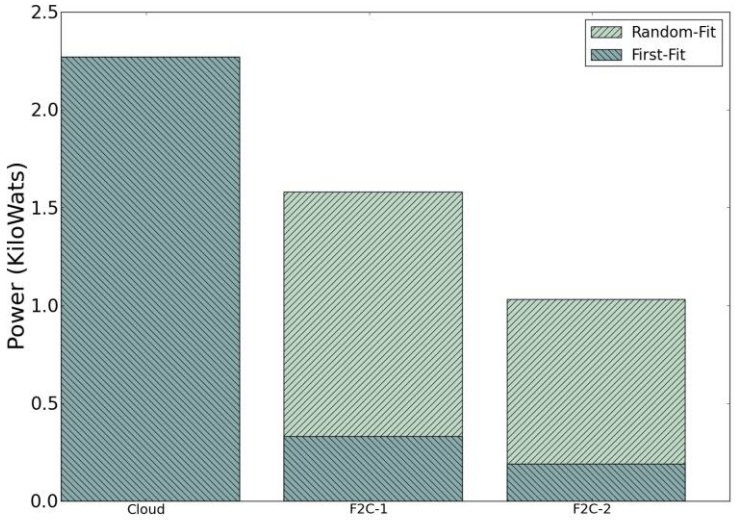
Network bandwidth occupancy

A conventional cloud infrastructure is commonly placed at the network core of Internet Service Providers (ISPs), where dedicated high capacity servers provide plenty of computing resources to cope with the demand of cloud users. Moreover, ISPs also provide the networking resources required to accommodate the traffic generated by cloud users. Recognized the fact that cloud users' traffic cannot be neglected, it turns out that this traffic may be large enough to significantly stress ISPs network core. In response, ISPs must re-dimension their networks to handle the traffic generated by cloud users, which increases their CAPEX and OPEX. Fortunately, the advent of fog opens the door to ISPs as a cost-efficient way to reduce the traffic generated at their network core. In the following lines, we evaluate how efficient fog may be in this regard.

Figure 5.13(a) shows the percentage of the total amount of traffic generated at the network core of an ISP for both Random-Fit and First-Fit, solely for F2C-1 and F2C-2 architectures. It is worth highlighting the reduction obtained with the introduction of the Fog layer 2. Indeed, the total traffic generated at the network core is 46.8% and 28.1 % for F2C-1 and F2C-2 architectures respectively, using Random-Fit. For a First-Fit strategy, the total traffic generated at the network core is even lower, 14.6% and 10.5% for F2C-1 and F2C-2 architectures respectively.



(a)



(b)

Figure 5.12: (a) Average SRT; (b) Energy consumed.

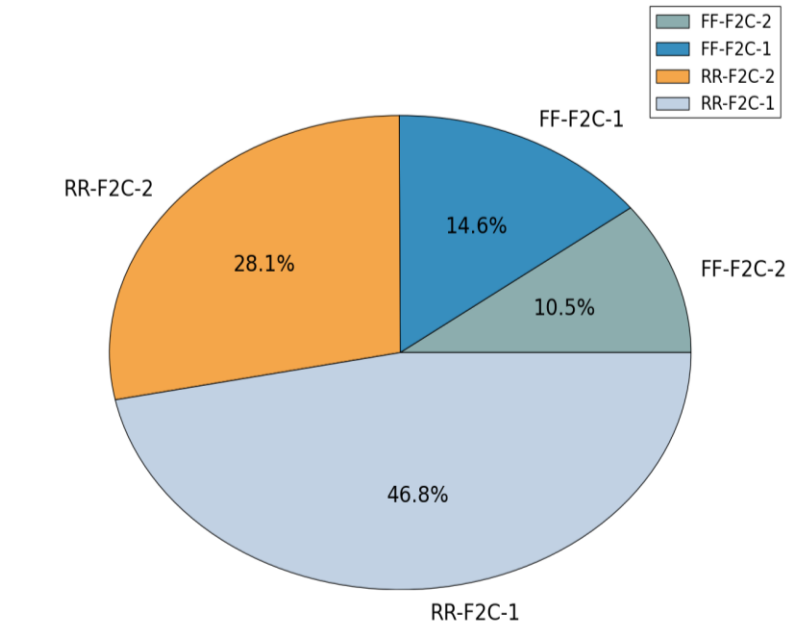
Based on the obtained results, it can be concluded that a Random-Fit strategy might be more appropriate to achieve a balance between the network traffic generated at the network core, and the one generated at the access domain (fog layer). By taking into consideration that end-users building the fog may not want to exhibit a degradation of their quality of service (QoS), the First-Fit strategy shows that it overuses the Fog layers for service allocation. At the same time, it should be considered that end-users might only want to share their processing computing features as long as their daily activities such as web navigation or chatting are not affected.

Service disruption probability and mobility evaluation

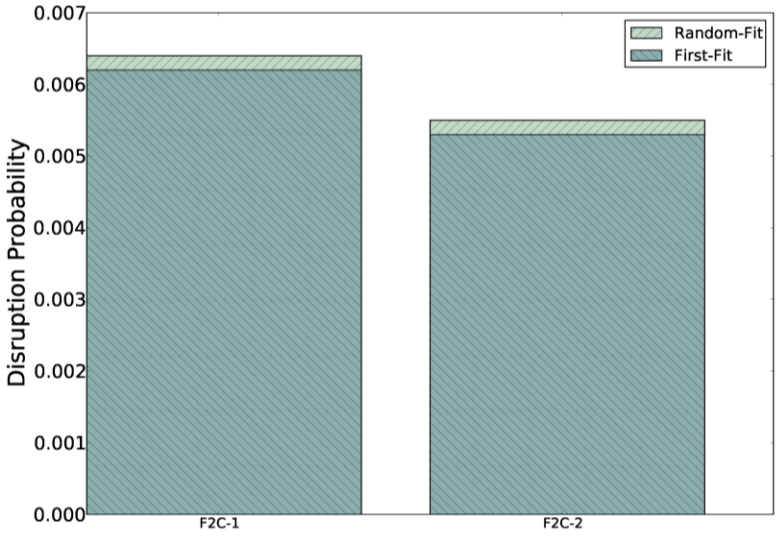
The disruption probability refers to the rate of services disrupted during their execution due to the lack of resources. To compute the disruption probability, the mobility of the nodes must also be considered. Therefore, the Freeway mobility model presented in [84] was adapted as the mobility pattern of the fog layer 1 domain. To that end, we consider a three lanes highway with 10 moving vehicles constituting a fog. Every time a vehicle exits a highway lane –and consequently the fog– a new vehicle enters the highway and join the fog within a random time given by an exponential distribution with mean of 6 time slots. Further, the lane length is assumed to be 1 Km and vehicles are traveling the lanes with a speed range between 50 and 80 Km per hour. We also consider a parking lot to model the mobility pattern of the fog layer 2 [85], where all cars in the parking lot build a fog in the fog layer 2. Similar to [85], we consider that flow of vehicles arriving at the parking lot is a Poisson process with inter-arrival time as well as the parking time given by an exponential distribution with a mean of 15 and 60 time slots respectively.

Figure 5.13(b) shows the disruption probability for both F2C-1 and F2C-2 scenarios. Notice how the benefits of F2C scenarios are diminished in terms of disruption probability. Indeed, the disruption probability of F2C-2 is lower than F2C-1, whereas the performance of First-Fit and Random-Fit regarding disruption probability is very similar.

It should be noted that the cloud is more robust to the disruptions in comparison to fog since the resources composing the latter present high mobility while the former's infrastructure is populated by static resources. Therefore, it can be concluded from the results shown in Figure 5.13(b) that increasing the layered structure of the F2C by adding the second fog layer decreases the disruption probability.



(a)



(b)

Figure 5.13: (a) Traffic at network core; (b) Disruption probability.

6. Service protection mechanisms for F2C computing systems

6.1.Challenges on service recovery in dynamic scenarios

Fog-to-Cloud enables the employment of cloud and fog paradigms in a coordinated way, by controlling and managing the large set of heterogeneous and volatile resources brought by the combination of both models. In fact, F2C is intended to provide a coordinated fog and cloud resources allocation, thus enabling the distributed execution of IoT services in fog, cloud, or both, while simultaneously leveraging the heterogeneity perceived on the combination of fog and cloud computing systems. Albeit the benefits of such F2C architectures have been clearly presented in Chapter 5, the negative issues in terms of performance that may come from a possible failure affecting the fog or cloud commodities have not been considered [86].

It is worth noting that previous IoT systems have employed cloud computing as a suitable solution for offering the demanded infrastructure in order to address the ever increasing requirements of IoT services related to both computing and storage capabilities This assessment is rooted on the fact that the huge computational capacity offered by cloud premises enables the deployment of IoT services with high requirements, including Video-on-demand or DC backup solutions [87].

On one hand, one key characteristic provided by cloud computing is the high reliability and robustness, enabling high availability of resources for services executed by end-users. On the other hand, whist the advent of new computing paradigms, such as fog computing, promises to overcome the negative issues of the cloud in terms of transmission latency, energy consumption, and network core load, among others, the services disruption probability introduced by the volatility observed at edge resources undoubtedly must be overcome.

Although distinct strategies have been successfully employed in cloud computing for failure protection on data and network resources, this is yet an open challenge in fog computing, further inherited by F2C computing systems. Therefore, in order to withstand possible failures, it is required the design of novel protection strategies specifically designed for distributed computing scenarios. For instance, the high dynamicity perceived at edge devices requires novel mechanisms for failure recovery, specifically designed for distributed scenarios. In such scenario, distinct recovery approaches may be assessed leveraging the distinct resources in the hierarchical F2C architecture, enabling the deployment of protection

resources in cloud or distinct fog layers in order to implement distinct failure recovery approaches matching distinct IoT services requirements. For instance, sensitive services requiring real time processing, such as healthcare or smart transportation, are further requiring low-delay failure recovery, i.e., the execution of a sensitive service at a device located at a short distance to an end-user shall also require the allocation of protection resources nearby, enabling a low delay recovery.

It must be remarked that a key aspect on the design of protection strategies for such distributed scenario is to enable service protection with low service allocation delay while minimizing the so-called protection cost. We consider the protection cost as the amount of computing resources devoted for service protection as well as the recovery delay. It is with no doubt that more research efforts must be devoted to distill the challenges related to service protection in F2C scenarios. To fill this gap, this section introduces and formulates the so-called Protected Service Allocation (PSA) problem in F2C scenarios. The main goal of the PSA problem is to minimize the service allocation time and protection cost.

There are several research studies dealing with distinct resilience aspects of cloud and fog computing. On one hand, studies available in [88], [89], [90] discuss the cloud and fog computing protection from a data security perspective. These studies put major focus on the integrity and protection of the IoT data processed by the computing commodities. On the other hand, among the studies dealing with protection in cloud or fog scenarios from a networking perspective it can be mentioned the work available in [91], where cloud resilience is achieved by means of network virtualization. Other contributions, such as [92], consider a cloud scenario based on an optical network core where an analogous technique to shared-protection so-called shared-path shared-computing (SPSC) protection is used against single link of node failure.

Regarding the resilience in Fog scenarios, the work presented in [93] aims at recovering from Mobile Edge Computing (MEC) failures. Albeit authors present a strategy for offloading workload to neighbor resources due to failure or overload, the availability of neighbor resources are not guaranteed by a protection strategy.

Other contributions focus on assessing proactive and reactive recovery strategies. In fact, both proactive and reactive recovery techniques are commonly employed on distinct scenarios, such as DC [94], where plenty of reliable computing and network resources are employed, or WSN [95], where a reduced amount of unreliable resources are employed. The work presented in [96] makes use of a proactive strategy for network failures recovery by defining backup paths in network switches, where the employment of priorities for distinct paths releases the controller from participating on the recovery process. On the fog computing side, authors in [69] replicate the service execution between the edge device and a remote server. Albeit the main purpose of the redundancy proposed by authors is not protection provisioning, the proposed proactive strategy yields a fault-tolerant service execution.

Despite the wealth of studies available related to cloud computing resilience, there are still open issues dealing with the negative effects that may come with the volatility inherent to fog commodities. In an attempt for addressing these challenges, in this chapter, we formalize, by means of linear programming, several protection approaches to cope with commodities failures.

Our goal is to enable enough computing resources to withstand single commodities failures. However, we assume a F2C scenario where only failures at the fog infrastructure are considered. The rationale driving this assumption is to show the vulnerability of fog nodes and its impact on performance and protection cost. It is worth mentioning that, contrary to cloud, fog commodities are mainly mobile devices with limited energy availability, both issues having a direct effect on the availability of fog resources.

6.2.Protection strategies assessment

The F2C architecture is able to jointly manage cloud and fog resources in a coordinated fashion, taking advantage of both high computational capacities offered by cloud data centers and low latency offered by fog resources located at the edge of the network. Nevertheless, possible failures in computing commodities may be prohibitive for the achievement of the envisioned performance in F2C computing systems. In order to withstand possible failures, it is required the design of novel protection strategies specifically designed for distributed computing scenarios.

In this section, we study the impact of distinct protection strategies on several aspects, such as the delay for service allocation and failure recovery, as well as the amount of computing resources employed for protection purposes. The strategies modeled in this chapter, as well as the respective attained results, have been already validated by the contributions published in [97] and [98].

6.2.1. Scenario description

The F2C architecture considered in this chapter is based on the one presented in section 5, where F2C resources are organized into 3 layers, consisting in one cloud layer and two fog layers building a hierarchical topology. Therefore, the first layer is the so-called fog layer 1, which is mostly constituted of resource-constrained devices, such as smartphones and vehicles in movement, presenting high-mobility and located at a 1-hop distance from the end-user. The second layer is the so-called fog layer 2, which embraces edge devices presenting lower mobility, higher resource capability and availability and higher distance from the end-user, if compared to the ones in fog layer 1. Examples of resources in this layer include vehicles in a parking lot and fog premises deployed in public buildings. Finally, the third layer is the cloud layer, which is formed by robust cloud data centers offering high computing capacity and availability in contrast with the high communication latency.

Moreover, like in the previous chapter, the total resource capacity offered by fog and cloud nodes as well as the capacity demanded by services are both measured in terms of slot units. Therefore, since an slot is the minimum unit for resource allocation, the amount of slots required by one single service is static and their allocation may be done either in one single fog or cloud offering the required amount of slots, or in a distributed fashion where service slots may be allocated in distinct F2C nodes and even in distinct F2C layers. Consequently, any failure recovery strategy shall require, at least, as many protection slots as the compromised node is employing for service execution when the failure occurs. It is worth clarifying that, the slots allocated for service execution are hereinafter called primary slots, whereas secondary slots refers to the slots selected for protection purposes. The described scenario is shown in Figure 6.1, where a service allocation distributed into 2 distinct fog nodes in fog layer 1 (see primary slots, in blue) have slots reserved for protection in fog layer 2 (see secondary slots, in green).

Distinct from the model presented in section 5.3, we consider, for the sake of simplicity, the allocation of only one resource type, i.e., only one type of resource slot is requested by services and offered by F2C nodes. This assumption aims at the simplification of the PSA problem modeling, allowing this work to focus on the description and comparison of the protection strategies.

6.2.2. Proactive vs Reactive failure recovery

In this section, the recovery of failures in F2C nodes is assessed through both proactive and reactive strategies, which are modeled as a Multidimensional Knapsack Problem (MKP). In addition, we study the impact of each one on several aspects, such as service allocation time and protection cost by providing simulation results, as well as an extensive analysis of the employment of proposed strategies in a F2C scenario.

Proactive and Reactive strategies

In this subsection, the proactive and the reactive failure recovery strategies are presented. For both strategies, it is assumed a horizontal allocation of protection resources. Therefore, protection slots are always reserved in the same architectural layer where the resources employed for service execution are, for instance, the protection resources for a fog in fog layer 1 are located in a distinct fog node in fog layer 1. The rationale behind this approach is that horizontal allocation may ease the allocation of resources with similar characteristics, respecting SLA even after failure events.

Moreover, for the sake of simplicity, we consider the failure of one single fog node. In other words, only one fog may become inaccessible on the first or second fog layers. In the cloud layer, it is assumed that cloud service providers can withstand with internal failures through their own protection schemes, hence, the cloud resources are always available. The assessed strategies work as follows.

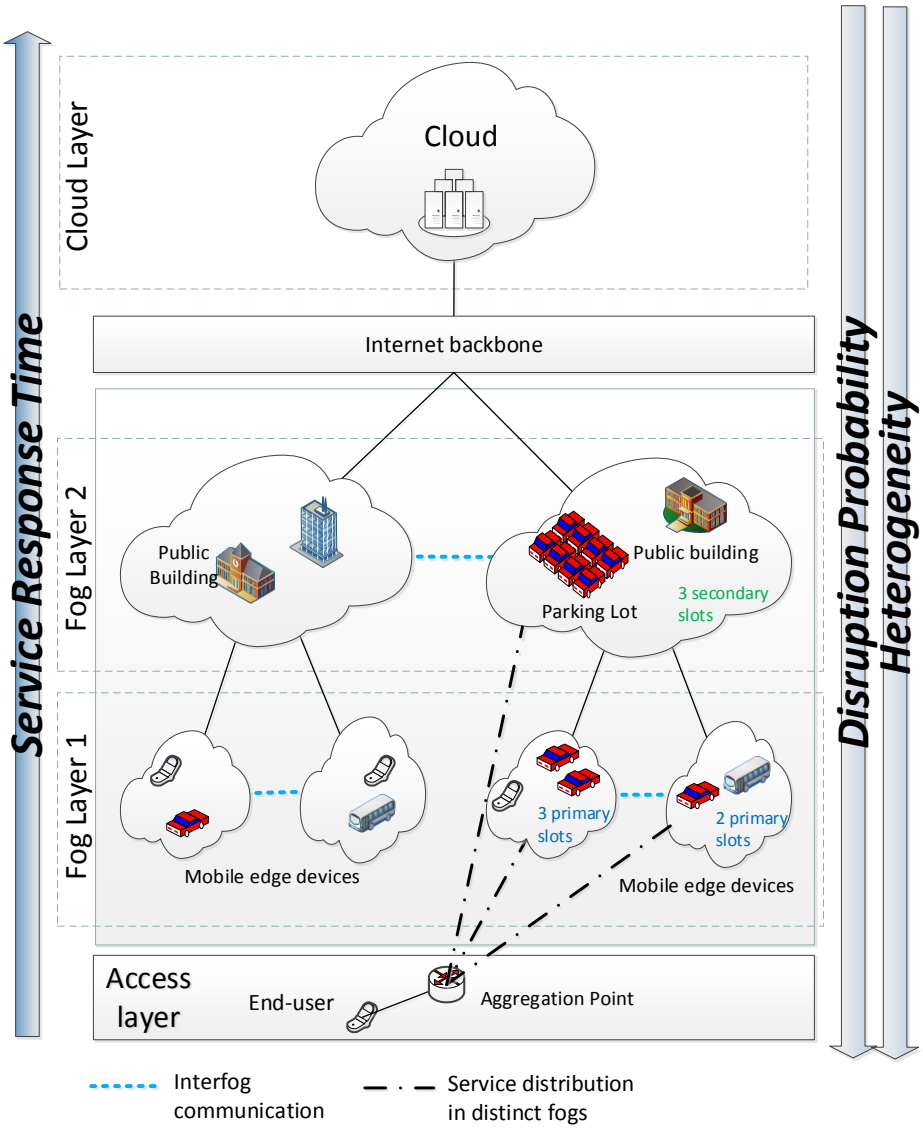


Figure 6.1: PSA scenario.

- Proactive recovery: in this strategy, protection resources (or secondary) are pre-allocated for each primary resource allocation. If the primary resource becomes unavailable, the protection may be used with no extra allocation delay at the cost of reducing the availability of primary resources at each fog layer, as shown in Figure 6.2(a).

- Reactive recovery: in this strategy, the protection resources are not allocated to a specific service until failure occurrence. However, a set of protection resources are reserved to react to an eventual failure, hence, when a failure takes place, the availability of resources delegated for recovery of the affected services is guaranteed. Indeed, a reactive strategy allows a set of protection resources to be shared among primary resources allocated in distinct fogs, diminishing the resource underutilization whilst delaying the recovery of services affected by a failure. This strategy is illustrated by Figure 6.2(b).

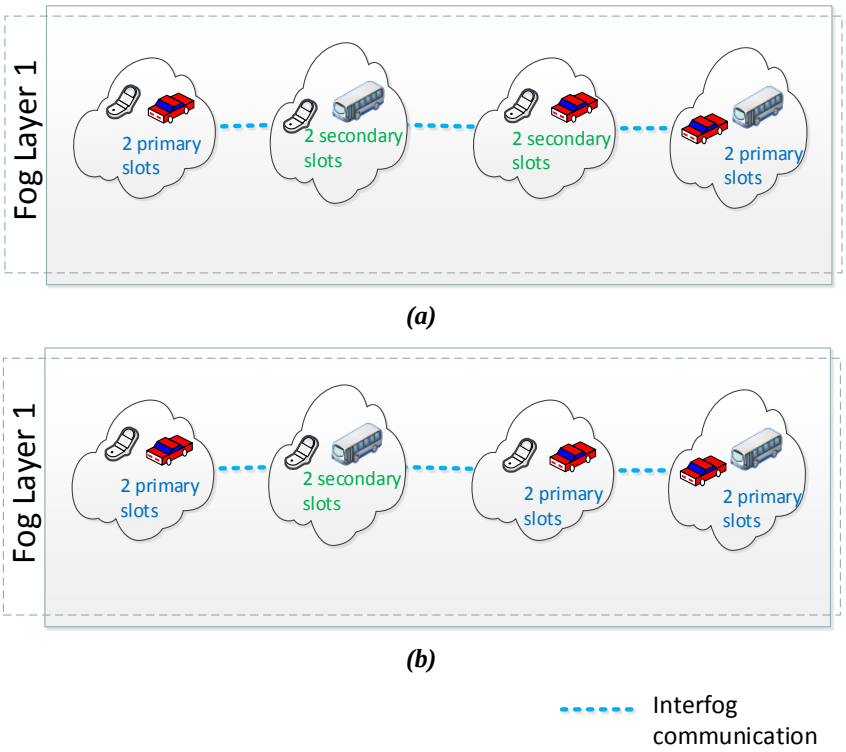


Figure 6.2: PSA strategies: (a) proactive; (b) reactive.

Mathematical model for proactive and reactive protection

In this section, the PSA problem is modeled by means of a formal description of the proactive and reactive protection strategies. The PSA is modeled as a MKP aiming at

decreasing the delay for allocation of each service as well as the protection cost, which can be achieved through the reduction of slots consumed for protection and the diminishing of recovery latency by the provisioning of low-delay protection slots using resources located in lower F2C layers. Therefore, the objective function, as described in Equation (6.1), is to minimize the sum of computed delay for the allocation of each service in the set of services S and the sum of the delay for communication with each secondary slot reserved in distinct F2C resources in the set R. All symbols used in the presented model are defined on Table 6.1.

$$\text{Min:} \quad \sum_{i \in S} D_i + \sum_{r \in R} P_r \tag{6.1}$$

Table 6.1: Symbols definition for PSA model

Symbol	**Definition**
D_i	Delay for the transmission of the service *i* (all primary slots required by the service)
P_r	Recovery delay offered by resource *r*
S	Set of services to be executed
U_i	Total number of slots required to execute service *i*
R	Set of F2C resources, i.e., set of distinct cloud and fog nodes
K_r	Set of slots provided by F2C resource *r* for both primary and secondary use
L_n	Set of fogs available at fog layer *n*
F_r	Set of fog nodes in the same fog layer of fog *r*
T_r	Allocation delay of a slot in F2C resource *r*, according to its F2C layer

Therefore, in order to represent the primary and secondary slot allocation into the available F2C resource, the respective linear programming integer variables Y and X were defined as follows.

$$Y_{i,r,k} = \begin{cases} 1, & \text{if service } i \text{ is allocated in resource } r \\ & \quad \text{consuming slot } k \\ 0, & \text{otherwise} \end{cases}$$

$$X_{r,k} = \begin{cases} 1, & \text{if resource } r \text{ has its slot } k \text{ reserved as} \\ & \text{a secondary slot (protection)} \\ 0, & \text{otherwise} \end{cases}$$

In order to properly accomplish the optimal allocation of primary and secondary resources, the objective function is subject to a set of constraints. Therefore, in other to compute the sum of allocation delays for all services in *S*, as specified by the first term of Equation (6.1), Equation (6.2) defines the allocation delay for each service allocation as the sum of primary slots allocated at each F2C node multiplied by the allocation delay at the respective node. In the second term or Equation (6.1), i.e., sum of the recovery delay introduced by each node employed as secondary resource, the protection delay of each node is defined by Equation (6.3) as the number of slots consumed for protection multiplied by the allocation delay offered by this node.

$$\sum_{r \in R} \sum_{k \in K_r} Y_{i,r,k} * T_r = D_i \, , \forall i \in S \tag{6.2}$$

$$\sum_{k \in K_r} X_{r,k} * T_r = P_r \, , \; \forall r \in R \tag{6.3}$$

It is worth noting that, in this model, the protection cost may be reduced either by diminishing the amount of secondary slots reserved (X) or by reserving secondary slots offering lower transmission delay (T). On the other hand, the primary allocation delay can be diminished only through the selection of resources offering lower delay (T), since the amount of primary slots allocation cannot be decreased. This constraint is defined by Equation (6.4), which aims at ensuring the allocation of the required amount of primary slots for each service in S. In addition, this constraint enables the distributed service allocation into distinct F2C resources and layers, as envisioned by F2C computing.

$$\sum_{r \in R_s} \sum_{k \in K_r} Y_{i,r,k} = U_i \, , \; \forall i \in S \tag{6.4}$$

Equation (6.5) and Equation (6.6) are, jointly with Equation (6.4), responsible for the coordinated allocation of primary and secondary slots according to service requirements and resources availability. The constraint imposed by Equation (6.5) guarantees that each resource capacity, in term of slots, is not surpassed. This is achieved by considering the capacity of each F2C node and its allocation of both primary slots demanded by distinct services and secondary slots required by protection strategies. Furthermore, Equation (6.6) is responsible for constraining the capacity of each slot to, at maximum, one allocation. That is to say, it ensures that each consumed slot is allocated as either one primary slot or one secondary slot.

$$\sum_{i \in S} \sum_{k \in K_r} Y_{i,r,k} + \sum_{k \in K_r} X_{r,k} \leq |K_r| \, , \; \forall r \in R \tag{6.5}$$

$$\sum_{i \in S} Y_{i,r,\mathrm{k}} + X_{r,\mathrm{k}} \leq 1 \, , \; \forall r \in R \; \wedge \; \forall k \in K_r \tag{6.6}$$

In order to cope with both proactive and reactive protection strategies analyzed in this section, two strategy-specific constraints are introduced. Equation (6.7) ensures the allocation of secondary slots following the proactive strategy, whilst Equation (6.8) guarantees the proper allocation of secondary slots in a reactive recovery strategy. The horizontal allocation approach defined for the proposed recovery mechanisms is also ensured by both equations. Therefore, on each primary slot allocated in one layer, Equation (6.7) enforces the reservation of a protection slot in the same layer, whilst, in Equation (6.8), the goal is accomplished by guaranteeing that the amount of secondary slots on each fog layer is not lower than the amount of primary slots allocated in any individual fog node located in the same layer.

$$\sum_{i \in S} \sum_{r \in L_n} \sum_{k \in K_r} Y_{i,r,k} = \sum_{q \in L_n} \sum_{k \in K_q} X_{q,k} \,, \quad \forall n \in \{1,2\} \tag{6.7}$$

$$\sum_{i \in S} \sum_{k \in K_r} Y_{i,r,k} \leq \sum_{q \in F_r - \{r\}} \sum_{k \in K_q} X_{q,k} \,, \quad \forall r \in L_1 + L_2 \tag{6.8}$$

Results

In this section, we assess the impact of both presented strategies for the PSA problem on the overall service allocation delay, failure recovery delay, and F2C resources load. The tools employed for executing the model are PuLP [75] and Gurobi Optimizer [76]. The attained results considered the service parameters and resource parameters shown, respectively, in Table 6.2 and Table 6.3.

Table 6.2: PSA model parameters (services)

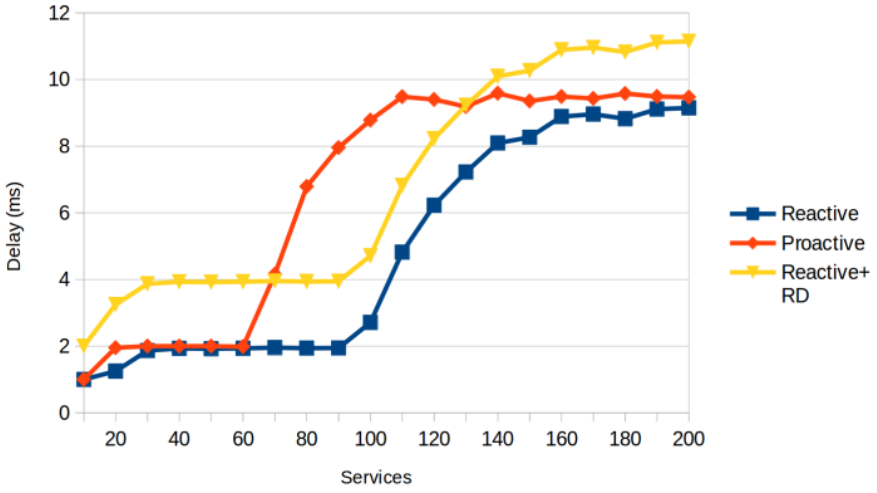
Parameter	**Value**
Number of requested services	From 10 to 150
Ratio between amount of service types: low consuming / high consuming services	90 (low) / 10 (high)
Slots demanded by low consuming services	2
Slots demanded by high consuming services	20

Table 6.3: PSA model parameters (resources)

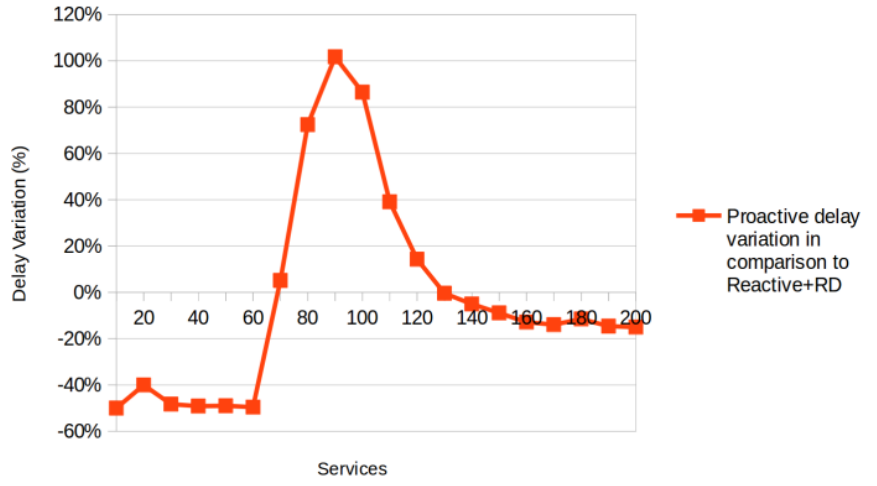
Parameter	Fog layer 1	Fog layer 2	Cloud
Number of F2C nodes per layer	8 fogs	4 fogs	1 cloud
Available resources per node	10	100	Unlimited
Allocation delay per F2C layer	1 ms	2 ms	10 ms

For the obtained results, the resource allocation for each service at distinct F2C nodes is assumed to be performed in a parallel fashion. Hence, the total latency for the complete allocation of one complete service is considered to be the maximum delay among all the allocated primary slots.

Both proactive and reactive protection strategies are compared in Figure 6.3(a) in terms of the observed impact on the delay for service allocation. This comparison evidences a lower delay on the service allocation with reactive protection in comparison to the proactive protection. However, this single results confrontation may be unfair since proactive strategies consume more secondary resources in lower layers leading to a premature allocation of primary resources in higher layers, resulting on a higher average latency. To cope with this situation, the overall delay for reactive strategies, considering both the service transmission to the primary resource and delay added by the service recovery after one eventual fail, is further calculated. Albeit the delay for service recovery in reactive strategies cannot be overlooked, this delay is not applicable to proactive strategies, since it can offer seamless failure recovery, as previously discussed in this section. Therefore, Figure 6.3(a) also includes the overall delay for a reactive strategy including the fail recovery delay, represented by the so-called reactive+RD. Therefore, a more realistic comparison, confronting the delays for proactive and reactive+RD, reveals that, in failure scenarios, each protection strategy may present better performance in distinct cases according to the number or services to be executed. Moreover, the sudden increase on the delay observed in this figure can be easily explained by the shift on the primary slots allocations from fog to cloud due the lack of available resources in both fog layers. As a consequence, even taking into account the added delay due an eventual reactive recovery, the early depletion of lower layer resources, observed in proactive strategies, results in a higher average delay of this strategy under medium load. A direct comparison between proactive and reactive+RD is shown in Figure 6.3(b), where an increase from 50% lower to 100% higher delay of the proactive strategy in comparison with reactive+RD can be observed, being followed by a decrease on the difference with the increasing on the amount of services.



(a)



(b)

Figure 6.3: Proactive vs reactive recovery strategies: (a) absolute and (b) relative comparison.

In Figure 6.4 it is presented the allocation of primary slots in the cloud so that the resource utilization in the third F2C layer can be analyzed. For the sake of comparison, the result attained by the exclusive employment of cloud resources is also included, i.e., the deployment of a traditional cloud computing system with no fog layers. This comparison shows that distributed allocation strategies, envisioned by F2C, enable an average load reduction in cloud resources as high as 50%. It is worth mentioning that, in this figure, we do not show the protection allocation in cloud commodities, since we assume that cloud servers implement their own protection techniques as previously exposed in this chapter. Moreover, we highlight that one additional benefit of the horizontal protection approach, employed on the described strategies, is that the fog layers are not tied to the cloud for the provisioning of protection slots.

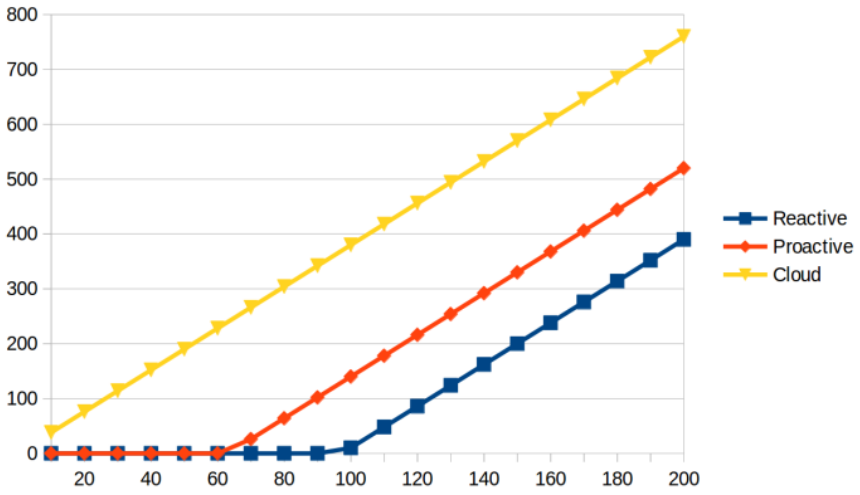


Figure 6.4: Primary resource allocation in cloud.

The analysis of the results presented in this section shows that both proactive and reactive protection strategies are feasible on F2C architectures, albeit the higher underutilization of edge resources observed in the proactive strategy shall hinder its full deployment in scenarios where huge amounts of service requests occurs in a short-term basis, such as ITS, where reactive strategies may be preferred.

Future lines of work include the assessment of the tradeoff between network and computing resources usage in proactive recovery strategies in F2C computing systems. Indeed, sensitive services requiring real-time fault-tolerant execution may leverage a desirable redundancy that may be provided by proactive strategies. However, the allocation of edge devices for primary and backup service execution requires the continuous transmission of the service execution state, such as checkpoints, in order to enable the backup execution to take place with minimum impairment in the eventual occurrence of

failures**.** In addition, this redundancy may drain the already scarce energy resources in several devices at the edge of the network.

6.2.3. Horizontal vs Vertical failure recovery

Albeit the scenario considered for the introduction of horizontal and vertical protection strategies is very similar to the scenario considered so far, this section introduces a key difference. To that end, two failure scenarios are considered: i) failure of one single fog node, where any fog node at either fog layer 1 or fog layer 2 may become inaccessible (similar to the scenario previously considered); and ii) general failure in one area, affecting one fog located in the second fog layer and all the underlying fogs in the first fog layer directly connected to the compromised fog in layer 2 (hierarchical failure). As an example of hierarchical failure, let's consider the scenario shown in Figure 6.5, where during a service execution (1), a general failure in the area (2) would cause disruption of all services being executed in the affected area (3), however, the employment of protection resources (4) may guarantee the service recovery. As in the previous section, it is assumed, for the sake of simplicity, that cloud service providers implement their own protection mechanisms and, thus, can handle eventual failures. Therefore, it is considered that cloud resources are always available.

Protection strategies

In this subsection, distinct strategies for the PSA problem are presented. Distinct strategies may use distinct approaches for defining in which F2C architectural layer the protection resources should be placed, considering the layer where primary resources are allocated. On one hand, the protection strategies so-called horizontal, vertical and hybrid, assume that the failure of one fog node does not affect the accessibility of other fog nodes (single node failure). On the other hand, hierarchical strategies assume a general failure scenario, pointed earlier in this section, where the failure of one node results on the inability for accessing underlying fog nodes, if any. The employed strategies are described as follows.

- Horizontal protection: this strategy considers the allocation of protection resources in the same fog layer where the primary resources are allocated in the F2C topology. The benefit observed in this strategy is the deployment of recovery strategies able to comply with SLA by guaranteeing the availability of protection resources with similar characteristics in comparison to the primary ones. This is the basic approach for the strategies considered in section 6.2.2.

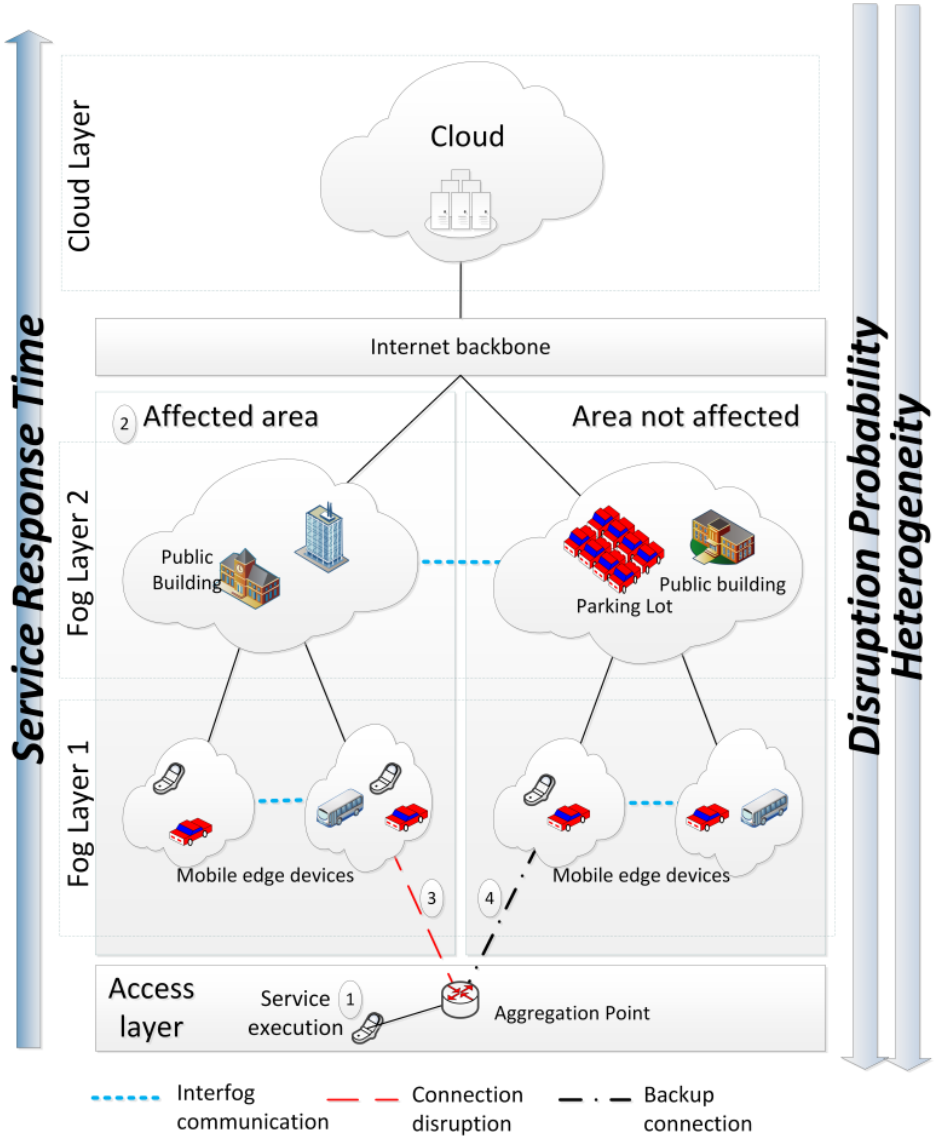


Figure 6.5: General failure scenario.

- Vertical protection: unlike the first strategy, the vertical protection ensures the reservation of protection resources in the F2C layer immediately above the fog node affected by the failure. For instance, a fog located in fog layer 1 affected by a failure relies on protection resources in fog layer 2, while a fog layer 2 failure relies on cloud secondary resources. The employment of this strategy benefits from an even higher availability of resources devoted to primary allocation in lower layers,

- Hybrid protection: this strategy, do not constrain the selection of protection resources to the deployed F2C topology, but to the resource availability. Therefore, as long as resources are available, the protection allocation is always performed in the same layer where the primary resource is. On the other hand, secondary resources are allocated in the layer immediately above. As an example, in low load scenarios, idle fog resources in fog layer 1 could be employed for protection, whilst, in high load scenarios, when resources with low latency are demanded for primary allocation, protection could be allocated at higher layer resources. With this approach, it is possible to reduce the resource underutilization perceived in vertical protection.

- Hierarchical horizontal protection: this strategy is designed to withstand failures affecting a local area, where the non-affected area must have enough secondary resources for recovering the services allocated at the compromised fog node as well as all underlying nodes in lower layers within the affected area. Similar to horizontal strategy, the computing resources selected for protection are positioned in the same layer of the primary resources, but distinct from that strategy, hierarchical protection is placed exclusively on fog nodes positioned in a distinct area (area not affected, as shown in Figure 6.5).

- Hierarchical vertical protection: this strategy is also designed to cope with failures affecting a local area. Similar to vertical protection, computed resources devoted for protection are located at higher layers. However, in this strategy, the amount of protection slots reserved in the upper layer must be enough for the allocation of all primary resources allocated within the affected area.

Extended mathematical model

In this section, the PSA model presented in section 6.2.2 as a MKP problem is adapted in order to consider the protection strategies considered in this section. However, the PSA objective remains the same as presented in the previous model, hence, the model aims at reducing the delay for primary resources allocation whilst decreasing the protection cost. That said, in this section, it is considered the same objective function described by Equation (6.1). Furthermore, the base constraints for the model are also kept. Therefore, Equations (6.2) to (6.6) are also employed in the model assessed in this section.

It is worth mentioning that, in this section, it is not considered the proactive protection approach previously presented. In fact, all strategies modeled in this section employ a reactive protection approach. In order to model the distinct protection strategies assessed in this section, new constraints are introduced specifically for each strategy. Furthermore, new symbols employed by the constraints introduced in this section, besides the ones presented in Table 6.1, are described in Table 6.4.

Table 6.4: New symbols for the extended PSA model

Symbol	Definition
H_r	Set of fog nodes positioned immediately under F2C resource *r* considering F2C hierarchy
G_r	Set of all F2C resources in the layer immediately above fog *r*

For the deployment of both horizontal protection strategies (for single and hierarchical failures), Equation (6.9) ensures that the number of protection slots on each fog layer is not lower than the amount of primary slots allocated in one individual fog node located in the same layer. However, albeit this constraint is enough to guarantee the availability of horizontal protection for the recovery of any single fog node, the hierarchical horizontal protection demands one extra constraint for the proper allocation of secondary slots, which is afforded by Equation (6.10). This equation, which affects exclusively the fog nodes positioned in the second fog layer (represented by L_2), ensures the availability of protection slots for hierarchical horizontal recovery of any fog failure in fog layer 2, i.e., recovery of the compromised node in layer 2 and the nodes in layer 1 affected by this failure.

$$\sum_{i \in S} \sum_{k \in K_r} Y_{i,r,k} \leq \sum_{q \in F_r - \{r\}} \sum_{k \in K_q} X_{q,k}, \quad \forall r \in L_1 + L_2 \tag{6.9}$$

$$\sum_{i \in S} \sum_{q \in H_r} \sum_{k \in K_q} Y_{i,q,k} \leq \sum_{q \in L_1 - H_r} \sum_{k \in K_q} X_{q,k}, \quad \forall r \in L_2 \tag{6.10}$$

In an analogous fashion, vertical strategies demand extra constraints in order to ensure the allocation of protection slots on the upper layers. The constraint presented in Equation (6.11) ensures that the number of protection slots in a layer is enough for the recovery of any fog node failure in the F2C layer below it. Albeit this constraint may be employed by both vertical protection strategies, in the vertical protection for single node failure, the constraint is applied for all fog resources, i.e., the nodes positioned in fog layer 1 and fog layer 2. On the other hand, in hierarchical vertical protection, this constraint is partially applied, i.e.,

only fog resources positioned in fog layer 1 are subject to it. Thus, the allocation of secondary resources for hierarchical protection of fog layer 2 primary resources is enabled by an additional constraint defined by Equation (6.12), which ensures the availability of protection slots for hierarchical recovery of any fog layer 2 node failure.

$$\sum_{i \in S} \sum_{k \in K_r} Y_{i,r,k} \leq \sum_{q \in G_r} \sum_{k \in K_q} X_{q,k}\,, \quad \begin{array}{ll} \forall r \in L_1 + L_2\,, & \text{if vertical strategy} \\ \forall r \in L_1\,, & \text{if hierarchical vertical strategy} \end{array} \tag{6.11}$$

$$\sum_{i \in S} \sum_{q \in H_r + \{r\}} \sum_{k \in K_q} Y_{i,q,k} \leq \sum_{q \in G_r} \sum_{k \in K_q} X_{q,k}\,, \quad \forall r \in L_2 \tag{6.12}$$

Another protection strategy implemented in this work consists in a hybrid of horizontal and vertical protection strategies. This approach, defined by Equation (6.13), consists in prioritizing the allocation of primary slots in resources located in lower layers (vertical approach), but also allowing the use of spare resources in the same layer (horizontal approach) for protection slots, avoiding resources underutilization in lower fog layers.

$$\sum_{i \in S} \sum_{k \in K_r} Y_{i,r,k} \leq \sum_{q \in R} \sum_{k \in K_q} X_{q,k}\,, \quad \forall r \in L_1 + L_2 \,|\, q \neq r \tag{6.13}$$

Results

In this section, we evaluate the impact of distinct modeled protection strategies on the overall resource allocation delay and protection cost. The presented results were attained through the employment of optimization tools, such as PuLP [75] and Gurobi Optimizer [76]. The service and resources parameters considered in the performed simulations are respectively shown in Table 6.5 and Table 6.6.

Table 6.5: Parameters (services)

Parameter	**Value**
Number of requested services	From 10 to 150
Ratio between amount of services: low consuming / high consuming	90 (low) / 10 (high)
Slots demanded by low consuming services	3
Slots demanded by high consuming services	30

Table 6.6: Parameters (resources)

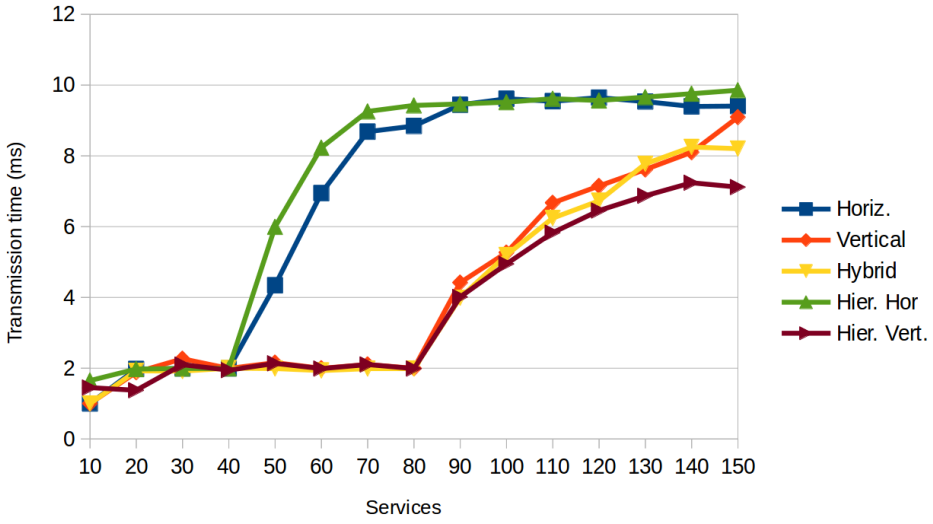
Parameter	Fog layer 1	Fog layer 2	Cloud
Number of F2C nodes per layer	4 fogs	2 fogs	1 cloud
Available resources per node	20	200	Unlimited
Allocation delay per F2C layer (t.u. = time unit)	1 ms	2 ms	10 ms

The impact for each PSA strategy in terms of average service allocation delay is illustrated by Figure 6.6, which confronts the strategies modeled in terms of primary and secondary resource allocation delay. As shown in Figure 6.6(a), the employment of horizontal protection strategies in the simulated scenario resulted on a considerable increase on the average delay for service allocation, even for a relatively low number of services. On the other hand, the delay resulting from the allocation of secondary resources (i.e., slots reserved for protection), as shown in Figure 6.6(b), uncovers the tradeoff observed on protection strategies. Indeed, albeit horizontal protection strategies cannot offer low impact on resource allocation latency for a medium to high number of services, the recovery delay is considerably lower than in other strategies, even when increasing the total number of requested services.

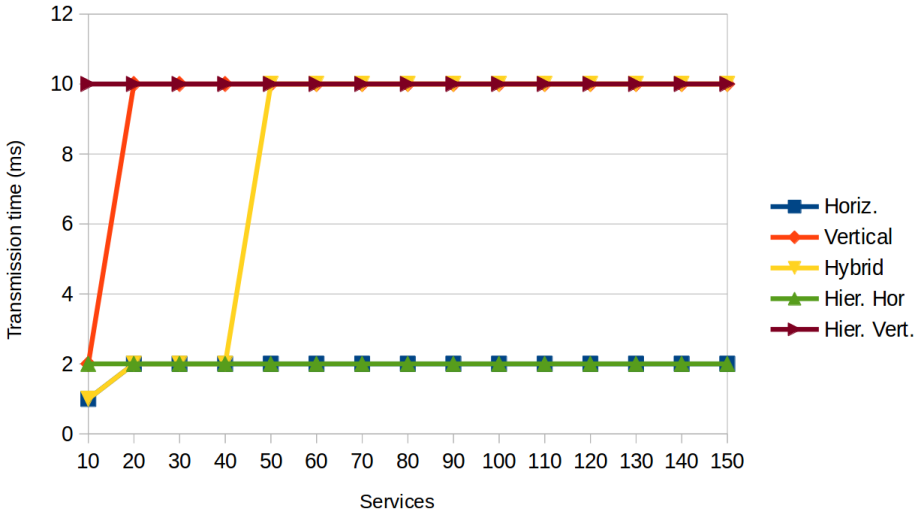
On the other hand, vertical protection strategies offer lower impact on the primary resource allocation at the cost of a higher recovery delay. It is worth noting that the primary allocation delay observed in the hybrid strategy is comparable to the one offered by vertical strategies and, albeit service recovery delay is not constant, a low recovery delay may be achieved by this strategy under reduced load.

A comparison of the resource allocation in the lower fog layer is depicted in Figure 6.7. By analyzing the primary and protection resources distribution, it can be explained the high recovery delay observed in vertical strategies. Notice that, due the nature of vertical strategies, all resources slots offered by fog layer 1 are allocated as primary use, even when not all resources in this layer are allocated, inhibiting a low delay recovery, while the hybrid allocation strategy employs spare resources for protection.

Moreover, Figure 6.7 shows that resource utilization in fog layer 1 can be as low as 50% for hierarchical horizontal protection, whilst a utilization ratio of 75% is achieved for horizontal protection (for single node failure), also endorsing the previously presented analysis regarding the low recovery delay in horizontal strategies, which is enabled by the high availability of secondary resources in lower layers.



(a)



(b)

Figure 6.6: Allocation delay for (a) primary and (b) secondary slots.

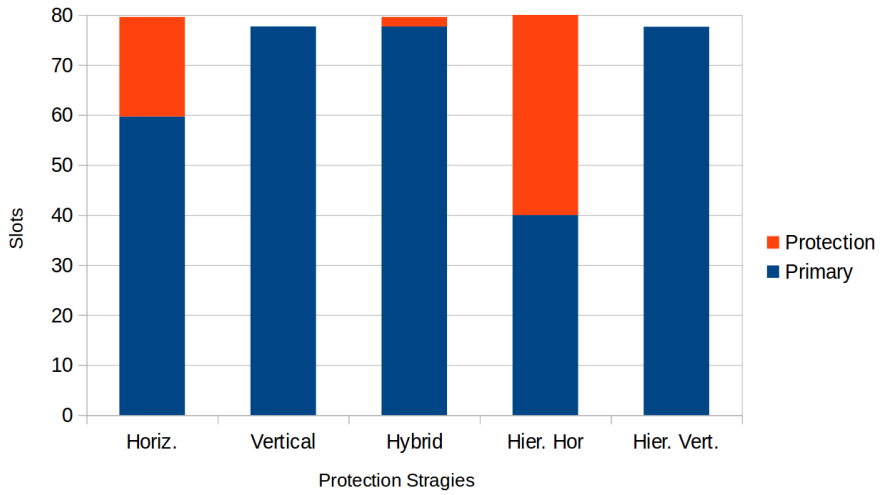
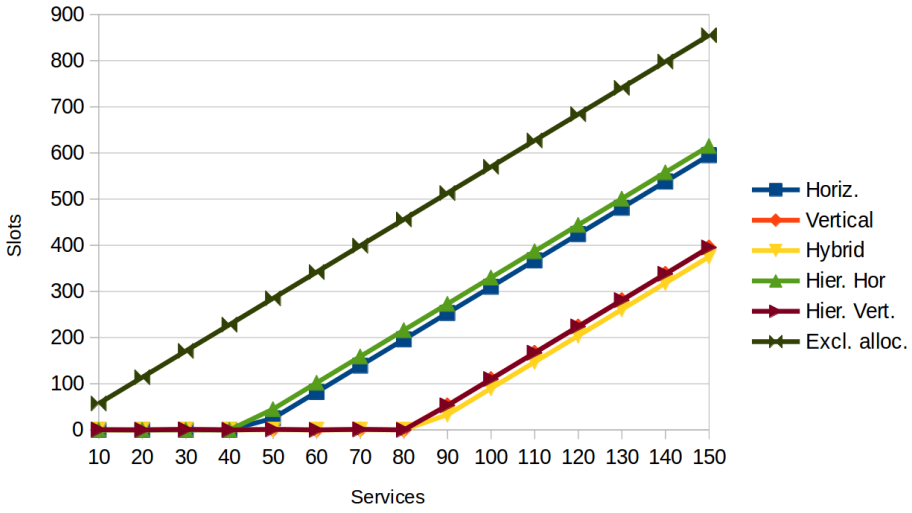


Figure 6.7: Average resource allocation in fog layer 1.

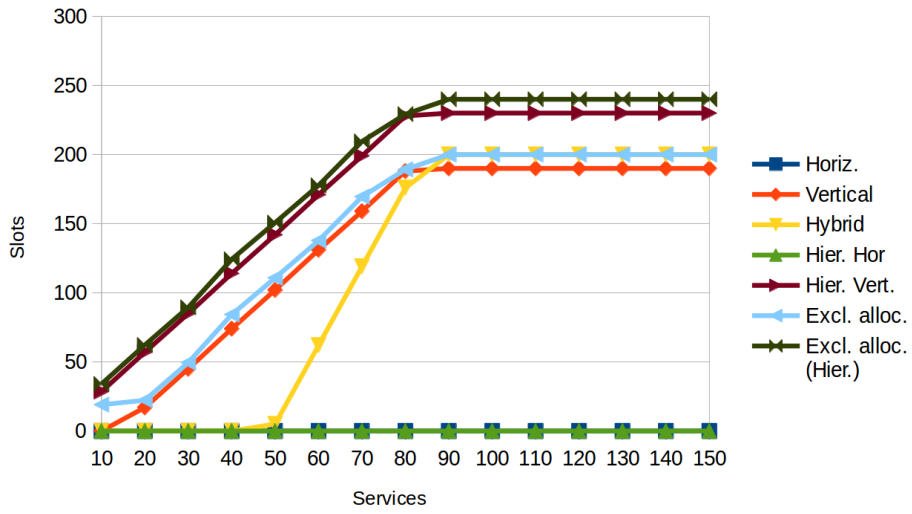
On the other hand, Figure 6.8 presents the allocation of primary and secondary slots at cloud. In Figure 6.8(a), for the sake of comparison, it is further included the results attained by exclusive cloud allocation, i.e., the employment of a conventional cloud computing scenario, with no fog layers, showing that the distributed allocation strategy envisioned by F2C enables a reduction on cloud resources load higher than 50% in comparison to vertical protection strategies, even when several service requests are considered. Albeit the load reduction presented by horizontal allocation strategies is lower, these strategies do not demand additional cloud resources for secondary slots allocation, as shown by Figure 6.8(b). Indeed, this is due to the assumption that failure recovery at cloud is transparent to the F2C system, since cloud providers implement their own resilience strategies.

For the sake of comparison, two distinct approaches for the exclusive cloud allocation are also shown in Figure 6.8(b), where the first –Excl. alloc.– refers to the allocation of secondary slots demanded by the recovery of any single fog failure, while the second –Excl. alloc. (hier.)– considers the recovery of a failure affecting one area, as earlier discussed in this section.

Finally, albeit the combined F2C architecture can significantly reduce both the consumption of resources in higher layers and the service allocation delay, it can be concluded from the presented results that distinct failure scenarios may require distinct protection strategies to be considered aiming at the deployment of the most suitable one, regarding to specific demands of each scenario. For instance, fog nodes providing sensitive services requiring real-time recovery, such as healthcare and Intelligent Transportation Systems (ITS), might deploy horizontal protection strategies, whilst fog nodes supporting no sensitive and high resource consuming applications, such as video streaming, might deploy vertical or hybrid strategies. Notice that, both sensitive and no sensitive services can be executed in a reliable way, while preventing cloud resources overload.



(a)



(b)

Figure 6.8: Resource allocation at cloud: (a) primary; (b) secondary slots.

7. Impact of control topology on QoS-aware service allocation

Distinct control topologies and the impact on QoS provisioning in cloud domains have been assessed in several works in the literature, such as [99] and [100], where the scalability of SDN controllers in data centers is discussed. Nevertheless, QoS researches in the fog arena, such as [101], are focused on data-plane and, since the F2C architecture proposes a distributed management model aiming at real-time resource provisioning for the execution of sensitive services, such as e-Health or Smart Transportation, the negative impact on QoS due to the delay generated by control decisions cannot be neglected. Therefore, we advocate for the need of assessing the distributed F2C control plane topology in order to enable the successful deployment of this novel architecture. Hence, in order to enable the establishment of low latency end-to-end communication, a key point is the reduction on the amount of controllers demanded for taking control decisions, while diminishing the average delay provided by each one of them. Furthermore, in the layered F2C control plane, the scalability undoubtedly depends on the number of layers, on the total amount of entries in the database of each controller –i.e., the number of devices managed by each one–, and on an efficient allocation of heterogeneous controllers within each control plane layer.

In this chapter, we assess the distributed F2C control topology and its impact on QoS-aware end-to-end communication. Therefore, the main goal of this chapter is threefold: (1) briefly discussing the most important features of the F2C architecture; (2) evaluating the impact of the amount of entries in the controllers' resource database, on the time expended by controllers to resolve a query; and (3) assessing the impact of the control topology on QoS, in terms of latency, especially for services requiring very high performance. The ideas introduced in this chapter, as well as the presented results, were previously validated and published in [102].

7.1. Background

Due to the novelty of F2C, there are no contributions assessing the control plane topology suiting the F2C architecture. Taking this into account, we revisit in this section distinct contributions also coping with distributed control scenarios, paying special attention to recent contributions on the SDN sphere.

The work presented in [99] assesses distinct SDN/OpenFlow controllers in terms of performance, scalability, reliability and security. For performance and scalability analysis,

authors evaluate the impact of the amount of underlying hosts on each considered controller. Albeit the amount of connected hosts has presented low impact on some implementations, fog controllers are not expected to offer enough robustness to manage the huge number of edge resources envisioned by future IoT scenarios. Moreover, the service-oriented communication approach, which is considered in this thesis as a trend for upcoming real-time applications, shall add extra complexity on the lookup table match in comparison to the observed in current host-oriented network applications. The work presented in [100] assesses the scalability of distinct topologies for an SDN control plane, be it centralized, decentralized –with local or global view– or hierarchical. The impact of the number of connected hosts is shown by authors through a mathematical model, which resulted in a considerable impact, even for hierarchical topologies under some of the described network configurations.

The work presented in [101] exploits the deployment of MDCs in fog resources for achieving low delay execution of sensitive services such as remote healthcare and monitoring. The strategy presented by authors is dedicated on preprocessing raw data, enabling low service processing and response through the employment of fog resources.

Finally, the work in [15] addresses SDN scalability concerns considering both centralized control, such as data centers, and distributed control architectures, as required by services deployed for distributed execution. After discussing scalability trade-offs, authors conclude that distributed services must be delay-tolerant, since the communication among distributed controllers shall result in a delay inherent to the communication of geographically distributed network elements.

7.2.Control as a Service (CaaS)

The service-oriented resource allocation, focus of this thesis, is illustrated in Figure 7.1 establishing communication between two devices at the edge of the network through a layered F2C control plane. Upon receiving a service request from an end-user –for instance, looking for the closest available ambulance–, the lower controller layer (CL1) may either select a resource at its table of underlying resources or forward the service request to its parent controller located in the second control layer (CL2) for the service provisioning according to the suitability of existing resources. In the latter case, a new decision must be taken among employing underlying resources or forwarding the request to parent controller, and so on. It is worth highlighting that this approach, which is adopted for the rest of this chapter, does not consider the horizontal communication between controllers. Furthermore, each controller at CL2 must carry the aggregated information of resources managed by underlying controllers at CL1. Hence, when controller CL1-a forwards a request to CL2-a, this controller checks –through the aggregated resource information– the availability of the requested resource in CL1-b and, if available, it connects to CL1-b, which is responsible for mapping the requested service into the most suitable resources.

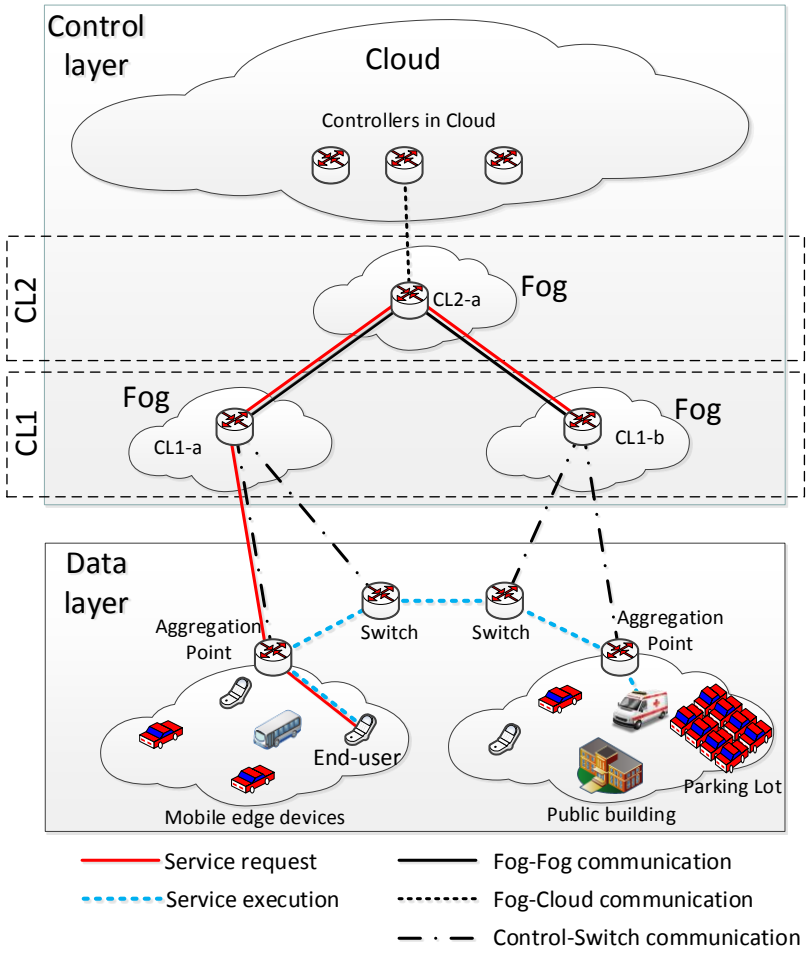


Figure 7.1: Service request in a service-oriented F2C scenario.

Leveraging the fog computing, SDN and SOA concepts, the F2C distributed control plane may be deployed at idle edge resources, enabling low-delay control decisions by placing controllers as close as possible to end-users. Therefore, it is positioned in this section the Control as a Service (CaaS) concept, where edge resources –ideally resources offering high reliability and low volatility– can play a controller role through the employment of processing and storage resources for storing resource databases, mapping service requests into the most suitable resources, among others controllers tasks.

Albeit the controllers placed on lower layers can offer lower network latency due to the proximity to end-user, the processing of service requests latency is dependent on the amount of underlying resources managed by each controller, thus, the low capacity observed on these devices, both in terms of processing and storage, hinders the management of the large number of resources at the edge of the network, as well as the information regarding offered services and devices characteristics. Consequently, the employment of these constrained

edge devices for managing resource databases containing several thousands of entries will, undoubtedly, increase the overall delay. In this scenario, it is important to assess both network and processing delay, considering the load per controller, the amount of controllers per control layer, and the number of control layers in the F2C topology. Moreover, the deployment of this concept requires several challenges to be addressed, including optimal selection of controllers according to resources suitability, migration strategies, and robustness, just to name a few.

In such a dynamic scenario, potentially embracing billions of devices, we envision a QoS provisioning strategy based on the dynamic setup or tear down of new controllers according to a given control topology and the information regarding edge resources. For instance, the number of controllers, as well as the control topology, may be automatically tailored as the amount of edge resources increase or decrease. Let us consider the scenario presented in Figure 7.2, where the first presented control topology consists of a cloud controller responsible for the management of underlying resources in two distinct areas. The transition illustrated by the blue arrow 1 illustrates the creation of lower control layer (CL1), as well as the deployment of fog controllers in distinct areas in CL1, in order to deal with the high control delay arisen from the communication with cloud. Furthermore, with the growth on the amount of edge resources, existing controllers may benefit from the deployment of new ones, as shown by arrow 2, preventing controllers overload. However, as the amount of controllers in CL1 increases, the inter-controller communication tends to increase as well, thus impacting negatively on the average delay due to the need for forwarding requests to the cloud controller –the reader should remember that horizontal communication is not considered in this work. In order to deal with this effect, a new control layer (CL2) may be deployed, as illustrated by arrow 3, preventing frequent communication with cloud controller.

It is worth noting that, if the communication among nodes within the same area in a control layer L, located immediately below the cloud layer C, is frequently set, a new controller may be deployed along with an additional layer located between layers L and C. For instance, as shown in Figure 7.2 (arrow 3), even when the amount of controllers in CL1 is relatively low, the deployment of CL2 containing one single controller per area shall eliminate the need for forwarding intra-area requests to cloud, thus diminishing the average communication delay in the considered area.

7.3.Delay modeling

In order to model the overall delay, we consider two sources of delay when setting the communication between the service client and the most suitable resource for the service provisioning: 1) controllers processing delay; and 2) network delay. In this section, the most important considerations regarding de delay modeling are presented.

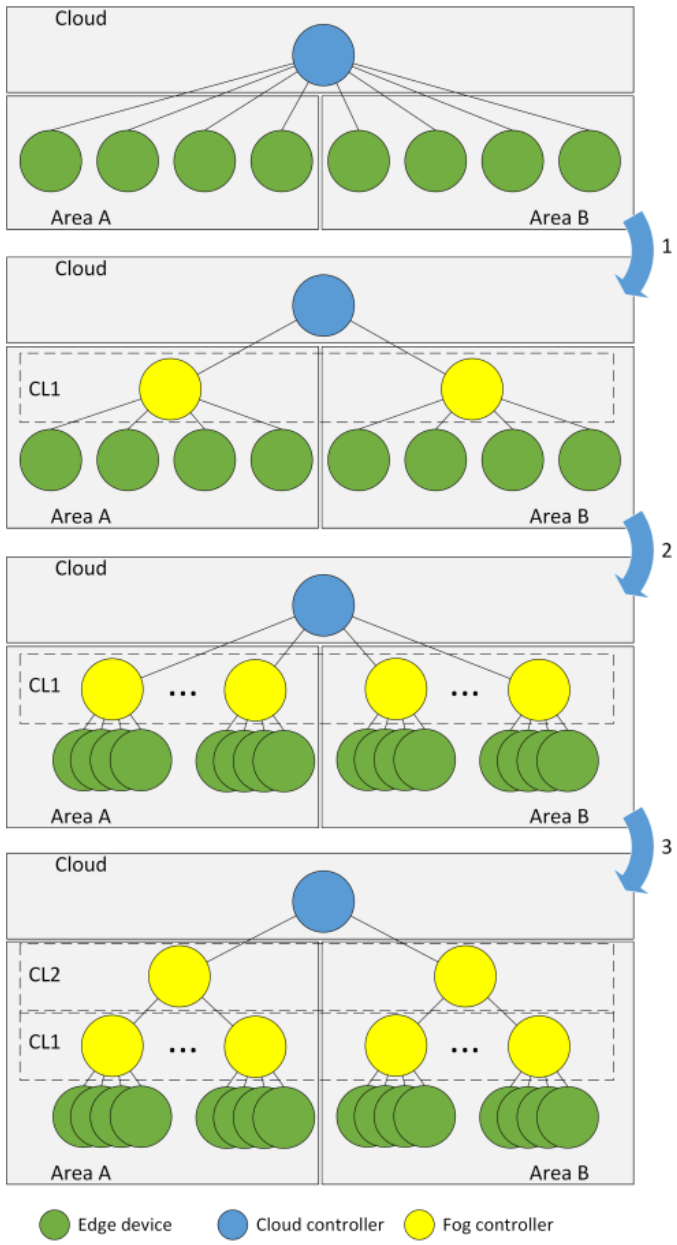


Figure 7.2: Deployment of controllers.

Processing delay

Let the processing delay for each controller be the lookup time aiming at mapping the requested service into the most suitable underlying resource in its database, if any. If no suitable resource is found, the request is forwarded to its parent controller, where the procedure is repeated, and so on. In order to contemplate both heterogeneity and amount of database entries on controllers at the edge of the network, the processing time is estimated through an experiment considering several databases with the amount of entries –each entry representing one resource– varying from 10^3 to 10^6 deployed at 3 distinct devices with distinct processing capacities: 1) Intel® Core™ i5-4570 CPU @ 3.20GHz; 2) Intel® Xeon® CPU E5-2620 v2 @ 2.10GHz; and 3) ARMv7 Processor rev 4 (v7l) @ 1.20GHz. The operating system used in all devices is Ubuntu 16.04, the databases were deployed in MySQL v5.7.17, and each plotted value is an average of 20 distinct queries taking into consideration distinct attributes of the resources table, such as resource type and geographical position where it is located.

The experimental results, depicted in Figure 7.3, show that the variation in the queries processing latency according to the amount of entries in each device is linear. Therefore, through the simple linear regression model, the trend line for each device is also plotted in Figure 7.3, given by $ax+b$, where b is the base delay defined for each controller, a is the delay added by each entry in the controller database, and x is the database size in terms of number of entries. Therefore, for the further results presented in this chapter, 3 heterogeneous resource types in terms of computing capacity are defined by employing the 3 obtained pair-values a,b, as shown in

Table 7.1. Hence, once these linear constants are defined for each resource type, the processing delay d is given as a function of the variable x, $d=f(x)$. Moreover, the inverse function $f^{-1}(d)$ can be employed in order to determine, given the maximum delay accepted in one single controller, the maximum number of resources it is able to manage. It is worth mentioning that, albeit the capacity of each controller in terms of storage is a key factor for determining the maximum resource database size, this model does not take the storage capacity into account. This may be explained due to the fact that the main goal of the model is to assess the delay on distinct F2C control topologies and, albeit a huge number of entries in the resource database will surely demand additional storage, it is assumed, for the sake of simplicity, that the controller has enough capacity to store the complete database, with all resources information relevant to the mapping functionality. With this approach, the database size shall impact exclusively on the processing time. Indeed, the rationale behind this assumption is to show that even with extended storage capacity, the constrained processing capacity at edge resources is still a bottleneck for the deployment of a static control topology providing QoS-aware end-to-end communication in a scenario presenting high dynamicity.

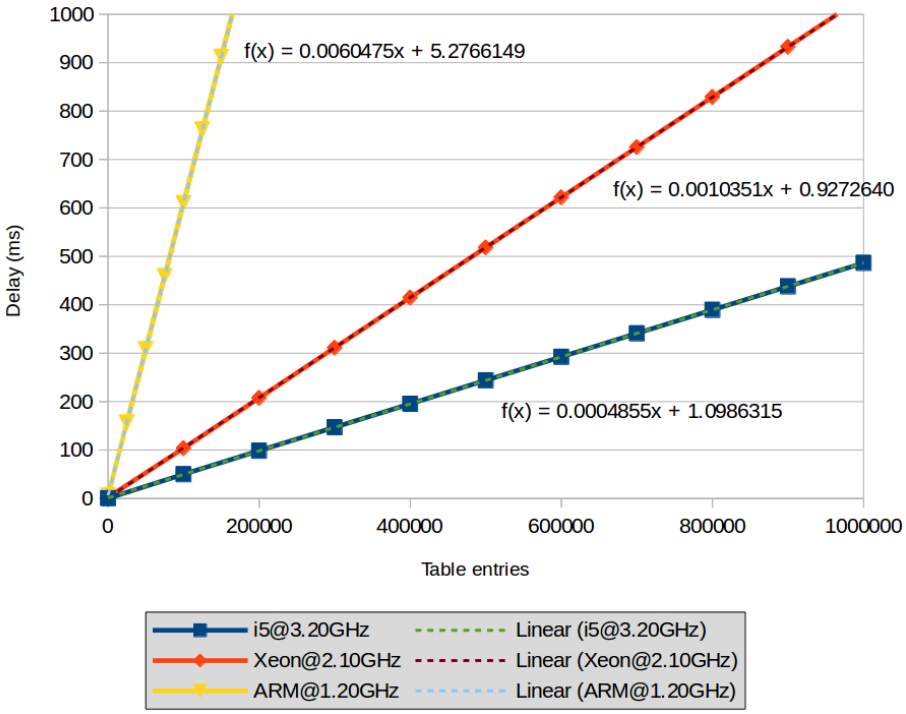


Figure 7.3: Impact of database size on processing delay.

Table 7.1: Linear constants for processing delay estimation

Device	**a**	**b**
i5@3.20GHz	0.0004855	1.0986315
Xeon@2.10GHz	0.0010351	0.9272640
ARM@1.20GHz	0.0060475	5.2766149

Network delay

Although the network delay is dependent on the communication technology employed on each controller network link, we assume, for the sake of simplicity, that the delay for links between 2 adjacent fog controllers –short range communication– is homogeneous and considerably lower than the delay observed on fog-cloud controllers communication –long range communication. In order to estimate the average delay for each communication type, the Zenmap 6.40 network analyzer tool [103] is used to evaluate the communication delay with hosts in distinct real networks, including hosts placed in traditional cloud servers (long range communication) as well as servers at the *Universitat Politècnica de Catalunya* (UPC), where this work is carried out (short range communication). Considering the average delay attained in each assessed communication range, shown in Table 7.2, it is assumed the communication delay between two fog controllers to be 1ms, while the communication delay between fog and cloud controllers is assumed to be 20ms.

Table 7.2: Average network delay

Cloud providers	**Delay (ms)**	**UPC hosts**	**Delay (ms)**
Amazon	45.246	Host 1	1.202
Google	9.959	Host 2	0.536
IBM	13.066	Host 3	1.250
Microsoft	13.143	Host 4	1.205
Average	*20.353*	*Average*	*1.048*

Taking into account the defined parameters for the network delay model, the overall network impact must be calculated according to the control topology. On one hand, each layer added to the control topology impacts negatively on the overall latency for edge to cloud controller communication, since it requires an additional fog-fog controller communication. On the other hand, it reduces the load at each controller as well as the end-to-end delay for controllers either within the same area or in areas geographically close to each other, since it is not necessary to communicate with higher layer controllers. In addition, albeit not considered in this work, it might be assumed that, in real IoT applications, the probability of connection between neighbor edge devices is higher than between devices located at further distance, which would result in an additional benefit on the deployment of topologies with more layers.

7.4.Results

In this section, the attained results concerning the impact of distinct control topologies on QoS in terms communication delay is presented. For each simulated scenario, the average communication delay takes into account the end-to-end delay from all-to-all edge nodes, i.e., for each device at the edge of the network, the average communication delay with each remaining edge device is calculated, taking into account both processing and network delay of used controllers. We define 4 resource types, referred to as A, B, C and D, where types A, B and C can play as controllers, whilst type D refers to all resources that cannot be employed as controllers –for instance, devices dedicated to data collection or storage. When resource types A, B or C are playing the controller role, the processing delay concerning to control decisions is based on the linear constants obtained, respectively, for 3 distinct devices evaluated in the previous section. Concerning the network delay, the latency values attained by the experiment presented in the previous section were used as base values for both considered communication ranges, as shown in Table 7.3.

Table 7.3: Control delay parameters

Parameter	**Value**
Processing delay linear constants a,b Resource type A (i5 based)	a = 0.0004849177, b = 1.4849688695
Processing delay linear constants a,b Resource type B (Xeon E5 based)	a = 0.0010358677, b = 0.4007560311
Processing delay linear constants a,b Resource type C (ARMv7 based)	a = 0.0060473114, b = 5.2790401437
Maximum processing delay per controller type	A = 4ms B = 4ms C = 10ms
Network delay for fog-cloud communication (long range communication)	20ms
Network delay for fog-fog communication (short range communication)	1ms

In order to evaluate the impact of distinct control topology configurations, two set of simulations are performed. In the first set, the maximum accepted processing delay per

controller node is modified in order to assess how the controllers' database size affects the end-to-end communication delay. On one hand, a lower processing latency can be achieved on each controller through the reduction of the amount of edge resources managed by each one of them. On the other hand, a reduced number of entries on the controllers database demands frequent communication among distinct controllers, which has a negative impact both on network delay and overall processing delay. This is justified by the fact that, in average, a higher number of forwardings may occur, requiring more controllers to process one single service request.

In Figure 7.4, this tradeoff is presented for distinct scenarios in terms of the number of available resources of each type and their maximum processing delay. For all scenarios presented in Figure 7.4, the total amount of edge resources is 2^{24} (=16777216). Moreover, the number of resources of types A, B and C, which may be eventually used as controllers, is shown in the figure for each one of the 4 considered scenarios. Notice that, for each scenario, we fix the maximum allowed processing delay for 2 resource types, while the one showed in the column "Varying" of the figure's legend is varying from 1 to 30ms according to axis *x* (see Figure 7.4). In addition, for each scenario, the resources types using fixed maximum processing delay are set to static values shown in Table 7.3 –i.e., 4, 4 and 10ms for resource types A, B and C, respectively. These values are chosen based on the best delay combinations obtained based on several simulations performed in this scenario.

The results show that an increment on the maximum delay per controller, for any simulated scenario, leads to a gradual increase on the end-to-end communication delay. However, the adoption of a very low controller delay yields a quick growth on the network delay, since the number of control layers is increased due to the low capacity of controllers, highly impacting on the overall communication latency. Furthermore, a confrontation of the communication delays illustrated by the blue and red line in Figure 7.4 shows that, albeit the inclusion of extra resources able to play a controller role may reduce the overall delay, the employment of big control databases—i.e., high controller delay—may void the benefits brought by the increased resource availability.

In the second set of simulations, the average control plane delay is evaluated in terms of the amount of nodes at the edge of the network. Unlike the previous simulations, for the sake of simplicity, only type A resources are allocated as controllers, and the amount of controllers required is determined considering the maximum processing latency for each one of them. As shown in Figure 7.5, as long as the controllers have capacity to manage the amount of edge resources, only a slight growth is perceived on the delay as the number of edge devices increases. However, when considering a maximum processing delay for controllers as low as 2 or 3ms, either the deployment of extra controller layers or the exploitation of cloud controllers is unavoidable, since the increasing on the number of edge devices requires the deployment of new controllers.

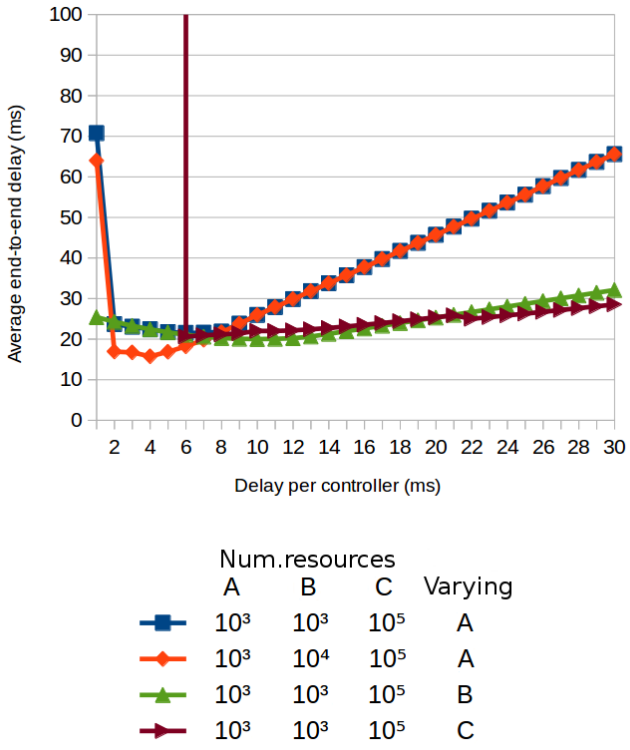


Figure 7.4: Impact of controllers' delay on QoS.

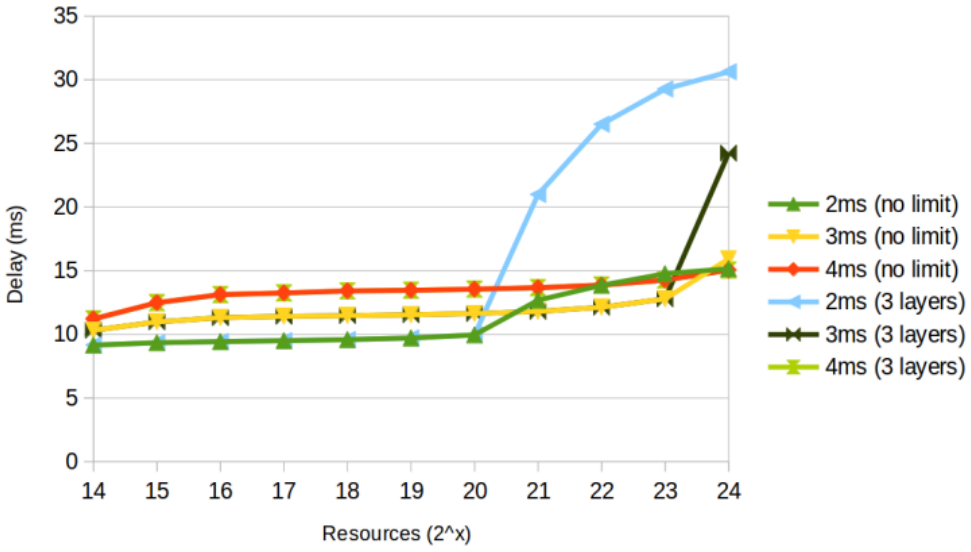


Figure 7.5: Impact of network size on QoS.

In order to confront both alternatives, the simulation is executed with a limitation on the maximum number of control layers to 3 –i.e., 2 control layers in the fog and 1 in the cloud. By preventing the deployment of the third fog layer, the controllers in layer 2 must forward requests to cloud controller as the amount of underlying edge resources increase. As it can be noticed by the presented values –represented in Figure 7.5 by the series tagged with "3 layers" caption–, with more than 2^{20} and 2^{23} resources when the maximum processing delay is respectively 2 and 3ms, the increase on the overall communication delay due to the cloud network latency is considerably higher than the addition of a new layer. On the other hand, when a controllers processing delay may be as high as 4ms, there is no need for the deployment of an additional controller for the number of underlying resources considered.

It is worth mentioning that, with the growth on the number of underlying resources, even the controllers allowing a 4ms processing delay shall require the deployment of new controllers and, consequently, additional layers. As a conclusion, strategies aiming at the deployment of controllers in such layered architecture must consider several variables to determine, in a highly dynamic scenario, the optimal decision for guaranteeing QoS-aware end-to-end communication. These decisions may include either tailoring the amount of control layers, or relaxing the maximum delay allowed per controller, or even exploiting cloud controllers.

In addition, albeit we consider that idle resources at the edge of the network may be dynamically employed as controllers –described in this chapter as the Control as a Service (CaaS) concept–, in a real scenario, it would be necessary, as a first step, the discovery of the set of available resources able of playing the controller role, later gathering required information for setting possible layers for each resource, and finally creating the controllers topology by setting the hierarchical relationship among them. Furthermore, new challenges regarding the management of edge controllers shall arise due to their inherent constraints, such as dynamicity and volatility, just to name a few. Finally, distinct services may require distinct controller characteristics, including distinct control topologies.

8. Conclusions

The increasing amount of devices at the edge of the network along with the advances on their hardware capabilities, such as processing power, memory and energy availability, number and type of sensors, among others, in addition to the network infrastructure advances, providing ubiquitous connectivity, has enabled the so-called Internet of Things (IoT). In this thesis, we envision a future IoT scenario where the traditional host-oriented communication model will not be able to cope with all potential services requirements. Whilst the immersion of the SOA concept into network research, especially in an IoT scenario, shall enable the deployment of innovative services, where the end-points executing each task related to the main service do not need to be known in advance, the efficient resource allocation in this model brings up several challenges, especially when taking into account characteristics, such as the dynamicity and heterogeneity, inherent to devices considered in IoT scenarios.

In order to achieve an efficient service-oriented resource allocation in this scenario, the employment of two distinct baseline technologies are considered in this thesis. In a first approach, the Path Computation Element (PCE) is extended to the Service-oriented PCE (S-PCE), as described in a comprehensive manner in Chapter 2, resulting in a centralized model for service orchestration, where both service and network requirements are taken into consideration, through traditional and new parameters added to PCEP messages, aiming at selecting the more suitable resources according to service request and paths between end-points. However, the centralized approach adopted by S-PCE, yields a non-negligible network delay, especially services requiring real-time service allocation, such as e-Health or ITS.

On the other hand, the inception of a new computing paradigm, so-called fog computing, envisioned the deployment of fog nodes at the edge of the network through a highly virtualized environment leveraging idle resources shared by end-users. As a consequence, low communication delay between the end-user and the virtualized resources constituting micro data centers was enabled, as well as lower energy consumption, reduced network load, enhanced end-user security, among other.

Taking these advantages into account, in a second approach, the Fog-to-Cloud (F2C) concept is considered, where the benefits of both cloud and fog computing are combined through an architecture for coordinated management of resources distributed at fog and cloud premises. Nevertheless, since F2C is still on its infancy, the service allocation in this

architecture requires several issues to be addressed. In this thesis, the main contributions regarding the service allocation in F2C are discussed in Chapter 5, including the efficient hierarchical resource distribution and assessment of distributed and parallel service allocation into distinct F2C layers, taking into account distinct aspects of the service allocation, such as type of required and available resources, load balancing, energy consumption and resources mobility. The protected service allocation is further discussed in Chapter 6, where several approaches for failure recovery are presented and modeled, showing that distinct services may employ distinct protection strategies considering specific requirements, such as resource availability and recovery delay tolerance. On the other hand, Chapter 7, rather than service transmission delay, focuses on control decisions delay in the distributed F2C control plane, assessing the impact of distinct control topologies on controllers overload, number of layers on the hierarchical F2C control plane and, consequently, the service-oriented resource selection delay.

In short, the contributions of this thesis are:

1. Extending the PCE routing technology to a Service-oriented architecture, so-called S-PCE [10], where, rather than only selecting the best path in a host-oriented communication model, the proposed architecture aims at selecting both the most suitable resources for the execution of a service requested by an end-user, and the best path for reaching out the selected resources. The service/resource mapping is accomplished through the employment of an extended ID/LOC Split Architecture, consisting in a 2-step mapping, where the first step maps a service IDs into the most suitable host IDs, considering both service and network requirements and resources capacity, whilst the second step maps the selected hosts IDs into currently used LOCs, preventing service disruption, caused by the dynamicity observed in IoT scenarios. The presented architecture scalability is assessed regarding the mapping latency in terms of the amount of edge resources, outperforming traditional mapping schemes globally employed, such as DNS.

Outcome:

- V. B. C. Souza, X. Masip-Bruin, E. Marin-Tordera, W. Ramírez and S. Sánchez-López, "Towards the scalability of a Service-oriented PCE," in *2015 20th European Conference on Networks and Optical Communications - (NOC)*, London, 2015.
- W. Ramirez, V. B. Souza, E. Marin-Tordera and S. Sanchez, "Exploring potential implementations of PCE in IoT world," *Optical Switching and Networking*, vol. 26, pp. 48-59, November 2017.

2. Proposing, for the first time, the set of attributes to be considered on the distribution of F2C resources into distinct cloud and fog layers, i.e., resources capacity, vicinity, and reachability to end-users [55]. Moreover, it is assessed, for the first time, the distributed service allocation considering the cloud layer as well as the

heterogeneous fog layers in terms of overall capacity and access delay –as a consequence of the distinct characteristics of the resources constituting each layer. Obtained results show that the deployment of lower fog layers offering low delay, in a tradeoff with the limited overall capacity, may provide QoS-aware allocation of sensitive services, whose requirements are often low in terms of resource capacity, whilst inflexible in terms of accepted delay. In addition, distinct failure recovery mechanisms have been assessed and it was shown that distinct protection mechanisms may be employed according to distinct service categories.

Outcome:

- V. B. C. Souza, W. Ramírez, X. Masip-Bruin, E. Marín-Tordera, G. Ren and G. Tashakor, "Handling service allocation in combined Fog-cloud scenarios," in *2016 IEEE International Conference on Communications (ICC)*, Kuala Lumpur, 2016.
- V. B. Souza, X. Masip-Bruin, E. Marin-Tordera, W. Ramirez and S. Sanchez, "Towards Distributed Service Allocation in Fog-to-Cloud (F2C) Scenarios," in *2016 IEEE Global Communications Conference (GLOBECOM)*, Washington, DC, 2016.
- X. Masip-Bruin, E. Marín-Tordera, A. Gómez, V. Barbosa and A. Alonso, "Will it be cloud or will it be fog? F2C, A novel flagship computing paradigm for highly demanding services," in *2016 Future Technologies Conference (FTC)*, San Francisco, CA, 2016.
- W. Ramirez, X. Masip-Bruin, E. Marin-Tordera, V. Souza, A. Jukan, G.-J. Ren and O. G. d. Dios, "Evaluating the Benefits of Combined and Continuous Fog-to-Cloud Architectures," *Computer Communications*, vol. 113, pp. 43-52, 15 November 2017.
- V. B. Souza, X. Masip-Bruin, E. Marín-Tordera, W. Ramírez and S. Sánchez-López, "Proactive vs reactive failure recovery assessment in combined Fog-to-Cloud (F2C) systems," in *2017 IEEE 22nd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, Lund, Sweden, 2017.
- V. B. Souza, W. Ramírez, X. Masip-Bruin, E. Marín-Tordera, S. Sánchez-López and G.-J. Ren, "Towards Service Protection in Fog-to-Cloud (F2C) Computing Systems," in *2017 Future Technologies Conference (FTC)*, Vancouver, Canada, 2017.

3. Proposing the Control as a Service (CaaS) concept, envisioning the allocation of resources at the edge of the network for control purposes (controllers) in order to enable control decisions to be taken closer to end-users, especially for services requiring real-time allocation. The benefits on the deployment of controllers on-demand in a highly dynamic scenario, such as IoT, is demonstrated by assessing distinct control plane topologies according to the amount or resources to be managed at the edge of the network [102].

Outcome:

- V. B. Souza, A. Gómez, X. Masip-Bruin, E. Marín-Tordera and J. Garcia, "Towards a Fog-to-Cloud control topology for QoS-aware end-to-end communication," in *2017*

IEEE/ACM 25th International Symposium on Quality of Service (IWQoS), Vilanova i la Geltru, Espanha, 2017.

9. Glimpse into future work

9.1.Service placement and execution

According to what it has been discussed and presented in this thesis, it is clear that fog computing and cloud computing, when deployed in a collaborative fashion, shall bring up remarkable benefits. Aligned to this, the combined F2C computing has been recently proposed as a powerful framework to make the most out of both distributed and centralized resource allocation. It is worth mentioning that, *distributed* and *centralized* placement do not necessarily mean allocation in fog and cloud respectively. We consider that, a *distributed* allocation occurs when a service is placed in several resources, whilst the *centralized* allocation occurs when the service is placed in one unique resource, be it either in cloud or fog.

F2C aims at optimizing the execution of IoT services with high requirements in terms of both processing and network delay through a perfect matching between resources capacities and services demands. Therefore, we define service placement as the management decision regarding where a service (or atomic service—as described section 5.3) is to be allocated for execution. The strategy to be deployed for services placement must consider specific demands of a service, be it atomized or not, as well as the set of available resources at both fog and cloud premises. Notice that distinct atomic services may have distinct characteristics in terms of accepted delay, required network and IT resources, input and outputs, holding time, security, interdependence, failure tolerance, etc. The service placement in a real-scenario must consider all these aspects.

Albeit simple strategies, such as Random-Fit and First-Fit, presented in Section 5.4, have shown good results in comparison to the approach based exclusively on cloud placement, enhanced placement strategies considering decisions regarding atomization, distribution, and parallel execution should be further assessed. Indeed, while still in its infancy from the research perspective, the collaborative model envisioned by the F2C model for service placement and resource allocation is a salient feature. In line with this, we propose two distributed approaches to be considered for service execution and placement in combined F2C systems, in addition to a centralized approach, where a service would be completely placed and executed at either cloud, or one single fog edge-device.

In order to illustrate the preliminary steps for the service placement, which include decisions regarding which of the three mentioned approaches should be considered for a

proper service placement, let us assume a service that is complex enough to be atomized, that is, a service implementing the primitives and signaling mechanisms required for its atomization. The workflow shown in Figure 9.1 illustrates possible decisions to be taken for service atomization, placement and execution, which are described as follows.

The service request is initially sent to an F2C management system, which is in charge of placing the service. Thus, as a reaction to the received request, two decision-making processes may be sequentially issued, where the first one is triggered immediately after the service request is received and is responsible for deciding whether the service should be atomized or not. This analysis considers the information regarding both requested service –for instance, type and capacity of required resources for its execution– and available resources –for instance, type, processing capacity, network load and the fog/cloud they are currently contributing with. If the service atomization is chosen, the second decision is responsible for deciding if the service should be executed sequentially or in parallel. Since the service has been atomized after the first decision, the second one may be handled through the analysis of the atomic services execution graph created as the atomization process output.

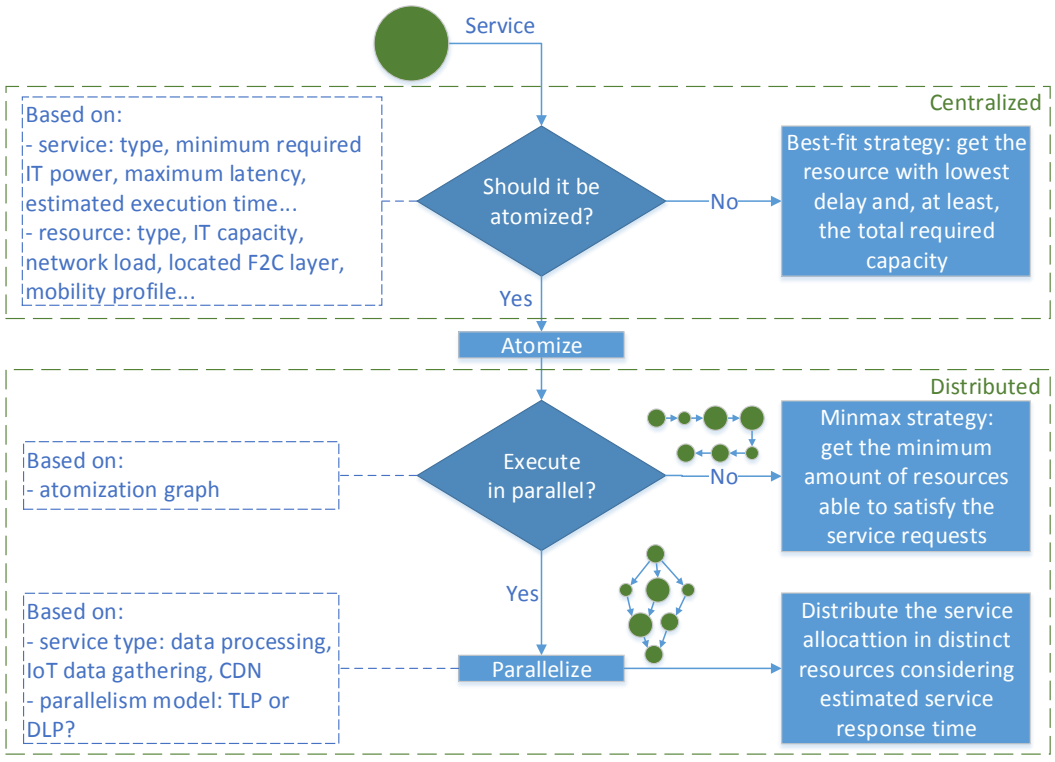


Figure 9.1: Decisions for service placement.

Notice that a centralized service execution may, in fact, be the best strategy in some scenarios, stemming from the complete service placement on one resource, be it cloud or

fog. Taking as example a centralized Best-Fit strategy, the selection of one unique resource offering the lowest network delay, whilst complying with all service requirements, may provide acceptable QoS for a range of services. Indeed, such centralized strategy may reduce the signaling and atomization effort, as well as the network load. For instance, a service collecting all required data from one single data source may benefit from a centralized strategy, whilst, on the other hand, no single resource can meet multiple constraints regarding service quality for more complex service scenarios, where centralized placement would result in suboptimal service execution.

Regarding the employment of the distributed execution approach, let us assume the sequential service execution as the next choice, where multiple resources are sequentially involved on the service execution. For example, the resource allocation can make use of distinct data sources, such as sensors or datasets, where each source is selected according to the immediately preceding processing outcome. The employment of MINMAX heuristics (as shown in Figure 9.1) may be used in some cases, where the sequential allocation may be accomplished by the allocation of the minimum set of resources that may deliver an efficient service execution, diminishing network communication overhead. As an example of possible implementation requiring sequential service placement, let us assume the deployment of cameras in distributed areas of a smart city, where an object location service is implemented, while a specific car moving through an urban area needs to be traced in real-time. In such service, the employment of edge resources for image processing can avoid the need of uploading each picture to a dedicated server at cloud. Furthermore, the employment of resources at the edge of the network will also enable the low delay selection of the region where further data should be collected, for instance, according to the direction where the target car is moving. Therefore, the strategy employed for distributed sequential processing needs to consider an efficient allocation of processing resources able to process locally raw data close to each data source, reducing network load and delay. Since the minimization of network load can be further achieved by the reduction on the communication among processing resources, an important challenge of this strategy is to assess the tradeoff between proximity to data sources and amount of employed edge resources.

In addition, Figure 9.1 further depicts the distributed parallel service execution, which is executed according to the atomized service execution graph, aiming at improving the overall service performance. The rationale behind this approach is to benefit from both simultaneous obtaining of data from distinct sources, and simultaneous processing of large raw data that may be split into nondependent data slices. Notice that, both parallel and sequential approaches selection are highly dependent on the service to be executed. Indeed, services presenting data level parallelism (DLP), for instance, may rely on the processing capacity of upcoming edge devices; however, notice that, an important requirement for the efficient parallel execution of such atomic services is the minimum interdependence among them –i.e., no distributed shared memory (DSM) and minimum control messages among allocated resources– due to the unreliability and high latency offered by the transmission medium.

It is worth noting that, the employment of distributed parallel placement and execution strategies opens up new issues and challenges. For instance, the processing latency for distinct atomic services may present a high difference from each other due to many factors, such as the heterogeneous type of tasks that may compose a service, amount of data processed by each atomic service, or allocated resources differences, among others. Therefore, let us consider a service whose completion depends on the outcome generated by a set of atomic services, possibly running in parallel, with distinct execution times. Assuming that the F2C control can estimate the execution latency for each atomic service, it might handle the allocation of two or more lower time consuming services at one single resource, enforcing their sequential execution in the same time interval of one time consuming service allocated to be executed in parallel in another resource. Moreover, strategies for message passing reduction employed in other distributed scenarios may further be evaluated and tailored to the F2C architecture, as well as the tradeoff between network load and QoS obtained from distributed parallel execution strategies.

9.2.Multidimensional control

According to what have been presented in Chapter 5 and Chapter 7, F2C layers distribution may be performed according to distinct characteristics or available resources, such as processing capacity, energy availability, mobility profile, and communication technologies, just to name a few. Moreover, the 3-layered F2C topology shown in Figure 4.3 considers the mobility and proximity to end-user as the main attributes for layers definition. Hence, the lower layer (fog layer 1) is mainly composed of mobile resources, providing low network latency at the cost of a higher disruption probability, whilst the upper fog layer (fog layer 2) embraces permanently or temporarily static resources whose network latency may be higher than lower layer resources, albeit being considerably lower than the communication delay offered by cloud resources, which compose the upper layer (cloud layer) offering high reliability and processing capacity at the cost of a higher network latency.

Albeit the F2C topology may, in general, follow the characteristics showed in Figure 4.3, distinct services may present distinct requirements. Therefore, the virtual topology for the service execution may assume distinct configurations according to the service requirements and the services provisioned by each one of the edge resources. We believe that one logically centralized controller, containing all control data demanded by future IoT applications may not be able to guarantee near-optimal resource selection and service orchestration. Therefore, distinct control topologies, simultaneously deployed as a multidimensional architecture, may be able to optimize the management of heterogeneous services requirements and resources offers.

It is worth mentioning that, albeit some works, such as [15] and [100], have addressed scalability concerns in conventional SDN networks, conventional SDN control plane is

responsible for the communication among distinct networks. Therefore, the information regarding each end-point of a network is usually not taken into account when setting a path between networks and, consequently, information kept by controllers regarding resources is limited to conventional network data, such as network addresses, interfaces, and costs, among others. On the other hand, the F2C architecture envisions the management of resources enabling the selection of the most suitable ones according to several service requirements. Novel services shall be deployed leveraging both fog computing and service-oriented communication model, taking advantage of new sets of available edge resources – offering, for instance, processing and storage capacities, as well as network features, such as bandwidth–, enabling both the execution of network services and the offloading of a large range of services deployed on end-user devices.

However, the successful mapping of requested services into the most suitable resources may be prohibitive when performed exclusively through network information traditionally used such as network communication technologies, available network interfaces, bandwidth, among others. In addition to that, the mapping shall be performed taking into account characteristics inherent to each resource type, such as: (1) Processors architecture, clock rate, number of shared cores, cache size; (2) Storage type, capacity, read/write velocity; and (3) Sensor/actuator type, data/action produced, as well as the inherent characteristics of each type, such as range, resolution, sensitivity, just to name a few.

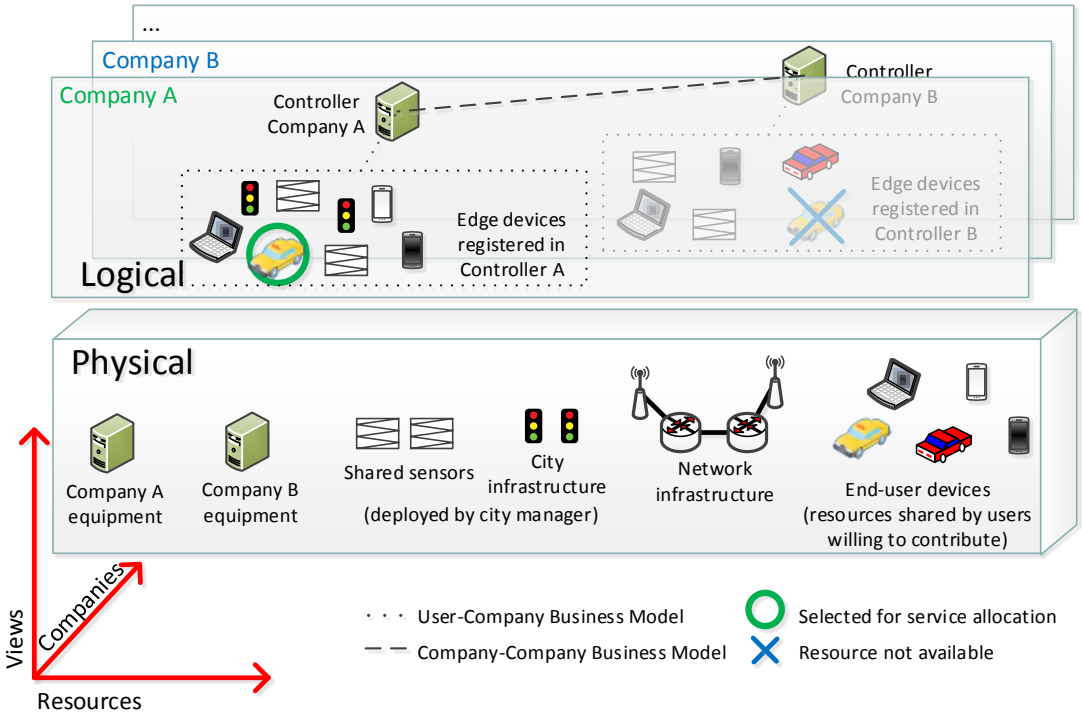
Moreover, authors in [100] assess distinct control topologies in an SDN environment, such as centralized, decentralized and hierarchical topologies with different number of layers. Since each assessed topology presents pros and cons regarding distinct aspects, such as simplicity, scalability, cost, response time, and manageability, the envisioned multidimensional control architecture leverage distinct topologies, according to the requirements of diverse services, rather than using one single control topology for the management of myriads of heterogeneous edge resources, including their respective characteristics as previously mentioned, in order to support the demands of diverse services enabled by IoT and fog computing. The deployment of distinct control plane topologies (or control plane instances) may result in the creation of specific resources lookup tables, containing information about resources able to support distinct service requirements. For instance, whilst one control instance is responsible for managing resources providing high performance computing, one distinct instance may manage resources offering green computing capacity. The optimal deployment of distinct control instances is a challenge requiring several approaches to be considered.

It is worth mentioning that each resource may be interesting for more than one control instance, making the relationship between control and resource not exclusive. Moreover, each control instance may be composed by distinct controllers organized into distinct topologies aiming at the optimal management of available resources. Therefore, each device playing the controller role must create and keep an updated view of underlying resources, including their relevant characteristics for service/resource matching. On the other hand,

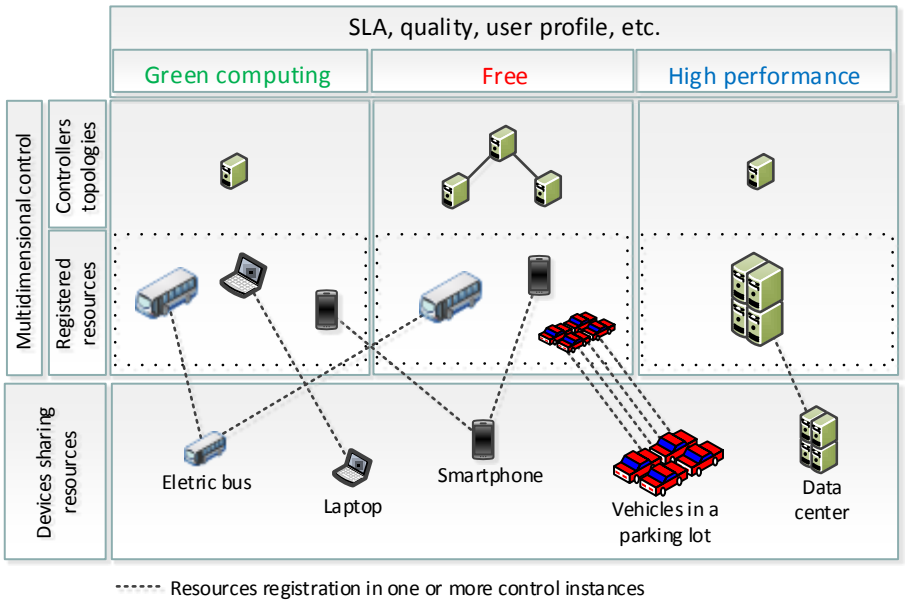
devices not playing a controller role shall collect and provide to controllers local information regarding the status of their respective resources, such as energy, compute and storage resources availability, among others. The type of information provided by non-controllers devices may also consider the category of services managed by each controller. In order to illustrate the potential benefits of the multidimensional control architecture from distinct perspectives, let us consider three distinct scenarios described as follows.

1. **Distinct companies employing shared resources:** Let us consider a smart city providing deployed IoT infrastructure, so that SPs may rely on them, rather than deploying their own edge resources. In addition, consider that the available resources may include both the ones deployed by city manager and idle resources shared by users willing to collaborate. Therefore, the infrastructure required to be deployed by a SP for leveraging the available edge resources and executing the offered service would be limited to the controller in charge of managing the resources used by the offered service. It is worth noting that the usage of shared resources by distinct companies opens opportunities for new business models, both between distinct companies and between companies and users. In this scenario, contracts established between parties must define which companies can use each edge device, resources provided, priorities, prices, and policies, among others. As illustrated in Figure 9.2(a), distinct control plane instances shall be deployed and, whilst several physical resources are shared by distinct SP, some of them are only available for one of them. This is defined through previously established contracts. On the other hand, notice that the yellow cab is initially making its resources available for both providers, however, the complete allocation by Company A (see Figure 9.2(a)), results on the unavailability of resources related to this device to Company B. This situation demands the assessment of distinct strategies to keep an updated view of the available resources to both SPs, considering scalability, required accuracy, business models, overhead, etc. Furthermore, updating strategies must consider that distinct SPs may be competitors, hindering the deployment control planes with inter communication.

2. **Distinct SLA provisioning:** Distinct from the first scenario, this one assess the benefits attained from the deployment of multiple control instances by one single SP aiming at the resource provisioning according to distinct SLAs. Notice that the provisioning of distinct SLAs shall require distinct information regarding edge resources in order to meet distinct service requirements. We advocate that the deployment of one single control instance for the management of myriads of edge resources in a specific area turns to be unfeasible, since it would demand the deployment of several controllers, resulting in a communication overhead in order to keep a synchronized view of underlying resources. On the other hand, the amount of resources managed by controllers may be considerably diminished by the deployment of distinct control instances, each one offering distinct QoE, according to available resources under predefined SLAs. For instance, distinct control instances shall provide distinct service



(a)



(b)

Figure 9.2: Multidimensional control architecture.

requirements, such as green computing, high security communication, high performance computing, and free usage, by employing the most suitable resources, as shown side by side in Figure 9.2(b), for sake of simplicity. It is worth noting that controllers may assume distinct topologies for each SLA according to a number of factors, such as amount of underlying resources, their categories, and capacities, among others. Moreover, although some resources may be managed by more than one controller, the synchronization of resources lookup table among distinct controllers may result in a lower overhead, since those tables are considerably reduced. The synchronization in this scenario also benefits from the fact that there is no competition among distinct control planes and inter-controller communication is not an issue, unlike the first scenario where distinct SPs are responsible for each control plane instance.

3. **Control as a Service (CaaS):** As introduced in [102], and described in details in Chapter 7, the CaaS concept aims at the deployment of controllers on-demand in highly dynamic scenario requiring low resource selection delay. This is achieved through the employment of shared resources at the edge of the network as controllers, leveraging fog computing concepts to make control decisions to be taken closer to end-user. However, the constrained capacity inherent of edge devices is a considerable limitation on the number of distinct services that can be managed by these edge controllers simultaneously, further limiting the type of resources managed by each controller. As a consequence, resource and service categorization strategies must be defined in order to create specific resource databases compliant to distinct service requirements, while enabling edge resources with low processing and storage capacity to play the controller role. Consequently, the resource selection and provisioning may be performed with a reduced latency, once the categorization is done, due to reduced resources database size [102]. In such scenario, strategies for the selection of the most suitable resources to play the controller role must be used for each control instance, taking into account the most important characteristics required by the service provided by such instance. It is worth noting that, as a number of devices may be logically included in more than one control instances, a device allocated as controller in one instance may or may not be employed as controller in other instances. As a consequence, on single device may have part of its shared resources employed as a controller in one control instance while, simultaneously, spare resources are used for execution of services managed by other control instances.

It must be remarked that designing such multidimensional control architecture shall arise several challenges for its successful deployment. These challenges are described in the following in order to provide some opportunities for future research in this subject.

- The scalability assessment is a key requirement in such multidimensional architecture, since shared resources may be managed by controllers in distinct control instances, demanding strategies able to provide an up-to-date view of

available resources by controllers in distinct dimensions whilst reducing signaling overhead

- The definition of the optimal control plane topology for each control instance shall take into account different characteristics according to the respective service managed by each instance. For instance, centralized, distributed and hierarchical control topologies present advantages and disadvantages and may be employed for distinct dimensions according to service needs and resources characteristics [100].

- The optimal selection of controllers in highly dynamic scenarios requires runtime strategies taking into consideration several aspects of candidates. Furthermore, the weight of each considered aspect can be modified according to several factors, such as provided service, control topology, layer where the controller is expected to be deployed, etc. An extra challenge regarding the coordination of controllers is observed when considering both intra and inter-control instances coordination.

- Since distinct services are provided by distinct control instances, strategies must be assessed in order to define how potential service clients can discover the controllers responsible for the provisioning of the required service. Typically used strategies, such as deployment of brokers, although effective, may have negative impacts, such as extra delay, which can be unacceptable for services with strict requirements.

- The collaborative model employed by F2C requires the creation of new business models to enable interested parties to get along with each other. This will include not only the relationship between SPs and service clients, but also among distinct SPs. For the former, SLA between each pair client-provider must comprise distinct aspects, such as expected QoS and user preferences regarding shared resources, schedules, privacy, and other preferences that shall be available in user profile. For the latter, besides directives for sharing their own private resources, SLA must include rules for data sharing while respecting clients' preferences, such as privacy. In such collaborative model, the creation of business models is a key requirement.

- The management of resources shared by devices requires the implementation of either an application to be installed on each device or a functionality on operating system built on devices with higher simplicity, such as some sensors.

- Conventional SDN design is mainly focused on controlling the communication among distinct networks by offering interfaces to control programmable switches (or forwarding devices). Therefore, information regarding devices at the edge of the network, such as services offered by them –usually used for

service-oriented resource selection–, are not kept by conventional SDN controllers, since the main goal is to establish connection between networks, rather than end-to-end connection. In addition, the amount of information regarding the edge resources whose controllers should keep, in a future service-oriented IoT scenario, tends to increase according to the amount of services offered by these resources.

- The deployment of such collaborative model arises several security concerns, which must be addressed in order to allow its adoption. Some of the main concerns include, but are not limited to, authentication, identity management, access control, integrity, privacy, and availability.

Finally, besides the challenges arisen from the deployment of such multidimensional architecture, several challenges are still not addressed in combined F2C computing. Therefore, resource discovery and monitoring, service allocation and efficient orchestration, and mapping services requirements into the most suitable resources are just some of the line of work that still deserve more efforts in order to enable the successful deployment of this novel architecture.

Bibliography

[1] L. Atzori, A. Iera and G. Morabito, "The Internet of Things: A survey," *Computer Networks,* pp. 2787-2805, 2010.

[2] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communication Surveys & Tutorials,* vol. 17, no. 4, pp. 2347-2376, Fourthquarter 2015.

[3] R. v. d. Meulen, "Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016," Gartner, 7 2 2017. [Online]. Available: www.gartner.com/newsroom/id/3598917. [Accessed 22 08 2017].

[4] J. Gubbi, R. Buyya, S. Marusic and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems,* vol. 29, pp. 1645-1660, 2013.

[5] B. Dorsemaine, J. P. Gaulier, J. P. Wary, N. Kheir and P. Urien, "Internet of Things: A Definition & Taxonomy," in *2015 9th International Conference on Next Generation Mobile Applications, Services and Technologies*, Cambridge, 2015.

[6] D. Miorandi, S. Sicari, F. D. Pellegrini and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks,* pp. 1497-1516, 2012.

[7] P. Suresh, J. V. Daniel, V.Parthasarathy and R. Aswathy, "A state of the art review on the Internet of Things (IoT) History, Technology and fields of deployment," in *2014 International Conference on Science Engineering and Management Research (ICSEMR)*, Chennai, 2014.

[8] M. Baldauf, S. Dustdar and F. Rosenberg, "A survey on context-aware systems," *International Journal of Ad Hoc and Ubiquitous Computing,* vol. 2, no. 4, pp. 263-277, 2007.

[9] K. B. Laskey and K. Laskey, "Service oriented architecture," *Wiley Interdisciplinary Reviews: Computational Statistics,* vol. 1, no. 1, pp. 101-105, 2009.

[10] W. Ramirez, V. B. Souza, E. Marin-Tordera and S. Sanchez, "Exploring potential implementations of PCE in IoT world," *Optical Switching and Networking,* vol. 26, pp. 48-59, November 2017.

[11] N. Abid, P. Bertin and J.-M. Bonnin, "A comparative cost analysis of identifier/locator split approaches," in *2014 IEEE Wireless Communications and Networking Conference (WCNC)*, Istambul, 2014.

[12] Y. Wu, J. Tuononen and M. Latvala, "An Analytical Model for DNS Performance with TTL Value 0 in Mobile Internet," in *TENCON 2006 - 2006 IEEE Region 10 Conference*, Hong Kong, 2006.

[13] Q. Song, I. Habib and W. Alanqar, "On the performance evaluation of distributed dynamic routing in GMPLS optical networks," in *GLOBECOM '05. IEEE Global Telecommunications Conference, 2005*, St. Louis, MO, 2005.

[14] W. Ramirez, X. Masip-Bruin, E. Marín-Tordera, M. Yannuzzi, A. Martinez, S. Sánchez-López and V. López, "An Hybrid Prediction-based Routing approach for reducing routing inaccuracy in optical transport networks," in *2014 19th European Conference on Networks and Optical Communications - (NOC)*, Milano, 2014.

[15] S. H. Yeganeh, A. Tootoonchian and Y. Ganjali, "On Scalability of Software-Defined Networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 136-141, February 2013.

[16] A. Farrel, J.-P. Vasseur and J. Ash, "A path computation element (PCE)-based architecture," RFC4655, 2006.

[17] F. Bonomi, R. Milito, J. Zhu and S. Addepalli, "Fog Computing and Its Role in the Internet of Things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, Helsinki, 2012.

[18] 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, October 2016.

[19] R. Muñoz, R. Casellas, R. Martínez and R. Vilalta, "PCE: What is It, How Does It Work and What are Its Limitations?," *Journal of Lightwave Technology*, vol. 32, no. 4, pp. 528-543, 15 February 2014.

[20] R. Casellas, R. Munoz and R. Martínez, "Path Computation Elements (PCE): Applications and Status," in *Optical Fiber Communication Conference/National Fiber Optic Engineers Conference 2013*, Anaheim, 2013.

[21] J. Vasseur and J. L. Le Roux, "Path Computation Element (PCE) Communication Protocol (PCEP)," RFC5440, 2009.

[22] J. P. Vasseur, R. Zhang, N. Bitar and J. L. Le Roux, "A Backward-Recursive PCE-Based Computation (BRPC) Procedure to Compute Shortest Constrained Inter-Domain Traffic Engineering Label Switched Paths," RFC 5441, 2009.

[23] A. Giorgetti, F. Cugini, N. Sambo, F. Paolucci, N. Andriolli and P. Castoldi, "Path state-based update of PCE traffic engineering database in wavelength switched optical networks," *IEEE Communications Letters*, vol. 14, no. 6, pp. 575-577, June 2010.

[24] R. Martínez, A. Castro, R. Casellas, R. Muñoz, L. Velasco, R. Vilalta and J. Comellas, "Experimental validation of dynamic restoration in GMPLS-controlled multi-layer networks using PCE-based global concurrent optimization," in *2013 Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference (OFC/NFOEC)*, Anaheim, CA, 2013.

[25] E. Crabbe, J. Medved, I. Minei and R. Varga, "PCEP extensions for stateful PCE," IETF

Draft-IETF-PCE-Stateful-PCE-09 (work in progress), 2014.

[26] E. Crabbe, S. Sivabalan, I. Minei and R. Varga, "PCEP Extensions for PCE-initiated LSP Setup in a Stateful PCE Model," IETF Draft-Crabbe-PCE-PCE-Initiated-LSP (work in progress), 2013.

[27] P. Martinez-Julia and A. F. Skarmeta, "Beyond the separation of identifier and locator: Building an identity-based overlay network architecture for the Future Internet," *Computer Networks*, vol. 57, no. 10, pp. 2280-2300, 5 July 2013.

[28] W. Ramirez, X. Masip-Bruin, M. Yannuzzi, R. Serral-Gracia, A. Martinez and M. S. Siddiqui, "A survey and taxonomy of ID/Locator Split Architectures," *Computer Networks*, vol. 60, pp. 13-33, 26 February 2014.

[29] V. B. C. Souza, X. Masip-Bruin, E. Marin-Tordera, W. Ramírez and S. Sánchez-López, "Towards the scalability of a Service-oriented PCE," in *2015 20th European Conference on Networks and Optical Communications - (NOC)*, London, 2015.

[30] N. Fernando, S. W. Loke and W. Rahayu, "Mobile cloud computing: A survey," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84-106, January 2013.

[31] I. Tomkos, S. Azodolmolky, J. Solé-Pareta, D. Careglio and E. Palkopoulou, "A tutorial on the flexible optical networking paradigm: State of the art, trends, and research challenges," *Proceedings of the IEEE*, vol. 102, no. 9, pp. 1317-1337, September 2014.

[32] D. G. Costa and L. A. Guedes, "The Coverage Problem in Video-Based Wireless Sensor Networks: A Survey," *Sensors*, vol. 10, no. 9, pp. 8215-8247, 2010.

[33] U. Lipowezky and I. Vol, "Indoor-outdoor detector for mobile phone cameras using gentle boosting," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, San Francisco, 2010.

[34] B. Girod, V. Chandrasekhar, D. M. Chen, N.-M. Cheung, R. Grzeszczuk, Y. Reznik, G. Takacs, S. S. Tsai and R. Vedantham, "Mobile Visual Search," *IEEE Signal Processing Magazine*, vol. 28, no. 4, pp. 61-76, July 2011.

[35] A. Kansal and F. Zhao, "Location and mobility in a sensor network of mobile phones," in *Proc. ACM SIGMM 17th Int'l Workshop on Network and Operating Systems Support for Digital Audio & Video*, 2007.

[36] S. N. Srirama, M. Jarke and W. Prinz, "Mobile web service provisioning," in *Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services (AICT-ICIW'06)*, Guadelope, 2006.

[37] R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu and W. Hu, "Ear-phone: an end-to-end participatory urban noise mapping system," in *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, Stockholm, 2010.

[38] A. Balasubramanian, R. Mahajan, A. Venkataramani, B. N. Levine and J. Zahorjan, "Interactive wifi connectivity for moving vehicles," in *SIGCOMM '08 Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, Seattle, 2008.

[39] L. Li and L. Sun, "Seer: trend-prediction-based geographic message forwarding in sparse vehicular networks," in *2010 IEEE International Conference on Communications*, Cape

Town, 2010.

[40] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Transactions on Networking (TON)*, vol. 11, no. 1, pp. 17-32, February 2003.

[41] G. Urdaneta, G. Pierre and M. V. Steen, "A survey of DHT security techniques," *ACM Computing Surveys (CSUR)*, vol. 43, no. 2, art.8, January 2011.

[42] A. Martinez, X. Masip-Bruin, W. Ramirez, R. Serral-Gracià, E. Marin-Tordera and M. Yannuzzi, "Toward a new addressing scheme for a service-centric Internet," in *2012 IEEE International Conference on Communications (ICC)*, Ottawa, ON, 2012.

[43] S. Deb, A. Srinivasan and S. K. Pavan, "An improved DNS server selection algorithm for faster lookups," in *COMSWARE 2008. 3rd International Conference on Communication Systems Software and Middleware and Workshops, 2008*, Bangalore, 2008.

[44] J. Jung, E. Sit, H. Balakrishnan and R. Morris, "DNS Performance and the Effectiveness of Caching," *IEEE/ACM Transactions on Networking*, vol. 10, no. 5, pp. 589-603, October 2002.

[45] Y. Wu, J. Tuononen and M. Latvala, "An analytical model for dns performance with TTL value 0 in mobile internet," in *TENCON 2006 - 2006 IEEE Region 10 Conference*, Hong Kong, 2006.

[46] A. Botta, W. d. Donato, V. Persico and A. Pescapé, "On the Integration of Cloud Computing and Internet of Things," in *2014 International Conference on Future Internet of Things and Cloud*, Barcelona, 2014.

[47] C. Doukas and I. Maglogiannis, "Bringing IoT and Cloud Computing towards Pervasive Healthcare," in *2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, Palermo, 2012.

[48] W. He, G. Yan and L. D. Xu, "Developing Vehicular Data Cloud Services in the IoT Environment," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1587-1595, May 2014.

[49] A. Beloglazov, J. Abawajy and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755-768, May 2012.

[50] A. Wolke, B. Tsend-Ayush, C. Pfeiffer and M. Bichler, "More than bin packing: On dynamic resource allocation strategies," *Information Systems*, vol. 52, pp. 83-95, August–September 2015.

[51] Z. Xiao, W. Song and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1107-1117, June 2013.

[52] J. Buysse, M. D. Leenheer, L. M. Contreras, J. I. Aznar, J. R. Martinez, G. Landi and C. Develder, "NCP+: An integrated network and IT control plane for cloud computing," *Optical Switching and Networking*, vol. 11, no. B, pp. 137-152, January 2014.

[53] 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. Studies in Computational Intelligence, vol. 546, Cham, Springer, 2014, pp. 169-186.

[54] OpenFog Consortium Architecture Working Group, "OpenFog Reference Architecture for Fog Computing," 2017.

[55] V. B. C. Souza, W. Ramírez, X. Masip-Bruin, E. Marín-Tordera, G. Ren and G. Tashakor, "Handling service allocation in combined Fog-cloud scenarios," in *2016 IEEE International Conference on Communications (ICC)*, Kuala Lumpur, Malaysia, 2016.

[56] "H2020 EU mF2C Project," [Online]. Available: http://www.mf2c-project.eu/. [Accessed 25 09 2017].

[57] M. Aazam and E.-N. Huh, "Dynamic resource provisioning through Fog micro datacenter," in *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, St. Louis, MO, 2015.

[58] A. Nahir, A. Orda and D. Raz, "Resource allocation and management in Cloud Computing," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, Ottawa, ON, 2015.

[59] P. Donadio, G. B. Fioccola, R. Canonico and G. Ventre, "Network security for Hybrid Cloud," in *2014 Euro Med Telco Conference (EMTC)*, Naples, 2014.

[60] S. M. Parikh, "A survey on cloud computing resource allocation techniques," in *2013 Nirma University International Conference on Engineering (NUiCONE)*, Ahmedabad, 2013.

[61] W. Lin, B. Peng, C. Liang and B. Liu, "Novel Resource Allocation Model and Algorithms for Cloud Computing," in *2013 Fourth International Conference on Emerging Intelligent Data and Web Technologies* , Xi'an, 2013.

[62] J. Dai, B. Hu, L. Zhu, H. Han and J. Liu, "Research on dynamic resource allocation with cooperation strategy in cloud computing," in *2012 3rd International Conference on System Science, Engineering Design and Manufacturing Informatization*, Chengdu, 2012.

[63] O. Rogers and D. Cliff, "A financial brokerage model for cloud computing," *Journal of Cloud Computing: Advances, Systems and Applications*, 2012.

[64] R. Kaewpuang, D. Niyato, P. Wang and E. Hossain, "A Framework for Cooperative Resource Management in Mobile Cloud Computing," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 12, pp. 2685-2700, December 2013.

[65] P. Simoens, L. V. Herzeele, F. Vandeputte and L. Vermoesen, "Challenges for orchestration and instance selection of composite services in distributed edge clouds," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, Ottawa, ON, 2015.

[66] G. Tanganelli, C. Vallati and E. Mingozzi, "Energy-efficient QoS-aware service allocation for the cloud of things," in *2014 IEEE 6th International Conference on Cloud Computing Technology and Science*, Singapore, 2014.

[67] A. Mukherjee, H. S. Paul, S. Dey and A. Banerjee, "ANGELS for distributed analytics in IoT," in *2014 IEEE World Forum on Internet of Things (WF-IoT)*, Seoul, 2014.

[68] R. Deng, R. Lu, C. Lai and T. H. Luan, "Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing," in *2015 IEEE International Conference on Communications (ICC)*, London, 2015.

[69] Y.-W. Kwon and E. Tilevich, "Energy-Efficient and Fault-Tolerant Distributed Mobile Execution," in *2012 IEEE 32nd International Conference on Distributed Computing Systems*, Macau, 2012.

[70] R. K. Balan, D. Gergle, M. Satyanarayanan and J. Herbsleb, "Simplifying cyber foraging for mobile devices," in *MobiSys '07 Proceedings of the 5th international conference on Mobile systems, applications and services*, San Juan, Puerto Rico, 2007.

[71] V. B. Souza, X. Masip-Bruin, E. Marin-Tordera, W. Ramirez and S. Sanchez, "Towards Distributed Service Allocation in Fog-to-Cloud (F2C) Scenarios," in *2016 IEEE Global Communications Conference (GLOBECOM)*, Washington, DC, 2016.

[72] W. Ramirez, X. Masip-Bruin, E. Marin-Tordera, V. Souza, A. Jukan, G.-J. Ren and O. G. d. Dios, "Evaluating the Benefits of Combined and Continuous Fog-to-Cloud Architectures," *Computer Communications*, vol. 113, pp. 43-52, 15 November 2017.

[73] N. Meghanathan, D. Nagamalai and N. Chaki, "Advances in Computing and Information Technology," in *Proceedings of the Second International Conference on Advances in Computing and Information Technology (ACITY)*, Chennai, India, 2012.

[74] 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, 1 September 2017.

[75] "Optimization With PuLP," [Online]. Available: http://www.coin-or.org/PuLP. [Accessed 2 10 2017].

[76] "Gurobi Optimization," [Online]. Available: http://www.gurobi.com. [Accessed 2 10 2017].

[77] S. Kosta, A. Aucinas, P. Hui, R. Mortier and X. Zhang, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *2012 Proceedings IEEE INFOCOM*, Orlando, FL, 2012.

[78] H. Li and S. Thompson, "Safe Concurrency Introduction through Slicing," in *PEPM '15 Proceedings of the 2015 Workshop on Partial Evaluation and Program Manipulation*, Mumbai, India, 2015.

[79] M. Alpuente, D. Ballis, F. Frechina and D. Romero, "Using conditional trace slicing for improving Maude programs," *Science of Computer Programming*, vol. 80 part B, pp. 385-415, 1 February 2014.

[80] R. M. Arasanal and D. U. Rumani, "Improving MapReduce Performance through Complexity and Performance Based Data Placement in Heterogeneous Hadoop Clusters," in *Distributed Computing and Internet Technology. ICDCIT 2013. Lecture Notes in Computer Science*, vol. 7753, Berlin, Heidelberg, Springer, 2013, pp. 115-125.

[81] V. Valancius, N. Laoutaris, L. Massoulié, C. Diot and P. Rodriguez, "Greening the internet with nano data centers," in *CoNEXT '09 Proceedings of the 5th international conference on Emerging networking experiments and technologies*, Rome, Italy, 2009.

[82] "Omnetpp simulator," [Online]. Available: https://omnetpp.org/.

[83] A. R. Curtis, W. Kim and P. Yalagandula, "Mahout: Low overhead datacenter traffic management using end host based elephant detection," in *2011 Proceedings IEEE INFOCOM*, Shanghai, 2011.

[84] F. Bai, N. Sadagopan and A. Helmy, "IMPORTANT: a framework to systematically analyze the Impact of Mobility on Performance of Routing Protocols for Adhoc Networks," in *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies*, San Francisco, CA, 2003.

[85] M. Caliskan, A. Barthels, B. Scheuermann and M. Mauve, "Predicting Parking Lot Occupancy in Vehicular Ad Hoc Networks," in *2007 IEEE 65th Vehicular Technology Conference - VTC2007*, Dublin, 2007.

[86] M. Naldi, "The availability of cloud-based services: Is it living up to its promise?," in *2013 9th International Conference on the Design of Reliable Communication Networks (DRCN),*, Budapest, 2013.

[87] L. Heilig and S. Voß, "A Scientometric Analysis of Cloud Computing Literature," *IEEE Transactions on Cloud Computing*, vol. 2, no. 3, pp. 266-278, 2014.

[88] J. K. Liu, K. Liang, W. Susilo, J. Liu and Y. Xiang, "Two-Factor Data Security Protection Mechanism for Cloud Storage System," *IEEE Transactions on Computers*, vol. 65, no. 6, pp. 1992-2004, 1 June 2016.

[89] Y. Li, W. Dai, Z. Ming and M. Qiu, "Privacy Protection for Preventing Data Over-Collection in Smart City," *IEEE Transactions on Computers*, vol. 65, no. 5, pp. 1339-1350, 1 May 2016.

[90] S. J. Stolfo, M. B. Salem and A. D. Keromytis, "Fog Computing: Mitigating Insider Data Theft Attacks in the Cloud," in *2012 IEEE Symposium on Security and Privacy Workshops*, San Francisco, CA, 2012.

[91] I. B. B. Harter, D. A. Schupke, M. Hoffmann and G. Carle, "Network virtualization for disaster resilience of cloud services," *IEEE Communications Magazine*, vol. 52, no. 12, pp. 88-95, December 2014.

[92] C. Natalino, P. Monti, L. França, M. Furdek, L. Wosinska, C. R. Francês and J. W. Costa, "Dimensioning optical clouds with shared-path shared-computing (SPSC) protection," in *2015 IEEE 16th International Conference on High Performance Switching and Routing (HPSR)*, Budapest, 2015.

[93] D. Satria, D. Park and M. Jo, "Recovery for overloaded mobile edge computing," *Future Generation Computer Systems*, vol. 70, pp. 138-147, May 2017.

[94] I. F. Akyildiz, A. Lee, P. Wang, M. Luo and W. Chou, "Research challenges for traffic engineering in software defined networks," *IEEE Network*, vol. 30, no. 3, pp. 52-58, May-June 2016.

[95] M. Younis, I. F. Senturk, K. Akkaya, S. Lee and F. Senel, "Topology management techniques for tolerating node failures in wireless sensor networks: A survey," *Computer Networks*, vol. 58, pp. 254-283, 15 January 2014.

[96] A. Sgambelluri, A. Giorgetti, F. Cugini, F. Paolucci and P. Castoldi, "OpenFlow-Based Segment Protection in Ethernet Networks," *Journal of Optical Communications and Networking*, vol. 5, no. 9, pp. 1066-1075, 2013.

[97] V. B. Souza, X. Masip-Bruin, E. Marín-Tordera, W. Ramírez and S. Sánchez-López, "Proactive vs reactive failure recovery assessment in combined Fog-to-Cloud (F2C) systems," in *2017 IEEE 22nd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, Lund, Sweden, 2017.

[98] V. B. Souza, W. Ramírez, X. Masip-Bruin, E. Marín-Tordera, S. Sánchez-López and G.-J. Ren, "Towards Service Protection in Fog-to-Cloud (F2C) Computing Systems," in *2017 Future Technologies Conference (FTC)*, Vancouver, Canada, 2017.

[99] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov and R. Smeliansky, "Advanced study of SDN/OpenFlow controllers," in *CEE-SECR '13 Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia*, Moscow, 2013.

[100] J. Hu, C. Lin, X. Li and J. Huang, "Scalability of control planes for Software defined networks: Modeling and evaluation," in *2014 IEEE 22nd International Symposium of Quality of Service (IWQoS)*, Hong Kong, 2014.

[101] M. Aazam, M. St-Hilaire, C.-H. Lung and I. Lambadaris, "MeFoRE: QoE based resource estimation at Fog to enhance QoS in IoT," in *2016 23rd International Conference on Telecommunications (ICT)*, Thessaloniki, 2016.

[102] V. B. Souza, A. Gómez, X. Masip-Bruin, E. Marín-Tordera and J. Garcia, "Towards a Fog-to-Cloud control topology for QoS-aware end-to-end communication," in *2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*, Vilanova i la Geltru, Espanha, 2017.

[103] "Zenmap webpage," [Online]. Available: https://nmap.org/zenmap/. [Accessed March 2017].

Publications

- W. Ramirez, X. Masip-Bruin, E. Marin-Tordera, S. Sanchez-Lopez and V. Souza, "Handling outdated information in optical PCE environments," in *2015 International Conference on Optical Network Design and Modeling (ONDM)*, Pisa, 2015.
- V. B. C. Souza, X. Masip-Bruin, E. Marin-Tordera, W. Ramírez and S. Sánchez-López, "Towards the scalability of a Service-oriented PCE," in *2015 20th European Conference on Networks and Optical Communications - (NOC)*, London, 2015.
- W. Ramirez, V. B. Souza, E. Marin-Tordera and S. Sanchez, "Exploring potential implementations of PCE in IoT world," *Optical Switching and Networking*, vol. 26, pp. 48-59, November 2017.
- V. B. C. Souza, W. Ramírez, X. Masip-Bruin, E. Marín-Tordera, G. Ren and G. Tashakor, "Handling service allocation in combined Fog-cloud scenarios," in *2016 IEEE International Conference on Communications (ICC)*, Kuala Lumpur, 2016.
- V. B. Souza, X. Masip-Bruin, E. Marin-Tordera, W. Ramirez and S. Sanchez, "Towards Distributed Service Allocation in Fog-to-Cloud (F2C) Scenarios," in *2016 IEEE Global Communications Conference (GLOBECOM)*, Washington, DC, 2016.
- X. Masip-Bruin, E. Marín-Tordera, A. Gómez, V. Barbosa and A. Alonso, "Will it be cloud or will it be fog? F2C, A novel flagship computing paradigm for highly demanding services," in *2016 Future Technologies Conference (FTC)*, San Francisco, CA, 2016.
- S. Kahvazadeh, V. B. Souza, X. Masip-Bruin, E. Marín-Tordera, J. Garcia and R. Diaz, "Securing combined Fog-to-Cloud system Through SDN Approach," in *Crosscloud'17 Proceedings of the 4th Workshop on CrossCloud Infrastructures & Platforms*, Belgrade, Serbia, 2017.
- V. B. Souza, X. Masip-Bruin, E. Marín-Tordera, W. Ramírez and S. Sánchez-López, "Proactive vs reactive failure recovery assessment in combined Fog-to-Cloud (F2C) systems," in *2017 IEEE 22nd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, Lund, Sweden, 2017.
- V. B. Souza, A. Gómez, X. Masip-Bruin, E. Marín-Tordera and J. Garcia, "Towards a Fog-to-Cloud control topology for QoS-aware end-to-end communication," in *2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*, Vilanova i la Geltru, Espanha, 2017.
- V. B. Souza, W. Ramírez, X. Masip-Bruin, E. Marín-Tordera, S. Sánchez-López and G.-J. Ren, "Towards Service Protection in Fog-to-Cloud (F2C) Computing Systems," in *2017 Future Technologies Conference (FTC)*, Vancouver, Canada, 2017.
- S. Kahvazadeh, V. B. Souza, X. Masip-Bruin, E. Marín-Tordera, J. Garcia and R. Diaz, "An SDN-based Architecture for Security Provisioning in Fog-to-Cloud (F2C) Computing System," in *2016 Future Technologies Conference (FTC)*, Vancouver, Canada, 2017.

- W. Ramirez, X. Masip-Bruin, E. Marin-Tordera, V. Souza, A. Jukan, G.-J. Ren and O. G. d. Dios, "Evaluating the Benefits of Combined and Continuous Fog-to-Cloud Architectures," *Computer Communications*, vol. 113, pp. 43-52, 15 November 2017.

Ongoing Work

- V.B. Souza, X. Masip-Bruin, E. Marin-Tordera, S. Sánchez, J. Garcia, G.-J. Ren, A. Jukan and A. Juan, "Towards a Proper Service Placement in Combined Fog-to-Cloud (F2C) Architectures," *Future Generation Computer Systems*, 2018. (submitted, under review)
- V.B. Souza, X. Masip-Bruin, E. Marin-Tordera, J. Garcia, S. Sánchez, A. Jukan and G.-J. Ren, "A Multidimensional Control Architecture for Fog-to-Cloud Computing," *Wireless Communications Magazine*, 2018. (to be submiteed)