Programa de Doutoramento em Informática
das Universidades do Minho, de Aveiro e do Porto

Universidade de Aveiro
Departamento de Eletrónica,
Telecomunicações e Informática

2019


Flávio
Silva Meneses

# Gestão Virtualizada de Mobilidade para Redes Futuras Baseadas em Particionamento de Rede

# Virtualized Mobility Management for Future Slicing-based Networks

Universidade de Aveiro
2019

Departamento de Eletrónica,
Telecomunicações e Informática

**Flávio**
**Silva Meneses**

# Gestão Virtualizada de Mobilidade para Redes Futuras Baseadas em Particionamento de Rede

# Virtualized Mobility Management for Future Slicing-based Networks

Tese apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Doutor no âmbito do programa doutoral MAP-i, realizada sob a orientação científica do Doutor Daniel Nunes Corujo, Investigador Doutorado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor Rui Luís Andrade Aguiar, Professor Catedrático do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

Este doutoramento foi realizado com o apoio do Instituto de Telecomunicações - Pólo de Aveiro (UID/EEA/50008/2019), e com o apoio dos projetos CENTRO-01-0145-FEDER-000010 e PTDC/EEI-TEL/30685/2017.

**o júri / the jury**

presidente / president

Doutor Armando da Costa Duarte
Professor Catedrático, Universidade de Aveiro

vogais / examiners committee

Doutora Marília Pascoal Curado
Professora Associada com Agregação, Universidade de Coimbra

Doutor José André Rocha Sá Moura
Professor Auxiliar, Instituto Universitário de Lisboa

Doutor Jorge Miguel Matos Sousa Pinto
Professor Associado com Agregação, Departamento de Informática da Escola de Engenharia da Universidade do Minho

Doutor João Paulo Silva Barraca
Professor Auxiliar, Universidade de Aveiro

Doutor Daniel Nunes Corujo
Investigador Doutorado, Universidade de Aveiro

**agradecimentos / acknowledgements**

A tese apresentada marca o término de um ciclo, que não seria possível sem a colaboração e ajuda de várias pessoas e entidades, sendo com elas que partilho os méritos que dela possa receber. Agradeço a todos aqueles que contribuíram de forma decisiva para a concretização deste estudo.

À Universidade de Aveiro e Instituto de Telecomunicações de Aveiro manifesto o meu apreço pela possibilidade de realização do presente trabalho e por todos os meios colocados à disposição, assim como a excelência da formação prestada e conhecimentos transmitidos, ambicionando que o trabalho desenvolvido dignifique ambas as instituições.

Aos orientadores desta tese, Doutor Daniel Corujo e Professor Doutor Rui Aguiar, agradeço não só toda a disponibilidade, colaboração e conhecimentos transmitidos, mas principalmente a capacidade de estímulo e confiança ao longo de todo o trabalho. Deixo também o meu reconhecimento a todos os colegas com os quais me cruzei ao longo destes anos e que de uma forma ou de outra contribuíram para esta tese. Em especial a José Quevedo e Carlos Guimarães, agradecendo a partilha de conhecimentos, mas principalmente por tornarem o tempo de laboratório em companheirismo.

Um especial e sentido abraço com profundo reconhecimento aos meus pais. À minha esposa, Sara de Meneses, por me ter acompanhado sempre ao longo desta caminhada, com o seu ombro amigo e estabilidade emocional. Deixo também um agradecimento especial ao casal Fátima e António, por todo o apoio e amizade partilhada, e essencialmente por me acolherem como família desde o início.

Por último, a todas estas pessoas resta-me dizer: Muito obrigado.

**Palavras Chave**

Redes Definidas por Software, Virtualização de Funções de Rede, Orquestração, Particionamento de rede, Gestão de Mobilidade

**Resumo**

O aumento do número de dispositivos móveis e a forma como são utilizados para o acesso à Internet tem estado na origem de diversos desafios de rede. Estes dispositivos têm vindo a evoluir e são agora capazes de consumir e produzir fluxos de vídeo em tempo real enquanto se movem e exploram múltiplos acessos a tecnologias sem-fios. Paralelamente, operadores de rede têm apostado em mecanismos de redes definidas por *software* (do inglês, SDN) e de funções de rede virtualizadas (do inglês, NFV), de forma a possibilitar a rede a adaptar-se de uma forma dinâmica e flexível a diversos tipos de requisitos de tráfego e casos de uso. Além disso, o 3GPP projecta a próxima rede de quinta geração (5G), como uma arquitetura holística capaz de agregar diferentes tecnologias de acesso, e servir múltiplos verticais (ex., indústria automóvel e saúde) com diferentes requisitos de tráfego (ex., baixa latência e elevados picos de transmissão de dados). Neste sentido, o particionamento de rede (do inglês, *network slicing* ou *slicing*) aparece como decisivo, permitindo a divisão da infra-estrutura de rede em redes logicamente isoladas, de forma a suportar determinados tipos de serviço. No entanto, isto levanta novos desafios relacionados com a gestão de mobilidade nesta arquitetura holística de redes altamente dinâmicas e flexíveis. Alinhada com esta visão, esta tese propõe um mecanismo de gestão virtualizada de mobilidade para redes futuras baseadas em particionamento de rede, capaz de adaptar a rede (através de mecanismos de *slicing*) aos requisitos de comunicação do utilizador, enquanto possibilita o *handover* transparente entre diferentes tecnologias de acesso. Este mecanismo foi suportado pelo desenho de uma arquitetura baseada em tecnologias SDN e NFV, sendo depois evoluída de forma a acomodar mecanismos de *slicing* e orquestração para instanciação e orquestração de *network slices*. A tese implementa e avalia provas de conceito dos diferentes componentes de rede, com os resultados a mostrar a sua viabilidade e providenciando indicadores para futuros melhoramentos e implementações comerciais.

**Keywords** Software-Defined Networking, Network Function Virtualization, Orchestration, Network Slicing, Mobility Management

**Abstract** The increasing number of mobile devices and their usage for Internet access has been at the origin of new networking challenges. In fact, mobile devices have evolved and are now able to both consume and produce live stream videos, while moving and taking advantage of multiple wireless access technologies. In parallel, network operators have been relying on Software Defined Networking (SDN) and Network Function Virtualization (NFV) mechanisms for enabling the network to dynamically and flexibly adapt to the different traffic requirements and use cases. Furthermore, 3GPP envisions the upcoming fifth generation (5G) of networks, as a holistic architecture able to aggregate different access technologies, and serve multiple vertical use cases (e.g., automotive and eHealth) with different traffic requirements (e.g., low latency and high peak data rates). Here, network slicing appears as the game changer, by allowing the network infrastructure to be divided into logically-isolated networks for supporting certain service types. Nevertheless, this rises new challenges related to the mobility management on such a highly dynamic and flexible holistic architecture. Aligned with this vision, this thesis proposes a virtualized mobility management mechanism for future slice-based networks, capable of providing the means to flexibly adapt the network (through slicing mechanisms) to the user's communication requirements, while enabling seamless handovers between different access technologies. This mechanism was supported by the design of an SDN/NFV-based architecture, that was further evolved to accommodate slice and orchestration mechanisms for the dynamic instantiation and orchestration of network slices. The thesis provides proof-of-concept implementation and evaluation of the different network components, with results showing its feasibility and providing indicators for further enhancement and commercial deployment.

# Contents

# List of Figures

# List of Tables

# Acronyms

| | |
|---|---|
| **3GPP** | 3rd Generation Partnership Projec |
| **4G** | fourth generation |
| **5G** | fifth generation |
| **5G-PPP** | 5G Infrastructure Public Private Partnership |
| **AAA** | Authentication, Authorization and Accounting |
| **AF** | Application Function |
| **AKA** | Authentication and Key Agreement |
| **AMazING** | Advanced Mobile wIreless Network playGround |
| **AMF** | Access and Mobility Management Function |
| **AMQP** | Advanced Message Queuing Protocol |
| **AP** | Access Point |
| **API** | Application Programming Interface |
| **ARP** | Address Resolution Protocol |
| **AUSF** | Authentication Server Function |
| **BS** | Base Station |
| **CDF** | cumulative distribution function |
| **COE** | container orchestrator engine |
| **CN** | Correspondent Node |
| **CVNF** | containerized VNF |
| **CPE** | Customer Premises Equipment |
| **CSMF** | Communication Service Management Function |
| **DHCP** | Dynamic Host Configuration Protocol |
| **DMM** | Distributed Mobility Management |
| **DNS** | Domain Name System |
| **E2E** | end-to-end |
| **EAP** | Extensible Authentication Protocol |
| **eCDF** | empirical CDF |
| **EDCA** | Enhanced Distributed Channel Access |
| **eMBB** | enhanced Mobile Broadband |
| **eNB** | evolved Node B |
| **EPC** | Evolved Packet Core |
| **EPS** | Evolved Packet System |
| **ETSI** | European Telecommunications Standards Institute |
| **FA** | Foreign Agent |
| **GW** | Gateway |
| **GRE** | Generic Routing Encapsulation |
| **GTP** | GPRS Tunneling Protocol |
| **HA** | Home Agent |
| **HD** | High Definition |
| **HSS** | Home Subscriber Server |
| **HTTP** | Hypertext Transfer Protocol |
| **IaaS** | Infrastructure as a Service |
| **ICMP** | Internet Control Message Protocol |
| **IETF** | Internet Engineering Task Force |
| **IMSI** | International Mobile Subscriber Identity |
| **IP** | Internet Protocol |
| **IPSec** | Internet Protocol Security |
| **ISP** | Internet Service Provider |
| **IT** | Instituto de Telecomunicações |
| **KPI** | Key Performance Indicator |
| **KVM** | Kernel-based Virtual Machine |
| **LCM** | Life-cycle Management |
| **LMA** | Local Mobility Anchor |

| | |
|---|---|
| **LTE** | Long-Term Evolution |
| **LXC** | Linux Containers |
| **MAC** | Media Access Control |
| **MAG** | Mobile Access Gateway |
| **MANO** | Management and Orchestrator |
| **MAR** | Mobile Access Router |
| **MEC** | Multi-access Edge Computing |
| **MIP** | Mobile IP |
| **MME** | Mobility Management Entity |
| **mMTC** | massive Machine Type Communications |
| **MN** | Mobile Node |
| **MNO** | Mobile Network Operator |
| **N3IWF** | Non-3GPP InterWorking Function |
| **NaaS** | Network as a Service |
| **NAT** | Network Address Translation |
| **NB** | NorthBound |
| **NEF** | Network Exposure Function |
| **NF** | Network Function |
| **NFV** | Network Function Virtualisation |
| **NFVI** | NFV Infrastructure |
| **NFVO** | NFV Orchestrator |
| **NIC** | Network Interface Controller |
| **NRF** | Network Repository Function |
| **NS** | Network Service |
| **NSaaS** | Network Slice as a Service |
| **NSI** | Network Slice Instance |
| **NSMF** | Network Slice Management Function |
| **NSSF** | Network Slice Selection Function |
| **NSSI** | Network Slice Subnet Instance |
| **NSSMF** | Network Slice Subnet Management Function |
| **NSST** | Network Slice Subnet Template |
| **NST** | Network Slice Template |
| **OAI** | OpenAirInterface |
| **ODL** | OpenDaylight |
| **OF** | OpenFlow |
| **ONAP** | Open Network Automation Platform |
| **ONF** | Open Networking Foundation |
| **ONOS** | Open Network Operating System |
| **OS** | Operating System |
| **OSA** | Open System Authentication |
| **OSM** | Open Source MANO |
| **OTT** | Over-the-Top |
| **OvS** | Open vSwitch |
| **OVSDB** | Open vSwitch DataBase |
| **PaaS** | Platform as a Service |
| **PCF** | Policy Control Function |
| **pCPE** | physical CPE |
| **PCRF** | Policy and Charging Rules Function |
| **P-GW** | Packet Data Network Gateway |
| **PMIP** | Proxy Mobile IP |
| **PMIPv6** | Proxy Mobile IPv6 |
| **PNF** | Physical Network Function |
| **PoA** | Point of Attachment |
| **PoD** | Point of Deployment |
| **QoE** | Quality of Experience |
| **QoS** | Quality of Service |
| **RAN** | Radio Access Network |
| **REST** | Representational State Transfer |
| **RPC** | Remote Procedure Call |
| **RTP** | Real-time Protocol |
| **RTT** | round trip time |
| **SB** | SouthBound |
| **SBA** | Service-Based Architecture |
| **SD** | Standard Definition |
| **SDN** | Software-Defined Networking |
| **SDR** | Software-Defined Radio |
| **SFC** | Service Function Chain |
| **S-GW** | Serving Gateway |
| **SIM** | Subscriber Identification Module |
| **SLA** | Service Level Agreement |
| **SliMANO** | Slice Management and Orchestration |
| **SMF** | Session Management Function |
| **SSID** | Service Set Identification |
| **TCP** | Transmission Control Protocol |
| **TLS** | Transport Layer Security |

| | |
|---|---|
| **TOSCA** | Topology and Orchestration Specification for Cloud Applications |
| **UDP** | User Datagram Protocol |
| **UDM** | Unified Data Management |
| **UDR** | Unified Data Repository |
| **UE** | User Equipment |
| **UHD** | Ultra High Definition |
| **UPF** | User Plane Function |
| **URLLC** | Ultra Reliable Low Latency Communications |
| **vAP** | virtual AP |
| **vCPE** | virtual CPE |
| **VDU** | Virtual Deployment Unit |
| **vGW** | virtual Gateway |
| **VIM** | Virtual Infrastructure Manager |
| **vPoA** | virtual PoA |
| **VM** | Virtual Machine |
| **vMN** | virtual MN |
| **VNF** | Virtual Network Function |
| **VNFM** | VNF Manager |
| **VPN** | Virtual Private Network |
| **vUE** | virtual UE |
| **WMWG** | Wireless and Mobile Working Group |
| **WNV** | Wireless Network Virtualization |
| **WPA** | Wi-Fi Protected Access |
| **XaaS** | Anything as a Service |

CHAPTER 1

# Introduction

> "*One of the main contributing factors to growing traffic is consumer video use. One of the main solutions to meet the demands of the increasing demand for bandwidth has long been leveraging Wi-Fi networks, which enables operators to scale capacity to meet their subscribers' needs.*"
>
> — Cisco

*The 5G research community has been looking for flexible, dynamic and low latency network architectures in order to meet the growing traffic consumption and user service demands. Technologies such as Software-Defined Networking (SDN) and Network Function Virtualisation (NFV) have been promising to offer such capabilities, by facilitating the instantiation and re-instantiation of Network Functions (NFs) in different network domains. Following such characteristics, network slicing has been considered a game changer for future network architectures, allowing operators to optimise and share resources, in a era where wireless devices have surpassed wired ones in the total amount of data shared in the network, evidencing the need for developing mechanisms that not only optimise the use of network resources, but also to seamlessly handover wireless devices among heterogeneous sliced wireless networks. This chapter presents the motivations that led to the study of mobility management for future networks, and how network virtualization and softwarization can contribute for a new level of abstraction in heterogeneous environments. This is followed by a problem statement, where the research challenges are discussed along with the adopted methodology. Finally, the achievements resulted from this research are presented.*

## 1.1 Motivation

Internet interconnects people all around the world, allowing its users to share information in multiple ways (e.g., text, voice and video), enhance productivity and commerce (e.g., e-productivity and e-commerce), and socialize through social media networks and applications. Although networks began by being designed for "human-to-human" interactions, the technological evolution shaped by the users' interests and types of network traffic consumption has patched the network to support highly-diverse services. Also, initially developed over fixed access networks, the "freedom" of wireless access networks allied with its constant capabilities enhancements (e.g., bandwidth and latency) and current mobile devices capability of simultaneously connect to multiple access networks while moving, have been catching users' attention. In fact, in 2017, wired devices already accounted for less traffic than wireless and mobile devices. The use of wireless and mobile devices to access the Internet is expected to keep growing, with forecasts pointing that by 2022 this type of devices accounts for 71% of the total traffic [1]. In addition, the average usage of smarthpones for data consumption has been increasing, where mobile video already accounted for more than half of all mobile data traffic in 2017. Also, the increase of such consumptions has triggered the development of mobile offloading mechanisms from fixed networks to Wi-Fi and femtocells, with the amount of mobile offloaded data traffic actually exceeding the cellular one. Finally, in 2017, fourth generation (4G) network connections accounted for more than 70% of the mobile data, and in 2022, fifth generation (5G) connections will represent 3.4% of mobile networks, but generating 2.6 times more than an average 4G connection, resulting in 11.8% of all mobile data [2].

In parallel, requirement demands for 5G are significantly higher than in current networks. As such, 5G requirements envision an explosion of connected devices, a massive increasing of traffic volume and the increasing of data rates with lower than ever latency [3]. Notwithstanding, not all future use cases require a self-adapting and ultra-fast network. As a result, the network needs to be built in a flexible way so that speed and capacity can be allocated in logical partitions of the network (i.e., network slices) in order to meet the demands of each specific use case [4]. Also, the vertical architecture of current networks makes it difficult to scale operator networks and adapt them to the changing subscribing demands. This becomes more complicated when considering the global increase of mobile devices, which are not only increasing video traffic consumption, but also able to access it from several wireless access technologies (e.g., Long-Term Evolution (LTE) and Wi-Fi) while on the move [2][5].

In this context, in its first steps, 5G research has been building its base upon SDN and NFV, providing a more flexible network environment, and enabling the network

providers to run their infrastructure more efficiently. The adoption of these technologies "allow vertical systems to be broken into building blocks, resulting in a horizontal network architecture that can be chained together, both programmatically and virtually, to suit the services being offered and scaled" [4]. Here, SDN centralizes the network control and the remote management and control of forwarding devices through Application Programming Interface (API), supported by protocols such as OpenFlow (OF)[6] and Open vSwitch DataBase (OVSDB) [7]. NFV takes advantage of virtualization and cloud technologies for deploying NFs in generic hardware. Thus, instead of separately deploying isolated NFs (e.g., mobile tracking, authentication, etc.) in dedicated servers, they are instantiated as Virtual Machines (VMs) or containers over shared computation, storage and virtualized network substrates. Lately, the term network slice has been widely associated with 5G systems. Maintaining its key enablers as SDN and NFV technologies, network slicing promises to enable operators to run their Network as a Service (NaaS). In this line, 5G has been addressing Network Slicing to allow the operator to slice one physical network into multiple, virtual, end-to-end (E2E) networks, where each of them are logically isolated, including device, access, transport and core network and dedicated for different types of services with different characteristics and requirements [3]. Nevertheless, despite initial rollouts, the upcoming 5G architecture is still under heavy research and development by both academic and industrial efforts, in parallel with standardization efforts. Moreover, important mechanisms for fully supporting a 5G vision are still lacking, such as the clear separation of slices per use case, the isolation between them, its levels and the types of resources that should be shared between slices, or even how mobility management will be achieved are still open aspects [3].

As such, the outcome of this thesis focuses on the study of Network Slicing issues associated to mobility management operations. The thesis exploits the Network Slicing concept, through the application of SDN and NFV technologies, for mobility management in wireless environments, bringing a more flexible and intelligent mobility management dimension to the next mobile network generation framework, able to better support existing and upcoming scenarios, contributing to national and international research efforts on 5G. Nevertheless, 5G studies encompass levels of specification and enhancement, ranging from hardware and radio to software and architectural aspects. In this line, this thesis is focused in mobility management leveraged by enhanced architectural aspects, softwarization and virtualization of the network functions and devices.

## 1.2 Problem Statement

Mobile devices have been experiencing different technological evolutions, increasing their features and capabilities, enabling them to take advantage of multiple wireless access technologies (e.g., Wi-Fi, LTE, etc.) while on the move. The upcoming 5G network aims to tackle this scenario in more complex environments (e.g., holistic architecture with highly dynamic traffic), considering the amount of traffic generated and the need to optimize network resources in respect to Quality of Experience (QoE) expectations. In addition, network operators have to search for ways to increase operational efficiency, reduce CAPEX and OPEX costs, and to agilize services instantiation. In the light of this, Network Slicing aims to virtualise core building blocks of the network, enabling the creation of communication system partitions for each different use case, typically supported by SDN and NFV concepts. Also, the network slice should be provided only for the necessary traffic treatment of each use case, thus not all slices will contain the same functions. Despite initial rollouts, 5G architectures are still very incipient, particularly considering the support of mobility management procedures. In this regard, this thesis aims to develop mechanisms that enable mobility management procedures to be jointly orchestrated with slice management and control, allowing mobile users to maintain connectivity using different interfaces, in environments where slices will be short-lived and extremely dynamic. This results in a framework which envisages a highly flexible and manufacturer-independent network architecture design, simplifying deployment and allowing the control of virtualized network resources and services running on top of the infrastructure (e.g., routing, firewalls, etc.). Next, the research questions and the adopted methodology for the development of the thesis are presented.

### 1.2.1 Research Questions

The thesis explores the feasibility of mobility management procedures through the exploitation of SDN and NFV mechanisms, while focusing on network slicing architectural aspects. In this way, the thesis aims to bring answers to the following questions:

Q1. What will be the actual benefits of SDN flow-based mobility management control in 5G environments?

Q2. Which NFV architectural mechanisms should be deployed (and devised) to support the integration of SDN in wireless environments?

Q3. How will a composed SDN and NFV solution perform for wireless and mobile environments?

Q4. How would such a solution evolve into a pure distributed design, while still maintaining its cloud-based behavior?

Q5. Which would be the key architectural aspects of a SDN/NFV-based wireless/mobile enhanced control infrastructure that would be transited into a network slice architecture, and which would be the requirements and benefits involved?
Q6. How should mobility management be implemented in a Network Slice architecture and which resources should be shared between slices?
Q7. What level of isolation should mobility management have, should it be an independent building block?
Q8. How will a mobility management building block communicate with the remaining blocks of a network slice?

In this line, the thesis presents a framework whose features are improved along the chapters of the thesis for assessing the self-imposed questions, with the answers being provided in the end of chapter 3, chapter 4 and chapter 5, and summarized in chapter 6.

### 1.2.2 Methodology

The thesis work initiated with a state-of-the-art review, developing both theoretical and practical studies, where the candidate was able to gather next generation mobile networks bibliographical references, as well as to study and propose evolutions to 5G-enabling technologies, in particular applied to wireless environments. Incrementally, a selection of relevant existing solutions (e.g., SoftRAN [8], Cloud-RAN [9], CellSDN [10]) was done, in order to allow the candidate not only to become familiarized with the important academic and industrial proposals in the area, but also to exploit the mechanisms of both technologies to contribute with new experimental data and improvements to existing solutions. As such, keeping in mind that both SDN and NFV are considered the key enablers of the network slicing concept, the thesis work started by trying to find answers to questions Q1-Q4, exploring how these key enablers enhance wireless networks and devices' mobility management. In this regard, the work developed for addressing questions Q1-Q4 resulted in multiple scientific publications in book chapter, journal and conference proceedings. Such publications evaluate an SDN domain extension up to the Mobile Node (MN), as well as a qualitative comparison between the proposed approach and other existing IP-based mobility protocols (such as Mobile IP (MIP), Proxy Mobile IP (PMIP), Distributed Mobility Management (DMM) and SDN-based solutions). Also, both SDN and NFV mechanisms (i.e., OpenFlow and OpenStack, respectively) were used to enhance wireless mobility, by virtualizing not only the Access Point (AP) but also the MN (or User Equipment (UE), used interchangeably along the thesis). This study is also reflected in chapter 3 and, as mentioned above, answers to these questions will be provided in chapter 6.

The following phases focused on the progression of use cases and architectural evolutions considered from the leading 5G visionary organizations in the world (i.e.,

5G-PPP, 5G Americas, IMT-2020 and 5G Forum), directly targeting the shortcomings of future 5G slice-based approaches for the support of mobility management mechanisms. The integration of the network slice concept in a network operator presents different challenges, especially when considering the ever-growing number of connected devices and their connectivity requirements, constraints and level of isolation. Focusing in the mobile broadband use case, the candidate explored different utilisation scenarios such as real-time video and latency tolerant data, in order to enable the network to optimally support each content type with dynamic on-demand slices. In fact, in such dynamic mobility management environments, the definition of a mobility management core function and the level of core function sharing between slices is itself a challenge. The designed framework implements different core building blocks (e.g., mobility management and slice manager), while exploring the different types of isolation between network slices. In this context, the usage of both SDN and NFV mechanisms, allowed the building and configuration of the framework. Additionally, the developed mobility management block tackles the issues pointed out by the 3rd Generation Partnership Projec (3GPP) (and by 5G Infrastructure Public Private Partnership (5G-PPP) projects), such as heterogeneous access technologies and specific mobility management requirements. The framework exploits solutions for control abstraction (e.g., OpenFlow) and new control interfaces, such as a northbound API for mobility management, able to operate in highly-dynamic slicing environments. This allows the network to adapt itself, according to the service, user and network utilization needs, in terms of mobility management. Moreover, aiming towards an heterogeneous network, the framework takes into consideration different link technologies, mostly wireless technologies. At this point the candidate explored the Network Slicing mobility issues, such as intra-technology and inter-technology seamless handovers, aiming to tackle the research questions Q5-Q8, by proposing a slice-based network architecture. Thus, this study was performed as an enhancement of the previous framework, and is reflected in chapter 4.

In the end, in chapter 5, the framework adopts orchestration mechanism to fulfill requirements of the upcoming 5G networks, such as the inter-technology convergence, enabling seamless handovers among different wireless technologies (e.g., Wi-Fi and LTE) and the dynamic instantiation of network slices. As result, the thesis contributes to 5G research by developing a mobility management framework, exploiting the usage of extremely dynamic network slices. The work considers both intra-technology and inter-technology handovers (e.g., mobile networks and Wi-Fi), providing not only a mobile offloading service, but actually a convergence architecture for heterogeneous wireless networks. For this, the necessary mechanisms to allow user devices to roam among different slices were developed, as well as to simultaneously obtain services from one or more specific network slices. In this respect, the thesis studies the impact in

mobility management control and data plane procedures when mobility signaling needs to traverse and impact different slices, as well as when different radio access slices are involved. Also, the developed mechanisms were implemented in the Advanced Mobile wIreless Network playGround (AMazING) testbed and in an in-house data-center, with the framework deployment stages being shared in open-source communities (e.g., GitHub), furthering the dissemination of the PhD outcomes. Finally, the work presented in this document builds upon scientific publications authored by the candidate. The reused material is properly referenced at the end of each chapter.

## 1.3 Achievements and Contributions of the thesis

In this section the achievements and contributions of the thesis are presented in terms of research projects, open-source projects and dissemination in scientific forums and publications. In this line, the candidate participated in multiple national and international research projects both internal and external of the institution. Next, the research projects that the candidate was involved are described, presenting the work developed by the candidate.

- AMazING[1]: The AMazING is an internal project that developed an outdoor wireless testbed in the Instituto de Telecomunicações (IT) de Aveiro. The AMazING testbed enables the control and reproduction of the experiments for wireless network scenarios. The candidate played a role on the hardware and software maintenance, as well as on the current development and migration of the testbed to an NFV and SDN enabled framework.
- 5GCONFIG[2]: The 5GCONFIG (Convergent Core Architecture for Next Generation Networks) was an international project that aimed to add a degree of flexibility to 5G networks through its modularisation allowing the tailoring of the network to be oriented to vertical scenarios. Also, 5GCONFIG proposed an access-agnostic 5G core network for heterogeneous wireless access integration. Here, the candidate developed efforts on the integration of heterogeneous wireless network in a single core network, evidencing its benefits on the mobility management for cross-technology handovers (i.e., vertical handover) [11]–[13].
- SOCA[3]: The SOCA project focuses on the sensing of individuals considering their physical context enabling personalized and predictive responses. Here, the candidate developed mechanisms for wireless context acquisition for further computation in data-centres. Mechanisms for context migration and offload to/from edge data-centers were developed as well [14]–[17].

[1]AMazING: `http://amazing.atnog.av.it.pt`
[2]5GCONFIG: `https://www.5g-control-plane.eu`
[3]SOCA: `http://soca.av.it.pt`

- Mobilizador 5G[4]: The Mobilizador 5G is a national project that aims the development, integration and validation of products for 5G networks. With focus on the access network, the candidate was involved in proposals and prototypes of virtualized costumer premises equipment (vCPE) and development of network control mechanisms [18]–[20].
- 5GCONTACT[5]: The main objective of the 5GCONTACT project is the development of flexible and optimised solutions for 5G networks, using slicing mechanisms in fixed, mobile and wireless networks, for optimising the network in smart assisted living scenarios. The candidate has been contributing to this project by developing mechanisms for wireless context acquisition to be further processed and used to optimised the network, as well as mechanisms to tailor network characteristics towards vertical smart assisted living scenarios [21]–[23].
- 5Growth[6]: The 5Growth project empowers vertical industries by proposing an end-to-end solutions supported by slicing and virtualization techniques. The candidate has been developing SDN-based slicing and virtualization mechanism for enhancing mobility management in heterogeneous sliced wireless networks [24].

For a matter of convenience, Table 1.1 presents the list of contributions of the candidate in terms of scientific publications. The work progress was also disseminated through posters in national forums:

- SDN-based End-to-End Mobility Management in Wireless Environments, Proc. Conferência sobre Redes de Computadores (CRC), Évora, Portugal, Nov 2015
- SDN-based End-to-End Mobility Management, Encontro Ciência 2016, Lisbon, Portugal, Jul 2016
- 5G-VCoM - 5G Virtual Cloud Mobility, Research Day, Aveiro, Jun, 2017
- CONFIG - 5G Control Plane, Research Day, Aveiro, Jun, 2017

In addition, the software developed and demonstrators of above presented works are available as online contribution to GitHub projects:

- *OpenFlow within the Mobile Node*[7] explains how to deploy an OF-based UE for managing its wireless connectivity remotely;
- *5G-Virtual Cloud Mobility (5G-VCoM)*[8] explains how to reproduce and configure a similar framework where the wireless context of the UE is virtualized in the cloud;

[4]Mobilizador 5G: `https://www.it.pt/Projects/Index/4524`
[5]5GCONTACT: `https://5gcontact.av.it.pt`
[6]5Growth: `http://5growth.eu`
[7]OpenFlow within the Mobile Node (GitHub): `https://atnog.github.io/of_mobilenode/`
[8]5G-VCoM (GitHub): `https://atnog.github.io/5G-VCoM/`

- *Slice Management and Orchestration (SliMANO)*[9] offers a Python-based open-source framework for end-to-end slice management and orchestration.

Finally, besides the referred contributions, the expertise acquired during the development of this thesis were leveraged by the candidate to provide guidance in the development of the following master theses:

- A software defined network controller quantitative and qualitative analysis, Pedro Bispo, University of Aveiro, Portugal (2017);
- IoT traffic flow management in Software Defined Networks, Sofia Marques, University of Aveiro, Portugal (2018);
- Offloading Mechanisms for Mobile Networks using SDN in Virtualized Environments, Rui Silva, University of Aveiro, Portugal (2018);
- Service deployment on Multi-access Edge Computing environments, David Santos, University of Aveiro, Portugal (2018);
- Software Defined Networks and Network Functions Virtualization for critical and reliable communications in 5G environments, João Filipe, University of Aveiro, Portugal (2018);
- Cloud-based virtual customer premises equipment, Tiago Vieira, University of Aveiro, Portugal (2019);
- Assessing traffic prioritization in Software-Defined Networks using Open vSwitch, Marlene Gomes, University of Aveiro, Portugal (ongoing).

[9]SliMANO (GitHub): `https://atnog.github.io/SliMANO/`

**Table 1.1:** List of publications.

| Type | Year | Title | Venue | Ref. |
|---|---|---|---|---|
| Conference | 2017 | An abstraction framework for flow mobility in multi-technology 5G environments using virtualization and SDN | IEEE NetSoft | [12] |
| Conference | 2018 | Handover Initiation Comparison in Virtualised SDN-basedFlow Mobility Management | IEEE ISCC | [13] |
| Journal | 2018 | Deviceless Communications: Cloud-Based Communications for Heterogeneous Networks | Springer WPC | [15] |
| Book Chapter | 2018 | Experimental Wireless Network Deployment of Software-Defined and Virtualized Networking in 5G Environments | Springer | [14] |
| Conference | 2018 | SDN-based Mobility Management: Handover Performance Impact in Constrained Devices | IFIP NTMS | [11] |
| Conference | 2018 | Using sdn and slicing for data offloading over heterogeneous networks supporting non-3gpp access | IEEE PIMRC | [16] |
| Conference | 2018 | SDN-based End-to-End Flow Control in Mobile Slice Environments | IEEE WS MOBISLICE | [22] |
| Journal | 2019 | Micro and Macro Network Slicing: An Experimental Assessment of the Impact of Increasing Numbers of Slices | Springer WPC | [21] |
| Conference | 2019 | A Performance Comparison of Containers and Unikernels for Reliable 5G Environments | IEEE DRCN | [18] |
| Journal | 2019 | An integration of slicing, NFV, and SDN for mobility management in corporate environments | Wiley ETT | [17] |
| Journal | 2019 | Dynamic network slice resources reconfiguration in heterogeneous mobility environments | Wiley ITL | [23] |
| Conference | 2019 | Traffic-aware Live Migration in Virtualized CPE Scenarios | IEEE WS MOBISLICE | [19] |
| Conference | 2019 | Dynamic Modular vCPE Orchestration in Platform as a Service Architectures | IEEE CloudNet | [20] |
| Conference | 2019 | SliMANO: An Expandable Framework for the Management and Orchestration of End-to-end Network Slices | IEEE CloudNet | [24] |

## 1.4 Thesis structure

The remainder of the thesis is organized as follows: Chapter 2 introduces the state of the art of the base technologies of the upcoming 5G networks and consequently the developed study, as well as the related work in the mobility management of such architectures. Chapter 3 presents a first approach of the developed architecture, where both Point of Attachment (PoA) and UE were virtualized in the cloud, enabling a greater degree of abstraction when considering wireless mobility. Chapter 4 introduces network slicing to

the framework architecture and how slicing impacts in the user experience, as well as how the framework enables inter-slice mobility. Chapter 5 evolves from the previous framework architecture and implements mechanisms for dynamic slice instantiation and orchestration able to use Management and Orchestrator (MANO) frameworks for Virtual Infrastructure Manager (VIM) management and orchestration. Chapter 6 concludes the thesis and presents future research directions, namely Service-Based Architecture (SBA) for 5G networks.

CHAPTER 2

# State of the art and development tools

> "*The main domains of the 5G system are wireless access, transport, cloud, applications, and management including orchestration.*"
>
> — Ericsson

*The thesis presents an architecture framework for mobility management in sliced-networks. Notwithstanding, despite upcoming 5G networks encompass different levels of hardware, radio and network enhancements, this thesis focus solely in architectural enhancements leveraged by the softwarization and virtualization of the core network components. In this context, this chapter introduces the 5G system architecture, and evolves towards the considered 5G key enablers, namely SDN and NFV technologies. As proposals from the 5G research community evolves, new terms and networks schemes leveraged by such key enablers, started to emerge. Concretely, Network Slicing is considered as the game changer for future network architecture, allowing operators to optimize and share resources, while tailoring the network infrastructure to vertical use case scenarios. Lastly, evaluation tools and frameworks for virtualized slice-based networks are presented.*

## 2.1 5G System Architecture

The 5G system architecture has been under heavy standardization effort, with the 3GPP's Release 15 presenting its specifications in both 5G New Radio and Core Network, being followed by Release 16 and Release 17 whose specification is already underway. 5G aims towards a more efficient and cost effective network, leveraged by SDN and NFV mechanisms, as well as service-based interactions among network functions for allowing independent scaling and evolution of both control and data planes [25], [26]. The 5G system reference architecture for non-roaming scenarios is depicted in Figure 2.2. Also, 5G has been made efforts for a holistic network, supporting a unified authentication framework for converging 3GPP and non-3GPP access. In this line, Figure 2.3 illustrates the 5G reference architecture for non-3GPP accesses. Next, the NFs of the 5G system architecture are presented, and compared with the NFs of the current Evolved Packet System (EPS) (i.e.,4G network) presented in Figure 2.1.

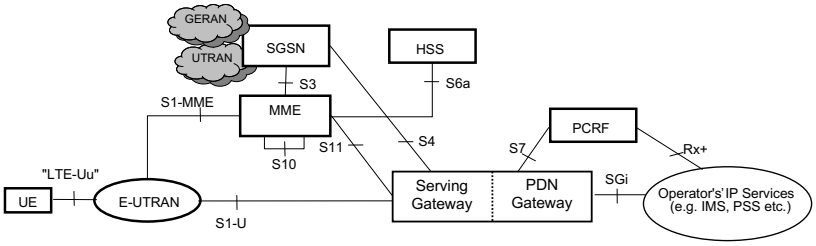



**Figure 2.1:** 4G System architecture (single gateway configuration option) [27].

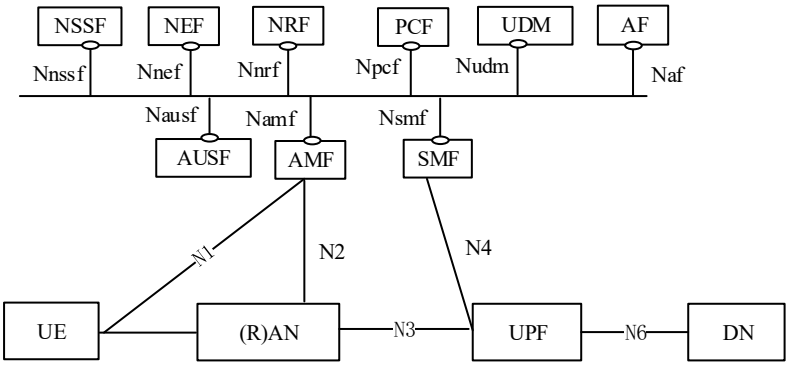



**Figure 2.2:** 5G System architecture [25].

- **Access and Mobility Management Function (AMF):** it connects to the Radio Access Network (RAN) control plane interface (N2) and it can be seen as the equivalent of the Mobility Management Entity (MME) from the EPS. Thus, the AMF manages the registration, connection, reachability and mobility of the UE.

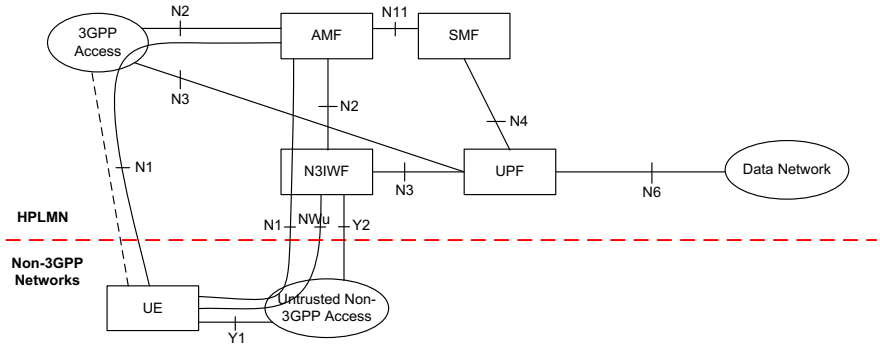



**Figure 2.3:** Non-roaming architecture for 5G Core Network with non-3GPP access [25].

Other functionalities involve the access authentication and authorization, as well as the security context management. Additionally, it supports the authentication of UEs connected via non-3GPP access (more details in Non-3GPP InterWorking Function (N3IWF));

- **Session Management Function (SMF):** when compared with EPS network functions, the SMF replaces the Service and Packet Gateways (Serving Gateway (S-GW) and Packet Data Network Gateway (P-GW), respectively), as well part of the MME. Thus, the SMF manages the UE's session and configures the traffic steering. It also ensures the roaming functionality;
- **User Plane Function (UPF):** it is the gateway for data traffic from the Access Network (AN), being an equivalent of the combined data plane of the S-GW and P-GW of the EPS. As such, a GPRS Tunneling Protocol (GTP) tunnel connects the AN to the UPF, which anchors the traffic for intra- and inter-RAT mobility;
- **Unified Data Management (UDM):** similarly to the Home Subscriber Server (HSS) network function of the EPS, the UDM verifies the user identification and access authorization based on subscription data and management;
- **Application Function (AF):** It enables application influence on traffic routing, provides access to the NEF and interacts with the policy framework for policy control.
- **Network Slice Selection Function (NSSF):** it selects the set of network slice instances serving the UE, while determining the AMF set to serve the UE;
- **Network Exposure Function (NEF):** it enables 3GPP NFs to expose their capabilities and events to other NFs, while handling the masking of network and user sensitive information to external AFs according to the network policy.
- **Network Repository Function (NRF):** it supports service discovery function by providing (upon request) the information (profile and supported services) of the discovered NF instances.

- **Policy Control Function (PCF):** similarly to the Policy and Charging Rules Function (PCRF) of the EPS, the PCF accesses the subscription information relevant for policy decisions in a Unified Data Repository (UDR) and provides it to the control plane functions its enforce.
- **Authentication Server Function (AUSF):** acting as an authentication server, it assumes part of the HSS of the EPS.
- **Non-3GPP InterWorking Function (N3IWF):** it establishes the tunnel with the UPF, while serving as local mobility anchor within untrusted non-3GPP access networks. It also communicates with the AMF for authentication and access authorization of the UE to the 5G core network in non-3GPP accesses.

In this thesis, the proposed frameworks were defined without explicitly considering the 3GPP standards, focusing instead on the assessment of the specific slice-based mobility management procedures themselves. Moreover, the framework was designed in such a way that allows it to be flexibly integrated into a 3GPP architecture. This is particularly verified in the framework presented in chapter 4, where the framework presented in chapter 3 was evolved and integrated in a SDN/NFV-based EPS complaint with 3GPP standards, maintaining (whenever possible) the existing interfaces used between 3GPP building blocks. Nevertheless, although the use of the EPS, following the current 3GPP standardization for 5G [25], proposed enhancements can be integrated as part of the above presented network functions.

## 2.2 Software Defined Networking and Network Function Virtualization as key enablers

As mentioned in previous section, 5G network systems have been addressing the holistic vision of a converged architectural solution. Such architecture should be flexible enough to support different network requirements for different network usage scenarios, namely enhanced Mobile Broadband (eMBB), Ultra Reliable Low Latency Communications (URLLC) and massive Machine Type Communications (mMTC). Additionally, this architecture should support optimized device mobility in heterogeneous wireless environments, developing mobility management procedures able to offer an optimal QoE for users. In this context, SDN and NFV appeared as the key enablers for such flexible and dynamic network. Next, SDN is introduced by presenting its architecture and its applicability to wireless environments. This is followed by an NFV introduction and the main research work related to wireless is discussed.

### 2.2.1 Software Defined Networking

Presented as a building block for upcoming 5G networks, SDN promises to introduce a greater degree of flexibility into the network architecture by decoupling the control-

path from the data-path, allowing the network to better adapt to more dynamic environments. As such, the network control operations are centralized in a high-level entity (namely, the SDN controller) able to dynamically coordinate and program the network forwarding entities (i.e., SDN switches) behaviour through a software API. The SDN controller became the central point of the network and uses NorthBound (NB) and SouthBound (SB) APIs to communicate with SDN applications and SDN switches, respectively.

Figure 2.4 illustrates the SDN architecture overview, where three layers are presented: (i) infrastructure layer; (ii) control layer; and (iii) application layer. Contrary to traditional networking where all thee layers are implemented in the firmware of the forwarding devices, in SDN the infrastructure layer is responsible for forwarding the user traffic according to the decision provided by the control layer. Forwarding devices are then able to expose their capabilities to the control layer through SB APIs and protocols such as OF [6]. The control layer uses SB APIs to acquire information from the infrastructure layer and apply management and controller decisions to the forwarding devices. The fact that in SDN the control layer is implemented through software allows the development of more dynamic solutions, specially when considering its integration with the application layer through NB APIs such as Representational State Transfer (REST). By doing so, besides monitoring the infrastructure layer, the control layer interfaces application and infrastructure layers, providing information to the SDN applications and performing policy management to the forwarding devices in order to meet applications' requirements and Service Level Agreement (SLA) [28]. In this regard, Open Networking Foundation (ONF) [29] claims that SDN should abstract network resources and state to external applications, as well as to ensure interoperability based upon open control layer interfaces. Also, it should not only support management interfaces for resource policies establishment, but also the co-existence with existing business and operations support systems.

As mentioned above, SDN provides open interfaces (i.e., SB and NB APIs) for the development of mechanisms capable of controlling and managing network resources connectivity, by performing traffic inspection and modification. The OF [6] protocol has been under standardization by the ONF and is considered the de-facto open-source SDN SB protocol instantiation, allowing the reconfiguration of network forwarding devices on the fly. In addition, due to its flexibility for accommodating new scenarios and applications, the OF protocol has evolved from research networks to a core component of network virtualization mechanisms and cloud-based architectures [30], becoming a key cornerstone for the upcoming 5G architectures [31].

In this sense, ONF's Wireless and Mobile Working Group (WMWG) [32] identifies SDN as an asset to wireless and mobile networks, contributing for RAN optimization and

overall QoE improvement. In [33], OpenFlow sheds its light onto wireless environments, allowing a wireless router to act as an OF switch. Also, Mobileflow [34] and SoftCell [35] complement the routing controller with a dedicated mobility engine, and the distribution of mobile networks data flows for MNs, according to a set of SDN policies. Also, works such as [36], [37], apply SDN to wireless traffic control, facilitating dynamic reconfiguration. Notwithstanding, it relies on complex procedures and dedicated protocols, hindering heterogeneity. In [38] SDN capabilities were conceptually extended to the MN, enabling and simplifying mobility-based scenarios. In [39], SDN is deployed within the MN providing it with optimized bandwidth estimation. In [40] an OpenFlow wireless extension to TDMA optimizations for IEEE 802.11 is experimentally evaluated. Experimental studies on flow-based mobility using OpenFlow in MN were done in [41] and [42], with the first only considering network initiated handovers and the later not focusing on the impact of the handover process. In [43], SDN was used in the MN, focusing on multi-homing protocols and requiring changes to the network endpoint on the receiver side, rendering the solution only operable in intra-domain scenarios.

Notwithstanding, above works did not considered slice-based architectures and how to dynamically manage the life-cycle of network slices for handovering the users' traffic to the optimal slice resources. In this line, this thesis virtualizes the UE allowing to network to acquired the UE's wireless context, while enabling it to dynamically anchor the UE's traffic and use SDN-based mechanisms for performing inter-slice mobility.

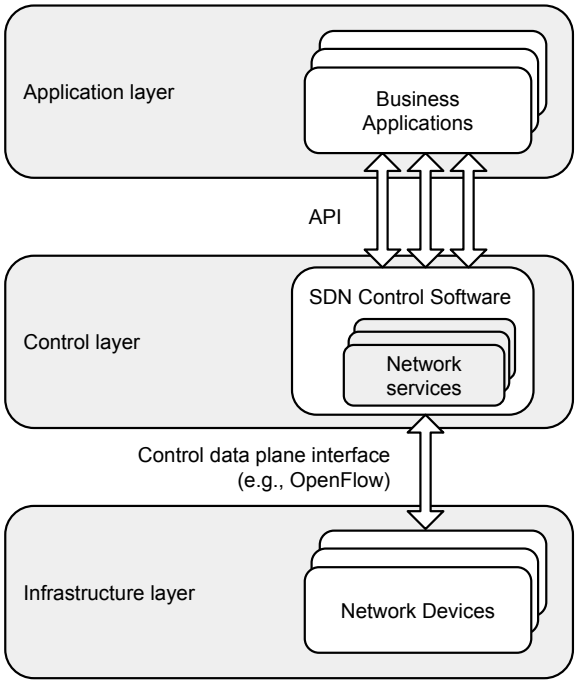



**Figure 2.4:** SDN architecture overview.

#### *2.2.1.1 OpenFlow protocol*

The connection between an OF switch to an OF controller is performed through an OF channel interface, that can run directly over Transmission Control Protocol (TCP) or encrypted using Transport Layer Security (TLS). The OF controller, uses this OF channel to configure and manage the OF switch via OF messages, which in turn are divided in three categories.

- *Controller-to-switch* messages are initiated by the controller and used to directly manage or inspect the state of the OF device. It may or may not require a response from the OF switch;
- *Asynchronous* messages are initiated by the switch and used to update the controller of network events and changes to the switch state;
- *Symmetric* messages are sent without solicitation by either the controller or switch.

Nevertheless, as can be seen in Figure 2.5, independently of the OF message type and version, its header serves the following structure: (i) a *version* field, specifying the version used in the communication; (ii) a *type* field indicating the message type and how it to interpret the payload; (iii) a *length* field indicating the end point of the message in the byte stream; and (iv) a *transaction identifier (xid)* for matching request and response OF messages.

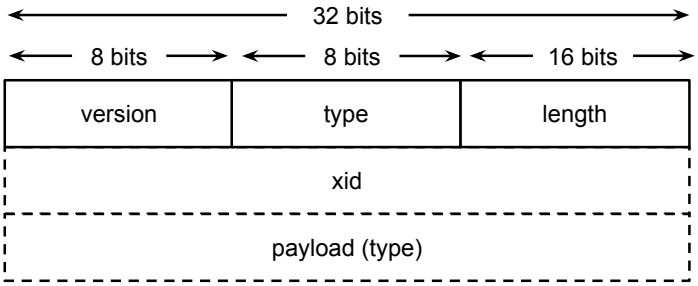



**Figure 2.5:** OpenFlow packet structure.

The first OF version was released in 2009 and it was limited to a single flow table with three components in a flow entry, namely Header Fields, Counters and Actions. Due to these limitations, multiple tables and group tables were introduced in OF v1.1, and basic support for IPv6 in OF v1.2. In 2012, OF v1.3 introduced Quality of Service (QoS) capabilities with the "Meter table" feature and provided the optional support for encrypted TLS communication and a certificate exchange between the switches and the controller. OF v1.4 introduced an unidirectionally or bidirectionally synchronization capability for flow tables, as well as "bundles" for grouping state modifications. "Scheduled Bundles" were introduced in OF v1.5, allowing synchronization among multiple switches [44]. At the time of this writing, despite OF

v1.5 being the latest version, vendors are still focusing in providing OF v1.0 and v1.3 in their products. Nevertheless, the OF mechanisms adopted in the framework proposal are backward compatible from OF v1.0. Due to the elevated number of messages in the current OF version, only the ones used in this thesis for acquiring wireless context from the MN and performing mobile offloading are described next.

- **Echo:** This symmetric message starts the connection between the MN and Controller by performing a *handshake*. It is started by the MN, and can be used to exchange information about latency, bandwidth and liveness;
- **Description/Features:** In these controller-to-switch messages, the MN announces its ports information and Media Access Control (MAC) addresses, allowing the Controller to identify the necessary characteristics of the MN and its connections to the network. An OF connection with the Controller for each interface is required, allowing the MN to determine which interfaces it wants to integrate into mobility procedures;
- **Status:** The OF status is an asynchronous message which can be used by the MN to inform the controller link up/down events of its network interfaces;
- **Packet_in**: The *packet_in* is an asynchronous message sent by the OF switch to the controller resulted from a mismatch flow or output action. In the proposed handover mechanism (section 3.1.2.2), the *packet_in* was generated by an output action in a flow entry with a specific matching, which was done through destination Internet Protocol (IP) address, allowing the MN to inform the controller of specific events such as a link going down;
- **Flow_modification (or flow_mod)**: The "modify flow entry message" is a modify-state message (sub-group of controller-to-switch messages) used by the controller to manage the OF switches state. It is used to add, delete and modify flow entries in the OF switch, allowing to redirect data flows from/to specific output ports. In SDN-based multiple interfaced MNs, it allows to redirect data flows from one wireless technology to another.
- **Barrier messages**: The *barrier_request* and *barrier_reply* are controller-to-switch messages used by the controller to ensure the completion of precedented operations before new ones. In a vertical handover, where the MN offloads the data traffic from one PoA to another, in the implemented mechanism two *flow_modification* messages are required. As such, the controller uses these messages to ensure that the first *flow_modification* is completed before sending the second one.

### 2.2.2 Network Function Virtualization

As discussed above, 5G research has been building its foundation in new architectural approaches, such as SDN and NFV, in order to tackle challenges such as massive traffic volumes, the proliferation of connected devices and sustainable integration of heterogeneous networks in mobile environments [45]. Standardization organizations, such as European Telecommunications Standards Institute (ETSI)[1] and ONF[2], are promoting an evolution from the traditional networks to others that leverage cloud-based mechanisms, by moving specific network functions to data-centers. In fact, the abstraction layer provided by SDN enables the deployment of a virtualized network independent of the underlying transport technology and network protocols. In this line, NFV contributes to this by providing fundamental mechanisms to decouple the NFs from the hardware, allowing their deployment into generic hardware. Thus, NFV allows the deployment of NFs as software instances running on servers through software virtualization techniques, instead of specialized and dedicated hardware [46]. Otherwise, by decoupling the data-path from the control-path, SDN allows routes to be dynamically setup through a standardised API. Also, it becomes possible to virtualize, in the cloud, IP network functions (such as load balancers, firewalls, security, and others) and Evolved Packet Core (EPC) network functions (such as, the MME, HSS and S-GW).

The adoption of SDN and NFV mechanisms not only allows improving the economics of deploying and managing services (i.e., CAPEX and OPEX), but actually provides new possibilities by bringing the virtualization concept to network operations. For example, in plain bare-metal scenarios, operators had to over-provision hardware by multiple orders of degree, in order to allow the network to face against peak-hours utilization demands. With virtualization, a more dynamic approach is able to be performed, with resources scaling according to real-time needs. In this context, while NFV is used to virtualize NFs, SDN improves network flexibility and (re)configurability.

Architectures such as Odin [47] and CloudMAC [48] already employ some of these concepts into wireless networks, by virtualizing APs in data centers. While Odin explores the implementation of high level services, such as Authentication, Authorization and Accounting (AAA), CloudMAC deals with the MAC frames of a specific AP in the cloud. Nevertheless, both architectures focus in the IEEE 802.11 standards, disregarding integration with cellular networks. Other approaches such as SoftRAN [8] and CloudRAN [9], further evolve from concepts that apply SDN in cellular networks, by exploring the usage of both SDN and NFV therein. Concretely, SoftRAN introduces a SDN-based centralized control plane for RANs which abstracts Base Stations (BSs) from a local area (SD-RAN), and provides different control algorithms for network operation.

[1] ETSI - http://www.etsi.org/

[2] ONF - https://www.opennetworking.org/

On the other hand, Cloud-RAN proposes to connect the SD-RANs to virtual BSs, providing a centralized control solution. Focusing on the wireless integration with NFV, OpenRadio [49] proposes a modular programming capability for the entire wireless stack, by decoupling the wireless protocol definition from the hardware, and ensuring that multi-core platforms are capable of implementing the protocols.

In this context, NFV has been identified as a solution for telecommunication service providers to perform more efficient operations (by remotely maintaining and updating NFs), and increase utilization efficiency of resources by using existing network capacity for more user traffic. In this line, ETSI has been developing standards for NFV deployment, providing some use cases, a reference architecture and a MANO framework [50]. Figure 2.6 presents the ETSI's NFV reference architectural framework, which encompasses the NFV Infrastructure (NFVI) and MANO.

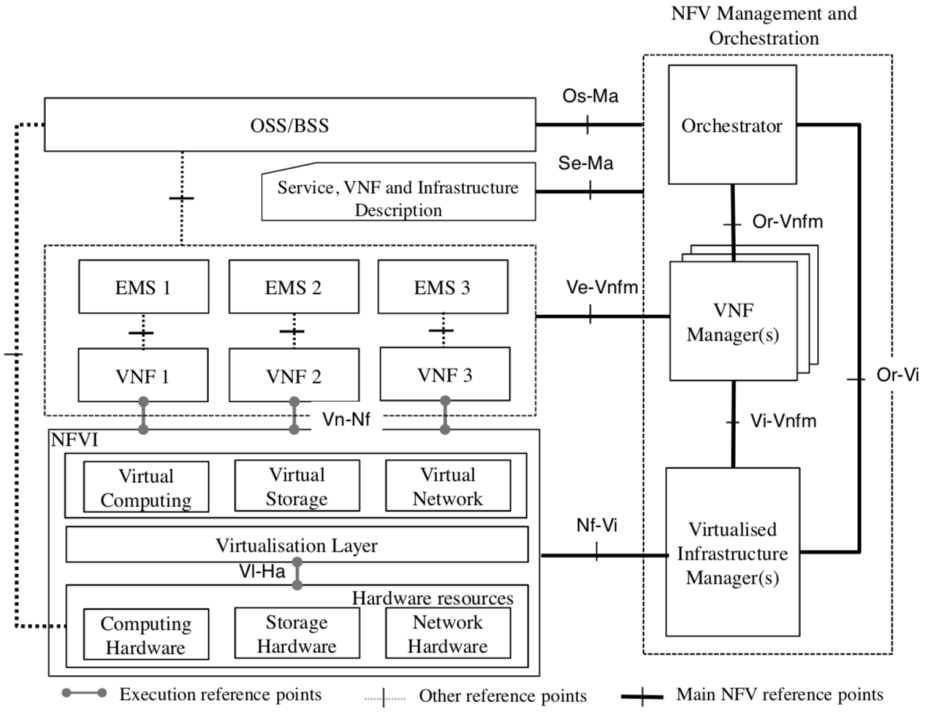



**Figure 2.6:** NFV reference architectural framework [51].

### *2.2.2.1 Management and Orchestration*

In virtualized networks, the MANO entity is a framework defined by ETSI responsible for the managing and orchestrating of resources in a virtualized data-centers. This includes compute, networking, storage, and VM resources. Also, the NFV MANO envisions a flexible instantiation, re-instantiation and deletion of network components. In Figure 2.6, MANO is composed by: (i) the NFV Orchestrator (NFVO) that is a key component of the NFV-MANO and manages network services by requesting, scheduling

and instantiating VNFs via VIM, while using the VNF Manager for VNFs' configuration; (ii) the VNF Manager (VNFM) that is responsible for the lifecycle management of VNFs, including instantiation, updating, scaling and termination of VNFs; (iii) the VIM that is responsible for controlling and managing the NFVI compute, storage, and network resources.

In the light of this, the NFVO orchestrates Virtual Network Functions (VNFs) by deploying and configuring them in a predefined set of VIMs, envisioning the creation of a network to support a certain service type. This concept of having a network deployed on top of another network, for supporting specific service types, is often presented as network slicing. Also, these virtualized networks are usually created by defining a datapath through the virtualized NFs (a chain of VMs), which results in a Service Function Chain (SFC).

In this context, the NFVO is one key component of MANO architectures, and it is in charge of the deployment and configuration of VNFs which are part of a network slice, leaving the interconnection (i.e., the chaining) of these VNFs out of its scope. However, this aspect still needs to be addressed, for enabling a truly automated slice orchestration. In this context, 3GPP proposes in their study for network slicing MANO [52] the following terms and definitions:

- **Network Slice Template (NST):** it describes the network slice and has the slice definitions and instance-specific information used as a skeleton to build-up network slices.
- **Network Slice Subnet Template (NSST):** it represents a slice template that is part of a higher level slice template.
- **Network Slice Instance (NSI):** it uses a NST as base and a set of custom parameters to build-up a network slice. The NSI includes all functionalities and resources (physical and logical) necessary to support certain set of communication services (thus serving certain business purpose). Also, the NSI contains NFs from both access and core networks, with the 3GPP management system containing the information of necessary interconnections between NFs (e.g., topology and QoS). Finally, a NSI may be fully or partly, logically and/or physically, isolated from another NSI.
- **Network Slice Subnet Instance (NSSI):** it is a NSI that is part of a higher level NSI. For example, for instantiation of a NSI that contains access and core network components, these components are instantiated as two NSSIs: one NSSI for the access network, and other for core. Thus, the targeted NSI is instantiated by combining both NSSIs. Nevertheless, a NSSI may be shared by multiple NSSIs and/or NSIs.

- **Network Slice Management Function (NSMF):** it manages and orchestrates NSIs. It derives network slice subnet related requirements from network slice related requirements and communicates with the Network Slice Subnet Management Function (NSSMF) and Communication Service Management Function (CSMF).
- **Network Slice Subnet Management Function (NSSMF):** it is responsible for the management and orchestration of NSIs. It communicates with the NSMF.
- **Communication Service Management Function (CSMF):** it translates the communication service related requirement to network slice related requirements. It communicates with the NSMF.

However, key issues for network slicing management, such as how services are requested via the 3GPP management system and how they are facilitated using and NSI, are still under discuss. Also, fully working solutions for slice orchestration for intra and inter administrative domains, as well as shared slice instance management, are still lacking.

## 2.3 The role of Network Slicing

Network Slicing (or slicing) is a concept that allows differentiating traffic traversing a shared network infrastructure, providing isolation to different scenarios (and their specific requirements). Currently, this is achieved through various mechanisms, such as VLANs or firewalls, adding complexity to networking configuration and requiring the deployment of special devices at appropriate network locations. More recently, the introduction of SDN allowed centralized network control, supported by protocols such as OpenFlow. This was leveraged by systems such as FlowVisor [53], which enable multiple applications to control a SDN network without interfering with each other, albeit requiring the interposition of complex hypervisor software at the management plane.

Lately, network slicing has been highlighted as a key enabler for the realization of the next generation of communications architecture (5G) [29]. The requirements in such environments are significantly more demanding than today's networks, envisioning an explosion of connected devices, massive increase of traffic and data rates with extremely reduced latency [3]. As such, the network needs to be built in a flexible cost-efficient way so that speed and capacity can be allocated on-demand in logical slices to meet the demands of each use case [4]. With the appearance of initial 5G architectural proposals, mostly based on SDN and NFV technologies, several key issues have surfaced, not only when considering the application of network slicing, but also the transition between mobile network and the novel 5G environments [54].

Network Slices are seen as the key for delivering differentiated service demands, as they are able to provide connectivity adapted and optimized for each different use case, application and user. In order to run the network in such resource efficiently way, researchers are laying its efforts into SDN and NFV technologies [4][3]. The level of network programmability provided by SDN allows several network slices, optimized for different service deployments, to be configured using the same physical and logical network infrastructure. "One physical network can therefore support a wide range of services and deliver these services in an optimal way" [4]. These features were exploited in [55] where in an SDN-based home network several slicing strategies were used for better user experience accordingly with the user demands and applications. On the other hand, NFV's flexibility allows the network functions to execute independently of location, being no longer bounded to a specific network node. The same network function can not only be executed in different places, but also be chained together as a logical network slice [4]. Slicing initiatives for mobile networks can be mainly divided into three groups [56]:

- **Spectrum-level (or radio) slicing:** it is the slicing of the spectrum though time, space or frequency multiplexing. Alternatively, the creation of an overlaid access (such as an Service Set Identification (SSID) in Wi-Fi) is also considered as a radio link virtualization. In this line, different solutions are proposed depending on the wireless medium access technology. In LTE, solutions such as [57], [58] propose an architecture for evolved Node B (eNB) virtualization with the objective of enabling infrastructure sharing among several operators. Also, [59] proposes an algorithm to dynamically allocate resource blocks depending on the slice demand. Considering the Wi-Fi, in [60] virtual APs with different SSIDs and security configurations are used for allowing a single AP to be shared among operators when deployed in popular locations (e.g., airports and hotels). Solutions such as [60] and [61], propose to configure Enhanced Distributed Channel Access (EDCA) parameters to improve throughput of the slices.
- **Infrastructure-level slicing:** it is the slicing of the physical network elements (antennas, base stations, memory) and accomplished mostly by virtualization. This allows network operators to share the infrastructure, which gains more impact when considering areas with limited coverage [62]. However, despite projects such as OpenAirInterface (OAI) and FlexRAN allowing the EPC virtualization and the eNB radio slicing, respectively, no fully working proposals were found in the literature for infrastructure sharing among operators.
- **Network-level slicing:** it is the slicing of all the network infrastructure as an end-to-end slice. Similarly to the previous one, fully working architectures capable of creating, managing and orchestrating end-to-end slices are still lacking.

Also, slicing implies the allocation of the required resources for independent services [56]. However, in wireless environments, the slice isolation of the wireless access medium adds particular challenges, especially when ensuring QoS and SLAs. In this context, not only SDN and NFV are seen as key enablers for resource slicing in wireless networks, but also for Wireless Network Virtualization (WNV) [56]. Towards slicing per service, user or application, WNV aims not only the sharing of the infrastructure, but also the radio spectrum.

Applying these concepts, works such as [63] propose through simulations where control mechanisms dynamically allocate network resources to different slices, in order to maximize users' satisfaction. However, the work focuses in QoE, disregarding mobility issues. In [64], the authors proposed a "5G Network Slice Broker" to facilitate on-demand resource allocation and perform admission control based on traffic monitoring and forecasting. Similar to this, in [65] the concept of hierarchical Network Slice as a Service (NSaaS) is introduced, allowing network operators to customize end-to-end networks as a service. Despite enabling operators to build network slices for vertical industries more agilely, these approaches disregard mobility management issues, such as the role of the MME in such architecture, or even the communication between the network slice and the MME. Next, initiatives for different slice dimensions are presented.

### 2.3.1 Slicing initiatives and implementation efforts for different slice dimensions

Different slicing initiatives and proof-of-concept frameworks have been proposed in recent years. However, the majority of such proposals often focuses in specific use cases, without presenting their integration with the global architecture, which in turn demands great flexibility in order to tailor a slice for the verticals. In addition, such existing works also address the subject assuming a single type of network slice (or at most, just distinguish between radio access or network services slices at the core). In this line, next recent slicing initiatives and proof-of-concept frameworks are reviewed, viewing them under the lens of the network slicing granularity classification and map them according to the architecture proposed in this thesis (presented in chapter 4).

#### *2.3.1.1 Operator*

Despite existing works (such as, [66], [67]) proposing infrastructure sharing among network providers, few actually experimentally implement such approach. For example, OAI [68] along with FlexRAN [69] allows to slice the Mobile Network Operators (MNOs) infrastructure, by virtualizing the EPC in a data-center and instantiate multiple radio slices, respectively. However, such frameworks do not allow the RAN sharing, requiring

each MNO to deploy its own RAN infrastructure (more specifically, eNBs). In this line, it is necessary to develop mechanisms allowing eNBs to be shared among MNOs, in order to allow full infrastructure sharing. Regarding to non-3GPP infrastructure sharing initiatives, works such as [16], [70] present proof-of-concept frameworks to dynamically instantiate Wi-Fi slices, which redirect traffic to the respective core networks, allowing APs deployed in public places to be shared among MNOs.

#### *2.3.1.2 Geographic area*

As above mentioned, frameworks such as FlexRAN [69] and OAI [68] enhanced the deployment flexibility of mobile networks. Multi-access Edge Computing (MEC) and fog computing are enablers for this type of slices, as the closer proximity of the network function deployment infrastructure to the end users, allows for geographical-based performance gains and resource usage optimization. In this line, by enhancing the modularity of such frameworks, it is possible to dynamically manage the control and data planes in order to instantiate a geographical slice, and use MEC to reduce latencies.

#### *2.3.1.3 Type of client*

Frameworks such as FlexRAN [69], allow the RAN slicing for MNOs, with instantiated RAN slices being attached to the MNO's EPC (as presented in section 4.2.3). Similarly, in [16], multiple Wi-Fi slices were instantiated with different QoS. Such approaches allow MNOs to instantiate 3GPP and non-3GPP slices for different classes of users, providing different QoS for each slice (as presented in section 4.2.2).

#### *2.3.1.4 Access technology*

Currently, proposed architectures in the literature provide mechanisms to create network slices for different access technologies, disregarding the interoperability between slices of different access technologies. In this line, [69] proposes the slicing of 3GPP networks, while works such as [70] slice non-3GPP PoAs for enhanced QoS and/or infrastructure sharing among operators. However, providing inter-operation capabilities between both slices requires both the development of new mechanisms as well as the enhancement of existing procedures. In [16], the candidate took the first steps towards this direction and developed a framework that dynamically instantiated a non-3GPP slice for the UE, allowing mobile video offloading while maintaining the QoE (presented in section 4.2.4.1).

#### *2.3.1.5 Node/UE*

Aligned with the interoperability between slices of different access technologies, the instantiation of slices per UE can be proposed. This allows the access technology currently in use by the UE to become transparent, enabling seamless handover scenarios. In this line, the use of a virtualized representation of the UE inside the network provider's

VNFs (as in chapter 4) anchors the UE to the network, allowing the network provider to flexibly move the slice among the different access technologies. Alternatively, the use of bond interfaces[3] allows the unification of access technologies in the UE, which also allows the transparency of multiple slices to the end-user.

#### *2.3.1.6 Interface*

Here, a different slice is instantiated depending if interfaces are from the same access technology, or not. For interfaces of the same access technology (e.g., dual-SIM smartphone), the MNO instantiates a slice per interface. For interfaces of different access technologies, the slice is similar to the access technology slice type. This type of slicing allows for greater UE connectivity resilience in case of network failure, and despite being supported by current slicing implementations, mechanisms for slice handover management in inter-domain scenarios need to be further developed.

#### *2.3.1.7 Flow*

This is the most frequent type of radio slicing. Works such as [71], [72] propose algorithms for the allocation of resource blocks depending on the traffic characteristics. Conversely, [22] noted that such approaches do not guarantee the uplink traffic demands in scenarios with "greedy" flows, and propose instead the use of SDN technologies within the UE, for dynamic queuing of flows.

#### *2.3.1.8 Packet*

This type of slicing aims to offer to the UE the capability to attach the same interface to multiple slices, allowing the parallel transmission of packets of the flow, with the benefit of enhanced throughput, redundancy and resilience. However, currently there is no slicing mechanisms in the literature that allows the UE to dynamically send packets of the same flow through different slices.

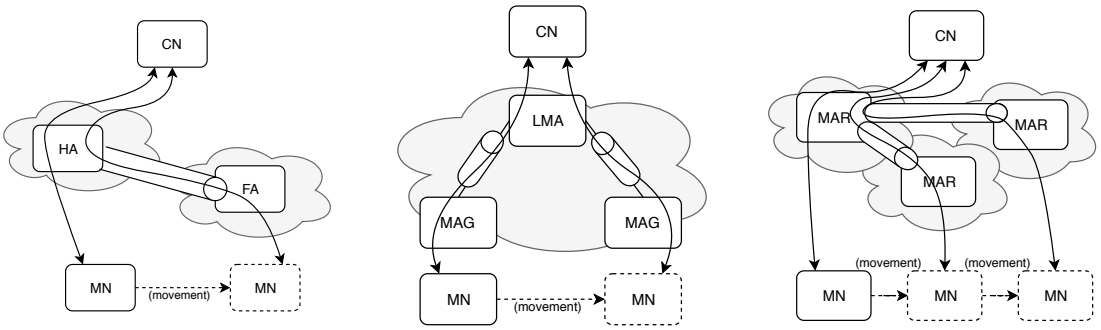
## 2.4 Mobility Management for 5G networks

As previously presented, wireless devices have been increasing its share in the global Internet data consumption [1], evidencing the need to develop mechanisms capable of performing seamless handovers over multiple access technologies (e.g., mobile and Wi-Fi). Notwithstanding, 5G studies show that despite of the increasingly large segment of users/devices, 5G solutions should not assume mobility support for all devices and services. Instead, mobility on-demand should be supported, ranging from very high

[3] Bond interfaces: Bonding network interfaces allows to have multiple network interfaces attached to a virtual interface (i.e., bond interface), which becomes the network interface used by the kernel for network access interfaces. Nevertheless, the bond interface physically uses the actual network interfaces (e.g, wlan0, wlan1, etc). This increases redundancy and availability for network access.

mobility, such as high-speed trains/airplanes, to low mobility or stationary devices such as smart meters [3].

Device mobility can be seen as changing PoA (namely, BSs or APs) along with associated procedures. MIP [73] was introduced by the Internet Engineering Task Force (IETF), to address mobility by mapping, in the Home Agent (HA), a global address with the device's current IP address. Thus, packets destined to the MN are received by the HA and tunneled towards the MN. However, even considering its IPv6 variant (MIPv6), IP-based mobility was not deployed Internet-wide [74] due to handover delay, triangular routing between HA and Foreign Agent (FA), signaling overhead and stack software modifications. In alternative, network-based mobility management protocols such as Proxy Mobile IPv6 (PMIPv6) [75] and DMM [76] were proposed, shifting mobility signaling to network nodes. Figure 2.7 compares the MIP, PMIP and DMM architectures. PMIPv6 still inherited issues such as non-optimal path communication between the MN and the Correspondent Node (CN), and tunneling. Nonetheless, PMIPv6 was deployed as a network-based mobility management protocol, handling the signaling on behalf of the MN, improving the localized routing handover delay, signaling cost, and Local Mobility Anchor (LMA) utilization [74]. Despite an evolution, PMIPv6 can be simultaneously deployed with MIPv6 [77], with the latter used globally and the former locally. Conversely, DMM was designed to be flexible and distributed [78], and supported by IETF and 3GPP, DMM complemented or replaced existing functions (e.g., LMA and Mobile Access Gateway (MAG) became a single entity, namely Mobile Access Router (MAR)) [76]. However, the inherent triangular routing and tunnels between MARs are still issues [78].




**(a)** MIP **(b)** PMIP **(c)** DMM

**Figure 2.7:** Mobility architectures comparison.

Mobility management studies for 5G networks started with initial SDN-based DMM architectures [46], [79]–[81] (Figure 2.8), resulting in a framework providing mobility and QoS for the end-user, in terms of latency and throughput. Here, mobility tunnels between MARs become unnecessary due to the added flexibility for dynamic datapath

reconfiguration offered by SDN's SB APIs, such as the OF protocol. However, 5G provides a new opportunity for the development of a fully converged architectural solution able to support optimized device mobility in heterogeneous wireless access environments. Thus, works, such as [33], applied OpenFlow capabilities to APs, further extended in [82], allowing SDN controllers to remotely configure flows traversing them. This established the foundation for using SDN and OpenFlow as mobility management mechanism, with [36] and [37] supporting an enhanced controller capable of managing handovers, using IEEE 802.21 and OpenFlow. OpenFlow-based mobility support procedures were also applied into mobile devices [38]. Experimental deployments were presented in [39] and [40] focusing on IEEE 802.11 technologies, and [41], [42] validating SDN mobility management capabilities in heterogeneous environments. This was progressed in [43] by deploying OpenFlow in end-to-end source mobility scenarios.

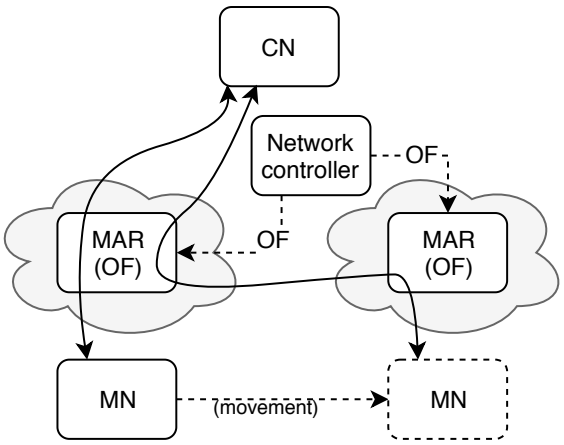



**Figure 2.8:** SDN-based DMM architecture.

Works such as [83]–[85] have provided mobility management enhancements enabled by the creation of virtual APs (vAPs) (as in [47], [48]). In [83] the vAPs is able to "follow" the MN, placing mobility procedures entirely inside the virtualized network of interconnected vAPs. As such, each MN is associated with its own vAP when it connects to the network, the latter moving along with its client, with the MN unaware of the physical handover. In [84] the authors take advantage of APs operating on multiple channels for seamless handovers. In [85] SDN-based procedures were proposed for creating, managing and migrating vAPs, without modifying MNs.

Nevertheless, the proliferation of new network services and end-devices that differ in terms of requirements and communication capabilities, creates the need to tailor the network to specific use cases and vertical industries (e.g., eHealth and automotive). In this context, "network slicing offers an effective way to meet all of the diverse use case requirements and exploit the benefits of a common network infrastructure, enabling operators to establish different capabilities, deployments, and architectural flavors for

each use case" [86]. As such, mobility mechanisms must be developed to enable the deliver of differentiated services over optimized connectivity and tailored to the use case, application and user. In the light of this, this thesis provides a framework that allows the dynamic instantiation of 3GPP and non-3GPP network slices, using SDN- and NFV-based mechanisms and considering the user's data requirements. For this, as mentioned before, the UE is virtualized, with the network preserving the user's wireless context in the vUE, and allowing it to partake and contribute to the optimization of the network by anchoring the UE's traffic and enabling the handover of traffic between slices (if necessary).

## 2.5 Evaluation tools, software and frameworks

A number of research efforts have focused on novel solutions for deploying virtualized and slice-based networks. Available solutions provide a reference and material to analyze and explore the concepts addressed along this thesis. As such, this sections presents possible technologies capable of implementing the desired framework, while providing an overview of them, highlighting their architecture, features and limitations.

### 2.5.1 Softwarization tools

OF switches can be either physical or virtualized, with Open vSwitch (OvS) being one of the most used open-source software (virtualized) switches. OvS not only supports standard management protocols (e.g., sFlow and NetFlow), but also open protocols such as OpenFlow and OVSDB, that expose the flow-based forwarding state and the switch port state, respectively. Also, OvS allows its use as an OF-hybrid switch, enabling the use of both OpenFlow and "conventional" Ethernet switching operations (i.e., L2 Ethernet switching, VLAN isolation, L3 routing). As such, OF-hybrid switches provide a classification mechanism outside of OpenFlow that routes traffic to either the OpenFlow or normal pipeline [87]. Regarding to SDN SB APIs, OpenFlow allows the remote and on-the-fly re-configuration of the flow tables (by adding, updating or even deleting a flow entry). Thus, each flow table is a set of flow entries, and each flow entry consists of match fields, counters, and a set of instructions to apply to matching packets. Nevertheless, when changes are required in the management plane, SDN-enabled forwarding devices implement different open protocols. OvS implements the OVSDB protocol for defining the configuration of the OvS itself (e.g., addition of a port to a OvS bridge, creation of a new OvS bridge) as well as the QoS policies.

#### *2.5.1.1 SDN Controller*

The SDN controller interfaces the application and infrastructure layers, offering open APIs for the development of SDN applications that helps the controller to manage

SDN-enabled forwarding devices. Doing this, the SDN controller abstracts the different levels of the network, enabling the network to evolve to a more programmable state, improving its functionality, flexibility and adaptability. Next, the most popular open-source SDN/OF controllers are presented.

- **OpenDaylight (ODL)[4]:** part of Linux Foundation Networking, it is a JAVA-based open-source SDN controller, driven by a global, collaborative community of vendor and user organizations that continuously adapts to support the industry's broadest set of SDN and NFV use cases.
- **NOX[5]:** considered the original OF controller, it was written in C++ and initially developed by Nicira Networks (side-by-side with OpenFlow) in 2009.
- **POX[6]:** presented as NOX's younger sibling, it brings faster development and prototyping of SDN/OF applications through Python.
- **Floodlight[7]:** it is an enterprise-class, Apache-licensed, Java-based OF Controller supported by a community of developers and maintained by Big Switch Networks.
- **Open Network Operating System (ONOS)[8]:** it is an open-source SDN controller for building next-generation SDN/NFV solutions, designed to meet the needs of operators wishing to build carrier-grade solutions. It is a JAVA-based open-source project distributed under the Apache 2.0 license, released in 2014 by Open Networking Lab (ON.Lab). Currently, it is maintained by Linux Foundation.
- **Ryu[9]:** it is a component-based SDN framework that provides software components with a well defined API that makes it easy for developers to create new network management and control applications. It is available under the Apache 2.0 license, written in Python and supported by Nippon Telegraph and Telephone (NTT).

Comparing the above presented SDN controllers, JAVA-based controllers (namely, ODL, Floodlight and ONOS) offer a similar feature set by being vendor-neutral and supporting open interfaces for SB an NB. Nevertheless, Python-based controllers (namely, POX and Ryu) also support such capabilities with the added benefit of offering a more agile software framework for a fast developing and prototyping, when compared with such "all-purpose" monolithic controllers. Moreover, performance improvements of JAVA-based controllers are only noticed for large scale networks [44]. In the light of this, the framework proposal in this thesis used Python-based controllers: initially using POX and then migrated to Ryu (for better features support, such as OF version and NB AP). Nevertheless, the framework is agnostic to the SDN controller choice,

[4]OpenDaylight: http://www.opendaylight.org/
[5]NOX: http://www.noxrepo.org/nox/
[6]POX: https://noxrepo.github.io/pox-doc/html/
[7]Floodlight: http://www.projectfloodlight.org
[8]ONOS: https://onosproject.org
[9]Ryu: http://osrg.github.io/ryu/

requiring the necessary application adaptations. Here, both POX and Ryu controllers offer APIs for developing applications through Python modules.

### 2.5.2 Virtualization Platforms

This section presents virtualization platforms as enablers for virtualizing network entities and functions.

- **OpenStack**[10]**:** OpenStack is an infrastructure platform which enables the deployment of private clouds. It allows developers to instantiate VMs on-demand with a complete Operating System (OS) stack (through an hypervisor). However, due to cloud security concerns, when a chaining of VMs to redirect flows is needed, it is required to encapsulate, perform Network Address Translation (NAT) or change the header of the packets.
- **ProxMox**[11]**:** Proxmox VE is an open-source Linux-based operating system that combines two different virtualization technologies, namely Kernel-based Virtual Machine (KVM) and Linux Containers (LXC), allowing users and developers to instantiate VMs. Contrary to OpenStack, ProxMox favors stability over flexibility.
- **Docker**[12]**:** Docker Engine provides containers for managing software workloads on a shared infrastructure, while isolating them via built-in OS features. Nevertheless, it allows direct access to the device drivers which makes I/O operations faster than with a hypervisor approach. Docker containers do not require a complete boot of a new OS, resulting in a lighter alternative for running applications on shared compute resources. Notwithstanding, Docker still has some limitations, such as isolation and security.

In this context, both OpenStack and ProxMox allow the deployment of a virtualization platform for private and public clouds. Despite ProxMoX VE being more stable, OpenStack is more automated and has a lively community ensuring new resources for new project developers and users. Regarding to the use of Docker containers, it significantly reduces the on-demand instantiation load of services, since containers' images do not have the guest OS. Also, the Docker Engine can be installed in a VM of a cloud provider, allowing to deploy a pre-built framework to be launched there, enabling the switch between OpenStack or bare metal environments whenever needed.

### 2.5.3 MANO frameworks

As presented in Figure 2.6, the MANO entity is composed by a NFVO, VNFM and VIM. In this context, Topology and Orchestration Specification for Cloud Applications

[10]OpenStack: https://www.openstack.org/
[11]ProxMox: https://www.proxmox.com
[12]Docker: https://www.docker.com/

(TOSCA) is a data model for orchestrating NFV services and applications and allowing to automate cloud infrastructure and resources through information models and templates. Conversely, ETSI standardized the functionalities of network components, with solutions for NFV orchestration from both academia and open-source organizations (such as ETSI itself and Linux Foundation) following these specifications. Next, well known MANO frameworks are presented.

- **Cloudify**[13]**:** it is an open-source cloud orchestration framework that enables to model applications and services, as well as to automate their entire life-cycle. Also, it is characterized by being a TOSCA-based technology. Initially developed by GigaSpaces Technologies, Cloudify currently belongs to a company with the same name.
- **Open Baton**[14]**:** it is a an extensible and customizable NFV MANO-compliant framework that integrates with existing VNFMs. Although OpenStack is the major supported VIM, it provides a driver mechanism for supporting additional VIM types. Open Baton was developed by Fraunhofer FOKUS and TU Berlin.
- **Open Network Automation Platform (ONAP)**[15]**:** it provides a unified operating framework for vendor-agnostic, policy-driven service design, implementation, analytics and life-cycle management for large-scale workloads and services, allowing network operators to synchronously orchestrate Physical Network Functions (PNFs) and VNFs. It is open-source and a Linux Foundation Networking project. Currently, ONAP is in the fourth release.
- **Open Source MANO (OSM)**[16]**:** it is an operator-led community hosted by ETSI that develops an Open Source NFV MANO software stack (aligned with ETSI NFV) for NFV networks. At the time of this writing, OSM is in release SIX.

### 2.5.4 Network slicing tools

As discussed in section 2.3, an end-to-end slice is composed by underlying radio and infrastructure slices. As such, different network slicing tools were used to accomplish dedicated and isolated networks deployed on top a single infrastructure. In this line, in this thesis, infrastructure slicing was accomplished through virtualization, where VNFs were deployed and chained creating SFCs and, ultimately, creating networks for supporting a certain service type. Regarding to radio slicing, tools for 3GPP and non-3GPP access networks are presented below.

[13]Cloudify: https://cloudify.co
[14]Open Batton: https://openbaton.github.io
[15]ONAP: https://www.onap.org
[16]OSM: https://osm.etsi.org

- **FlexRAN**[17]**:** it is a platform that separates the control and data planes of RAN (4G and 5G) runtime environment, and acts as an abstraction layer for the RAN. Also, FlexRAN allows not only the RAN slicing, but actually the development of RAN control applications for monitoring and coordinating the RAN infrastructure.
- **Hostapd**[18]**:** designed to run in the background for controlling the authentication of IEEE 802.11 APs, the hostapd software allows the creation of overlaid Wi-Fi access networks (i.e., SSID in Wi-Fi), which is also considered a radio link virtualization (as presented in section 2.3).

### 2.5.5 Wireless testbed

As previously presented, this thesis features a framework envisioning a holistic network for different access networks. As such, the framework was implemented over 3GPP and non-3GPP access networks testbeds, which are presented as follows.

- **3GPP network:** Regarding to 3GPP network deployment, the OAI[19] is a consortium of industrial and academia contributors that offers an open-source software and hardware development for the core and access networks 3GPP-complaint.
- **non-3GPP network:** The AMazING testbed [88] is an outdoor system composed by 24 fixed nodes distributed across $1200m^2$ sited in the rooftop of IT in Aveiro, Portugal. It allows controllability (for the experimenter) and high reproducibility of the tests, providing to the users a full access to node devices for expanding its capabilities by locating the core functions that eventually access the nodes' wireless interfaces.

## 2.6 Chapter Considerations

The current state-of-the-art presents solutions for a slice-based 5G architecture at several levels of deployment. Also, the existing work does not take into consideration implications that network slicing, as a key enabler for 5G, has on mobility management procedures involving different access technologies, such as Wi-Fi and mobile networks. They do, however, pave the way to enable a better understanding of how the existence of different slices can be considered, both at the access and the core level, having different use cases requirements in mind and providing a framework for flexible mobility management support in 5G environments. Finally, this chapter reuses partial material from the publications presented in Table 1.1 authored by the candidate.

[17]FlexRAN: http://mosaic-5g.io/flexran/
[18]Hostapd: https://w1.fi/hostapd/
[19]OpenAirInterface (OAI): https://www.openairinterface.org

CHAPTER 3

# A Virtualized SDN-enabled Framework for Mobile and Network Devices

> "*A digital transformation, enabled by mobility, cloud and broadband, is taking place in almost every industry, disrupting and making us rethink our ways of working.*"
>
> — Ericsson

*The increasing number of connected mobile devices exploring wireless data traffic through different technologies has been increasingly developing the potential for interoperability scenarios in mobile operators. The presence of different PoAs to the network provides users and services with added benefits, by increasing connectivity and optimization opportunities. However, the different access technologies operate in distinct ways and create a complex integration challenge for their combined exploitation. Upcoming 5G deployment scenarios have been further accentuating these scenarios, with the combination of novel access technologies into the telecommunications domain.*

*This chapter introduces a framework that integrates SDN and NFV to migrate operational aspects of both the network's PoA and MNs into the cloud. The SDN Controller is used as an enhanced mobility management entity by integrating the capability to instantiate in the cloud a virtual representation of the MNs, which not only act as anchors for the handover process of the physical MNs, but also act as proxies for the delivery of context information about the physical MNs wireless surroundings. Such information can be further used by the Controller for optimized handover decisions, resulting in a technology-agnostic flow mobility management mechanism for heterogeneous networks.*

## 3.1 Network Architecture

The way we use and access the Internet has been changing since its creation, evolving to more dynamic traffic patterns and following new consumption trends such as IP video in mobile scenarios. New requirements towards the underlying network rise, particularly regarding to mobility management support in wireless heterogeneous environments. Despite 5G network architectures have begun development empowered by SDN and NFV technologies, the application of such technologies in wireless mobility scenarios still has to deal with the specifics of each link layer technology.

In the light of this, the framework proposed in this chapter explores a holistic vision of SDN and NFV integration to provide flexibility and abstract mechanisms to diminish the necessary measures for supporting flow mobility management in heterogeneous scenarios. For this, NFV mechanisms are used to instantiate a virtual representation of the MN (i.e., virtual MN (vMN)) and the PoA (i.e., virtual PoA (vPoA)) in the cloud for abstracting wireless context (i.e., active links, signal level, neighbor cells) and means for optimizing link selection and heterogeneous mobility procedures. In addition, these virtual representations can be enhanced beyond the capabilities of their physical counterparts, and be coupled with the necessary logic that allows them to inter-operate or be controlled by network control processes, independently of the involved access technologies. Figure 3.1 illustrates the architectural concept of the framework, which is composed by thee main building blocks:

i. Enabling end-devices with SDN capabilities, allowing flow-based handovers without additional mobility protocols (section 3.1.2);
ii. PoA virtualization, exploring the cloudification of the network (section 3.1.3);
iii. Moving the wireless context to the cloud by virtualizing the MN (section 3.1.4).

The remainder of this sections individually evaluates the SDN-based mechanisms into the MN and the PoA virtualization, in section 3.1.2 and section 3.1.3, respectively. This is followed by the virtualization of the vMN, and the evaluation of the overall framework architecture in mobile offloading scenarios (section 3.1.4). Next, the network controller role in the framework is discussed.

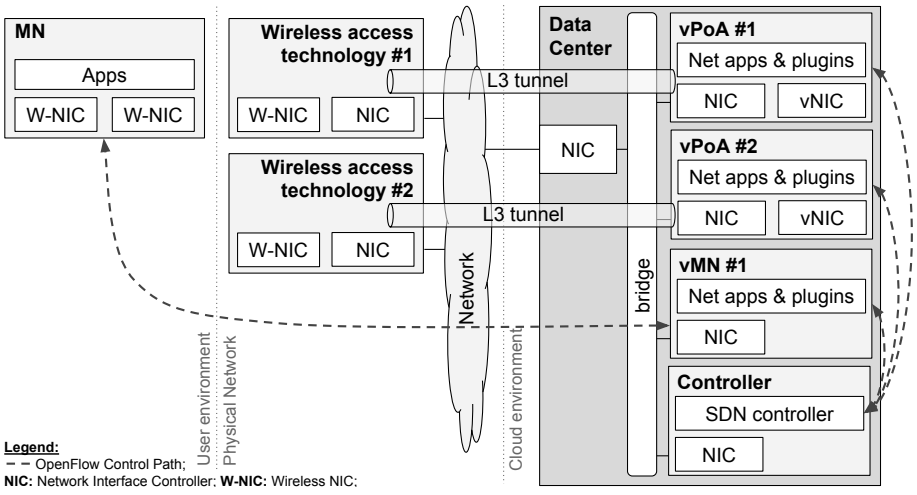



**Figure 3.1:** Framework Concept Overview [12].

### 3.1.1 Network Controller

In the proposed architecture, the SDN Controller is enhanced with management capabilities for all network nodes in regards to flow mobility and virtualization of network entities (e.g., PoA and MN). Also, the SDN controller is able to trigger the vMN instantiation in the cloud, and bind it to the MN (further detailed in section 3.1.4). It leverages OpenFlow to communicate with the virtualized equipment in the cloud, (re)configuring network resources via flow-based rules. For handover procedures, it maps the MN to the correspondent vMN, leaving the handover procedure of flows to the selected vMN. It can scale in the operator's data-center or cloud as needed, and integrate therein with other core decision entities. Figure 3.2 depicts the SDN controller architecture.

Here, the POX controller was used as SDN controller and integrated with a developed application (able to process OF messages sent from the vPoA and vMN), to add VMs instantiation and handover execution capabilities, over the base SDN Controller operations. Concretely, it allowed for flow-based rule operations by the SDN Controller to also encompass specific flow matching, through the usage of the OF *packet_in* message (which are typically used for flow mis-match with flow tables in OF switches). Additionally, this application identifies the type of network node (i.e., MN or switch) storing its information and instantiating a new vMN if required. Nevertheless, this application is a behavioral example that can be enabled in the controller by the framework, but the design does not impose a mandatory procedure. Other potential aspects, such as an enhanced NB API allowing other network entities to convey such features (through SDN applications) can be considered and developed. Moreover, despite the use of the POX controller, the framework was built independently of the SDN controller choice.

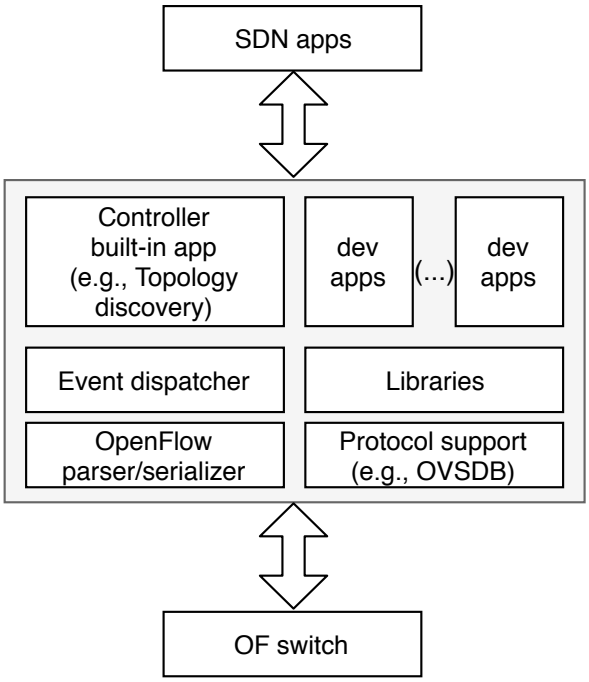



**Figure 3.2:** SDN controller architecture.

### 3.1.2 Enabling end-devices with SDN capabilities

This section explains which SDN mechanisms were extended up to the MN in this proposal for improving mobility management in heterogeneous wireless environments, by enabling flow-based handovers using OpenFlow and without requiring additional mobility protocols, contributing for a more homogenized network. For this, an OF controller (i.e., the network controller or the vMN, discussed in section 3.1.4) is able to interact, in a media-independent way, with the MN via OpenFlow, enabling the former to acquire link context information (e.g., link strength) from the MN. Such information can be further combined with other Key Performance Indicators (KPIs) obtained (e.g., signal strength, neighbor cells) from monitoring subsystems, generating a handover decision. Thus, this allows the network to enforce traffic optimization and policing, since by establishing a connection between the MN and an OF controller, the latter becomes capable of controlling how MNs connect to different access technologies, and implementing flow-level actions, such as packet redirection between network interfaces in offloading scenarios.

The MN architecture is illustrated in Figure 3.3, and features a multi-interfaced MN (as current dual Wi-Fi/Mobile smartphones), targeting flow-based mobility environments. Here, OF bridges (i.e., OF-BR) were instantiated in the MN and interconnected with patch ports (performing virtual connections that act as patch cable between switches), enabling the traffic redirection (for flow mobility) between wireless interfaces. In this way, it is possible to not only change packet characteristics (i.e., source and destination

addresses) before it leaves the node, but also from which interface (with the added benefit of this feature being remotely controlled by the network, via an OF controller). For this, the OvS was installed in a dual-interfaced MN, with an OvS bridge for each wireless network interface. Since MNs are characterized as both content producer and consumer, the OvS bridges are instantiated as Layer 3 (L3) between the Linux network stack and the physical network interface. The OvS L3 bridge allows the configuration of an IP address, enabling it to receive and send packet from/to kernel and from/to wireless interfaces.

In addition, since a dual-interfaced MN implements two OF bridges, it opens two OF connections with the OF controller, which in turn requires a mechanism for identifying network interfaces from the same MN. For implementation purposes, the framework proposal names the patch-ports with the MAC address of the destination bridge, simplifying the identification mechanism (cross-matching) of bridges of the same system by the controller. In this way, the creation of an OF system identifier, as well as the modification of OF standards is avoided. Here, the OF controller uses the OF *features* message (as discussed in section 2.2.1.1). Furthermore, in order to allow the MN to operate as a standard device in environments where the network has no OF controller, here MN's OvS bridges are configured in "standalone" fail-safe-mode and with a low priority "NORMAL" action. In addition, a higher priority action is set to allow the trigger for the handover to be sent when a controller is available.

Finally, as mentioned above, the handover procedure is handled by an SDN/OF controller (i.e., network controller or vMN) jointly with a developed application. This application is a behavioral example that can be enabled in the controller by the framework, but the design does not impose a mandatory procedure. Instead, it leverages on the SB and NB APIs of the SDN Controller allowing other network entities to interact with the controller and define the handover execution, increasing the flexibility of the solution. Next, the context acquisition and the handover procedure are discussed.

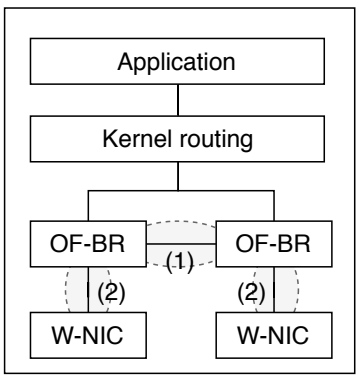



**Figure 3.3:** SDN-based mobile node overview. Connections in the figure: (1) patch port; and (2) physical port [22].

#### *3.1.2.1 Wireless Context Acquisition*

As present in section 2.2.1, works such as [36], [37], applied SDN-based mechanisms to the PoAs for acquiring wireless context (e.g., attachments and detachments of MNs). Here, the enablement of SDN-based mechanisms in end-devices not only allows flow-based handovers without additional mobility protocols, but also allows the network to acquire wireless context of the MNs in order to explore different signaling opportunities from both networks' PoAs and from the MN itself.

In this context, the MN can use OpenFlow not only to receive commands from an OF controller (not necessarily the network controller), but actually to contribute with information about network conditions from the MN's point-of-view, to further optimize network control and mobility decisions. However, in order to enable the MN to notify the controller towards potential network optimization, the MN needs to be capable of perceiving its network surroundings as well as its own traffic, to possibly inform the controller what is the problematic flow. Otherwise, keeping all monitoring in the network reduces not only the control signaling over-the-air, but also the consumption of MN resources, while enabling other metrics to be explored, such as the detection of the PoA overload, or even the knowledge of the number of users connected to the same PoA. However, the complexity of scalable solutions increases with the number of attached users. Table 3.1 features signaling opportunities for acquiring network context that can be coupled to OF messages for both the MN and PoA perspectives.

Finally, to enable the collection of such information and package them in OF event packets, applications (in both Bash and C) were developed for the MN and PoA. Note that these applications were developed for proof of concept and illustrative simplification, since the design of the framework aims to be generic and deployable independently of the underlying wireless access technology by providing the means to integrate with existing or novel monitoring methods. The applications are available as open-source tools in the GitHub repository.

**Table 3.1:** Signaling Opportunities [11].

| Network entity | Signalling opportunities | Disadvantages |
|---|---|---|
| MN | • Signal noise/strength;<br>• Packet loss detection;<br>• Discover connection to better networks; | • Higher MN complexity;<br>• Higher control communication delay;<br>• More control signalling over-the-air; |
| PoA | • Overload detection;<br>• Number of connected users;<br>• Hardware/ Link failure; | • MME (or, OF controller) loses MN's perspective;<br>• *Ping-pong* effect possibility; |

#### *3.1.2.2 Flow-based Handover*

As stated in section 2.2.1.1, OF messages can be used to perform a flow-based handover, offloading the specific flows from one wireless network interface of the UE to another. The handover procedure is illustrated in Figure 3.4, and its signaling explained as follows. Please note that MN and UE are used interchangeably along the document.

- **Packet_in**: It is used as the handover trigger and carries information such as the current active flows on the interface and/or the flow that should be offloaded.
- **Flow_modification (or flow_mod)**: If the *packet_in* results in an handover decision, the controller redirects the flow through flow-based rules via OF *flow_mod* messages for each MN interface involved in the handover. The first flow-based rule adds in the MN's OvS bridge a flow entry, which redirects the matching traffic from a specific port (i.e., the patch port) towards the network via the new transmitting wireless interface. In order to perform this forwarding, this rule should implement the *"MOD_SRC_ADD"* and *"MOD_DST_ADD"* OF actions, to fulfill the wireless handshake. Both actions change the source and destiny MAC address, respectively.
- **Barrier_request and barrier_reply messages**: After the first *flow_mod*, a *barrier_request* is sent, ensuring that the second *flow_mod* is sent only when the first is already executed, avoiding packet loss or *ping-pong* effects during the handover. Therefore, mimicking a *make-before-break* approach.
- **Flow_modification (or flow_mod)**: The second OF *flow_mod* message redirects the matching traffic towards the bridge responsible of the new transmitting network interface. Since the first flow-based rule is already implemented (which forwards the traffic towards the network), after implementing this rule the UE starts transmitting through the new interface.

The intelligence of this handover procedure is the SDN controller, and as such, an application was developed in order to allow the controller to adopt such behavior upon specific triggers. In this line, an application for the POX controller and another for the Ryu controller were developed. Despite both applications having the same logic, controller specific libraries required dedicated applications to be developed. Upon the attachment of a SDN-enabled device to the network, the SDN controller uses the OF *features* message to discover its proprieties and verify if it is a MN.

Upon a handover request trigger (sent as OF *packet_in* message), the controller generates an interruption, and after analyzing the source of the packet, starts the procedures to perform the handover. As described above, in a *make-before-break* approach, the first flow-based rule is sent to the new transmission bridge, and immediately after, OF *barrier* messages are exchanged, ensuring that the first flow-based rule is implemented

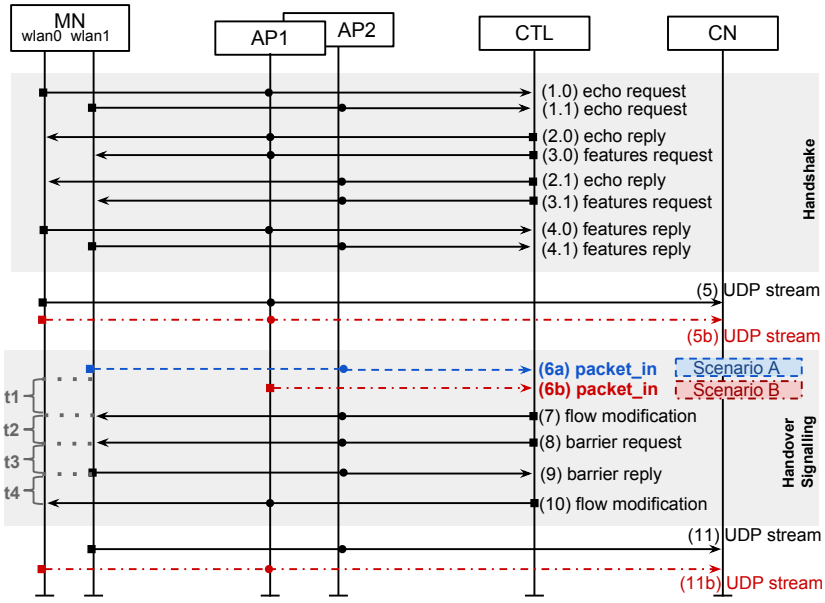



**Figure 3.4:** Handover procedure high-level message sequence [11].

before sending the second one. Similarly to the *packet_in*, the *barrier_reply* message, sent with the same ID as the *barrier_request*, will cause an interruption that motivates the sending of the second flow-based rule. Finally, specific aspects such as how the controller is aware of MN network attachments were also explored with applications being developed for the MN and PoA, exploring SDN SB and NB APIs. This will be further detailed in section 3.1.4.

### *3.1.2.3 OpenFlow Signaling Impact*

The handover procedure signaling is responsible for 550*bytes* (5 OF messages). As presented in section 3.1.2.1, the handover can be triggered from different network entities, namely the MN and PoA, *Scenario A* and *Scenario B* of Figure 3.4, respectively.

In *Scenario A*, these 5 messages are handled by the MN, while in *Scenario B*, the MN only handled 80% (i.e., 416*bytes*) of them, with the remaining 20% being part of the PoA signaling. This message is the handover request (i.e., the OF *packet_in* message), which has an impact of 134*bytes* (24% of the handover signaling). As such, *Scenario B* contributes towards a reduction of the overhead introduced in MN by the handover signaling. Notwithstanding, the major impact of the OF signaling regards to *keep-alive* messages exchanged every 5*s* between the controller and OvS (by default). In this context, in a 40*s* experiment, the MN exchanged $2422 \pm 48$*bytes*. These *keep-alive* messages have special impact on the MN since each bridge requires an OF channel and, therefore, the number of keep-alive messages is duplicated. Nevertheless, this overhead can be reduced by defining a higher period for such messages.

### 3.1.3 Points of Attachment virtualization

This section addresses the PoA virtualization, presenting different approaches and comparing them with legacy solutions. 5G mobility communication systems need to go beyond mobile network technologies, instead providing mobility management and control procedures in a technology-agnostic way. In this line, the vPoA represents the virtualization of PoA management and control services (e.g., authentication, association and Dynamic Host Configuration Protocol (DHCP)) in the cloud, exploiting resources therein to regard PoAs as virtualizable network functions that can be dynamically instantiated according to utilization needs. As such, the vPoA can be seen as working on behalf of the PoA, able to fully offload its operating mechanisms (Figure 3.5b) or only selected services and applications (Figure 3.5c). Figure 3.5 compares the different deployments.

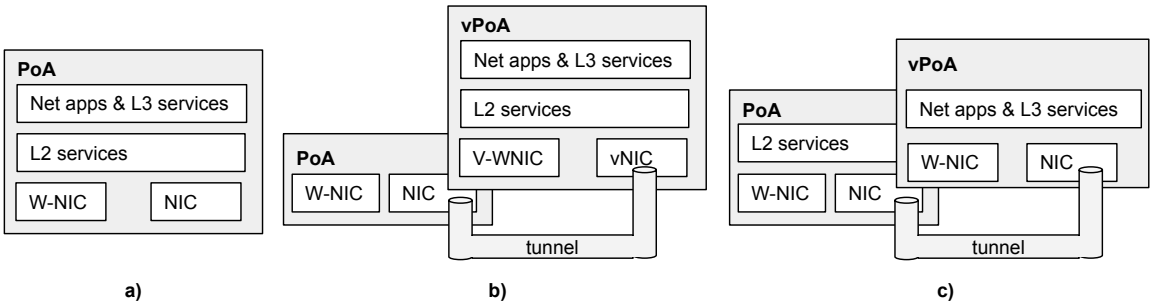



**Figure 3.5:** PoA high-level architectural overview for: (a) regular Wi-Fi PoA (without virtualization); (b) fully virtualized PoA; and (c) partially virtualized PoA [14].

Note that despite the framework of this chapter leverages PoA virtualization, the contribution is focused in the virtualization of the MN and the usage of SDN-mechanisms into the MN for enhancing mobility management for heterogeneous networks. Thus, here the candidate implements and evaluates different PoA virtualization strategies found in the literature for further integration in the final framework.

#### *3.1.3.1 Mobility enhancement enabled by the Point of Attachment virtualization*

PoA virtualization brings a trade-off between delay and both hardware and wireless mobility simplicity. By virtualizing the PoA in the cloud (i.e., vPoA), the physical PoA becomes a bare metal device, but creates a longer control and management path between the attached device and the vPoA. Table 3.2 summarize the deployment differences.

In a regular Wi-Fi (Figure 3.5a) network, where virtualization is not considered, the AP deals with all wireless management and control frames (e.g., authentication, association and probes), while at the same time manages L3 connections and services. Thus, current Wi-Fi AP configurations are simple, but in large scale environments,

complexity increases substantially. When compared with virtualization approaches illustrated in Figure 3.5, discarding processing time, it is expected that the "fully virtualized" approach has a higher delay for both L2 and L3 attachment, since besides the Wi-Fi round trip time (RTT), an Ethernet RTT is added. As such, the "partially virtualized" has a lower delay, since the Ethernet delay is inputted only for L3 messages. A regular AP deployment should have the lowest L2 and L3 attachments delay. Conversely, results presented in Figure 3.6 shows that the "partially virtualized" approach registered the lowest values of attachment delay (discussion in section 3.1.3.2).

Under a L2 perspective, virtualizing the PoA in the cloud enables traffic management to be offloaded to the vPoA, decoupling the intelligence from the hardware and deploying it in any point of the network. This facilitates handover management both for L3 and L2, since the attachment is processed in the virtual instance instead of the physical node. As such, a vPoA can be transmitting/receiving management frames from different physical nodes, and a handover between two physical PoAs does not necessarily mean an L2 or L3 handover, since they may be connected to the same vPoA. CloudMac [48] and Odin [47] identified that AP virtualization into a data-center improves intra-technology mobility at a cost in bandwidth and attachment delay. In contrast, physical PoAs become simpler, while vPoAs had increased computational power. Despite MEC scenarios where this approach may be well suited [89], wireless networks may experience performance issues due to the delay between the physical PoA and the corresponding vPoA. At the cost of losing seamless L2 handovers, this is overcome by keeping the L2 attachment in the physical equipment and virtualizing only L3 services (e.g., DHCP). In this case, the PoA handles L2 connectivity, allowing MN attachment and remaining L2 communications to be managed therein. Otherwise, L3 services are migrated to the cloud. Considering a handover in a Wi-Fi network, where a MN switches the physical AP, a new L2 connection is required. However, the MN may keep the L3 connection, facilitating handover management.

**Table 3.2:** Comparison of Wi-Fi access point approaches [14].

| | **Advantages** | **Disadvantages** | **Mobility impact** |
|---|---|---|---|
| Regular AP | Simple implementation; | Manual reconfiguration; | Complex mobility management;<br>Dedicated mobility protocols |
| L2 vPoA | Simpler network equipment; | L2+L3 attachment delay<br>Lost performance due to the RTT delay | Seamless L2+L3 handover |
| L3 vPoA | Simpler network equipment;<br>Reconfiguration on the fly; | L3 attachment delay | Seamless L3 handover |

#### *3.1.3.2 Evaluation of PoA virtualization strategies*

This section provides an experimental overview of different vPoA strategies (presented in Figure 3.5), using an Wi-Fi AP in Open System Authentication (OSA) as example. As such, hereafter the PoA and its virtual entity are presented as AP and vAP, respectively. The MN's network (re)attachment delay and bandwidth impact are studied.

Physical nodes (i.e., MN and PoAs) were deployed in single board computers with a 64-bit support AMD APU CPU with 1GHz dual-core and 2GB DRAM, while the corresponding virtual instances (i.e., vMN and vPoAs) deployed in an in-house data-center, using a VM with 1 vCPU and 2GB RAM. In both types of network nodes, Ubuntu 64-bit 14.04 was used. Following the presented architecture of Figure 3.5 in strategy: a) hostapd and dhcp-server were deployed in AP; b) hostapd and dhcp-server were deployed in vAP; and c) hostapd was deployed in the AP and dhcp-server in the vAP. As proposed by CloudMAC [48], to move the management capabilities (e.g., attachment of a user equipment) of the AP to the cloud (i.e., strategy b, fully virtualized PoA), the vAP was coupled with an enhanced *mac_80211_hwsim*, which emulated a wireless interface and the OpenFlow tool, Capsulator[1], was used to create a tunnel between the wireless monitor interfaces of both entities. For the partially virtualized PoA (strategy c), a similar approach was used, and a tunnel using the Capsulator tool, was instantiated between both the AP and the vAP.

Figure 3.6 compares a regular AP with different vAP deployments in terms of attachment delay and bandwidth. The "L2 attachment" regards to messages of an Wi-Fi OSA, while "L3 attachment" regards to DHCP messages. As a trade-off for hardware simplicity and L2 handover management, the fully virtualized (i.e., "L2 virtualization") approach has the worst result, with a delay increase of 50% for a L2 attachment and DHCP IP address. This is mainly due to the increase of RTT and overhead caused by increased forwarding of wireless management frames. This is reflected in the bandwidth, which decreased by 14%. By offloading only L3 services, not only the bandwidth value was re-established, but also the DHCP delay was improved, while maintaining the L2 attachment delay. This may be due to forwarding overhead decrease (since Wi-Fi management frames are processed in the AP), allied with the increased computational power of the vAP.

[1]Capsulator: http://archive.openflow.org/wk/index.php/Capsulator

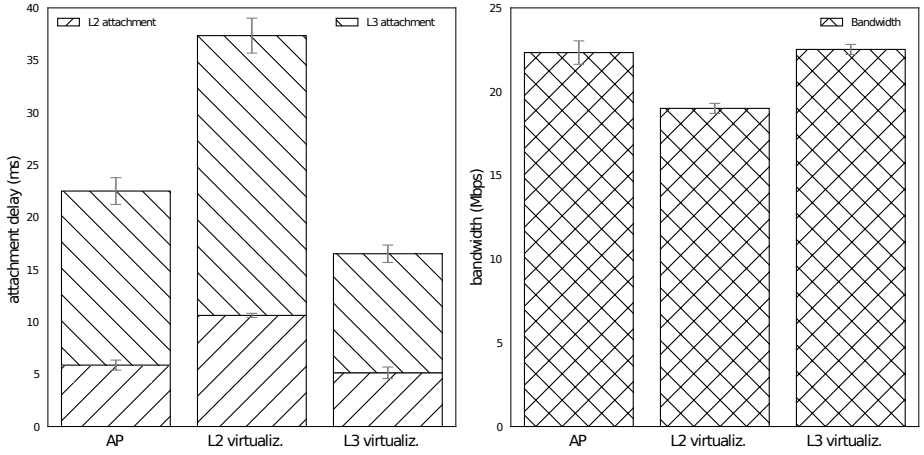



**Figure 3.6:** Attachment delay and bandwidth for the deployed AP approaches [14].

### 3.1.4 Integrating vMN and vPoA for bringing user context to the cloud

This section addresses end-node virtualization (as presented in section 3.1 and illustrated in Figure 3.1), presenting a framework for bringing user context to the cloud, while including the SDN-enabled MN (section 3.1.2) and PoA virtualization (section 3.1.3). This enables information about the MN's link to be collected in the core network by its virtual representation (i.e., vMN), enabling the Controller to adapt the network to each type of access technology. Current smartphones allow accessing online services while moving from one wireless technology to another. MN virtualization implies the virtualization of its network context into the cloud enabling the vMN to perform management decisions on behalf of the MN: the vMN not only allows to anchor the MN when offloading traffic from one wireless PoA, but actually allows to deploy a virtual representation on other devices (such scenario is further discussed and evaluated in section 3.2).

The interaction between the vMN and its physical counterpart allows the provisioning of information about the connectivity status of the latter. Thus, the network controller (which is involved in the instantiation of the vMN and its traffic flows configuration) gets feedback on the physical link conditions of the MN, allowing optimization. Moreover, the fact that the physical MN is connected to its virtual counterpart, means that all traffic associations between the vMN and other communication entities can be handled inside the data-center, minimizing handover control signaling. Moreover, the MN's perspective information, such as a list of detected neighboring network cells, their link quality, or even application and user preferences, can be sent towards the network, which can use it to enforce traffic optimization and policing, and to optimize mobility management. An example of such scenario, would be the detection that the current link does not have enough signal strength to support a video call, and thus trigger a

flow handover allowing the video flow to reach the MN via a different interface (i.e., Wi-Fi) with better conditions, while maintaining the audio in the original interface of the call. Moreover, despite acting as an anchor for the mobility process, the vMN also acts as a proxy for the Internet.

As a result, the framework architecture presented in Figure 3.1 can be updated to Figure 3.7, presenting how data-flows are routed in the architecture.

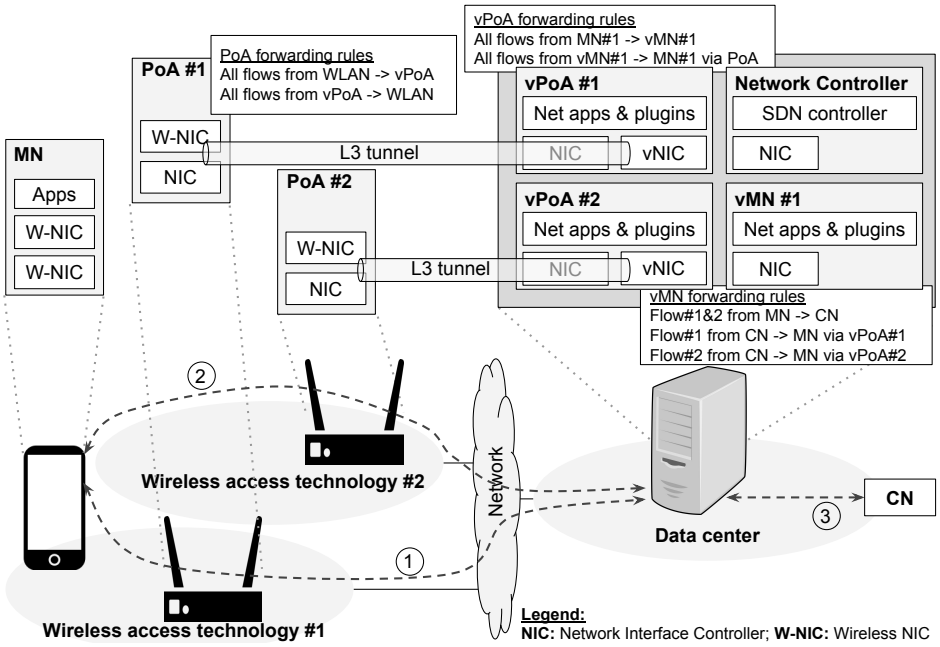



**Figure 3.7:** Framework architecture overview and forwarding rules [13].

### *3.1.4.1 User context transfer in the cloud*

This section explains the control and management communication of the framework. Binding the MN to the vMN via OpenFlow enables both entities to directly interact with each other. While the MN exploits the OF protocol to provide its own perspective of the network (i.e., provide context about the wireless link) and assist the network connectivity selection, the vMN manages the data traffic of the MN through the implementation of flow-level actions, such as packet redirection between mobile interfaces for an offloading scenario. For the vMN to assist in managing the MN, a mobility management application was developed to be used along with a SDN controller (e.g., POX and Ryu). This application stores the information regarding to the MN activity such as active links, MAC addresses and current flows. As described in section 3.1.2.1, such information is acquired via OF messages and is further used for handover procedures directly over the MN or to assist the SDN Controller for network optimization. Figure 3.8 illustrates control interactions among the framework entities, where different approaches can be considered for context updates (e.g., active links and neighbor cells):

*Case A)* event trigger in the vPoA: the PoA/vPoA monitors the different attached MNs and periodically updates the controller (or when requested), which updates the vMNs;

*Case B)* event trigger in the MN: the MN monitors the wireless medium, sending updates to the controller via direct OF control communication between both MN and vMN (SDN Controller is no longer an intermediate). For this, the MN can acquire the vMN's IP address via DHCP and it can use OF messages and events, as presented in section 2.2.1.1 and summarized as follows:

- *Features*: to provide initial information to the vMN (e.g., available wireless access technologies);
- *Packet_in*: for signaling specific events by MN towards vMN (e.g., handover opportunities);
- *Flow_modification*: to update the MN's OF flow-table for source-mobility scenarios;
- *Port_status:* indicate MN events to the vMN (e.g., link up/down).

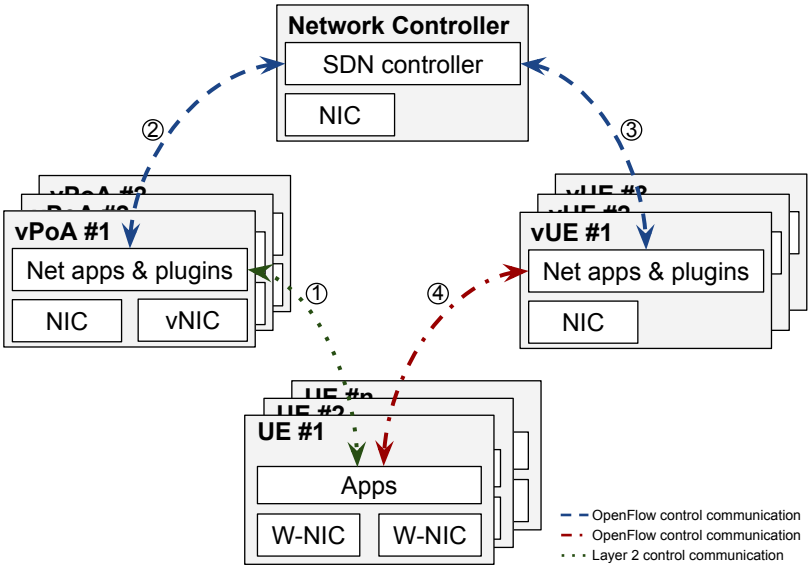



**Figure 3.8:** Control communication [14].

### *3.1.4.2 Framework procedures*

Similar to section 3.1.2.1, during the MN attachment, the PoA informs the controller of a new connection, using an OF *packet_in* message. MN attachment information is sent to the SDN controller, registering the necessary information for network (re)configuration. By saving the MN's MAC address and PoA, the controller verifies if the MN is already associated to an existing vMN or if a new one is required. Figure 3.9 depicts this process, which is described as follows.

#### *3.1.4.2.1 MN's attachment and vMN's instantiation*

When the MN attaches to the network (1), this attachment is performed in cloud and generates an event in the vPoA, triggering an OF message towards the Controller (2) with the MN's identification (i.e., the attached MAC address). Receiving the vPoA's message, an event is generated in the Controller, which verifies the existence of a vMN or if a new one is required (3). After the vMN's instantiation, the vMN establishes an OF connection with the SDN Controller, and informs the later, that it is ready to be added to the network (4). Thus, the Controller updates the flow tables on the vPoA (5), in order to redirect the MN's traffic for the respective vMN.

#### *3.1.4.2.2 MN's context update*

If by receiving the vPoA's message (2), the generated event in the Controller results in an already existence of a vMN, the Controller updates the context of both vMN and vPoA (5). Moreover, the vPoA monitors the link conditions of the MN via link layer messages. Changes are informed to the SDN Controller via OF messages (events, *case A*), which in turn updates the vMN's context. Specific MN-initiated events can also be communicated to the network (**case B**), with the MN sending a OF message towards the vMN (6-7). Note that this update (such as, MN's wireless connections) can be performed in two ways (i.e., *Case A* and *Case B* presented above).

#### *3.1.4.2.3 Handover signaling*

The framework uses the procedure presented in section 3.1.2.1. Thus, the OF control is extended to the MN and the handover is performed using a standard-based API for heterogeneous flow mobility management, contributing to a more homogeneous network management procedure, in terms of used protocols. Nevertheless, the procedure defers depending on the flow direction:

- *Downstream:* if the trigger is sent from the vPoA to the Controller, the latter verifies the network state and, after deciding which MN should be offloaded, updates the flow-table (9) with the IP address of the MN new receiving interface. For an MN-initiated mobility, the trigger is sent to the vMN, which updates its flow table. Besides OpenFlow, the MN can send the trigger via an User Datagram Protocol (UDP) packet, which is converted into an OF message in the vPoA, and sent to the Controller (as in scenario of section 3.2).
- *Upstream:* When the trigger is sent from the vPoA towards the Controller (10b), the latter verifies the network state, and informs the respective vMN. If the trigger is sent from the MN (10a), the message is sent to the vMN, responsible for

choosing which flow will be offloaded (11-14). In addition, an OF *barrier* message (12-13) is used to verify the correct reception of the first flow-rule, preventing packet loss due to the handover procedure (as discussed in section 3.1.2.2).

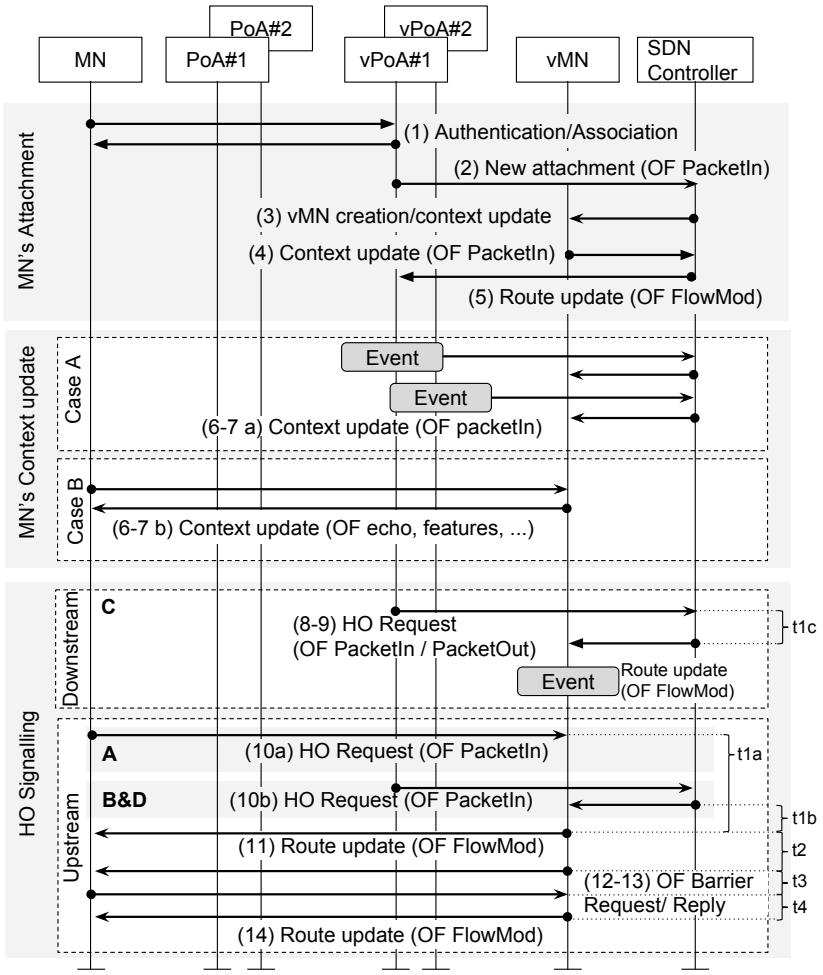



**Figure 3.9:** Control high-level signaling [13].

### *3.1.4.3 Proof-of-Concept evaluation of the framework*

This section implements and evaluates the architecture illustrated in Figure 3.7. The framework addresses SDN, NFV and cloud principles for abstracting mobility management procedures in heterogeneous environments. It goes beyond the state-of-the art, by considering the virtualization of the MN allowing the controller to reduce direct interaction with the end-nodes, interacting instead with their virtual counterparts. Gains can be expected here, particularly when the Controller and the virtual nodes representations are in the same data-center. Moreover, each vMN manages the wireless state of its respective MN, alleviating the Controller load.

The framework used the AMazING testbed for deploying the physical network entities PoA, CN and MN. Otherwise, the vMN, vPoAs and the SDN Controller were

deployed in VMs in an OpenStack data-center. The AMazING nodes were equipped with a 64-bit 1 GHz dual-core AMD APU CPU and 4GB of DDR3-1066 DRAM, while the OpenStack VMs were instantiated with 1 vCPU and 1GB of RAM, running Ubuntu 64-bit 14.04 LTS. Finally, OvS (v2.5.90) enabled to (re)configure the network on the fly via OF flow-based rules. Depending on the entity that triggers the handover (e.g., vPoA or MN), different handover initiation procedures can be done. Although not limited to, the framework has focused on the following ones, illustrated in Figure 3.9:

- *MN-initiated handover (MIHO):* The MN verifies its current connection with the PoA, and sends an handover request directly to its vMN, upon discovery of a PoA with better signal strength.
- *Network-initiated handover (NIHO):* The handover trigger is executed in the network, reducing the signaling over-the-air. Pre-configuring triggers in the vPoA (e.g., attachment, signal level to the MN and overloaded bandwidth), the vPoA sends a handover request to the Controller, which alerts the respective vMN.

The following evaluated scenarios are directly related to the use case scenarios and signaling presented in Figure 3.9. Videos were streamed using VLC[2] software via Real-time Protocol (RTP). The experiments were run 10 times, with their average being presented with a confidence interval of 95%. Figure 3.10 shows the throughput for each scenario, where "*original video*" is a calculated optimal video flow necessary to avoid any packet loss while transitioning from PoA1 to PoA2. As such, the proposed framework's performance is compared with this flow, to see if it is able to maintain the expected throughput. Otherwise, Figure 3.11 presents the measured handover delay (in the MN and the vMN) for the evaluated scenarios. Note that the MN's handover delay cannot be directly compared with vMN's delay, since it was accounted in different time windows.

#### *3.1.4.3.1 Scenario A (MIHO for upstream traffic)*

In Figure 3.10a), the MN was initially streaming a video[3] towards the CN via PoA1 (Figure 3.7- 1). As the MN moved away from PoA1, the link strength and quality decreased, and as such the MN issued a handover request at $25s$ (Figure 3.9- 10a). Once the handover procedure was completed (Figure 3.9- 11/14), the video stream was sent towards the CN (Figure 3.7-3) via PoA2 (Figure 3.7-2). The handover occurred when PoA1 had $-60dBm$ of signal strength against the $-20dBm$ of PoA2 (such thresholds are configurable). After the handover, the MN kept streaming towards the CN, but

[2]VLC: https://www.videolan.org/
[3]Caminandes 3 (video 1): http://www.caminandes.com/

via PoA2. Also, the flow offloading to the new PoA did not result in interruptions nor losses to the upload data rate of the video.

#### *3.1.4.3.2 Scenario B (NIHO for upstream traffic)*

In this scenario, the vPoA monitored the MN's link strength and prompted the Controller for a network optimization. Verifying the origin of this message, the Controller notified the correspondent vMN (Figure 3.9- 10b), which in turn proceeded the handover process. Figure 3.10b) illustrates the operation of the proposed mechanisms, without packet loss during the video transmission. Here, the handover request was sent by the vPoA1 at 25$s$. As in the previous scenario, the proposed mechanism was able to maintain data rate. Also, both scenarios $A$ and $B$ had similar traffic performance, with PoA1 registering 45% of the total video traffic, while the AP2 registered the remaining 55%.

#### *3.1.4.3.3 Scenario C (NIHO for downstream traffic)*

In Figure 3.10c), the MN was initially attached only to the PoA1, receiving a video stream via mobile network (Figure 3.7- 1 and 3). When the MN attached to the Wi-Fi network (i.e., PoA2), the vPoA2 triggered the controller (via OF message), which notified the vMN (at 25$s$). In turn, the vMN offloaded the video stream from the mobile network (vPoA1, Figure 3.7- 1) to vPoA2 (vPoA2, Figure 3.7- 2) via OF flow-based rules, reducing the video transport cost in the mobile network. The handover procedure did not result in packet loss, with the traffic towards the MN being offloaded to the new wireless access interface seamlessly, since the process occurred in the cloud (i.e., in the vMN), with the deployment of the flow-rule in the vMN instantly redirecting the flow towards the new MN interface. As such, the vMN receives the full traffic video from the CN and redirects it to the MN via the more appropriate wireless network. Finally, while PoA1 transmitted 45% of the video traffic, PoA2 transmitted the remaining 55%.

#### *3.1.4.3.4 Scenario D (NIHO for upstream traffic balancing)*

This scenario emulates a PoA overload. Thus, the MN simultaneously streamed two videos, which caused the PoA to overload and consequently originating packet loss. For this, the MN started streaming a video (video 1) towards the CN via PoA1. At about 9$s$ of experience, the MN started a second video[4] (video 2) streaming, also via PoA1. Receiving both video streams, PoA1 got overloaded, and informed the SDN Controller. Similar to the previous scenario, the SDN Controller notified the vMN, which in turn offloaded the flow of video 1 (at 24$s$). As such, the MN kept streaming, video 2 via

[4]Big Buck Bunny (video 2): https://peach.blender.org/

PoA1 and video 1 via PoA2, towards the CN. In this way, only one stream was moved to the new vPoA, performing a flow-based mobility. In Figure 3.10d) the video traffic received by the CN is compared with the traffic sent from each PoA. The stream analysis shows that if the streaming of both videos was kept in via PoA1, the lost would be about 55.80% of the remaining video packets from video 1, against 12.80% from video 2. Also, the handover process did not result in packet loss nor sequence errors, keeping both videos' original data rate, and revealing the feasibility of the proposed procedures. Finally, while PoA1 accounted for 75% of the total video traffic, PoA2 registered the remaining 25%.

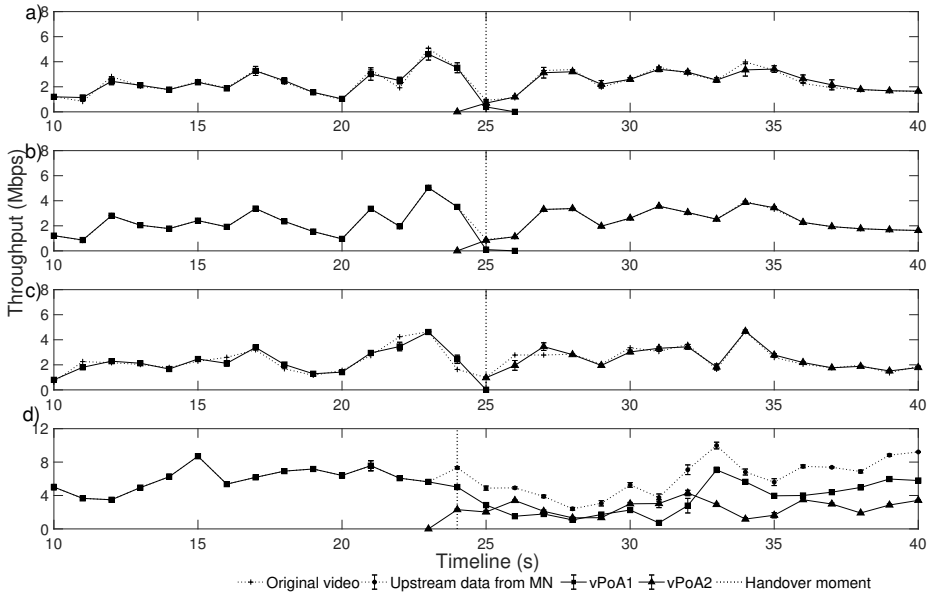



**Figure 3.10:** Video throughput for: **a)** Scenario A; **b)** Scenario B; **c)** Scenario C; and **d)** Scenario D [13].

#### *3.1.4.3.5 Handover delay for upstream scenarios (i.e., A, B and D)*

For the handover delay measured in the vMN, the MIHO (i.e., *Scenario A*) had better performance than NIHO (i.e., *B* and *D*). This occurred because, in MIHO scenarios, the MN notified the vMN directly, having one less control decision. In Figure 3.9, it is noted that for NIHO, the request was done from the vPoA to the SDN controller, which in turn notified the vMN. Thus, despite increasing the handover delay in vMN's point of view, NIHO scenarios reduce the handover delay in the MN's point of view in 50%, since the notification process becomes transparent for the MN. In this line, the handover delay in the MN was measured from the moment that it notices the handover, which only occurred when the first flow-rule (Figure 3.9- 11) was received.

#### *3.1.4.3.6 Handover delay for downstream scenarios (i.e., C)*

The handover delay in the vMN was measured from the moment that it received the notification of the Controller until the first redirected flow ($36.28(\pm 5.44)ms$). However, since this process was transparent to the MN, the latter only noticed the handover when it stopped receiving from one wireless interface and started receiving from the new one ($t5$). As such, the handover was seamless for the MN, and its delay ($13.65(\pm 3.46)ms$) was measured between the last video packet received from the initial wireless interface (i.e., before handover), and the first packet received by the new interface (i.e., after handover). However, the real handover time may be smaller, since the measured value depends on the time period between two consecutive packets of the video stream. This results from the fact that the mobility process occurs in the cloud between the SDN Controller and the vMN, with the implementation of the flow redirection in the vMN instantly redirecting the flow towards the new MN interface.

As a global note, the increase of data traffic in the vMN negatively affects the control communication, by increasing the handover delay in about 13%. This is due to the fact that control traffic was not prioritized over data, therefore competing for the same resources. However, this issue may be mitigated by separating both data and control traffic. As such, the vMN would deal with the control traffic, while an OF switch could be instantiated in order to create a path for the data traffic. This strategy was adopted in the framework proposed in chapter 4.

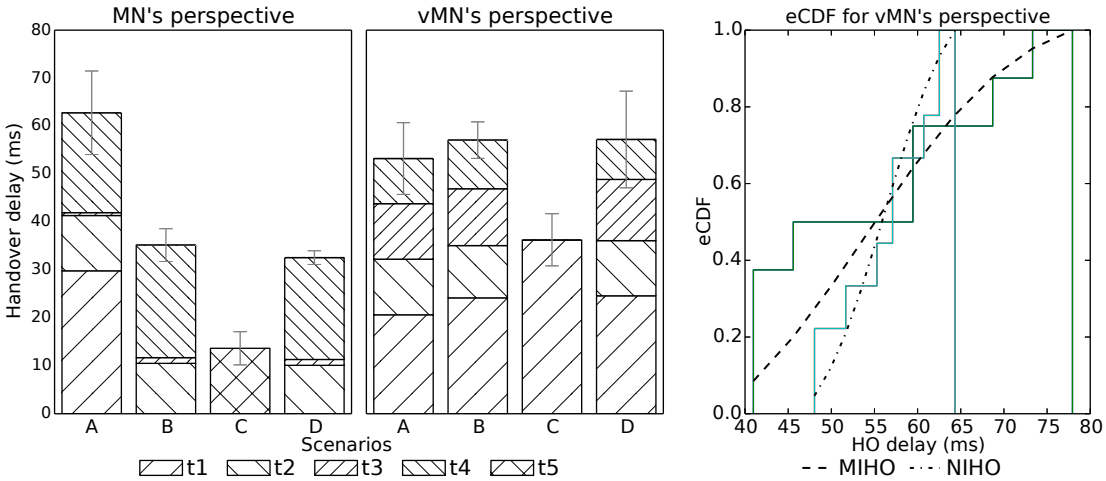



**Figure 3.11:** Handover delay for evaluated scenarios [13]. Time intervals (t1-t5) are related to signalling points depicted in Figure 3.9.

## 3.2 Towards a deviceless communication

This section further progresses the concepts introduced in the previous sections and applies them to a new framework that provides a deviceless communication approach, where data and media flows reaching a user can be individually shifted into nearby devices. To support this, the framework presented in the previous section enhances SDN and NFV concepts, allowing the opportunistic utilization of nearby devices as the user moves, while still being perceived as a single end-point towards external entities. Figure 3.12 presents the conceptual context and location infrastructure. Also, an experimental validation scenario is illustrated in Figure 3.13, showcasing a video stream being delivered to a nearby large TV screen, allowing the user to watch the video while answering a voice call.

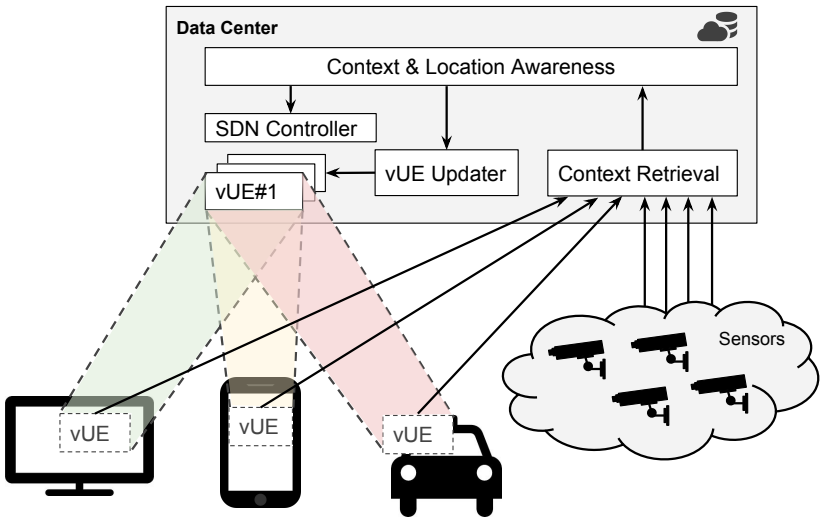



**Figure 3.12:** Conceptual framework for a deviceless communication [15].

### 3.2.1 Involved entities and their integration with virtualization concepts

The framework gathers information of the user's surrounding environment, and dynamically updates the virtual UE (vUE), allowing the user to communicate in scenarios where the user has only a localized probing device, by instantiating the user's context in nearby devices (e.g., smart TVs and smartphones). Nevertheless, given the privacy aspects of a generic location and context architecture, for a proof-of-concept prototype, the framework was simplified and a smartphone was used as UE and localized identifier, probing devices that listens the user's surrounding environment and retrieves the relevant information (such as, nearby devices and re-routing opportunities) directly to the vUE. In this way, nearby devices choose which information to share, and the user fetch permitted devices.

#### *3.2.1.1 User Equipment and its virtualization*

Nowadays smartphones' capabilities (represented in the proposal as the UE) allow us to explore scenarios such as accessing a video from Youtube[5] or Netflix[6], while jumping from one wireless technology to another (e.g., from LTE to Wi-Fi, or vice-versa). Here, the devices have a virtual representation on the cloud (vUE), and a context retrieval infrastructure collects data from sensors and devices, about all relevant context for the user and its nearby devices. This is passed to a *Context & Location Awareness Engine* that will control the flows for the vUE, and will perform a request to a *vUE updater* if required. The *vUE updater* will dynamically profile the vUE as the user context changes. This enables the user to keep the context of the UE in the cloud for further usage in different equipment with networking capabilities (such as, smart TVs). As such, the user is no longer attached to an unique UE, being able to explore communication scenarios where the user is not "carrying the device". To this end, while the UE represents a current dual-interfaced (i.e., mobile and Wi-Fi) smartphone, the vUE is created on-demand by the network Controller and consists on a dynamic representation of the UE in the cloud. Also, as presented in section 3.1.4, the vUE acts not only as a proxy for the Internet access, but also as an anchor for flow mobility procedures. As such, cross-technology handovers become transparent for both end-nodes, since both continue to send/receive the traffic to/from the same entity (i.e., vUE) regardless of the physical device that eventually terminates the flow.

#### *3.2.1.2 Smartphone context application*

To enable the smartphone to trigger an offloading event upon reception of a voice call, an Android application was developed and installed therein. The purpose of this application is to signal the network when the user is accessing an online video and simultaneously answering a call. As such, the UE is aware of nearby TVs, by constantly reading bluetooth signals from beacon devices[7] attached to them. The distance estimation between the UE and the TV is based on the received signal strength of each beacon device. When the UE is in a call and watching a video, if it detects a TV, it informs the network about the handover opportunity. Note that for a generic implementation of a deviceless communication environment, the UE would become a powerful device discovery probe. When the application detects that the user answered an incoming call and if there is, at least, one TV in range, a trigger is sent to the network towards the vUE, identifying the nearest TV. When the mobile voice call is ended, the application triggers the vUE to retrieve the video back to the UE. Finally,

[5]Youtube: `https://www.youtube.com/`
[6]Netflix: `https://www.netflix.com/`
[7]Bluetooth beacon device: ByteReal TagBeacon 2.0

while this application is running in the background, the user is visualizing the online video stream in the VLC application. Security (and authentication) aspects of these procedures (such as policy-based network decisions about which video flows can be offloaded to external devices or not) are outside the scope of this work.

#### *3.2.1.3 PoA and its virtualization*

Since the framework aims to be agnostic of the access technology, the BSs and APs are generically represented as PoAs. In this line, the vPoA represents the virtualization of the management and control services of the PoA (i.e., the fully virtualized PoA of section 3.1.3) in the cloud. Thus, for proof-of-concept implementation, the PoA/vPoA was implemented as an Wi-Fi AP/vAP, following the respective deployment presented in section 3.1.3, where the vAP is responsible for the all management traffic of the physical AP. Also, a *Link SAP* module to monitor the signal level of the attached MNs was developed and implemented in the vAP for monitoring signal strength and attachment/detachment, allowing it to perceive the MN's connectivity context.

#### *3.2.1.4 Network Controller*

As presented in section 3.1.1, the network Controller (i.e., SDN Controller) manages all network equipment in regards to flow mobility management, redirecting flows (via OF flow-based rules), and instantiating new network entities of recently attached devices (such as, vPoAs and vUEs) in the cloud. In this line, and framing a video re-routing context, the framework improves video QoE by implementing a technology-agnostic traffic flow mobility mechanism by bringing a virtual representation of both the UE and PoAs to the cloud. In this way, flow mobility management mechanisms are migrated to the data-center, benefiting the enhanced computational power, thus allowing the implementation of more complex management decisions, when the opportunity to redirect flows towards other devices arises. Regarding to UE's handover procedures, the controller maps the UE with the correspondent vUE, leveraging the handover of the flow to the selected vUE.

#### *3.2.1.5 Headend, Set-top-box and Correspondent Node*

In the simplified reference scenario (Figure 3.13), the video headend is used as a video server for streaming and transcoding a video. The set-top-box is connected to a TV and receives a small context of the UE therein, while the CN performs a voice call with the UE. Moreover, the set-top-box has bluetooth capabilities, allowing nearby devices locate it. The Set-top-box was deployed in a Raspberry Pi 3[8] connected to a TV, running the Raspbian Jessie. Finally, the CN performs the voice call with the UE over the cellular network.

[8]Raspberry Pi 3: `https://www.raspberrypi.org/products/raspberry-pi-3-model-b/`

### 3.2.2 Use case description

Figure 3.13 presents the simplified framework for the reference scenario, where a video is re-routed from the UE to a nearby TV (equipped with a set-top-box). When in its home network, the UE attaches to the Wi-Fi network and reads the bluetooth beacons sent from the set-top-box, estimating the distance of the nearest TV. Also, the Controller updates the necessary UE's context in the cloud (i.e., vUE). For simplification, in such scenario it is assumed that vPoAs are already instantiated in the cloud and connected to their physical counterparts (i.e., PoAs). Also, the implementation was simplified, and only reached the virtualization of the AP and UE, maintaining the cellular connection in its default configuration.

Figure 3.14 depicts the signaling regarding to this scenario. The scenario starts with the user visualizing a video stream in its UE via Wi-Fi network (Figure 3.14:1-2), when a voice call is received. Answering the voice call, the user triggers an event (Figure 3.14:3), originating a handover request towards the vUE. Upon reception of the handover request, the vUE updates the video stream destination, offloading it to the nearest TV (Figure 3.14:6). Ending the mobile voice call, another trigger is generated (Figure 3.14:7) notifying the vUE, which in turn redirects the video stream back to the UE (Figure 3.14:10). In this way, the user is able to keep visualizing the video on another device, despite answering the call on the mobile phone.

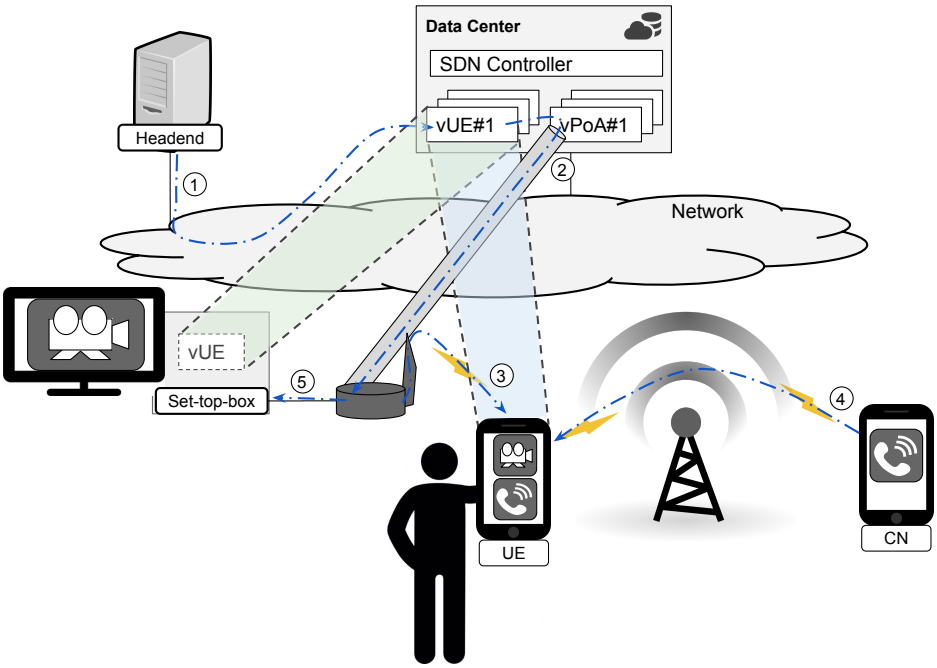



**Figure 3.13:** Overview of the deployed deviceless scenario [15].

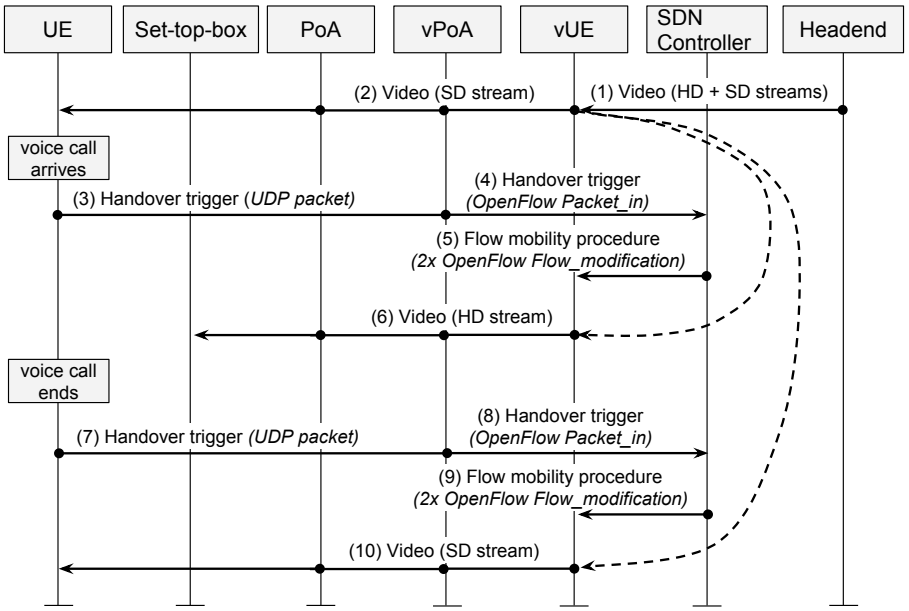


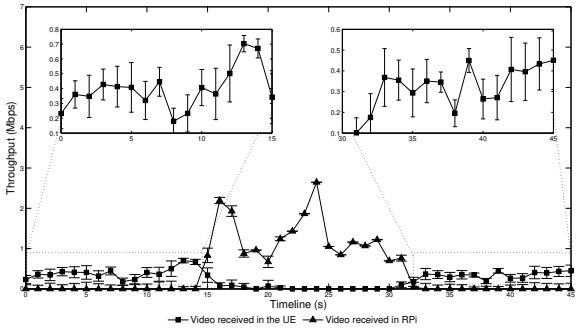

**Figure 3.14:** High-level message sequence of the deployed deviceless scenario [15].

### 3.2.3 Proof-of-Concept evaluation

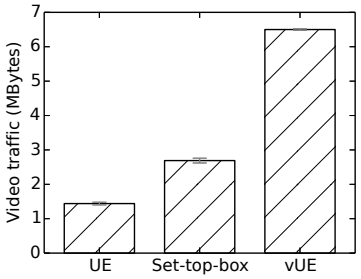
To experimentally evaluate the proof-of-concept prototype, the AMazING testbed was used to deploy the physical network entities PoA, video headend, and CN. A Nexus 5 smartphone with Android 6.0 was used as UE. The AMazING nodes were equipped with a 64-bit support AMD APU CPU with 1 GHz dual-core and 4GB of DDR3-1066 DRAM. The vPoA, vUE, and SDN Controller were virtualized in Docker containers (mimicking a data-center). The Docker software was running in a Virtualbox VM, with 1 vCPU, 4GB of RAM and Ubuntu 64-bit 16.04 LTS OS.

To evaluate the proof-of-concept scenario, the received and transmitted traffic was captured in the UE, vUE and set-top-box. Figure 3.15a shows the received video for the evaluated scenario. Initially the UE was accessing an online video[9] with Standard Definition (SD) quality (following the Android standards). As the UE received a voice call, the video switched to the nearest TV. Here, the video was upgraded to High Definition (HD) quality, which consequently increased the throughput. As the voice call finished, the video switched back to the UE with lower quality, decreasing the throughput to initial values. The scenario was evaluated during $45s$, with UE receiving the video in a total of $24s$ (i.e., $15s$ before and $9s$ after the voice call), and the set-top-box (connected to the TV) in remaining $21s$ . The flow mobility process to/from the UE was measured at the PoA in the time window between the offloading trigger and the first video data packet sent to the new destination, presenting a delay of $27.85(\pm 3.17)ms$.

[9]Caminandes 3: `http://www.caminandes.com/`




**(a)** Registered video throughput.




**(b)** Video traffic registered in each entity.

**Figure 3.15:** Video throughput over time and total video data amount for the deviceless scenario [15].

Figure 3.15b presents the amount of video traffic in the network, dividing it by the UE, set-top-box and vUE. The vUE received both streams (i.e., SD and HD), in order to be able to switch between them dynamically. While the UE received 22% of the video traffic, since the video transmitted to the TV is HD, the set-top-box received 41% of the video traffic, corresponding to 86% more video traffic compared to the UE.

Table 3.3 presents the signaling impact of the offloading procedure messages. As a framework that aims to operate independently of the physical layer, MAC headers were not considered. These messages had an impact of a $157bytes$ $(72 + 85bytes)$ in the UE for the trigger (i.e., UDP packets) and $325bytes$ in cloud entities (i.e., for the triggers to be sent as OF *packet_in* messages - $156bytes$ and $169bytes$, respectively). The flow mobility is completed after the implementation of two OF *Flow_modification* messages (i.e., the first to implement the new path and the second to remove the previous one) with an impact of $296bytes$ (i.e., $172bytes$ and $124bytes$, respectively) for the OF *flow_modification* messages). Despite triggers sent from the UE to update the context of its surrounding, the remaining flow management signaling and context triggers were retained in cloud. As such, while $917bytes$ for scenario management were accounted in the cloud, only $157bytes$ were sent over-the-air.

**Table 3.3:** Scenario signaling impact (bytes) [15].

| Trigger | type of message | impact on UE | impact on vPoA |
|---|---|---|---|
| arriving call | offloading opportunity | 72*bytes* | 156*bytes* |
| ending call | offloading opportunity | 85*bytes* | 169*bytes* |
| new UE attaching | context update | - | 142*bytes* |
| UE disattaching | context update | - | 142*bytes* |
| UE's Signal level crossing pre-established threshold | context update | - | 142*bytes* |

## 3.3 Chapter Considerations

This chapter addressed how the integration of software and virtualization mechanisms can enhance mobility management for upcoming 5G networks, by building systems with greater degree of flexibility and abstraction. In this context, this chapter discussed, implemented and evaluated three building blocks of the proposed framework: (i) SDN capabilities extension all the way to the UE; (ii) PoA virtualization approaches; and (iii) bringing the UE's context through virtualization.

As result, the proposed framework virtualizes both the UE and PoA, thus moving radio and core network functions, as well as UE's wireless context to the cloud. Here, the user context was brought to the cloud by virtualizing the UE therein and extending the OF control path up to the UE, allowing it to become part of the network control, assisting the controller in the mobility management by providing its perspective of the network conditions (such as, detected neighbor cells) and minimizing signaling delay. Also, the usage of SDN mechanisms within the UE, allowed for selective flow-based handover remotely controlled by the vUE. This was supported by proof-of-concept deployments and evaluations in experimental wireless testbeds.

Additionally, the proposed framework was evolved and deployed in a proof-of-concept scenario that provided the foundations to the concept of deviceless communication. This concept ultimately allows a user to be connected by any device on its surrounding. Here, and in order to avoid associated context privacy issues, this concept was simplified as a UE (smartphone) deciding which nearby device to use (for quality, energy, or convenience reasons). As such, here the interaction of the UE with other connected devices, such as TVs with network capabilities, is explored for selective flow handover, enhancing the user experience by redirecting data and content and taking advantage of the better features existing in such devices (i.e., larger screen).

Furthermore, the frameworks presented in this chapter provided successful indicators

of the benefits of a SDN flow-based mobility management (research question Q1), more specifically, the development of flow-based mechanisms agnostic to the wireless technology and without requiring additional and dedicated mobility protocols, thus contributing to a more homogenized network in terms of control protocols. Also, the introduction of NFV mechanisms (research question Q2) enabled the virtualization of both the UE and PoA, allowing control and management decisions to be performed in the cloud. Notwithstanding, wireless devices virtualization brings a trade-off between delay and both hardware and wireless mobility simplicity. The framework was implemented and evaluated considering different scenarios and network triggers, with results showcasing its feasibility allowing seamlessly performed flow-based handovers between different wireless network interfaces (research question Q3). Finally, this framework paved the way for a more distributed architecture design (research question Q4), by providing the foundations for a more flexible and dynamic network able to instantiate virtualized network entities in different points of deployment. Also, it enables the introduction of network slicing by provisioning the means for dynamically chain VNFs to perform a slice (discussed in chapter 4).

Finally, this chapter was written based in outcomes achieved through following publications authored by the candidate: book chapter [14], scientific journal [15] and international conference proceedings [11]–[13]. Moreover, the publication [11] was distinguished with a best paper award in the mobility track of the conference.

CHAPTER 4

# Inter-slice Mobility Management

> "*5G networks, in combination with network slicing, permit business customers to enjoy connectivity and data processing tailored to the specific business requirements that adhere to a Service Level Agreement (SLA) agreed with the mobile operator.*"
>
> — GSM Association

*5G networks aim not only to enhance traffic performance and allow efficient management, but also to enable it to dynamically and flexibly adapt to the traffic demands of different vertical scenarios. In order to support that enablement, the underlying NFs are being virtualized and deployed in cloud-based environments, allowing for a more optimized usage of the infrastructure resources. In addition, such resources can be sliced, allowing isolated provisioning to specific NFs allocated to disparate vertical deployments. As network slices are envisaged by operators to fulfill a small number of slices, able to cater towards essential 5G scenario demands (i.e., eMBB, mMTC and URLLC), the total amount of slices existing in a system is currently dictated by the underlying operational overhead placed over the cloud infrastructure. This chapter explores the challenges associated to a vision where the network slicing concept is applied with a much greater level of granularity, ultimately allowing it to become a core mechanism of the network's operation, with large numbers of co-existing slices. Here, the framework presented in chapter 3 is enhanced for instantiation of network slices among network providers, enabling it to instantiate sub-slices tailored to use cases and vertical tenants. As such, this chapter proposes a 3GPP-complaint SDN/NFV-based architecture supporting non-3GPP access, and capable of dynamically instantiate 3GPP and non-3GPP slices for inter-slice flow-based mobility.*

## 4.1 Slice-based network overview

The framework architecture presented in the chapter 3 was enhanced in order to enable a deployment of a network over a Infrastructure as a Service (IaaS) environment, allowing network and infrastructure providers to be different players. As such, in the architecture proposal presented in this chapter, network providers (such as, MNOs) request a network slice to the infrastructure provider, in order to deploy their access network. For this, infrastructure providers offer an service orchestrator that can be interfaced, allowing network providers to announce their requirements. This allows infrastructure sharing among multiple network providers, since these request a set of NFs (i.e., PNFs and VNFs) from the infrastructure provider, which in turn slices the infrastructure (i.e., BSs, APs, forwarding devices, data-centers) and chains the requested NFs. This is portrayed as Macro-slices, in the proposed architecture of this chapter. Additionally, network providers, when requesting resources to the infrastructure provider, tailor such requests towards the construction of a suitable network slice, as required by the vertical and traffic demands of each use case. As such, network providers offer a vertical manager, allowing each vertical to specify their requirements. These compose the Micro-slices, in the proposed architecture. Figure 4.1 illustrates the architecture proposal, with the different involved layers being described as follows.

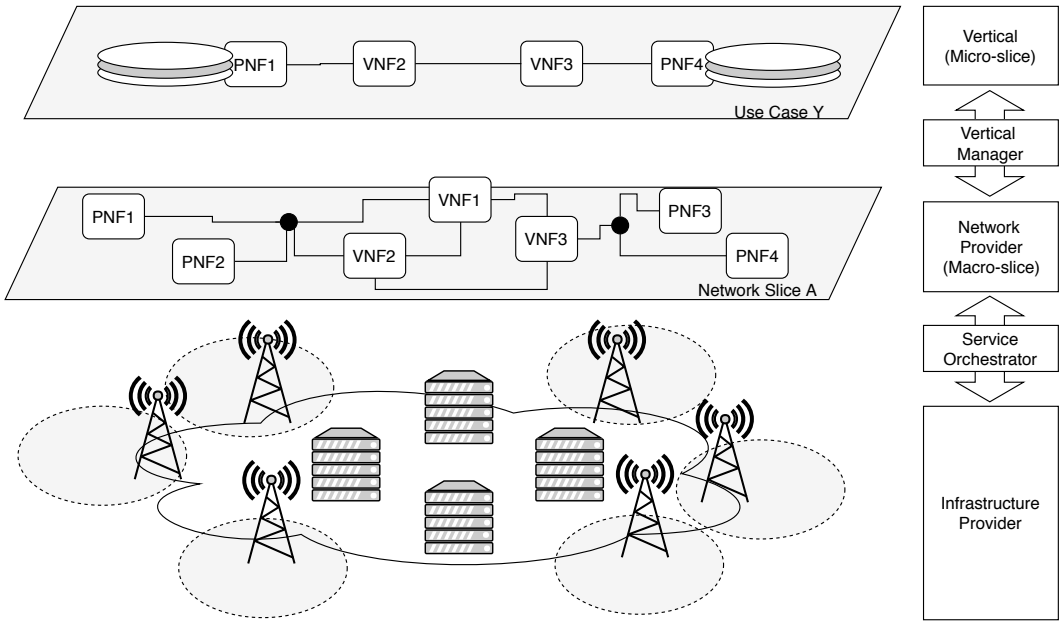



**Figure 4.1:** Slice-based architecture overview [21].

In the proposed architecture, the infrastructure provider is responsible for instantiating network slices when requested by network providers. For this, the infrastructure providers offer a service orchestrator where network providers can choose the necessary

PNFs and VNFs from a catalog, to meet a vertical's demands. Thus, the infrastructure is responsible for chaining the requested NFs in a logical isolated network.

In this context, network providers do not own the infrastructure (i.e., physical equipment). Instead, network providers (such as, MNOs) deploy their network over a shared infrastructure by requesting a slice to the infrastructure provider. Here, a network slice is defined as a set of isolated PNFs and VNFs chained together, resulting a logical isolated network. Also, network providers are able to instantiate sub-slices by reconfiguring their NFs, adapting the virtual access network to each vertical and/or use case.

Verticals are defined as specific industrial or commercial use cases such as automotive and eHealth. In this line, the network provider offers a vertical manager, allowing verticals to specify the necessary traffic requirements. Such information is then used by network operators to (re)configure their slices, in order to adapt the network to the specific traffic demands of each vertical (different types and possibilities of vertical slices are presented in section 4.1.1).

Regarded to the service orchestrator, it exposes the infrastructure layer to network providers, while coordinating the multiple different requests from such providers. As such, the service orchestrator creates a high-level business service to automate network slices instantiation, providing isolation and security among slices and network providers.

Lastly, the vertical manager exposes the network provider features and available resources to the vertical tenant. Here, verticals announce the requirements of the use case, and the SLAs are defined. The manager then (re)configures the network slice (e.g., chains of PNFs and VNFs) in order to tailor the slice to the traffic demands.

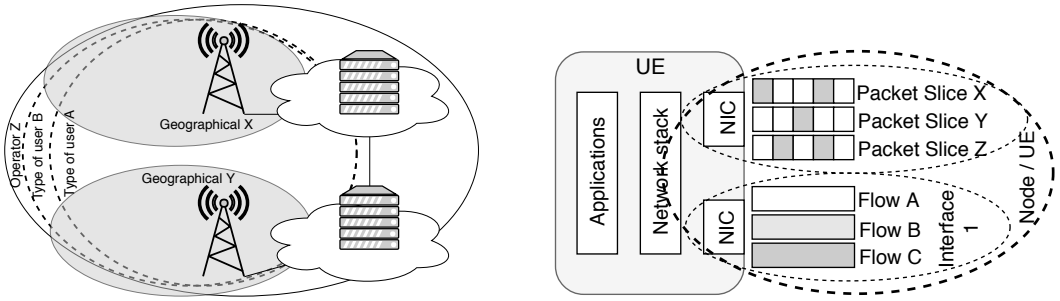
### 4.1.1 Vertical slices and use cases

As discussed previously, network providers, such as MNOs, request PNFs and VNFs to the infrastructure provider (forming a network slice) to deploy their network. Nevertheless, MNOs are able to (re)configure their virtual networks (i.e., network slices), in order to meet traffic demands of each vertical. In this section, and aligned with section 2.3.1 of chapter 2, network slice types are proposed and classified in terms of granularity, going beyond the currently accepted general notion of network slice. Figure 4.2 illustrates the proposed slices dimensions, while Table 4.1 summarizes them by providing use case examples.

In this line, the MNO requests an operator slice to the infrastructure provider, which in turn grants PNFs and VNFs. The MNO is responsible for the orchestration of its slice, allowing it to (re)configure the requested NFs. Thus, the MNO is able to dynamically instantiate sub-slices with different sizes and requirements, according to the use cases' and/or verticals' demands.

Figure 4.2a exemplifies macro-slices instantiated for covering bigger areas of the network. For example, to meet the sporadic or seasonal traffic demands of a sport event, the MNO may instantiate a geographical slice, in order to enforce better coverage and enhance QoS and QoE for its users during the event. Also, the MNO may create different classes of users (e.g., premium, regular and low cost) and instantiate a slice with different requirements and QoS for each class.

Current smartphones are able to simultaneously use multiple wireless access technologies (e.g., LTE and Wi-Fi), thus the MNO has the possibility of instantiating a slice per access technology. This allows the traffic redirection to the less crowded access network for QoE enhancement and/or the traffic offloading from the licensed to the unlicensed spectrum (i.e., from LTE to Wi-Fi). Notwithstanding, alternatively, for UE's redundancy and throughput enhancement, the MNO may instantiate a slice covering multiples access technologies of the UE. Also, despite UEs having multiple interfaces of the same access technology, a slice can be instantiated per interface, allowing a UE to be attached to different MNOs simultaneously.




**(a)** Macro-slices. **(b)** Micro-slices.

**Figure 4.2:** MNO's sub-slice dimensions [21].

Nevertheless, use cases such as eHealth and automotive require high reliability, with the presence of micro-slices becoming important (Figure 4.2b). In this line, scenarios where Network Interface Controllers (NICs) of the nodes (or UEs) attach to multiple slices simultaneously, allow the UE to prioritize certain types of flows. Also, in more specific scenarios, packets of the same flow are transmitted/received in parallel, allowing a greater degree of redundancy, reliability and/or throughput.

**Table 4.1:** Types of slices and use cases [21].

| Type of slice | Description | Vertical / Use case |
|---|---|---|
| Operator | Network slice instantiated by the infrastructure provider with multiple PNFs and VNFs. | MNOs. |
| Geographic area | Slice instantiated in geographical area for sporadic gathering. | Sport event or concert in a stadium. |
| Type of client | Instantiated by the tenant MNO over the operators slice by (re)configuring the requested PNFs and VNFs. | Definition of classes of users for QoS and QoE. |
| Access technology | The MNO offers network access through different wireless access networks (e.g., LTE and Wi-Fi), instantiating a slice per access technology. | Mobile video offloading. |
| Node/UE | Independently of the number of UE's network interfaces, the MNO instantiates one slice for the UE. | eMBB (high throughput, since the UE uses multiple interfaces simultaneously). |
| Interface | For UEs with multiple network interfaces for the same wireless access technology (e.g., dual-sim smartphones) a slice is instantiated per interface. | Resilience and redundancy |
| Flow | The MNO instantiates slices per types of flow and/or service, allowing UEs to attach to multiple slices simultaneously. | QoS and QoE per flow and/or service (traffic shaping). |
| Packet | The MNO instantiates multiple slices to which UEs attach allowing the parallel transmission of packet from the flow. | uRLLC and eHealth. |

## 4.2 Framework Architecture Enhancements

The proposed framework aims to inter-operate with 3GPP and non-3GPP networks, by instantiating a 3GPP slice and (when appropriate) dynamically instantiate a non-3GPP (i.e., Wi-Fi) slice for traffic offloading from the licensed to the unlicensed spectrum, while ensuring the traffic requirements (in terms of security, isolation and QoS) for end-users. For this, the architecture presented in chapter 3 was adapted for ensuring compatibility with 3GPP networks. Also, in order to allow the MNO to deploy its service over an IaaS, in this chapter, the framework architecture adopts a SDN/NFV-based 3GPP architecture and enhances it by instantiating new network entities: (i) Slice Creator; (ii) Slice Selector; (iii) Context Updater; and (iv) vUE. Figure 4.3 presents the proposed mobile network architecture as an enhancement of a SDN/NFV-based 3GPP architecture. For compatibility sake, this proposed framework maintains to the highest possible extent existing interfaces used between 3GPP building blocks, while mapping any new introduced protocol to 3GPP signaling whenever possible.

In the 3GPP standards (release 15 [90] and release 16 [27]) for 4G networks, the interoperability between 3GPP and non-3GPP networks is ensured by the MME and P-GW. The MME is responsible for the control of intra-3GPP and inter-3GPP (to non-3GPP technologies, e.g., from LTE to Wi-Fi) handovers, where the S-GW and the P-GW are used as data traffic anchors for intra-3GPP and inter-3GPP handovers, respectively. Notwithstanding, with the introduction of new technologies into the network (i.e., slicing, SDN and NFV), new procedures should be studied, along with the instantiation of new network entities and functions. In fact, for 5G networks, the 3GPP is considering such enhancements, and new network functions are being established under standardization [25] (presented in chapter 2, section 2.1). In this line, since at the time of this writing there is no open-source 5G core frameworks available, the architecture here presented uses a 4G network and proposes the usage of the vUE for managing the inter-3GPP mobility of its physical counterpart, alleviating the MME procedures. Moreover, although the proposed architecture focuses on the existing EPC, the current 3GPP standardization for 5G in [25] was followed, with the UE's context updater being able to be integrated in the AMF, allowing the network to follow the UE and update its context in the virtual instances. The building blocks of the proposed framework are described next, and depicted in Figure 4.3.

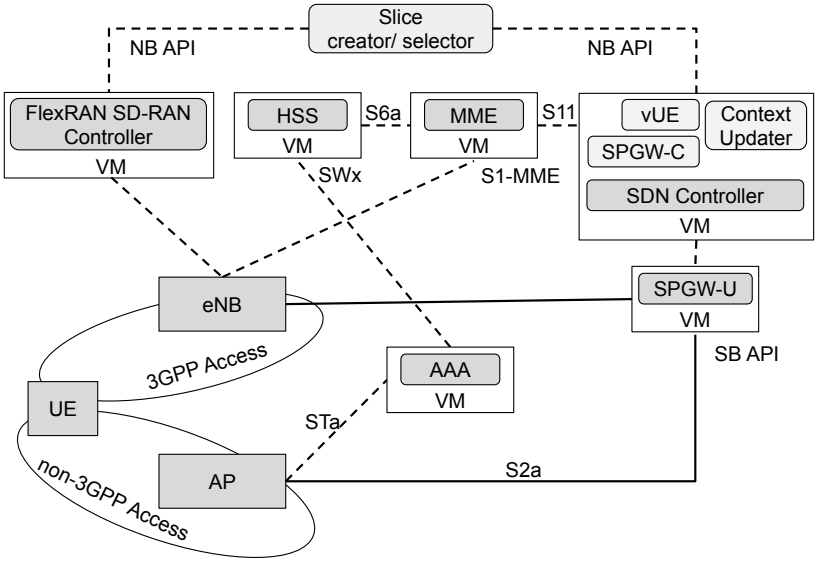



**Figure 4.3:** Deployed framework architecture.

- **Slice Creator:** The slice creator is in charge of dynamically instantiating 3GPP sub-slices (e.g., RAN slices) and non-3GPP slices. As such, it communicates with the context updater of each 3GPP slice. Since each slice is specific to each use case, the slice creator chooses the building blocks to be instantiated with the slice.

Here, the S/P-GW was instantiated for user plane and the Context Updater and vUE for control plane.

- **Slice Selector:** The slice selector maps the UEs to their corresponding slices. For example, while a regular user has its UE connectivity attached to a default slice, corporates may have a dedicated virtual EPC with different network requirements per UE. Also, and specific to the scenario discussed in section 4.3, a corporate may contract a MNO for dedicated non-3GPP slices for its collaborators to improve the remote working experience.
- **Context Updater:** The context updater communicates with the vUEs, updating its information and requesting the instantiation of non-3GPP slices to the slice creator. Here, the context updater will enhance the capabilities of the S-GW and the P-GW of the 3GPP by creating and updating the vUEs of the end-users.
- **vUE:** The vUE assumes the MME responsibility of controlling inter-3GPP handovers, but with a finer flow control, since each vUE is responsible for its UEs flows. As such, the vUE maps the UE in the network, keeping the information regarded to the physical UE's wireless link and active flows. Also, each vUE is able to verify if the QoS of each flow meets the requirements agreed in the SLAs.

Note that the above network entities are enhancements from the architecture proposed in chapter 3. Thus, *slice creator*, *slice selector* and *context updater* entities were introduced into this architecture to allow the collection of UE's context information (e.g., neighbor cells and APs) and to dynamically instantiate slices with the predefined KPIs (e.g., bandwidth and access networks) for supporting the UE's communications. Contrarily, despite already being introduced in chapter 3, here the vUE is a SDN application, decoupling the control and data paths, and allowing the anchoring of the UE's data in different network points.

### 4.2.1 Network procedures

The framework proposal considers different procedures and network entities interaction, depending on the type of access technology used by the UE. Figure 4.3 depicts the entities involved in the procedures for the 3GPP and non-3GPP attachment. The above procedures are described as follows.

#### *4.2.1.1 3GPP attachment and 3GPP slice*

When an UE attaches to the network via a 3GPP access the eNB requests credentials to the MME (via the S1-MME interface). The MME validates the International Mobile Subscriber Identity (IMSI) of the UE with the HSS (via the S6a interface), and then replies to the UE. After that, the UE requests a L3 attachment, which is agreed between the MME and the S-GW-C (control plane) (via the S11 interface). Upon attachment

completion of the UE, the S-GW-C notifies the Slice Selector, which in turn sends the result to the S-GW-C, the FlexRAN controller and the Context Updater of the correspondent slice. The S-GW-C then creates a tunnel between the attached eNB and the Gateway (GW) of the respective core slice. As presented in chapter 2 (section 2.5.4), the FlexRAN controller is capable of instantiating the 3GPP radio slices.

#### *4.2.1.2 non-3GPP attachment and non-3GPP slice instantiation*

As presented in chapter 2 (section 2.3 and section 2.5.4), the non-3GPP radio slicing can be achieved by wireless virtualization. Thus, the proposed framework deploys multiple SSIDs with different wireless security encryptions over the AP. For this, APs of the proposed framework were enhanced with an application for on-demand reconfiguration of the wireless networks with the necessary characteristics (e.g., security, traffic shaping).

Regarding to the non-3GPP access authentication, different procedures are considered.

- **Independent non-3GPP access:** when the user connects to a non-3GPP access, the AP authenticates the UE for wireless access (e.g., Wi-Fi Protected Access (WPA)), and the Internet Service Provider (ISP) and MNO (through the context updater and vUE) establish the slice requirements for the UE, with the former instantiating a slice for the UE (scenario presented in section 4.2.4.2).
- **Mobile core with support for non-3GPP access:** when an UE attaches to a non-3GPP access (i.e., Wi-Fi) deployed by the MNO, the AP authenticates the UE in the AAA (interface STa), which in turn verifies the IMSI of the UE with the HSS (interface SWx). DHCP is then requested by the UE, and the S-GW-C with slice selector connects (via tunnel) the AP with the respective slice S/P-GW (used in section 4.3).

#### *4.2.1.3 Inter-slice flow mobility*

Here, inter-slice mobility is processed as inter access technology handovers (i.e., between 3GPP and non-3GPP, and *vice versa*). However, in the proposed framework, such procedure was moved from the MME to the vUE. This allows a greater degree of granularity, since each vUE monitors the flows of its UE. Conversely, current MME implementations do not perform flow-based mobility, instead they offload all traffic, independently of the flow characteristic and the wireless link status of the UE. In this context, for inter-slice flow mobility, the vUE monitors the wireless link and active flows of its physical counterpart, in order to perform flow-based mobility adapted to each flow, while requesting for non-3GPP slices with the agreed QoS requirements with the tenant vertical. Regarding to intra-3GPP slice mobility, in the proposed framework,

the MME continues to be in charge of its management, according with the involved eNB and UE the transmission parameters.

### 4.2.2 Wi-Fi slices implementation and evaluation

For non-3GPP access, the framework was evaluated, as proof-of-concept, under the instantiation of Wi-Fi slices, more specifically, the IEEE 802.11n at 5GHz. As such, the AP was implemented using APU2C4 with wle600vx wireless modules. The Ubuntu 14.04 LTS OS was used with the hostapd v2.1 software for AP capabilities. For the non-3GPP slice instantiation an application was developed, which reconfigures the Linux *hostapd* daemon creating a new SSID with the necessary characteristics (e.g., security, traffic shaping) and implements traffic shaping through Linux traffic control when requested by the network controller. Finally, the new SSID was configured in IEEE 802.11n at 5GHz with a chosen protect access, including the EAP-AKA[1] authentication, which in turn allows the authentication using the Subscriber Identification Module (SIM) card.

#### *4.2.2.1 Slice instantiation delay*

As mentioned above, Wi-Fi slices were created exploiting *hostapd* features by instantiating an SSID for the UE, where different authentication methods can be set (e.g., WPA2 and EAP-AKA). As such, slice instantiation delay is the time interval between the slice creation request and the AP's reply after instantiation. Also, such delay can be decomposed as follows: (1) suitable AP discovery; (2) slice instantiation; (3) UE's connection to the AP; and (4) flow redirection. Figure 4.4 shows the cumulative distribution function (CDF) and empirical CDF (eCDF) of such procure (from 1 to 3).

The (1) suitable AP discovery took $9s$ ($\pm 2s$) and it was regarded to the UE's detection of an available AP[2] (for instantiating a slice) in range and informing it to the network (i.e., to the vUE). In the Linux UE deployment, an application was developed for enabling the UE to scan the wireless environments with a periodicity of $9s$ (similarly to iOS and Android OS [91]). The (2) slice instantiation took $10s$ and was the delay between the non-3GPP slice request and its instantiation. This delay was associated to the necessity of restarting the *hostapd* with the updated values (i.e., SSIDs, traffic shaping, and DHCP server). Then (3) the UE took about $7s$ ($\pm 1s$) to detect and connect to the dynamically instantiated SSID. Finally, (4) the data flow was redirected from the 3GPP to the non-3GPP slice.

In this line, while the instantiation of the non-3GPP slice took $10s$, the overall procedure took about $26s$. Finally, the proposed framework can explore location service

[1]EAP-AKA specifies an Extensible Authentication Protocol (EAP) method that is based on the Authentication and Key Agreement (AKA) mechanism used in 3GPP networks.

[2]This was implemented by discovering known SSIDs (section 4.3).

of current smartphones, and preemptively instantiate the non-3GPP slice, saving 20$s$ of delay (i.e., slice request and slice instantiation).

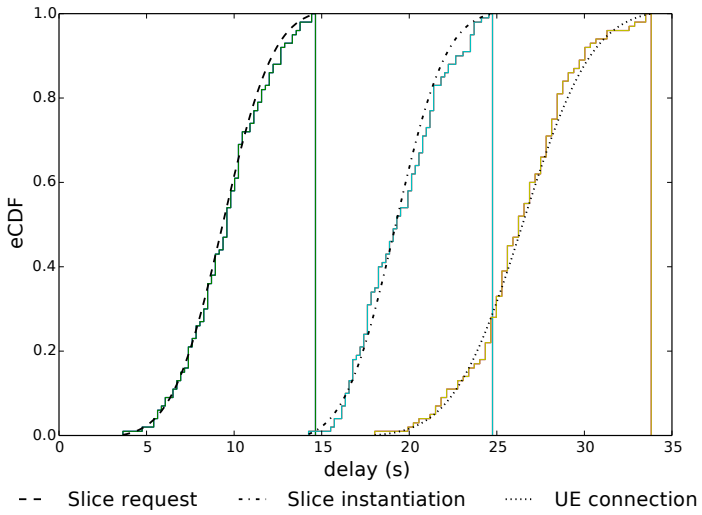



**Figure 4.4:** Instantiation of a non-3GPP slice [17].

#### *4.2.2.2 Slice throughput*

As an illustrative scenario, here the Wi-Fi bandwidth was shaped depending on the service or user type. As such, the AP (in a IEEE 802.11g at 2.4GHz) was sliced with an SSID per user (totaling 3 users), while dynamically establishing different bandwidth percentages per slice. Thus, it was defined: *slice 1* for premium users with 12$Mbps$; *slice 2* for regular users with 6$Mbps$; and *slice 3* for low-cost users with 2$Mbps$. Figure 4.5 shows the achieved TCP downlink throughput over time per slice. Results were obtained using the *iperf* tool. Initially, all slices/users were defined with the same allowed bandwidth. Thus, the 20$Mbps$ were divided among the existing slices, resulting in about 6$Mbps$ per slice. At 9$s$, the traffic shaping was established and the bandwidth was divided as previously described. As a final note, for UDP traffic, while for downlink the bandwidth is divided as described, for the uplink the UEs continuously compete for the bandwidth, showing the importance of radio slicing for E2E slicing mechanisms.

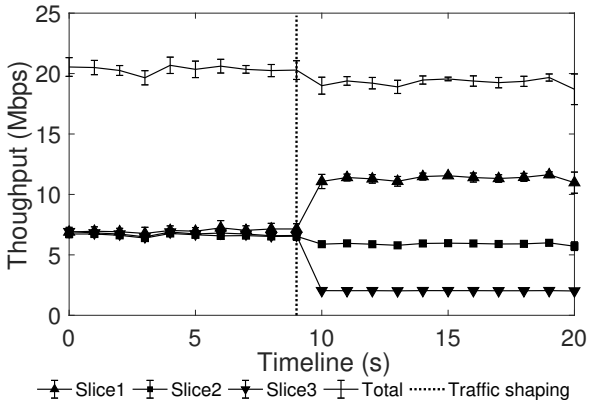
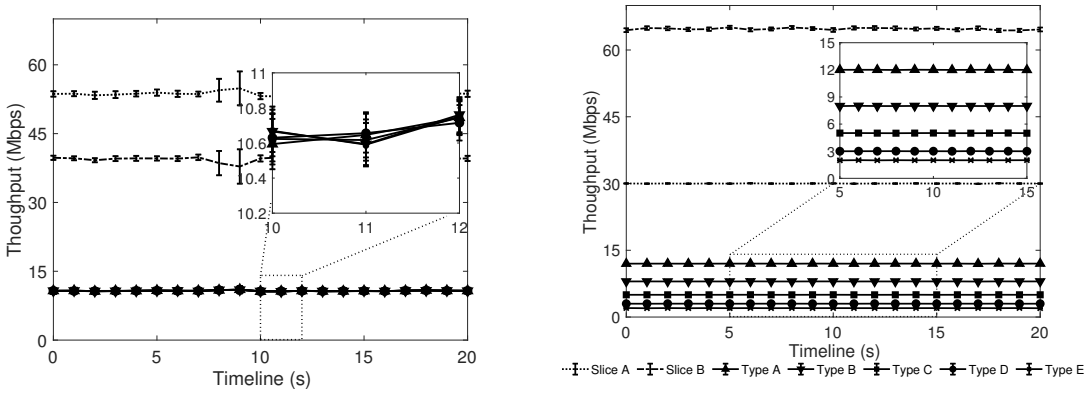



**Figure 4.5:** Impact of traffic shaping in slice throughput [16].

#### *4.2.2.3 Traffic shaping within the network slice*

Figure 4.6 illustrates the upstreaming UDP throughput for a sliced AP with two IEEE 802.11n at 5GHz wireless networks (i.e., *slice A* and *slice B*). Thus, the AP had approximately $100Mbps$ of bandwidth to be shared among active slices. In this line, Figure 4.6a illustrates the traffic for a scenario without E2E QoS applied. As such, the bandwidth was divided by both networks, independently of the slice requirements, resulting in approximately $50Mbps$ per slice. Also, with each slice achieving $50Mbps$, such bandwidth was divided per user flows. In the evaluated scenario, the UE had 5 active flows, resulting in approximately $10Mbps$ each, disregarding the flows characteristics (i.e., live video, web searching).

Conversely, in Figure 4.6b the UE was a SDN-enabled device, allowing its virtual counterpart to dynamically instantiate queues depending on the user demands while switching the traffic among queues in a flow-based way, resulting in a dynamic E2E QoS. Additionally, the network provider is able to dynamically establish the traffic bandwidth to each UE, and consequently, its slice. This allows to establish the dynamic bandwidth elasticity of the UE and slices. Here, *slice A* was limited with $30Mbps$, allowing *slice B* to achieve $65Mbps$. Nevertheless, not all flows have the same network demands, with examples such as online gaming and live video streaming having more restrict requirements in terms of delay and throughput than web content searching or cached video. As such, and as an illustrative scenario, in *slice A*, different QoS policies where configured, depending on the traffic characteristics. Thus: *type A* with $12Mbps$ for online gaming; *type B* with $8Mbps$ for live video streaming, *type C* with $5Mbps$ for video cache, *type D* with $3Mbps$ social media; and *type E* with $2Mbps$ for web searching. Comparing both scenarios (i.e., Figure 4.6a and Figure 4.6b) it is noted that by reducing the bandwidth in one slice, the other slice was able to consume the remaining bandwidth, allowing slice elasticity. Also, Figure 4.6b illustrates the feasibility of the proposed mechanism with the flows being shaped to the configured




**(a)** **(b)**

**Figure 4.6:** Wi-Fi slice throughput for: (a) without E2E QoS; and (b) SDN-based UE with E2E QoS [22].

bandwidth.

#### *4.2.2.4 Signaling impact for traffic shaping*

The signaling impact in terms data overhead and delay for the dynamically instantiation of QoS rules in the UE are presented in Table 4.2. As discussed previously, to implement SDN-based QoS policies both OF or OVSDB protocols can be used. In this line, OF *meter_modification* message creates a flow meter in the OF/OvS bridges, allowing the traffic policing above configured thresholds. OVSDB enables the management of OvS bridges, allowing the creation of QoS rules for traffic shaping. In this context, while an OF *meter_modification* message had 164 bytes of overhead and a delay of approximately 7ms, the creation of a QoS rule had approximately 7Kbytes of overhead and 60ms. This is mainly due to the fact that both the SDN controller and OvS bridge exchange information about the bridge status before and after the QoS creation. Finally, to apply the QoS policies, the flow must be redirected to the flow meter and/or QoS queue. This is performed via OF *flow_modification* messages. As final note, flow meter and queues are complementary to each other. However, flow meters are usually easier to create and modify at the runtime (via OF messages), while queues are more rigid and created out of band or by specific protocols (e.g., OVSDB).

**Table 4.2:** Signalling impact for traffic shaping within Wi-Fi slice [22].

| Function | Protocol | Data overhead | Delay |
|---|---|---|---|
| Flow meter | OpenFlow | 164 *bytes* | 7 *ms* |
| QoS and Queues | OVSDB | 6914 *bytes* | 59.10 *ms* |
| Flow redirection | OpenFlow | 236 *bytes* | 5 *ms* |

### 4.2.3 3GPP slices implementation and evaluation

The architecture of the instantiated MNO's slice is presented in Figure 4.3, where the EPC was instantiated in a in-house data-centre running OpenStack (Ocata). The deployed EPC was based in the OAI project, enhancing its flexibility by introducing a SDN controller capable of receiving information from NB SDN application helping in the network (re)configuration. In this line, 3GPP entities, such as the MME, HSS and S/P-GW, were instantiated in VMs with a 1 CPU core and 2 GB of RAM, running the Ubuntu 16.04 LTS OS. Regarding to the RAN, its flexibility was enhanced by deploying the FlexRAN framework. In this context, the proposed framework is composed by both the FlexRAN and the OAI, allowing the development of SDN NB applications to acquire context from both core and ran networks, and assist the SDN controller in the core and access networks management via NB APIs. The eNB was deployed in a physical machine running Ubuntu desktop 16.04 LTS OS with an Intel Core i7-7700K CPU and 32 GB of RAM, and an USRP B210 Software-Defined Radio (SDR) board with a LP0965 antenna.

The new proposed network entities, namely the slice creator and the slice selector, both were developed and implemented as Ryu SDN applications for assisting the SDN controller in the management of UEs' requirements. As such, both run along with the SDN controller and interacted with the remaining entities via UDP messages, and when applicable deploying OF messages through the SDN controller.

For simplification, due to the complexity of infrastructure sharing among MNOs (section 2.3.1), in the proposed scenario the slice for the MNO was already instantiated. In this context, the MNO reconfigures its building blocks (PNFs and VNFs) according to requests from verticals. Figure 4.7 illustrates the messages sequence for the reconfiguration of the framework upon the attachment of an UE.

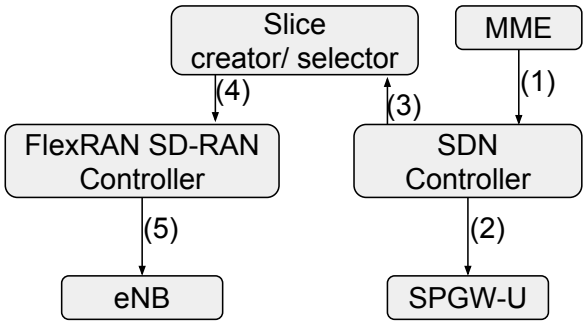



**Figure 4.7:** High-level sequence of messages for the scenario evaluation [21].

In Figure 4.7, when a UE attaches to the 3GPP network, the MME informs the SDN controller via NB APIs, through developed REST messages (Figure 4.7:1). The SDN controller then reconfigures the S/P-GW-U (user plane) (Figure 4.7:2) via SB APIs (i.e.,

OF for flow based rules and OVSDB for tunnel creation), enabling the communication of the UE. Initially, the UE is attached to a default slice. Nevertheless, after attachment completion, the SDN controller informs the *slice creator/selector*, which in turn verifies the UE's characteristics (e.g., type of client, current flow in use, etc.) and (if necessary) requests to the FlexRAN SD-RAN controller to move the UE to the correct slice (or asks for a new one, moving the UE afterwards).

Next, the proposed framework is evaluated in terms of UE attachment, radio slice instantiation and slice handover delays, and the impact caused by the simultaneous existence of requests and/or increasing number of instantiated slices. Experiments were run 50 times, with the results being shown in Figure 4.8 and presented with a confidence interval of 95%.

#### *4.2.3.1 Slice radio instantiation delay*

As previously discussed, the deployed framework uses both OAI and FlexRAN. In this context, FlexRAN only allows to instantiate a maximum of 10 radio slices. Figure 4.8a) presents the instantiation delay of upon a request of multiple radio slices. The results show that the slice instantiation did not depend on the number of requested slices, as of the current version of the FlexRAN software, for the amount of simultaneous slices used in the evaluated experiments. In Figure 4.7, it is noted that the request was sent by the developed SDN NB application via REST (more specifically, Hypertext Transfer Protocol (HTTP) POST message). However, the FlexRAN SD-RAN controller sends an acknowledgment before the correct implementation of the slice, misleading the slice requester to move UEs to a slice before its correct implementation. To overcome this, a delay between the slice instantiation acknowledgment and the UE's slice handover was set.

#### *4.2.3.2 Slice handover delay*

From Figure 4.8b) it is noted that the slice handover of the UE was independent of the number of instantiated slices. As such, in Figure 4.7, when the requester asked for a slice handover of the UE, the network took about $2s$ to switch the radio slice that the UE was attached to. In this context, in scenarios where the network uses handovers from one slice to another as the means to fulfill the UE requirements (e.g., better latency), the slice handover delay requires to be much lower in order not to incur any impact to the experienced QoE (especially when considering URLLC scenarios). As such, enhanced mechanisms should be developed in order to increase handover performance.

#### *4.2.3.3 S/P-GW update delay*

Figure 4.8c) presents a scalability study of the developed SDN mechanism for S/P-GW flow control. For this, the MME emulated the attachment of multiple UE simultaneously and the delay for correct flow implementation for the requested number of UEs was measured. The experiment uses a worst case scenario, since requests were sent consecutively to the SDN controller. Results show that the developed mechanism escalated well, outperforming a linear scenario.

#### *4.2.3.4 UE attachment delay*

This experiment evaluates the total delay of the UE attachment in a dynamically instantiated slice. Thus, this delay results in the sum of the delay for slice radio instantiation, update of the S/P-GW and slice handover. Figure 4.8d) presents the eCDF of the delay for each stage completion (as such, each stage accounts the delay of previous stages). In Figure 4.7, when an UE attached to the 3GPP network, the MME notified the SDN controller (Figure 4.7:1) which in turns updated the S/P-GW-U (Figure 4.7:2). Such update took about $100ms$ and was related to the installation of two flow-based rules (for uplink and downlink UE's communication) in the OF switch. The *slice creator/ selector* then instantiated a new slice for UE (Figure 4.7:4, $600ms$), moving the UE to it afterwards (Figure 4.7:4, 2). In this context, the whole process took almost $3s$ to be completed.

#### *4.2.3.5 Messages impact*

Figure 4.8e) illustrates the message sizes for the different scenario messages. For the UE attachment and detachment (Figure 4.7:1), the REST API was used, more specifically, HTTP POST messages. As such, the former had an impact of $370bytes$ and the latter $310bytes$. When the SDN controller is notified of the UE's attachment, it updates the S/P-GW-U via OF messages (i.e., OF *flow_modification*). Thus two messages were sent (Figure 4.7:2): first for uplink (UL) communication ($170bytes$), and second for downlink (DL) ($178bytes$). Initially, the UE is attached to a default slice. In case, of specific UE's requirements, the *slice creator/ selector*, uses HTTP POST messages (Figure 4.7:4) to instantiate a slice ($558bytes$) and to handover the UE for the new slice ($66bytes$). Finally, for a slice delete and HTTP POST message ($237bytes$) is sent towards the FlexRAN SD-RAN controller.

Regarding to request slice messages, Figure 4.8f) shows how the HTTP POST message increased depending on the number of requested slices. Also, in scenarios where there are already 5 slices, and 3 more slices are needed, the HTTP POST contains the resulted 8 slices with the correspondent slice parameters (resulting in $1204bytes$).

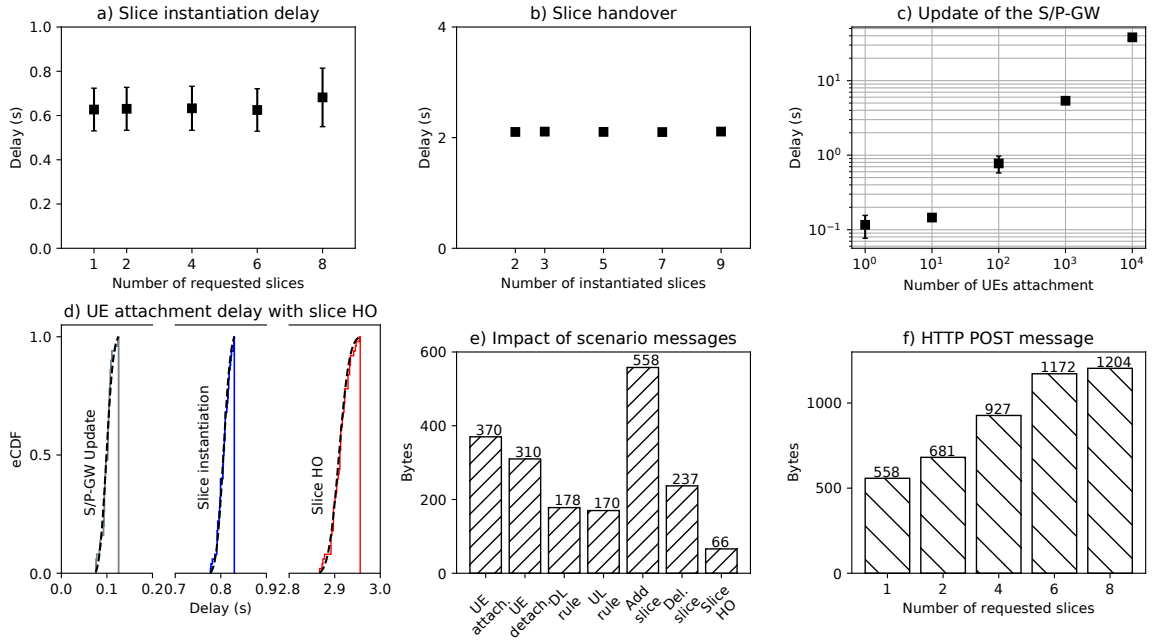



**Figure 4.8:** Experimental results for 3GPP network reconfiguration [21].

### 4.2.4 Proof-of-concept evaluation for mobility scenarios in dynamic slice environments

This section evaluates the proposed framework in terms of inter-slice mobility for mobile video offloading (section 4.2.4.1) and dynamic resource reconfiguration (section 4.2.4.2).

#### *4.2.4.1 Mobile video offloading for a non-3GPP slice*

This scenario dynamically instantiates non-3GPP slices, in order to allow the operator to offload mobile video from the licensed (i.e., LTE) to the unlicensed (i.e., Wi-Fi) spectrum, while keeping the user's QoE. Figure 4.9 depicts the high-level message sequence of the scenario. Here, the flow redirection to the non-3GPP (Wi-Fi) slice is decomposed in four stages. First, the vUE detects that the user started to visualize a live video. For this, the OF *flow_stats* message was used (with a periodicity of $5s$) to analyze the characteristics of the user's flow (e.g., protocol, port and bitrate). If such analysis results in an offloading decision, the vUE requests the creation of a slice. Secondly, the slice is created exploiting *hostapd* features by instantiating an SSID for the UE, whose authentication is ensured via IMSI. Thirdly, after the slice is created, the UE detects the SSID and attaches to it. Finally, the vUE implements a flow redirection in the chosen anchor (i.e., GW) via a OF *flow_modification* message. The offloading delay was measured from the moment the UE starts receiving the video via LTE until its redirection to the Wi-Fi slice, resulting in a average delay of $36s$.

The framework was implemented using the guidelines presented in sub-section 4.2.2 and sub-section 4.2.3, for non-3GPP and 3GPP, respectively. In this section the proof-

of-concept implementation was evaluated in terms of handover performance, throughput achieved in a non-3GPP slice, and overhead in over-the-air data traffic. The experiments were run 10 times with the results being presented with a confidence interval of 95%. Finally, an experiment was recorded and is available online in the project webpage[3].

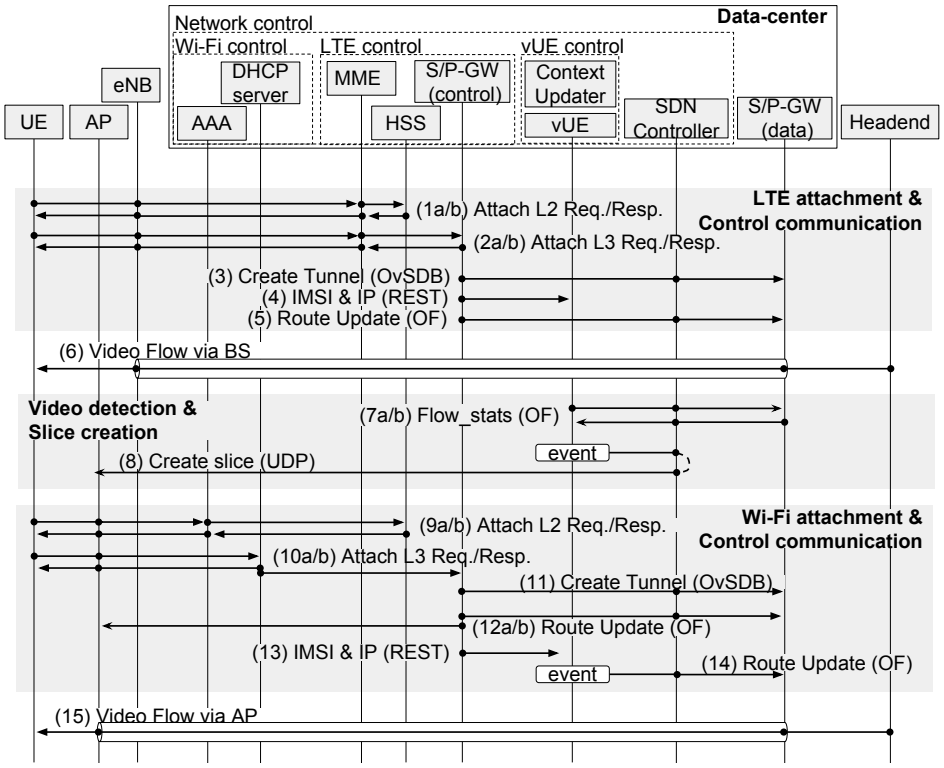



**Figure 4.9:** Mobile offloading for a non-3GPP slice high-level signalling [16].

#### *4.2.4.1.1 Scenario evaluation*

The evaluated scenario started with the user visualizing a live stream video on a mobile. Initially, the UE was receiving the video[4] via LTE, however in the meantime, the vUE requested (at 2s) to the SDN controller an Wi-Fi slice for mobile video offloading due to congestion on the network[5]. Still receiving the live video, the UE attached (at 22s), in background, to the dynamically instantiated Wi-Fi slice. Such attachment triggered the vUE, which in turn redirected (at 22s) the video flow from the LTE to the Wi-Fi seamlessly, switching from the licensed to the unlicensed spectrum. Figure 4.10 illustrates the throughput in the handover scenario, comparing it with having the video always received via the congested eNB. Here, despite the throughput being similar to both situations (only 2% of throughput loss), in the latter the UE received unsorted

[3]Mobile video offloading demo: https://atnog.github.io/5G-VCoM/demos/demo1.html

[4]Big Buck Bunny (video): https://peach.blender.org/

[5]Exploiting the OAI limitations, which works in transmission mode 1, the eNB was overloaded by adding more users requesting UDP data.

packets due to congestion, which degraded the user's QoE (this can be seen on the recording available online). In terms of bytes, in the 40s of the assessed video, 50% of its total cost (4 Mbytes) was offloaded to Wi-Fi. No lost packets were experienced using the developed mechanism, which was able to redirect the flow to the Wi-Fi slice maintaining the user's QoE.

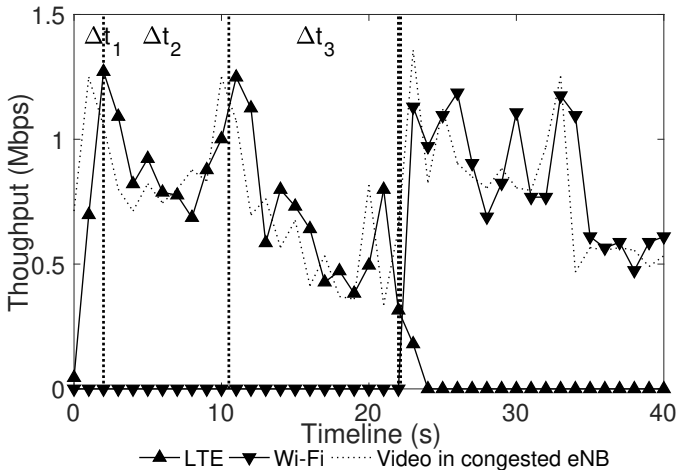



**Figure 4.10:** Video throughput over time for mobile offloading scenario [16].

Regarding to the signaling, table 4.3 presents the impact of the control signaling of the evaluated offloading scenario. Since the REST (used for interfacing SDN applications and the controller) imposes high overhead (e.g., sending 107 bytes of data required 410 bytes), the slicing creation message was implemented via dedicated UDP signaling. As such, for 14 bytes, this dedicated message had an impact of 60 bytes. For SB communication, OF and OVSDB were used for route updates and OvS configuration (create tunnels) on-the-fly, respectively. While each route update had an average impact of 178 bytes, tunnel creation had an impact of 20 Kbytes. This was due to the information exchanged between the OvS and SDN controller, where both share status before and after creating the new port/tunnel. Nevertheless, the tunnel is created only at the first communication of each eNB and/or AP. Regarding to the OF *flow_stats* message for data update in the vUE, it had an impact of 332 bytes for each message sent every 5s. As such, for this 40s scenario it had an impact of 2.6Kbytes. Moreover, as the UE's active flows increase, the payload of the OF *flow_stats* reply message (8b) increases.

**Table 4.3:** Impact of dedicated signalling messages for mobile offloading scenario [16].

| Function (in Fig. 4.9) | Protocol | Payload (bytes) | Total impact (bytes) |
|---|---|---|---|
| UE's (de)attachment (4/13) | REST | 107 | 410 |
| Create tunnel (3/11) | OVSDB | 19131 | 21298 |
| Create Slice (8) | UDP | 14 | 60 |
| Route update (5/12/14) | OF | 112 | 178 |
| UE's flows info: periodic (7a) | OF | 72 | 138 |
| UE's flows info: periodic (7b) | OF | 128 | 194 |

#### *4.2.4.2 Dynamic 3GPP slice reconfiguration*

This scenario focuses on network slice optimization for the highly variable UE's data traffic, by reducing resources consumed by network slices that are no longer in use after a UE handovers from them. Here, the framework of the previous section was slightly changed, allowing the 3GPP and non-3GPP access network being provided from different entities (MNO and ISP, respectively). For this, the Over-the-Top (OTT) virtualizes the UE connectivity context in the cloud, and communicates with telecommunication operators (MNO and ISP) to update the UE's connectivity status and current data requirements. In addition, the OTT uses the vUE to anchor the UE to the network and mask underlying radio and network slices into a E2E slice. Here, connectivity context is defined as the UE's current link conditions, active access technologies and surrounding (e.g., neighbor cells). Also, the OTT represents a third-party providing a service that ensures the optimal connection and QoE for its UE, by negotiating with the MNO and OTT to ensure the UE's communication requirements and the optimal network slice resource allocation, masking underlying slices in an E2E slice. This results in an SDN-based inter-technology handover transparent to the end-points. The network architecture for this scenario is presented in Figure 4.11

A proof-of-concept implementation was deployed in an in-house testbed and tested for TCP and UDP data traffic, for both upload and download streams. The control signaling for both protocols and stream direction are similar, and illustrated in Figure 4.12. With the purpose of facilitating the procedures flow, Figure 4.12 presents the core network of the three providers as single entity. The framework was implemented using the guidelines presented in sub-section 4.2.2 and sub-section 4.2.3, for non-3GPP and 3GPP, respectively. The experiments were run 25 times, with the results presenting their average with a confidence interval of 95%.

##### *4.2.4.2.1 Scenario signalling*

In Figure 4.12, initially the UE is connected to the LTE network, and downloading (or uploading) data (1). Passing by an AP, the UE attaches to the non-3GPP network (2)

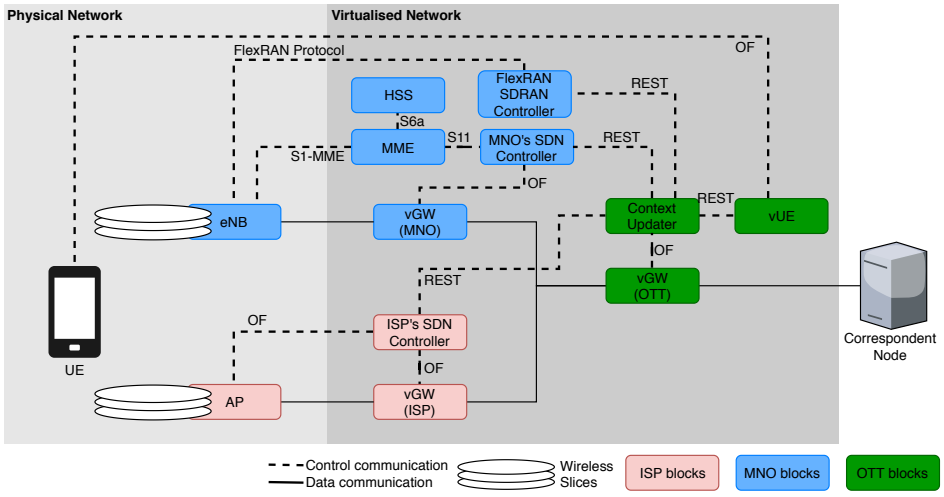



**Figure 4.11:** Network architecture considering an OTT [23].

and informs its virtual counterpart via an OF *packet_in* message (3). In turn, if it results in an handover policy, the vUE updates the flow tables of its physical counterpart (4) and in the virtual Gateway (vGW) (via context updater) via OF *flow_modification* messages. When the handover is completed, the OTT (through the context updater and via REST) notifies the MNO (5), which optimizes the wireless resources of 3GPP slices, by moving the UE for a slice with less capabilities. Since the UE's datapath is anchored in the OTT network by the vUE, masking underlying slices in a E2E slice, the handover process becomes transparent and seamless to both the UE and the CN (6).

Moving away from the AP, the link strength to the non-3GPP reaches a minimum threshold and the UE informs its virtual counterpart (7). In turn, the OTT notifies the MNO (8), allowing it to prepare and optimize the UE's slice requirements to receive the data by moving it to a slice with more capabilities. After switching the UE's attached slice, the MNO informs the OTT, which in turn updates the flow tables of the UE and vGW (9) via OF *flow_modification* messages. Again, as the UE's datapath is anchored in the OTT's network, underlying slices are masked in a E2E slice and the handover procedures become transparent to the UE and CN (10). As result, the OTT is able to communicate with telecommunication operators and dynamically handover the UE's data among heterogeneous wireless slices with the purpose of guaranteeing the optimal UE's connection and wireless resource allocation.

#### *4.2.4.2.2 Scenario evaluation*

Figure 4.13 illustrates the download and upload throughputs for both wireless networks. Also, the 3GPP network was divided in 2 slices, in order to switch the UE considering its current data communication requirements. The LTE *slice A* was configured with 25% of the total bandwidth (approximately 3.5 *Mbps* of bandwidth). Contrary, *slice B* was

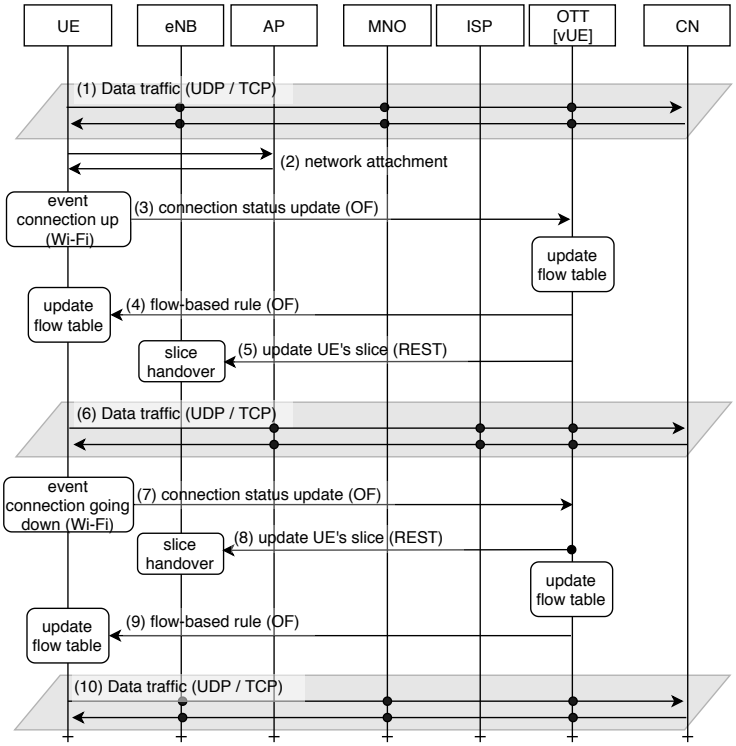


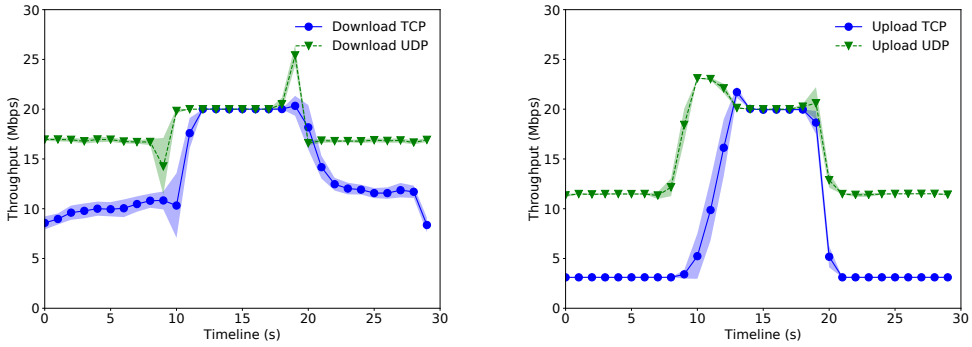

**Figure 4.12:** High-level message sequence for the deployed OTT scenario [23].

configured with 70%, accomplishing approximately 10.5 $Mbps$. Regarding to Wi-Fi, it allows approximately 50 $Mbps$, however the instantiated Wi-Fi slice was set to 20 $Mbps$. Finally, evaluated streams were generated using *iperf3* network tool.

Figure 4.13a illustrates the throughput over time for TCP and UDP data when a user was downloading data. In the figure, it is noted that both slices maximum allowed bandwidth was accomplished, and as the UE moved to the Wi-Fi slice (at 10$s$), the throughput increased from 10$Mbps$ (or 15$Mbps$ when UDP) to 20$Mbps$. On the background, as the UE offloaded the data to the Wi-Fi slice, the OTT notified the MNO, which in turn optimized its wireless resources by switching the UE to a slice with less priority and bandwidth (i.e., LTE *slice A*). Before disconnecting from the Wi-Fi slice (at 20$s$), the UE informed the OTT, which in turn notified the MNO that the UE soon will require an higher bandwidth. When the UE moved back to the LTE, it was already attached to *slice B*.

Finally, the throughput for uploading data is presented in Figure 4.13b. Here, the throughput behavior was similar to the downloading data. Nevertheless, the LTE network offers less bandwidth capacity for upload than for download. As such, the maximum throughput for the *LTE slice B* was approximately 4$Mbps$ and 11$Mbps$, for TCP and UDP, respectively. Contrary, the Wi-Fi slice kept its throughput on the pre-established 20$Mbps$. As in the downloading experiment, the MNO optimized its

wireless resources by switching the UE's attached slice, after and before the offloading to the Wi-Fi slice. Finally, the handover, for both experiments, from/to the LTE slice to/from the Wi-Fi slice took approximately $7(\pm 2)ms$ and $50(\pm 14)ms$, on the vUE and the UE, respectively.




**(a)** Download data throughput over time.

**(b)** Upload data throughput over time.

**Figure 4.13:** Experimental results of the proposed OTT architecture [23].

## 4.3 Network slicing for corporate environments

This section focuses on how network slicing enhances the remote access to private networks while on the move. Keeping this in mind, the framework proposed in this chapter, Figure 4.3, assumes that the MNO instantiates the necessary building blocks to support corporations with remote working communications. Such building blocks may include, for example, mobility management, slice mechanisms and corporation services (e.g., internal portals).

Figure 4.14 illustrates a simplified scenario, where the MNO provides a slice to the corporate, supporting inter-slice mobility, and with corporation services instantiated in the cloud. Here, the access to the corporation services are ensured via the "vCorp-GW" and monitored by the vUEs (along with the "context updater"). The vUE is a virtual representation in the cloud, of the physical UE of the corporations' employees, and readily provides input to the network flow mobility management entities on behalf of its physical counterpart. Also, the vUE allows the corporation to dynamically manage the physical UE by setting the requirements for QoS and security level of each collaborator.

When the user leaves the corporate network's Wi-Fi, this movement is perceived by the MNO (i.e., indicated by link events sent by the UE), which triggers the creation of a dedicated slice for the UE over this new network. This slice receives the redirected UE's flows to allow a transparent access the corporate-based services. Whenever the

UE changes to another network, the process is repeated, with a new slice being created therein, and the previous slice being released for resources optimization.

In this line, UEs of the corporation are able to remotely access corporation services both from the mobile (e.g., LTE) and the Wi-Fi networks without Virtual Private Network (VPN) configuration (through a VPN client). Also, allowing physical resource sharing (i.e., the AP) among different corporations and/or MNOs in public places, the proposed framework extends such features and dynamically instantiating dedicated slices to be accessed by the specific corporation's collaborators.

For implementation simplicity purposes only, the proof-of-concept exploits the fact that the MNO and ISP reside on the same entity and assumes a trustable network underlying the framework, with Generic Routing Encapsulation (GRE) tunnels being used without encryption. Nevertheless, this framework's architecture is designed to support secured tunneling protocols. In fact, OvS supports Internet Protocol Security (IPSec) tunneling and there are associated software patches in order to support GTP[6].

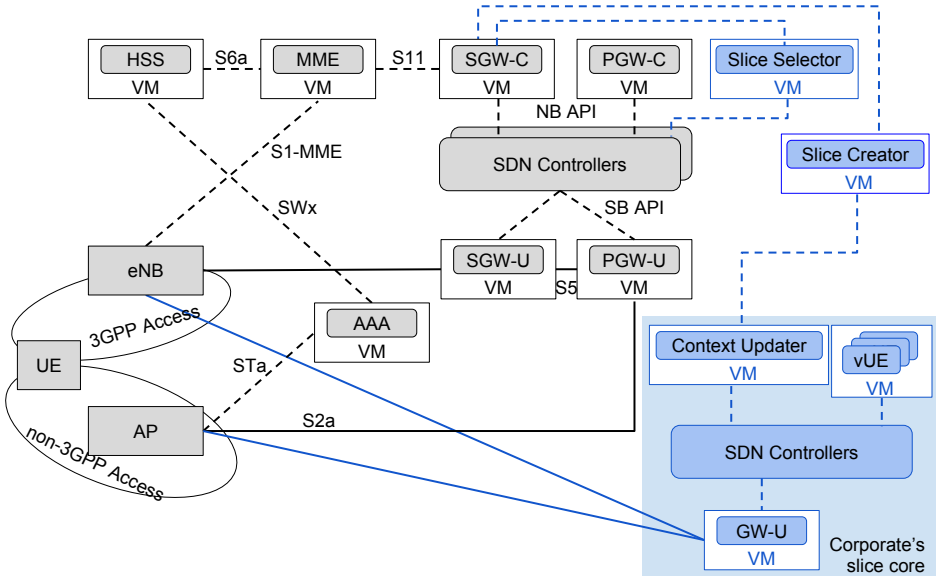



**Figure 4.14:** Proposed mobile network architecture for corporate environments [17].

### 4.3.1 Network entities interactions

This section presents the interactions among network entities for the proposed corporate environments. Figure 4.15 illustrates the high-level message sequence of the proposed scenario, which (similarly to the one presented in section 4.2.1) is described as follows.

[6]OvS pacth for GTP: https://patchwork.ozlabs.org/patch/579431/

When an UE attaches to the network via a 3GPP access (i.e., LTE) the MNO validates the UE and uses the slice selector and context updater, to establish the user data requirements.

When the UE is connected to LTE and detects a non-3GPPP (e.g., Wi-Fi) access of the operator in its surroundings (i.e., via a known SSID) with link quality above a pre-established threshold, the UE notifies the vUE (messages #2a/#2b) with the identification of the AP[7]. To notify its virtual counterpart, the UE sends an UDP message (#2a) towards the vCorp-GW, which in turn redirects to the context updater as an OF *packet_in* message (#2b). The context updater is responsible for updating the vUE information. If this results in an offloading decision by the network, the vUE (along with the context updater) requests a non-3GPP slice (#3) to the Slice creator via an UDP message with the AP and slice identification (i.e., which corporation and/or level of security). Then a dedicated slice (via a unique SSID) for the user (or group of users, depending of the security and mobility policies) is instantiated in the AP (#4). Here, the proposed framework explores wireless virtualization by deploying multiple SSIDs with different wireless security encryptions on-demand using a mobile core with support for non-3GPP access (as presented in section 4.2.1) .

Furthermore, in this scenario, the procedures for flow handover between slices is triggered by an offloading policy (e.g., UE connected to the Wi-Fi slice), where the context updater updates the flow table of the GW (message #5) redirecting the data flow from the licensed LTE to the unlicensed Wi-Fi access network, via an OF *flow_modification* message. Here, the trigger was the UE attachment to the non-3GPP slice, which was notified to the context updater by the MME via a REST message. When the link strength of the Wi-Fi connection crosses a pre-established threshold, the UE reports this to its virtual counterpart (messages #7a/#7b), triggering the redirection from the Wi-Fi to the LTE (message #8), in order to seamlessly keep the data connection (message #9).

In addition, Figure 4.16 presents an overview of the datapath for enterprise's authorized and non-authorized users. Since UE#1 is an authorized enterprise user, its data traffic is redirected towards the vCorp-GW (i.e., in the 3GPP slice), both from 3GPP and non-3GPP access. Also, the UE#1 is attached to the AP via a dedicated slice (i.e., unique SSID), allowing the physical resource sharing of the AP with the UE#2. The UE#2 can (eventually) connect to both 3GPP and non-3GPP, but being an unauthorized user, its data traffic is redirected to the default 3GPP slice, with no access to the corporation services.

[7]UE SSID detection: To enhance the UE with this capability an application was developed and implemented in the UE.

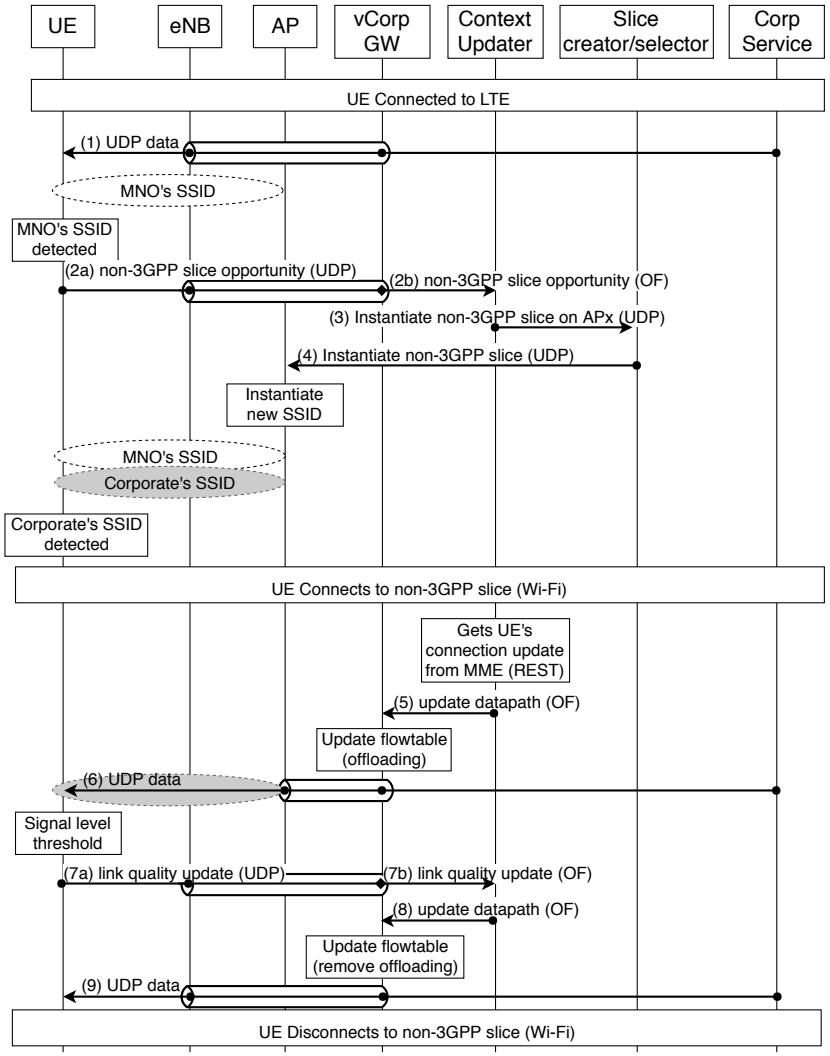



**Figure 4.15:** High-level message sequence for non-3GPP slice instantiation [17].

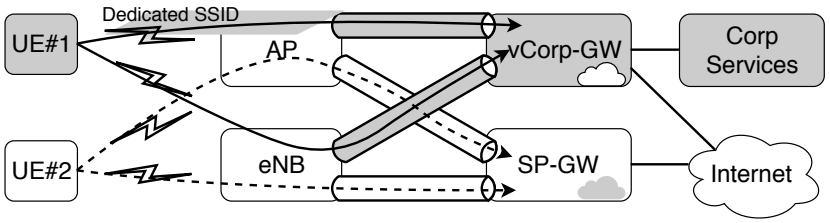



**Figure 4.16:** Deployed datapath simplification, where UE#1 is a registered UE of the corporation and UE#2 is a regular user [17].

### 4.3.2 Proof-of-concept scenario evaluation

This section evaluates and discusses the proof-of-concept implementation in terms of handover performance and overhead in over-the-air data traffic. The experiments were run 50 times with the results being presented with a confidence interval of 95%. Next, the overhead introduced by VPN clients (the OpenVPN was used as example) is

discussed and compared with the proposed framework, focusing on overhead introduced per packet.

#### *4.3.2.1 Packet overhead*

Figure 4.17 depicts the overhead per data packet introduced by the proposed framework in wireless and wired connections, comparing it to the usage of IPSec instead of GRE and to the use of OpenVPN for remote access. The use of a client such as OpenVPN in the UE improves communication security on the wireless medium, however increases the over-the-air overhead as a trade-off. For example, using OpenVPN in a TUN-style tunnel over UDP and default TLS options, about 69*bytes* of overhead are increased to each packet. In the proposed framework, the VPN tunnel stops at the AP and/or eNB, and uses a dedicated slice (i.e., a unique SSID when connected via Wi-Fi), saving overhead over-the-air. Additionally, the proposed mechanism also reduces the overhead in the wired connection, since a GRE tunnel encapsulation increases 24*bytes*. Also, as a security enhancement, IPSec can be used in tunnel mode with an overhead cost of 52*bytes*. In this context, there is a trade-off between security degree and overhead introduced per packet: despite that the overhead might not seem much, when considering highly scaled networks, this additional overhead can affect network equipments' performance when dealing with multiple VPNs at high speed connections. As such, extending the VPN service by dynamically instantiating a dedicated SSID, (while retaining the ability for the UE to authenticate to it using its SIM card information via EAP-AKA), reduces the overhead over the air and processing time by the forwarding devices, while (potentially) saving energy by avoiding the data encryption in the UE. As final note, when considering shared wireless environments (e.g., home and shopping center), the proposed framework has the advantage of aggregating users per server avoiding multiple VPN connections to the server, reducing overhead.

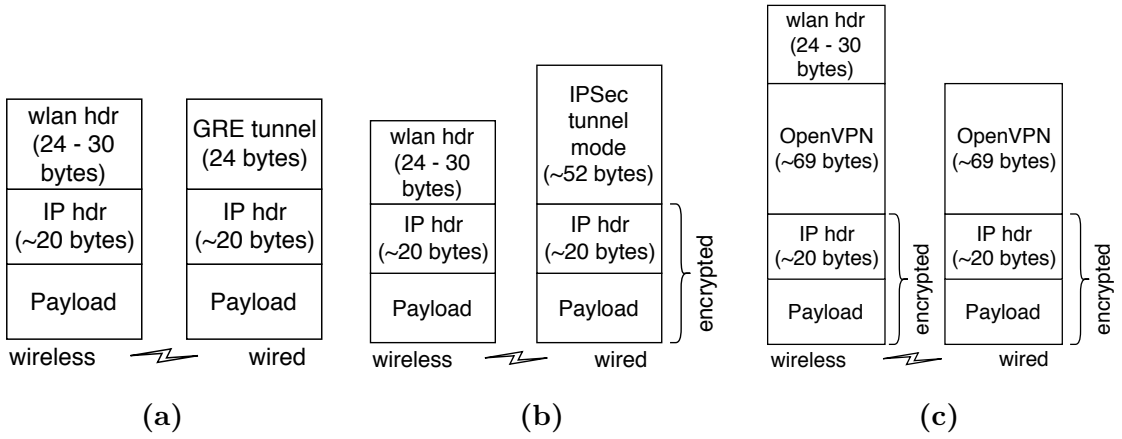



**Figure 4.17:** Packet overhead when considering: (a) GRE tunnel (b) IPSec in tunnel mode. (c) OpenVPN [17].

#### *4.3.2.2 Handover performance*

In order to assess this proof-of-concept in terms of handover performance (achieved throughput over time and downtime) corporation services were virtualized in the cloud and accessible through the vCorp-GW. Figure 4.18 illustrates an UDP stream handover mimicking a scenario with remote working while on the move. The iperf3 application was used in order to generate UDP flows with a pre-established bandwidth, facilitating the visualization of the impact caused by the proposed framework over the QoS.

In the evaluation tests, the UE acted as an UDP client and requested an enterprise service with a bandwidth of 50Mbps. Initially, the user was in the corporation perimeter and was connected to the corporate services via Wi-Fi with an SSID associated to a specific type of user: guest, premium (e.g., CEOs and CTOs) or staff. On its way home the user was able to telework via LTE (from $25s$ to $63s$), and, upon arrival to home (at $63s$), a dedicated SSID was dynamically instantiated in its home Wi-Fi. This avoided the necessity of a VPN client service in the UE, while saving signaling over-the-air. As discussed in section 4.2.2, the non-3GPP slice took about $10s$ to be instantiated and the UE took another $9s$ to detect and to attach the slice. For simplification, in the evaluated scenario, the network preemptively instantiated the non-3GPP slice in the user's home network, saving the instantiation delay in the offloading mechanism. For this, the network can exploit localization services of current smartphones or pre-schedule the slice instantiation. In this context, since the vUE works on behalf of the UE for network management decisions, the UE informs the vUE of neighbor cell and/or current location, allowing the network to preemptively take procedures to adapt (or instantiate) slices.

The evaluated scenario was compared with the throughput for a dual interface UE (Wi-Fi and LTE) where no handover mechanisms was applied. Here, the user was attached to a Wi-Fi and LTE networks simultaneously and was accessing the corporation services through VPN. As such, as the UE changes access network, the VPN requires a reconnection (for evaluation proposes $3s$ was assumed) to the server. In this line, when the UE disconnected from the Wi-Fi (at $25s$) the VPN tried to reconnect through the LTE, negatively affecting the achieved throughput. Moreover, when the UE connected to a new Wi-Fi network (at $62s$), another VPN reconnection was required in order to offload the traffic via Wi-Fi. Comparing to the proposed framework, the VPN client scenario originated a throughput loss of 8% between VPN reconnections.

Finally, it is necessary to look for ways to offload the traffic to non-3GPP access (such as the Wi-Fi) while keeping or enhancing the QoS. This gains more value when considering places of social gathering (e.g., shopping centers), since BSs support a limited number of LTE connections due to the finite spectrum that can be used at one

physical location, leading to degraded service quality for customers [92]. Also, studies, such as [93], show that Wi-Fi is typically more energy efficient than LTE, pointing out that offloading to Wi-Fi will usually reduce the UE energy consumption.

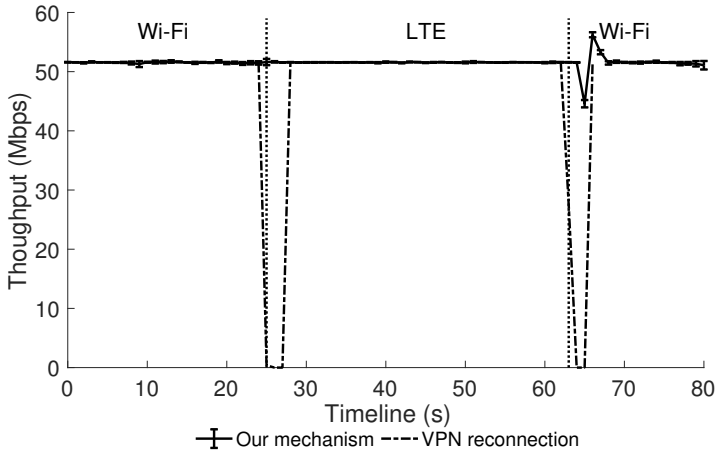



**Figure 4.18:** Throughput over time for the corporate scenario [17].

## 4.4 Chapter Considerations

This chapter introduced network slicing to the architecture proposed in chapter 3, while performing the necessary adaptations for compatibility with 3GPP networks. As such, leveraged by SDN, NFV and network slicing, the framework architecture proposed in this chapter is capable of abstracting the partitioning of the network to verticals, while providing the necessary requirements by dynamically reconfiguring the network and allowing inter-slice mobility in heterogeneous environments. The proposed architecture was implemented for proof-of-concept based in OAI and FlexRAN. Results showcased that, despite some of the mechanisms exposed to an increasing amount of slices were able to cope with the experimented scenario's demands, existing technical approached still need to be further enhanced in order to allow not only a larger number of multiple isolated slices, but also to efficiently handover UEs among slices.

In addition, exploring the core convergence of multiple access technologies (namely, LTE and Wi-Fi), the proposed framework was evaluate for remote working in corporate scenarios. In this context, the proposed framework instantiates a core network slice for a corporation and dynamically re-instantiates it at the target network, as the user moves away from the corporation, whether it is in a 3GPP or non-3GPP technology. This solution was further supported by creating a virtualized representation of the UE, the vUE, which provides support for the mobility process. In this way, the configuration of a VPN client in the UE becomes unnecessary as it is transparent to the user. Here,

a VPN service was defined as a mechanism that allows the remote access to private networks.

Moreover, this chapter incremented the validation of SDN flow-based mobility management mechanisms (research question Q1), supported by a virtualized mobile network architecture (research question Q3) which facilitates a distributed design by allowing dynamic instantiation of NFs (research question Q4) and network slices. These architectural aspects (research question Q5) were enabled by the introduction of the *slice creator* and *slice selector* entities, allowing the network's control entities (e.g., MME and HSS) to be shared among network slices, however still providing the necessary datapath isolation for serving multiple different service type communications (i.e., vertical use cases). Additionally, despite MME maintaining the intra-3GPP mobility responsibility, inter-3GPP mobility decisions were moved to the vUE, allowing (if necessary) to isolated the UE's context, while anchoring the user's data traffic, for both intra- and inter-3GPP handovers (research question Q6 and Q7). Nevertheless, the vUE communicates with the remaining network entities via REST messages, for acquiring user's data requirements (HSS and MME) and requesting the instantiation of network slices (slice creator and slice selector).

Finally, this chapter reuses partial material of the outcomes achieved through the following publications authored by the candidate: scientific journals [17], [21], [23] and international conference proceedings [16], [22].

CHAPTER 5

# Slice Management and Orchestration

"*Network management of entities in 5G systems will be able to automate and orchestrate a range of life cycle management processes, and will be capable of coordinating complex dynamic systems of applications, cloud, transport and access resources.*"

— Ericsson

*This chapter enhances the architecture discussed in the previous chapter, by creating mechanisms for dynamic VNFs and, ultimately, slice instantiation and orchestration. In the light of this, the developed framework leverages SDN, NFV and MANO mechanisms to dynamically instantiate virtual CPEs (vCPEs) and periodically monitor the data traffic in the vCPE for migrating the vCPE (with near-zero downtime) on the basis of predefined data requirements. This is followed by a slice management and orchestration framework (akin Slice Management and Orchestration (SliMANO)) entity for abstracting the instantiation of E2E network slices, which are composed by a chain of both PNFs and VNFs. In this line, the proposed SliMANO framework is a plug-in based system that requests network resources and coordinates the interaction among network orchestration entities for its instantiation and chaining in order to perform an E2E slice. These entities could range from MANO, network controllers and RAN controllers.*

## 5.1 Orchestrating Slice-based Points of Attachments

In telecommunications, the term Customer Premises Equipment (CPE) is usually defined as the equipment installed in the customer's premises to provide network services. Thus, a CPE can have multiple forms and features (e.g., telephone headsets, set-top-boxes or subscriber routers). In this context, here the CPE and vCPE are defined as the Wi-Fi PoA and vPoA (presented in previous chapters), respectively, and equivalent to a home router.

In light of this, and as presented in chapter 2, NFV and SDN have captured network operators' attention due to their flexibility and adaptability to new environments, while promising to overcome the need for proprietary networking hardware and open their approach for new verticals and monetization [94]. Also, service providers become able to optimally position virtualized service functions based on precise user's needs (e.g., entertainment and/or professional usage) while incorporating policies into the decision process [95]. However, as usage requirements associated to CPEs (i.e., from its user or users) vary over time, limitations can manifest from sub-optimal resource allocation or unexpected link usage (e.g., due to the unexpected rise in consumption of bandwidth-demanding applications). As such, it becomes critical to dynamically and flexibly monitor and adapt the CPEs resources based on real-time statistics and analytics. In this context, NFs currently hosted in the physical CPE (pCPE) can be moved to virtual environments, creating virtual counterparts (i.e., vCPEs) that implement the necessary NFs [95]–[97]. Furthermore, telco-operators' services started to be proposed for cloud-based deployment in Platform as a Service (PaaS) environments, allowing for a faster deployment as well as abstracting the developer from the infrastructure [98].

This section proposes a framework that dynamically instantiates vCPEs in a PaaS after the power up of a pCPE. In this context, the vCPE is composed by a chain of containerized VNFs (CVNFs) that can be added or removed as required through SDN mechanisms. Contrary to proposals of the literature, this allows the VMs to serve more than one pCPE without requiring the sharing of VNFs among vCPEs and still ensuring traffic isolation. Also, given this profile-dependency, along with associated functionalities and traffic requirements, the proposed framework integrates MANO-compliant mechanisms, and leverages standardized APIs for dynamic instantiation and removal of vCPEs on multiple Point of Deployments (PoDs) (e.g., core, edge or fog) considering predefined traffic requirements. Additionally, due to the highly dynamic traffic patterns, traffic requirements may change with time, leading to sub-optimal resource usage and service placement. To manage this, the framework is capable of migrating the vCPE to a cluster (or PoD) that best fits optimal requirements.

### 5.1.1 Framework overview

The framework proposal for allowing the vCPE deployment in PaaS environments is depicted in Figure 5.1, where the architecture can be defined in three layers: i) the VIM enables the implementation of an IaaS, allowing the sharing of data-center hardware resources; ii) the NFVO orchestrates the VMs that deploy the VNFs; and iii) the container orchestrator engine (COE) orchestrates and manages a cluster of VMs for deploying CVNFs, resulting in a PaaS environment. Finally, the vCPE is created by chaining multiple CVNFs.

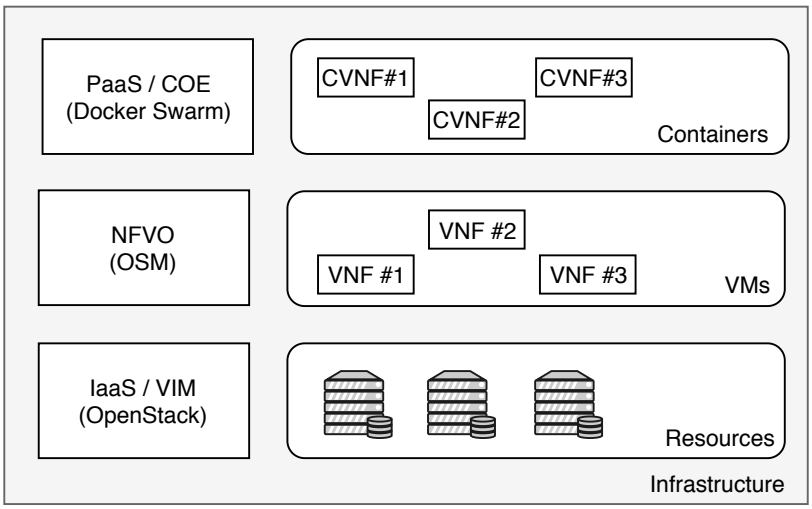



**Figure 5.1:** Proposed architecture for vCPE over PaaS [20].

This results in a framework where the infrastructure owner and the operator are two different entities (as proposed in chapter 4, Figure 4.1). In this context, the infrastructure owner is able to tenant platforms for service deployment to the network operator, avoiding the need for operators to own their own infrastructure. As such, the motivational scenario, considers a use case in which the operator requests a PaaS for deploying vCPEs with pre-established SLAs. In the light of this, the framework features a scenario where the operator provides, in a plug and play approach, a pCPE with predefined requirements (e.g., E2E delay and bandwidth). Therefore, as the pCPE powers up and gains connectivity, it establishes a connection with the network controller (i.e., SDN controller). On verifying the pCPE type and requirements, the controller, in turn, requests the instantiation of a new vCPE with a list of the necessary NFs (e.g., firewall, DHCP and Domain Name System (DNS)) and their location (i.e., type of cluster, edge or core) considering the initial predefined requirements. However, due to the dynamics of the different types of traffic usually imposed on the pCPEs/vCPEs, such requirements may vary over time. As such, the network controller periodically monitors traffic patterns transversing vCPEs, verifying if requirements are being met.

For a proof-of-concept scenario, a subscribing wireless router (hereafter, defined as pCPE) that initially has its CVNFs instantiated in the core's data-center was considered. However, as the end-users change its traffic patterns to services with constrained E2E delays, such as Ultra High Definition (UHD) live streams and/or augmented reality, the vCPE is migrated to the edge's cluster that better suits the necessary requirements (e.g., an edge deployment provides lower latency by employing network resources that are closer to the user).

#### *5.1.1.1 Framework Entities*

Here, a vCPE is defined as one or multiple chained CVNFs instantiated in a virtual environment. In this context, the framework features a scenario (Figure 5.2) where the infrastructure owner tenants a network operator by instantiating a network controller (i.e., the SDN controller) and a cluster of nodes for deployment of CVNFs (event #0). As such, since the operator does not have control over the infrastructure, the infrastructure manager offers a PaaS for the deployment of CVNFs. In this way, as the pCPE powers up and connects to the network, it triggers the SDN controller (event #1), which in turn requests a new vCPE to the service orchestrator (event #2). The VNF Manager triggers the instantiation of the required CVNFs (event #3) for the vCPE, with the SDN controller being notified of their deployment (event #4). Finally, the controller reconfigures the datapath via the SDN substrate, interconnecting the pCPE and the vCPE via a L3 tunnel (event #5&7). The involved network entities are described as follows.

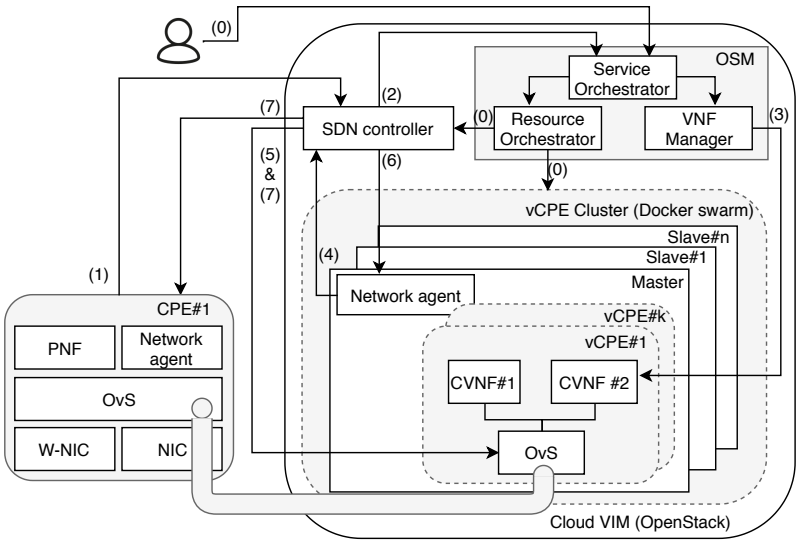



**Figure 5.2:** Architecture overview for vCPE over PaaS [20].

- **MANO:** it instantiates, configures and monitors VNFs in IaaS environments, via pre-built descriptors. It is composed by three fundamental building blocks: (i) the

Service Orchestrator, which manages network services by requesting, scheduling and instantiating VNFs to the Resource Orchestrator, while using the VNF Manager for VNFs' configuration; (ii) the Resource Orchestrator, which requests the instantiation of VNFs to the VIM (e.g., OpenStack); and (iii) VNF Manager, which uses the network descriptors to perform the necessary configuration over the VNFs.

- **Cloud-VIM:** it controls and manages the compute, storage and network resources of the NFV infrastructure. It can accommodate multiple tenants that are seen as different VIMs by the MANO. Here, edge and core cloud-VIMs are similar but providing different E2E delays. Also, it can accommodate multiple network operators, by deploying and offering a PaaS.
- **Network controller:** it ensures network operations features, and requests the instantiation of VNFs and CVNFs to the MANO, while dynamically reconfiguring the datapath and the chaining of CVNFs. As such, it is owned by the operator and manages its PaaS.
- **Cluster:** the cluster is composed of a set of nodes (VMs) enclosing CVNFs instances that have been requested by the network controller (i.e., SDN controller). The cluster of nodes can be seen as a PaaS environment, since it allocates the instantiated CVNFs requested by the operator (via SDN controller). Also, the cluster is able to scale the number of nodes up or down (as necessary), while monitoring their state and alerting the network controller in high-load and migration scenarios.
- **physical CPE (pCPE):** it stands as a physical node that implements PNFs for providing wireless capabilities. Thus, remaining NFs are offloaded and performed in cloud premises (as presented for the "partially virtualized" approach in section 3.1.3).
- **virtual CPE (vCPE):** a virtual entity that enhances its physical counterpart (i.e., the pCPE) capabilities by implementing CVNFs. These CVNFs can be deployed in the centralized cloud, in edge clouds or in the pCPE itself. In the proposed scenario, the vCPE is a chain of containerized VNFs (via SDN mechanisms), deployed over a PaaS.

#### *5.1.1.2 Proof-of-Concept Implementation and Deployment*

This section describes the implementation and deployment details of the proposed framework for proof-of-concept scenarios of section 5.1.2 and section 5.1.3.

The framework was implemented in an in-house data-center running OpenStack Queens as cloud-VIM and OSM (release 5) as MANO. OSM was running in a VM of the data-center with 4 vCPUs and 8 GB of 1 RAM. The cluster of nodes for

vCPE instantiation was implemented using Docker Swarm, enhanced with Prometheus[1] for metric monitoring and featured a developed application for containerized VNFs instantiation when triggered by the OSM. For proof-of-concept purposes the number of nodes of the cluster ranges from 1 to 5, each one with 1 vCPU and 1 GB of RAM.

Triggers to add or remove nodes (i.e., VMs) to the cluster, or to migrate the vCPE among cluster nodes use the CPU workload. Thus, to produce workload, the *stress-ng*[2] tool was used, obligating the CPU to cross the pre-established process usage thresholds and triggering the SDN controller for possible vCPE migration, or the MANO to add or remove cluster's nodes. In this context, the first trigger is sent to the SDN controller when the VM crosses the 85% of workload for possible vCPE migration (i.e., the SDN controller verifies which vCPEs are instantiated in the overloaded node, and depending on the SLA the vCPE can be migrated). When the cluster reaches 85% of workload (and maintains it during 5 minutes), the developed network agent alerts the MANO, which in turn instantiates a new VM and adds it to the cluster. Similarly, when the workload drops to less than 30% (and maintains it during 5 minutes), the MANO is triggered and removes the cluster's node with less load (after migrating the vCPE to another node).

For the scenario presented section 5.1.3, in order to mimic the edge and core data-centers, two different OpenStack tenants were used, providing the vision of different VIMs to the OSM. Also, to emulate different network characteristics for both edge and core, a traffic processing delay (increasing E2E delay) and traffic loss were added to the core's cluster using the *traffic control* Linux application.

The video headend is also specific for section 5.1.3 and was implemented in a VM with 8 vCPU and 16 GB of RAM, using the VLC application as video server. The VLC application allows to transmit and transcode a stored video as a livestream via HTTP. Here, the Big Buck Bunny video in HD and UHD formats was used. The proof-of-concept deployment was evaluated using the VLC application in an One Plus 6 smartphone and an Apple iPad Mini 4 for the UHD and HD video clients, respectively. As network controller, the Ryu SDN controller was used and a SDN application was developed for performing the instantiation, migration and triggering procedures above described.

Finally, the pCPE was implemented in an APU2C4 with 4 GB RAM running Ubuntu server 14.04 LTS. When the pCPE powers up it connects to the network, notifies the controller for initial setup configuration, creating a Wi-Fi network in IEEE 802.11n at 5GHz using the *hostapd* and *isc-dhcp-relay* software in a Docker container. Likewise, the vCPE is instantiated as a set of CVNFs (i.e., DHCP server, DNS and firewall)

[1]Prometheus: https://prometheus.io

[2]stress-ng: https://kernel.ubuntu.com/~cking/stress-ng/

attached to an OvS bridge. VNFs were based on Alpine Linux containers, using the *isc-dhcp-server* and *iptables*.

Next, two different scenarios are evaluated. Section 5.1.2 evaluates a migration scenario where a vCPE is instantiated in an already existent cluster's node and then (due to KPIs) migrated to other node of the cluster. Otherwise, section 5.1.3 initially instantiates a new node into the core's cluster to accommodate the vCPE, however due to the traffic requirements, the vCPE is migrated to the edge's cluster.

### 5.1.2 Migration among cluster's nodes

This sections describes the high-level sequence message for a dynamic instantiated vCPE over a PaaS environment scenario, followed by its evaluation and discussion in terms of instantiation and migration delays.

#### *5.1.2.1 Scenario description*

Figure 5.3 illustrates the high-level signaling for the instantiation of a vCPE upon its power up, and its migration upon a high-load notification from its cluster. Such procedures are described as follows. Note that the following signaling assumes the *day 0* (Figure 5.3, event #0) already occurred, thus the cluster was already instantiated and ready for vCPE deployment.

##### *5.1.2.1.1 vCPE instantiation*

Upon its power up, the CPE tries to establish communication to the SDN controller via OVSDB (message #1). Identifying the pCPE (through its *system_id*), the controller establishes such connection with it and requests to the MANO a node to instantiate the vCPE jointly with the required network functions (message #2). In turn, the MANO (message #3) verifies the load of each node of the cluster and instantiates the required VNFs (as Docker containers) in the node with more available resources at that time (message #4). If needed, the MANO may instantiate a new node attached to the cluster to better support current communications and the instantiation of new vCPEs. After the VNFs' instantiation, the cluster directly notifies the SDN controller (message #6), which in turn starts biding both the pCPE and the vCPE via SDN mechanisms (i.e., OvSDB and OF).

In this context, the controller instantiates a new OvS bridge in both the CPE and its virtual counterpart (message #7), attaches the requested CVNFs to the vCPE bridge (message #8), and creates a L3 tunnel between the pCPE and the vCPE (message #9). Finally, the vCPE is attached to the output bridge (message #10) and flow tables are updated, allowing to offload not only the Internet traffic to/from the pCPE to the vCPE, but actually to offload NFs and perform chaining of CVNFs (message #11).

#### *5.1.2.1.2 vCPE migration*

As described above, this framework monitors the workload of the cluster and scales the number of cluster nodes up and down. This allows the framework to maintain the SLAs initially defined between the infrastructure owner and the operator. As such, the cluster has a monitoring system for collecting metrics (e.g., CPU, memory and input/output load) and alerts the controller when a pre-established threshold is crossed.

For proof-of-concept purposes, a simulated high-load scenario is illustrated. In this line, when the load crosses a configurable predetermined threshold, the cluster alerts the SDN controller (message #12), which in turn verifies the pCPEs/vCPEs instantiated in such node, and if necessary requests to the MANO to migrate the vCPEs with higher priority (message #13). This migration is composed by two main procedures: (i) a new set of VNFs (message #18) attached to a new OvS bridge (message #18:21) is instantiated in a node with more available resources (creating a clone of the vCPE); and (ii) VNFs and the OvS bridge are removed from the old associated node (message #24:26). Between these two stages, the controller updates the tunnel end-point (message #22) and flow tables (message #23) in the pCPE, implementing a *make-before-break* approach.

Note that this procedure is a proof-of-concept for a high-load scenario and that the framework is extensible enough to accommodate more mature procedures and monitoring systems. Additionally, the framework also implements a scale up and down of the number of cluster nodes, even though this work focuses on the pCPE/vCPE architecture and its migration among the cluster's nodes.

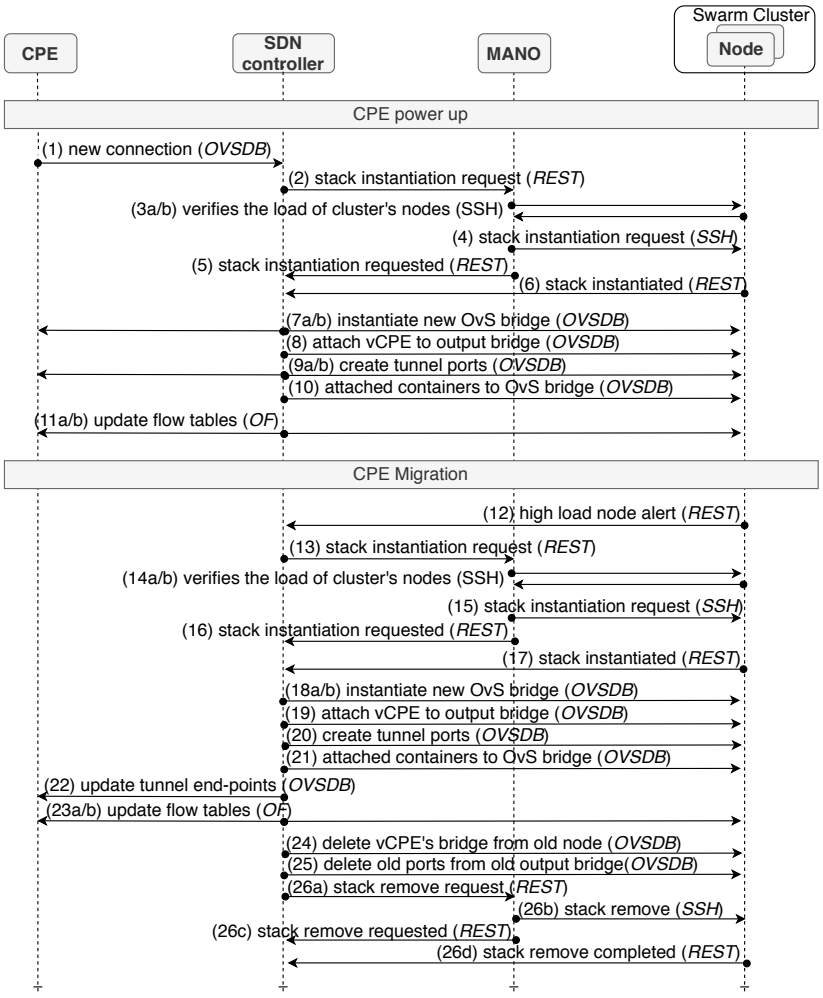


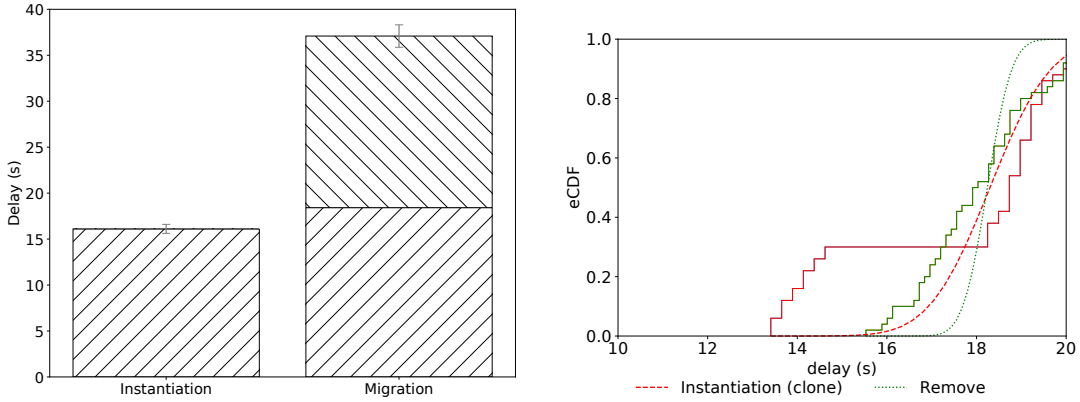

**Figure 5.3:** High-level signaling for instantiation and migration in PaaS architectures [20].

### *5.1.2.2 Scenario evaluation*

This section evaluates the framework proposal in terms of instantiation and migration delay, as well as the throughput. Experiments were run 50 times, with the results presenting their average with a 95% of confidence interval.

#### *5.1.2.2.1 Instantiation delay*

Figure 5.4a presents the overall delay of the instantiation of a vCPE, upon its trigger by the pCPE. Also, Table 5.1 decomposes such delay into the different stages of the instantiation. These stages were measured in the SDN controller and are directly related to the high-level signaling presented in Figure 5.3. To be fully operational, the vCPE took about 16$s$ (from message #2 to #11). This time accounted the instantiation (9.5$s$, from message #2 to #6), OvS bridge configuration (6.28$s$, from message #6 to #10) and messages update (0.14$s$, from message #10 to #11).




**(a)** Overall instantiation and migration. **(b)** vCPE migration delay.

**Figure 5.4:** Instantiation and migration delay of a vCPE [20].

#### *5.1.2.2.2 Migration delay*

The migration delay of the vCPE was measured at the SDN controller, with the overall delay being presented in Figure 5.4a and decomposed in Table 5.1. In addition, Figure 5.4b presents the CDF and eCDF of the migration delay decomposed in the two stages of the procedure (i.e., instantiation of a clone vCPE and removal of the old vCPE, as presented in the previous section). In this line, the complete migration procedure took 37$s$. Nevertheless, this delay can be decomposed in two sub-procedures: first, the instantiation of a clone vCPE in the cluster node with more available resources and the datapath reconfiguration (about 18$s$, from message #13 to #23d); and second, the removal of the original vCPE (about 18$s$, from message #24 to #26d). Thus, the vCPE became operational 18$s$ after the migration request, with the pCPE using this virtual entity as its counterpart to enhance its features and capabilities.

**Table 5.1:** Instantiation and migration delays [20].

| | Messages (Fig.5.3) | Delay (s) |
|---|---|---|
| Instantiation | from #2 to #6 | 9.46 (±0.25) |
| | from #6 to #7 | 2.63 (±0.29) |
| | from #7 to #8 | 0.14 (±0.00) |
| | from #8 to #9 | 0.66 (±0.01) |
| | from #9 to #10 | 2.85 (±0.06) |
| | from #10 to #11 | 0.14 (±0.00) |
| Migration | from #12 to #13 | 0.13 (±0.03) |
| | from #13 to #22 | 18.42 (±1.07) |
| | from #22 to #23 | 0.01 (±0.00) |
| | from #23 to #24 | 0.19 (±0.01) |
| | from #24 to #25 | 0.07 (±0.00) |
| | from #25 to #26a | 0.11 (±0.01) |
| | from #26a to #26d | 18.30 (±0.43) |

#### *5.1.2.2.3 Migration impact over throughput*

In order to measure the impact of the migration procedure in ongoing throughput, an experiment featuring both UDP and TCP streams (generated using the iperf3[3] tool) was conducted. This experimented was conducted 50 times, however since the migration trigger occurs in different points of experiment time, Figure 5.5 illustrates an aleatory selected run. As discussed in section 5.1.1.2, the migration is triggered by increasing the CPU load on the VM where the vCPE resides.

For the UDP traffic, the throughput was not negatively affected by the migration of the vCPE among VMs of the cluster. In Figure 5.5, the workload of the VM crossed the pre-established threshold at 12$s$ of the experiment runtime, with the cluster's network engine triggering the SDN controller at that time (message #12, in Figure 5.3). The migration was completed after 29$s$ of experiment runtime (message #23, in Figure 5.3). After this, the old vCPE was removed, however with the pCPE already using the "cloned" vCPE as its virtual counterpart.

Regarding to TCP traffic, Figure 5.5 shows that despite a "*make-before-break*", a minimal throughput decreased occurred at second 29 during 1.5$s$. This behavior was mainly due to the fact that after migrating the vCPE to a new cluster node, a gratuitous Address Resolution Protocol (ARP) was sent towards the network to update the new L2 location of the vCPE.

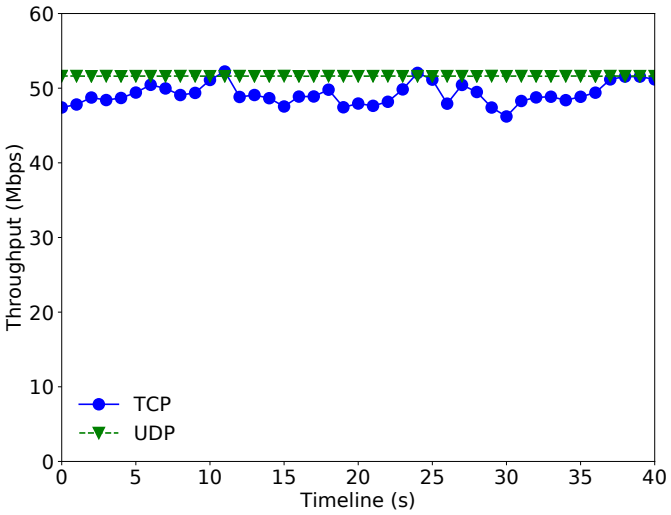



**Figure 5.5:** Throughput over time in a migration scenario [20].

### 5.1.3 Migration among Point of Deployment

In this section, the framework was extended in order to allow the vCPE migration among PoDs. As such, the instantiation process of vCPEs remains similar to the

[3]iperf3: https://iperf.fr

previous scenario, but with the added benefit of selecting the initial PoD (e.g., core) and migrating it to another location (e.g., edge) considering current user requirements. Figure 5.6 presents an overview of the proposed scenario. Next the high-level sequence message for a dynamic instantiation of a vCPE is described, followed by its evaluation in terms of impact over an on-going livestream video in a migration between PoDs event.

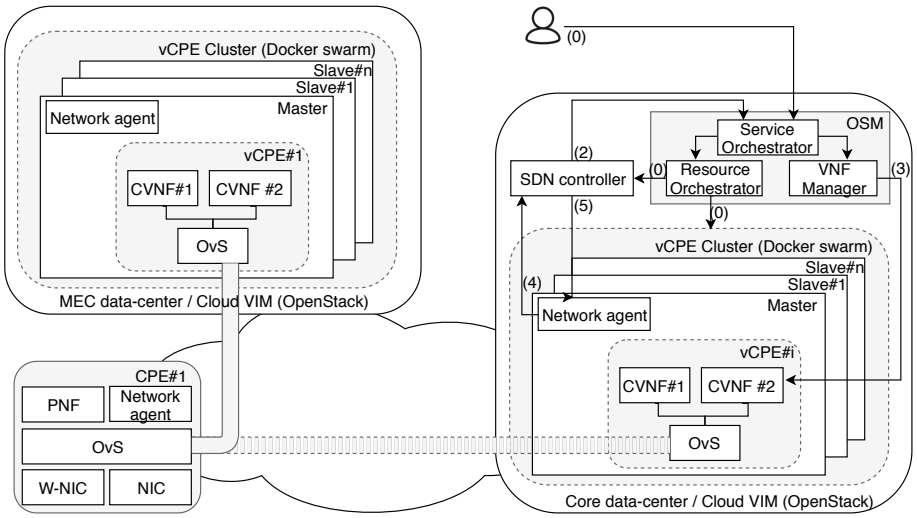



**Figure 5.6:** Migration among PoDs scenario overview [19].

### *5.1.3.1 Scenario description*

Figure 5.7 presents the instantiation and migration high-level signaling of a vCPE upon a new pCPE connection. The use case scenario assumes that the edge's cluster provides an environment optimized for traffic with more constrained requirements when compared to the core's cluster, by offering less E2E delay and less traffic and computational workload.

#### *5.1.3.1.1 vCPE Instantiation*

In a scenario where pCPE is predefined with less constrained traffic requirements, the SDN controller requests the dynamic instantiation of its NFs in the core network via MANO. Similarly to the previous scenario (section 5.1.2), first, the MANO requests the instantiation of a new node (message #2 and #3). Then, it waits for the new node (i.e., VM) to be joined to the cluster (message #4), and requests the CVNFs instantiation (message #5). After the instantiation of the CVNFs (message #8), the cluster notifies the SDN controller, which in turn proceeds to bind the pCPE and vCPE via SDN mechanisms (OVSDB and OF). Thus, an OvS bridge is instantiated in both pCPE and vCPE (message #9a/b) along with a tunnel to redirect data traffic from the

pCPE towards its virtual counterpart (message #10a/b). Here, the VNFs are deployed in containers and attached to the OvS bridge (message #11), while the service chain is ensured via OF flow-based rules (message #12a/b).

#### *5.1.3.1.2 Traffic awareness and migration*

Despite being initially deployed either in the core or edge data-centers, this framework allows vCPEs to be migrated among them. This is especially important when considering pCPEs/vCPEs with highly dynamic traffic. For example, in a subscriber router (pCPE) scenario, one may infer that it usually will be used for web browsing traffic, therefore initially deploying it in the core data-center. However, livestreaming and online gaming are gaining more followers, where E2E delay is a high valued requirement. In this line, users may infer that the imposed delay by virtualizing the pCPE in the cloud will downgrade the QoE. In this context, in a scenario where the vCPE is initially instantiated in a cluster with less restricted data traffic requirements (i.e., core's cluster), but due the high dynamic traffic consumption it may require more powerful resources or better E2E delay, this framework migrates the vCPE to a new cluster via SDN-based mechanisms, with near-zero downtime and without negatively affecting the QoE.

In the proof-of-concept, the end-user (i.e., UE) requests an UHD livestream (message #13a/b), while in the background the SDN controller monitors the traffic requirements of the vCPE via OpenFlow (message #14a/b). Detecting that data requirements are changing and that the vCPE is no longer capable of ensuring such requirements, the SDN controller migrates the vCPE to a cluster able to fulfill the necessary requirements (e.g., E2E delay). Such procedure is composed by two stages: first, the instantiation of a clone vCPE in the edge's cluster (from message #15 to message #23); second, the removal of the initial (old) vCPE of the core's cluster (from message #25 to message #26d), to free resources. Nevertheless, the livestream is redirected as soon as the edge's vCPE is ready (message #23a/b, where the traffic is redirected via OF flow-based rules). Next, the monitor and video detection mechanism for the proof-of-concept scenario is presented.

#### *5.1.3.1.3 Video traffic detection*

For proof-of-concept purposes, the livestream video detection was implemented leveraging SDN mechanisms. The SDN controller periodically (each 5s) monitors the traffic that is currently transversing the vCPEs via OF messages (OF *stats* message, Figure 5.7 message #14a/b). Therefore, the SDN controller requests the current flow statistics (e.g., bitrate, protocol, ports) and verifies if it matches with a livestream flow (triggering a vCPE migration). Nevertheless, note that this mechanism was developed for proof-

of-concept purposes, and that the framework does not mandate a specific detection procedure, being flexible enough to the accommodate more robust mechanisms for predictive data traffic detection.

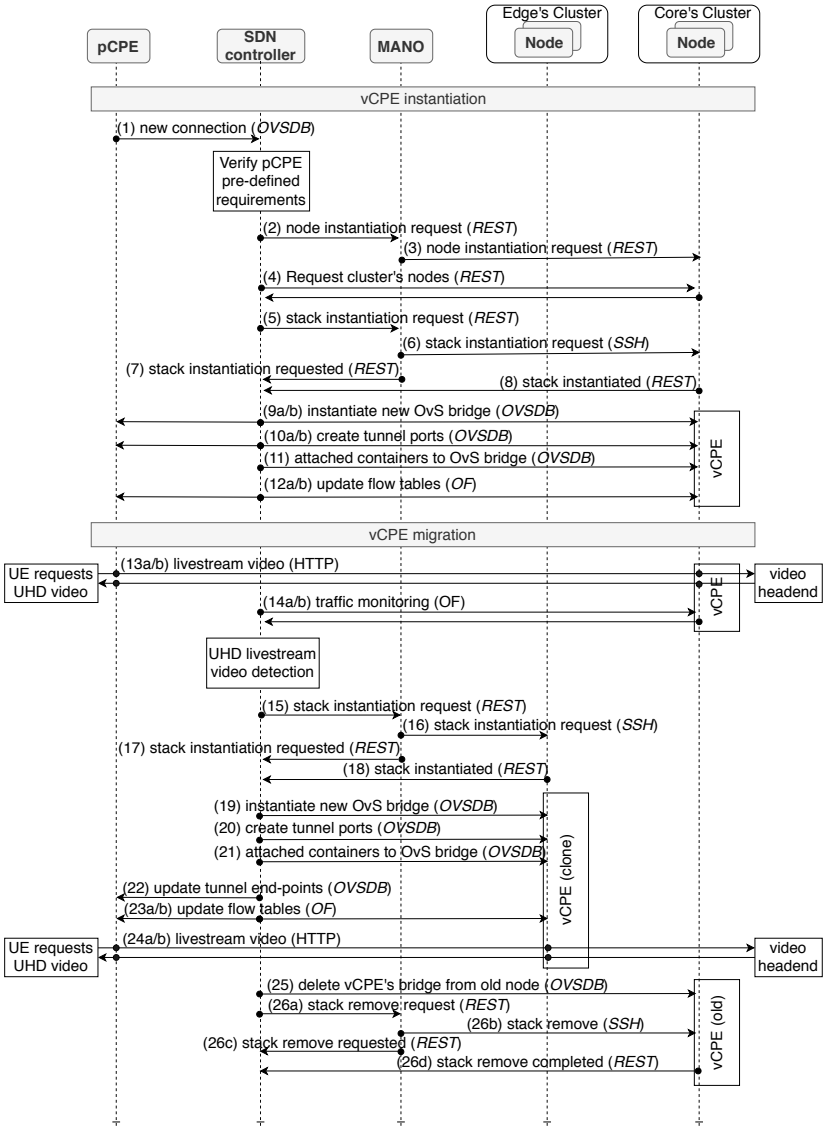



**Figure 5.7:** High-level signalling for vCPE instantiation and migration among PoDs [19].

### *5.1.3.2 Scenario evaluation*

This section evaluates the framework proposal in terms of instantiation and migration delay, as well as the vCPE live migration impact over the QoE of a HD and UHD livestream video. Experiments were run 50 times, with the results presenting their average with a 95% of confidence interval. Additionally, a demo[4] was recorded and is publicly available online.

[4]Demo: https://atnog.github.io/5G-VCoM/demos/vcpe.html

#### *5.1.3.2.1 Instantiation delay*

The instantiation delay of a new vCPE was measured in both edge and core clusters. Table 5.2 decompose such delays in three main procedures which are directly related to Figure 5.7: (i) cluster node (or VM) instantiation (from message #1 to #4b); (ii) the instantiation of the CVNFs (from message #5 to #8); and (iii) the service chain configuration via SDN mechanisms (from message #8 to #12a/b).

From Table 5.2 it is noted that more than 60% of the instantiation delay was related to the new cluster node (i.e., an OpenStack VM) instantiation (i.e., 63s). To this, it is added the CVNFs instantiation, accounting for 20% of the delay. Finally, the edge's cluster presented similar results. The main difference resides on the CVNFs instantiation, mainly due to the fewer configuration parameters.

#### *5.1.3.2.2 Migration delay*

For the migration delay, in order to avoid the $63s$ of the VM instantiation, an available node was preemptively instantiated in the edge's cluster for the vCPE's CVNFs deployment. In this line, the SDN controller took about $10s$ for video detection[5], while the vCPE clone instantiation and the video redirection (from message #15 to #24) took $18s$. Thus, the framework took about $28s$, since the user requested the video until the vCPE was migrated to the edge's cluster. Finally, the core's vCPE was deleted, and took almost $25s$.

**Table 5.2:** vCPE instantiation and PoD migration delays [19].

| | Messages (Fig.5.7) | Delay (s) |
|---|---|---|
| Instantiation | from #1 to #4b | 63.14 (±0.60) |
| into core's POD | from #5 to #8 | 22.25 (±0.53) |
| | from #8 to #12a/b | 16.34 (±0.42) |
| Instantiation | from #1 to #4b | 62.86 (±0.63) |
| into edge's POD | from #5 to #8 | 10.64 (±0.35) |
| | from #8 to #12a/b | 14.11 (±0.51) |
| Migration | from #13 to #14 | 10.54 (±1.09) |
| from | from #15 to #24 | 18.26 (±2.38) |
| core to edge | from #25 to #26d | 24.84 (±1.64) |

[5]Video detection delay: this delay results from the fact that this framework requests the flow statistics with 5s periodicity, and usually at least 2 messages are required for the detection.

#### *5.1.3.2.3 End-to-end delay*

As mentioned before, in our proof-of-concept deployment, the main difference between core and edge clusters is the E2E delay. Thus, the edge's cluster provides a reduced E2E delay (when compared with the core's cluster), making it more suitable for scenarios with more constrained delays, such as livestreaming, online gaming and augmented reality.

For proof-of-concept, and since both clusters were deployed in an in-house datacenter, a delay of $100ms$ and 5% packet loss on the core's cluster nodes (section 5.1.1.2) was imposed. As such, the RTT[6] between an attached UE and the video headend was measured, transversing both the core and edge clusters. The edge's cluster presented $3.96(\pm 12.66)ms$, while the core's presented $166.87(\pm 65.07)ms$. As expected, due to the random loss of packets in the core's cluster, the QoE of the end-user was negatively impacted (discussed in the next sub-section). Conversely, the E2E delay increase did not negatively impact the video QoE, nor significantly delayed it, with the UE's hardware being a key contributor to the performance of the decoding process.

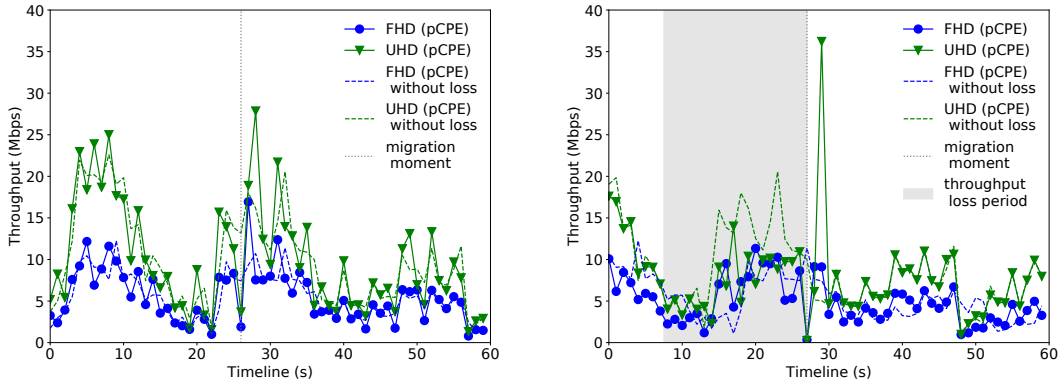
#### *5.1.3.2.4 Video throughput impact*

Since the migration was not triggered at the same experiment time in all the runs, Figure 5.8a illustrates the video throughput (captured in the pCPE for both HD and UHD) of just one run (the one recorded for the online demo) where the migration occurred at 26s of the experiment. Additionally, throughputs are compared, in Figure 5.8b, corresponding to a scenario where the core's cluster begins (at 7s) to randomly loss packets with a 5% of probability.

In this context, Figure 5.8a compares the achieved video throughput of a migration scenario with a scenario without packet loss. At the moment of migration (at 26s), the downtime, although near-zero, made the server re-adjust the throughput in the followed seconds (from 26s to 29s), and return to the regular throughput after stabilizing. Similarly, Figure 5.8b illustrates the impact of the packet loss at the core's cluster (from 7s to 27s), with the video throughput returning to expected values after the migration to the edge's cluster (at 27s).

Finally, for both HD and UHD livestreams, the migration of the vCPE to the edge's cluster, was performed with a near-zero downtime and did not negatively impact the QoE of end-users (as can be seen on the online demo). In fact, in the scenario of Figure 5.8b, the QoE was actually increased, since the videos throughput was resumed

[6]The RTT was measured by running 1000 Internet Control Message Protocol (ICMP) packets, with results showing their average delay and standard deviation. A symmetric delay for both directions was assumed.




**(a)** Video throughput of the online demo. **(b)** Video throughput imposing packet loss on the core's cluster.

**Figure 5.8:** Live migration impact in on-going HD and UHD livestreams [19].

to the expected values. In this way, the migration mechanism allows the framework to instantiate vCPEs with pre-established SLAs and when sub-optimal characteristics are detected, it migrates and re-dimensions the vCPEs as the service levels change over time.

#### *5.1.3.2.5 Final remarks*

Comparing the vCPE architecture with a regular CPE implementation, where all NFs are performed in the physical hardware, this proposal presents a greater degree of flexibility and modularization, since VNFs can be dynamically added and removed to/from the chain as needed. Notwithstanding, in terms of E2E delay, the proposed architecture presented an increase of 39% of delay ($3.35(\pm 8.95)ms$ of RTT), when compared to a similar architecture but without virtualization ($2.41(\pm 8.30)ms$ of RTT). Nevertheless, as illustrated in Table 5.3, the global registered throughput[7] was not negatively impacted for both UDP and TCP types of traffic.

**Table 5.3:** Comparison of virtualized- and non-virtualized CPE approaches.

| CPE approach | Bandwidth (Mbps) | TCP Re-transmissions |
|---|---|---|
| non-virtualised | 79.85 (±1.6) | 142 (±8) |
| virtualised | 78.20 (±0.75) | 90 (±4) |

[7]The throughput was measured using the iperf3 tool to generate the traffic. Results present the average with a 95% confidence interval of 100 runs.

The ETSI Industry Specification Group for NFV [99] has evidenced the need for on-demand deployment of network virtualization and softwarization capabilities, in order to enable the different service types. In this line, the NFVO orchestrates VNFs by deploying and configuring VNFs in a predefined set of VIMs, allowing the creation of a network slices. Also, ETSI standardized the functionalities of network components, with solutions for slicing orchestration from both academia and open-source organizations (such as ETSI itself and Linux Foundation) following these specifications (presented in section 2.2.2.1).

Works such as [100]–[102] envision automated mechanisms to deploy and manage network slices for cloud-based mobile networks. Moreover, existing solutions, such as OSM and FlexRAN, are limited to a specific scope: while some are more focused on RAN slice management, others are focused on slices based on network functions chaining within a data-center leaving PNFs out-of-scope. Notwithstanding, E2E network slices are composed by multiple slices (e.g., a radio slice chained with an infrastructure slice), which are orchestrated by different entities. As such, there is a need for a network entity able to interact and coordinate the overall network slicing life cycle management. In this context, the proposed architecture of chapter 4 already considered such interactions with the different entities to be performed by an SDN application along with the Ryu SDN controller. However, the diversity of SDN controllers and MANO frameworks creates the urge to develop a third-party entity for supporting such diversity.

In the light of this, this section proposes a framework that follows the 3GPP specification [52] for the management and orchestration of E2E network slices. The proposed framework, named SliMANO, is an ETSI-compliant E2E network slice manager and orchestrator that abstracts network slicing actions (e.g., instantiation, decommission and reconfiguration) from the responsible network orchestration entities (i.e., MANO, SDN controller and RAN controller). When compared with the architecture proposed in chapter 4, SliMANO replaces and enhances the *slice creator* and *slice selector* entities (which were implemented as SDN applications).

### 5.2.1 SliMANO's Overview

The automated deployment of E2E slices and the multi-domain of its service orchestration requires a new level of abstraction of the initial proposed framework. As such, SliMANO is presented as a plug-in system of network modules that enables the slice orchestration to be agnostic of both MANO and VIM frameworks and, ultimately, of SDN controllers. Figure 5.9 illustrates the motivational scenario, where for deploying E2E network slices over the physical network, different networks need to operate and

coordinate actions. For example, while the NFVO instantiates network services in data-centers and the RAN controller manages 3GPP radio slices, the SDN controller re-configures the datapath interconnecting PNFs and VNFs. Figure 5.10 depicts the SliMANO architecture divided in three main building blocks, namely the SliMANO Core, SliMANO Plug-in Framework and SliMANO Agents Framework.

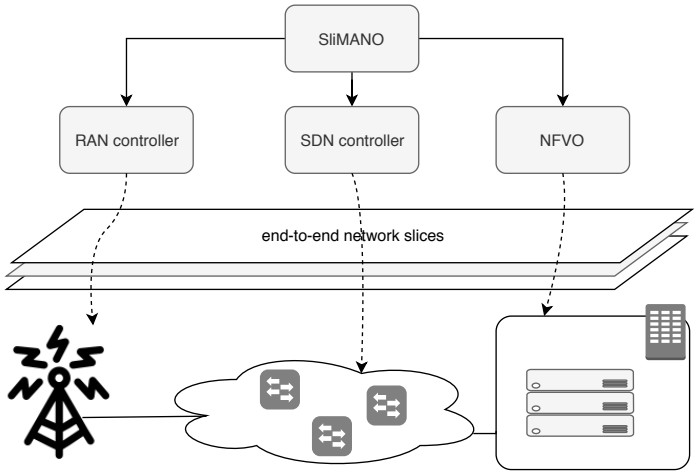



**Figure 5.9:** SliMANO's motivational scenario [24].

#### *5.2.1.1 SliMANO Core*

The *core* building block is responsible for coordinating SliMANO's functionalities. In this line, currently it implements three main components, described as follows.

- **Slice life-cycle management:** It handles operations related to network slice resources, such as their update (to modify the resource's behavior) and deletion.
- **Slice monitoring:** This module is responsible for monitoring the resources allocated to each network slice instance. In case of resource failure, it notifies the slice life-cycle management, which in turn verifies how to recover from the failure (if possible). In addition, the slice monitoring module manages the slice resources and its capability to meet the imposed QoS. This results in a highly scalable service assurance system and effective closed-loop life-cycle management.
- **Slice orchestration:** It verifies the availability of resources for the instantiation of a network slice. Also, it is responsible for requesting the necessary resources to the entitled entities (e.g., MANO, SDN controller, FlexRAN controller). Resources can be a NFVO Network Service (NS), a SDN application or a network slice in a RAN.

#### *5.2.1.2 SliMANO Plug-in Framework*

This block allows SliMANO to be a generic framework, by building an API between the *core* (via plug-ins) and *agents*, facilitating the continuous development of new plug-ins

and respective agents. Note that each plug-in has its correspondent *agent* for external communication.

- **NFVO plug-in:** It builds a contract to communicate with NFVOs. For example, in a scenario where OSM is the NFVO, the plug in uses the OSM agent for performing the necessary operations involving OSM (e.g., instantiate and delete NSs).
- **Network controller plug-in:** Similarly, it uses the respective agent to request operations to the network controllers (e.g., reconfigure the datapath and establish QoS), such as SDN controllers.
- **RAN plug-in:** It performs operations (via agents) on 3GPP-based network slices (e.g., re-dimension RAN slices).

#### *5.2.1.3 SliMANO Agents Framework*

As mentioned above, the agents framework is coupled with the plug-in framework. Thus, it performs the actions requested by plug-ins to the respective external network entities.

- **NFVO agents:** The NFVO agents perform the actions on the respective NFVO external entity. As such, it is required to develop an agent for each supported NFVO (e.g., OSM, ONAP, Cloudify, etc.). Usually, actions to the NFVOs are performed via REST APIs.
- **Network controller agents:** Similarly, a network controller agent for each supported controller (e.g., ODL, ONOS, Ryu), in order to request network operations, such as the reconfiguration of the datapath for interconnecting VNFs.
- **RAN agents:** The RAN agent requests actions to the 3GPP network. For example, the FlexRAN exposes a REST API for radio slice instantiation, reconfiguration and removal.
- **Northbound Interface (NBI):** Finally, the NBI allows a client (e.g., a network entity, network operator or network administrator) to request the instantiation of a network slice. This request is made via REST, with the SliMANO's NBI translating the request to the necessary network operations.

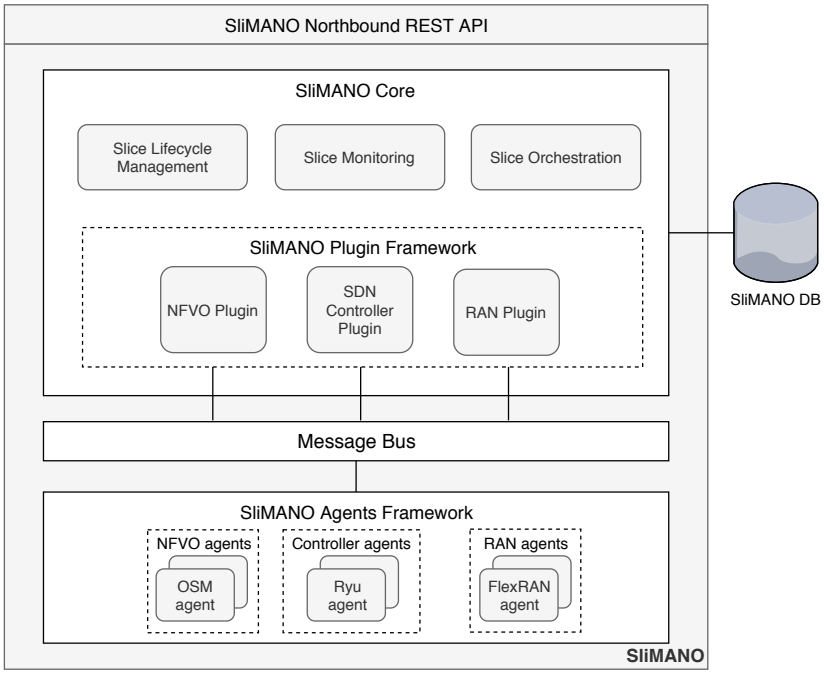



**Figure 5.10:** SliMANO's architecture [24].

### 5.2.2 High-level of sequence message for instantiation and delete action

This section presents a proof-of-concept scenario where the SliMANO instantiates a network slice in the data-center via a supported NFVO. Here, a network slice is defined as an E2E chain of NSIs. Figure 5.11 depicts the high-level signaling of such procedure.

#### *5.2.2.1 Network slice instantiation*

As explained in the previous section, the NBI translates the request to network operations. The instantiation of a NSI is divided into two main procedures, namely deployment and configuration.

First, on the engine side, SliMANO verifies the dependencies of the requested NSI with other NSIs, deploying such dependencies if necessary. As such, SliMANO verifies the availability of a plug-in for deploying the required NFV via a supported NFVO, and a plug-in for a support network controller to apply the required network actions. Fulfilling the dependencies, SliMANO internally employs a set of workers for the deployment of the necessary resources into the correspondent external entities (such as, the NFVO). Here, each resource deployment is managed by an engine worker, which in turn is also responsible for ensuring the necessary operations conformity. After each operation completion, the respective worker informs the result (success or fail) to the core engine. Note that, usually, each resource operation execution is done by only one worker, which communicates with a plug-in for applying actions to external entities (e.g., the NFVO). The resources request to the external network entities may involve its configuration.

Alternatively, it is possible to apply further configuration through actions defined in the payload of the NSI request.

Finally, the network slice's configuration is stored in the database, and feedback is retrieved to the network slice requester (i.e., client) using the user provided callback.

#### *5.2.2.2 Network slice deletion*

Similarly to the instantiation procedure, for deleting a NSI the client requests it through SliMANO's NBI via REST. The NBI validates the payload and proceeds with a request to the core's engine. The engine gets the NSI's information from the database, and starts a Life-cycle Management (LCM) instance for the delete action. In turn, the LCM instantiates a delete task for each resource of the NSI. These tasks contact the corresponding agent (via its core plug-in) to perform the delete action via the respective resource manager (i.e., the NFVO)[8].

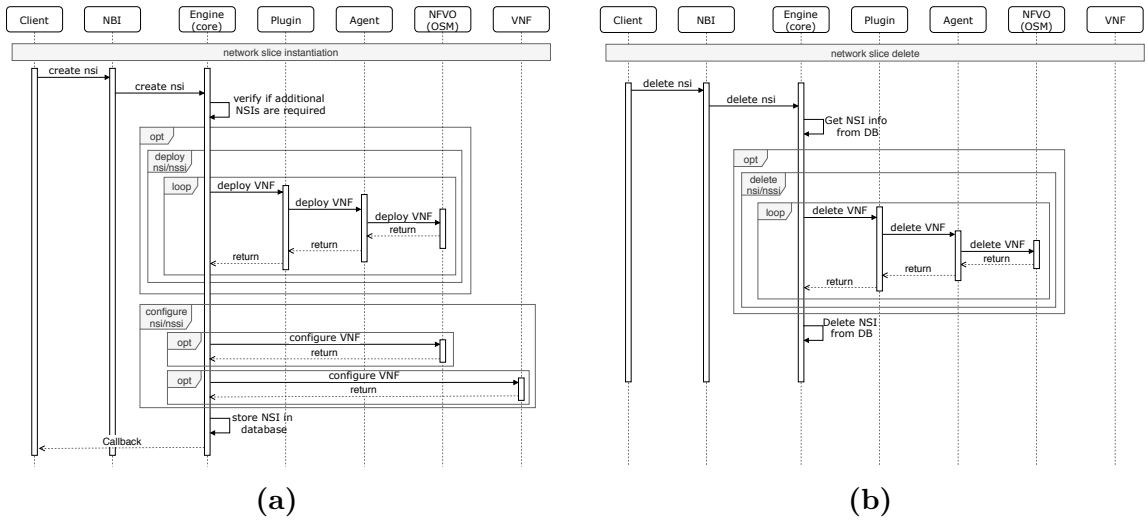



**Figure 5.11:** High-level signaling for NSI: (a) instantation; and (b) deletion [24].

### 5.2.3 Proof-of-Concept Implementation and Evaluation

This section presents the implementation details of the SliMANO architecture, followed by the proof-of-concept deployment of the scenario described in section 5.2.2 and illustrated in Figure 5.11.

The proof-of-concept scenario was deployed in an in-house data-center running OpenStack Queens as VIM and OSM release 5 as MANO[9]. Regarding computational resources, OSM was deployed in a VM with 4 vCPUs and 8GB of RAM, and SliMANO was deployed in a VM with 2 vCPUs and 4GB of RAM. For Docker containers orchestration, Docker Compose was used with the containers being deployed in a single

[8]In the current version, the delete action callback was not implemented. Thus, the client needs to verify the correct operation by requesting the available NSIs via SliMANO's NBI.

[9]ONAP was not tested due to system requirements restrictions.

host. Finally, in order to compare SliMANO's performance in terms of overall delay, the scenario was also deployed using the NetSlice feature of OSM introduced in release 5 [103], which allows the instantiation of network slices.

#### *5.2.3.1 SliMANO vs OSM NetSlice: Functional comparison*

SliMANO aims to be a generic and modular solution and in order achieve that, SliMANO base architecture was built upon microservices. Due to that fact, it provides a high degree of flexibility in component development. Hence, the possibility of deploying new plug-ins/agents to support new kinds of network entities requires low effort.

The previous mentioned characteristic is a key comparison point between SliMANO and OSM's NetSlice module. OSM NetSlice is an OSM internal module that only has an interface with OSM's internal components and, therefore, suffers from limited expandability. The module processes network slice templates, which are composed by a set of NSs and their interconnection definition. This interconnection can be made internally in the OSM target if all the NSs described in the template are located in that OSM instance. The interconnection could be made between different OSM instances if there are NSs that need to be deployed on different OSM instances. For that case, OSM has a component called WAN Infrastructure Manager (WIM) that is responsible for configuring a multi-site network to interconnect those NSs. At the time of this writing, in the tested OSM version, this component is in development stage and still cannot be used.

The SliMANO can achieve the same OSM WIM functionality by describing the interconnection in the slice template, with the former using the configured plug-ins/agents to configure it. Nevertheless, like the OSM WIM, this functionality is under development. The main advantage of SliMANO over OSM's NetSlice module is the fact that, as mentioned above, SliMANO is not dependent of any internal framework interface, component or framework. And as a consequence of that, SliMANO is more expandable than OSM's NetSlice module, as it has the capability to support different kinds of frameworks and network premises with a relatively low development effort.

Regarding to standards fulfillment, both solutions follow 3GPP slice management specification [52]. Thus, both of them are on par in that regard.

#### *5.2.3.2 SliMANO's implementation*

SliMANO was developed in Python and implemented following a microservices architecture. Here, the Nameko[10] framework was used, offering Remote Procedure Call (RPC) services over Advanced Message Queuing Protocol (AMQP). In this line, SliMANO's *core* and *agent* components were implemented in different Docker containers, and using

[10]Nameko: https://www.nameko.io

the RabbitMQ[11] message broker for asynchronous messaging. Finally, for the database, MariaDB[12] was used. The microservices architecture in conjunction with Docker containers increases deployment flexibility, allowing each SliMANO's component to be deployed in a single host or dispersed among hosts.

#### *5.2.3.3 Scenario Evaluation and Discussion*

This section experimentally evaluates SliMANO framework and compares the results with the NetSlice feature of OSM release 5. The proof-of-concept scenario deploys multiple OSM NSs with 1 VNF, which in turn is composed by 2 Virtual Deployment Units (VDUs) deployed on Openstack VIM as VMs. For evaluation purposes, the NS is a Kubernetes[13] cluster that contains a VNF. with a master and a worker node as VDUs. The experiments were run 50 times, with results presenting their average with a confidence interval of 95%.

##### *5.2.3.3.1 Network slice deployment delay*

Table 5.4 compares the SliMANO's overall instantiation and delete delays of a network slice with the NetSlice feature of OSM release 5.

As can be seen, for both instantiation and delete actions, and independently of the number of the NSs (and, consequently, VNFs with their VDUs) per slice, the SliMANO and OSM presented similar results. Nevertheless, despite SliMANO presented slightly faster deploys when more than 2 NSs were considered, it showed slightly longer values for a single NS deployment, and for network slice deletion. However, such delay increments were all under 3 seconds, which are negligible when compared against the overall procedure delay.

**Table 5.4:** Overall OSM and SliMANO delay for instantiation and delete of a network slice [24].

| Orchestrator | Action | 1 NS[seconds] | 2 NS[seconds] | 5 NS[seconds] | 10 NS[seconds] |
|---|---|---|---|---|---|
| OSM | Instantiation | 58.00±1.45 | 107.29±1.75 | 258.64±2.96 | 499.53± 2.33 |
| | Delete | 20.16±0.01 | 28.39±1.11 | 51.49±0.94 | 95.09±1.44 |
| SliMANO | Instantiation | 60.55±1.16 | 108.74±1.69 | 254.87±5.77 | 498.07±5.32 |
| | Delete | 22.70±0.06 | 29.99±1.47 | 53.71±0.42 | 94.13± 2.91 |

[11]RabbitMQ: https://www.rabbitmq.com
[12]MariaDB: https://mariadb.org
[13]Kubernetes: https://kubernetes.io/

#### *5.2.3.3.2 Internal components delay*

In order to evaluate the delay imposed by SliMANO in the overall network slice instantiation procedure, the time delay of SliMANO's internal components for different network slice dimensions was measured. Thus, Table 5.5 presents the instantiation and delete delays imposed by SliMANO's NBI and core components for network slices of 1, 2, 5 and 10 NSs. As mentioned above, each NS is composed by 1 VNF with 2 VDUs. Thus, for example, in the 10 NS scenario, SliMANO instantiates 10 VNFs and 20 VDUs.

**Table 5.5:** SliMANO's components delay for instantiation and delete of a network slice [24].

| Component | Action | 1 NS[ms] | 2 NS[ms] | 5 NS[ms] | 10 NS[ms] |
|---|---|---|---|---|---|
| NBI | Instantiation | 169.02±3.66 | 165.14±3.16 | 168.38±3.07 | 166.36± 2.76 |
| | Delete | 175.12±2.36 | 178.38±3.11 | 174.96±2.90 | 176.30±3.50 |
| Core | Instantiation | 274.48±5.06 | 291.74±4.81 | 463.54±7.63 | 719.58±25.42 |
| | Delete | 294.74±5.14 | 326.44±6.31 | 388.90±10.26 | 524.28±12.46 |

Comparing the values of instantiation and delete actions, it is noted that SliMANO imposed similar delays for both instantiation and deletion actions. This results from the fact that both actions have similar procedures (as illustrated in Figure 3.2.2). Additionally, the overall delay imposed by SliMANO is composed by the NBI and core engine. The delay of the SliMANO's NBI remained constant for the evaluated slice dimensions. Contrarily, the SliMANO's core increased its delay as the number of NSs (and consequently VNFs and VDUs) increased. This was mainly due to the fact that as the number of deployed NSs increases, the operations related with thread creation for each NS at the beginning and the ones related to the persistence at the end of deploying are taking more processing time to accomplish. Moreover, Nameko RPC framework adds some significant amount of delay, mainly because it sets up a new connection to the message broker for each remote service (in this case a service corresponds to one agent), which impacts efficiency.

#### *5.2.3.3.3 Final remarks*

OSM and ONAP are two well known open-source MANO architectures that recently added the slice management capabilities. However, such capabilities are restricted to their internal functionalities, such as its own NFVO. Contrarily, SliMANO offers an external solution agnostic of both the physical and virtual environment, allowing the integration of different NFVOs, SDN controllers and RAN controllers. Nevertheless, being an external solution, SliMANO adds a degree of delay, mainly due to the com-

munications among network entities[14]. However, this cost can be seen as negligible when compared with the overall instantiation delay of a network slices (Table 5.4). In contrast, SliMANO offers a lightweight and independent solution for slice management and orchestration that abstracts the operator from the NFVOs and other network resource managers.

## 5.3 Chapter Considerations

This chapter enhanced the architecture proposed in chapter 4 by adopting MANO mechanisms. This allowed the framework to flexibly and dynamically instantiate network entities and services. A proof-of-concept scenario where upon a connection of a SDN-enabled pCPE with wireless access capabilities, the framework dynamically instantiates the necessary NFs in the cloud over a PaaS. In this way, an infrastructure owner is able to define a SLA for a Docker Swarm cluster of VMs and tenant it to an operator. In turn, the operator has full control of the data-path configuration via SDN mechanisms to migrate the vCPE among nodes of the cluster (or even among clusters) with near-zero downtime, allowing to balance the workload of the cluster.

In addition, a new framework entity was introduced to the architecture presented in chapter 4 for replacement of the *slice creator* and *slice selector*. The network slice management and orchestration (akin, SliMANO) entity is a plug-in based system that follows the ETSI specifications in order to operate with multiple MANO and VIM entities, providing a new level of abstraction to the deployment of network slices. Initial results shown that delays result from the NBI and are associated to SliMANO operating as an entity that is external to the orchestrator. Nonetheless, such delays are negligible in regards to the total duration of an orchestration operation, and are a result from the added flexibility the framework provides in comparison with other solutions, enabling the development of new slice mechanisms agnostic to the underlying network, VIM and MANO entities and procedures.

In this context, this chapter integrated SDN, NFV and slicing solutions with MANO-complaint mechanisms enhancing the architecture design by enabling the dynamic instantiation of network entities and VNFs distributing the datapath in multiple PoDs (research question Q4). This can be also transported to the control path, by instantiating control entities (e.g., the vUE) in other PoDs. Lastly, the introduction of the SliMANO enhanced how network orchestrators (MANO, RAN and SDN controller) request network slices through REST communications (research question Q8), allowing mobility management entities (such as the vUE) to explore the existence of multiple

[14]As shown in results, the major delay was imposed by HTTP REST interfaces of both the NBI and SBI to communicate with the other components (e.g. NFVO, SDN controller).

slices for handovering users (or flows) between access networks and slices (research question Q6).

Finally, this chapter reuses partial material of outcomes achieved through publications in international conference proceedings [19], [20], [24] authored by the candidate.

CHAPTER 6

# Conclusion and Future Directions

"*A Cloud-Native 5G Architecture is Key to Enabling Diversified Service Requirements.*"

— Huawei Technologies Co.

*The main goal of this thesis was to contribute to the 5G research community by developing and architectural framework capable of abstracting mobility mechanisms of involved endpoints in heterogeneous and multi-slice wireless environments, leveraged by SDN, NFV and slicing technologies. Chapter 2 introduced such technologies, presenting as well the related work, specially when associated to wireless scenarios. The framework architecture basis was presented in chapter 3, and further extended to support slicing mechanisms and infrastructure orchestration, in chapter 4 and chapter 5, respectively. To conclude, this thesis made efforts to contribute towards the upcoming 5G networks by studying how SDN and NFV can be used to enhance and abstract mobility mechanisms in heterogeneous and multi-slice environments, as well as mask underlying slices into a single end-to-end slice.*

## 6.1 Review of Achievements

The thesis explored the feasibility of mobility management procedures through the adoption of SDN and NFV mechanism, while focusing network slicing architectural aspects. Also, MANO mechanisms were integrated envisioning a standardised management and orchestration of network resources, and a flexible deployment and maintenance of NFV services. As presented in chapter 1, this study allowed the candidate to contribute to national and intentional projects, as well to the research community through publications as book chapter [14], scientific journals [15], [17], [21], [23] and international conference proceedings [11]–[13], [16], [18]–[20], [22], [24], authored by the candidate.

In this line, the candidate initially applied softwarization and visualization concepts for an holistic architecture where the core network was access-agnostic. Here, the virtualization of the UE (i.e., the vUE) was proposed and evaluated in different use cases and triggered scenarios [12], [13]. Also, following the cloufication trend, not only the UE was virtualized but also the wireless AP, allowing the control and management of the APs to be realized in the vAP. The virtualization of both the AP and the UE added a greater degree of simplicity on the cross-technology handover, since most of the control signaling was performed in cloud, where UE's wireless context was analyzed, potentially benefiting from the enhanced computational power and the anchoring of the traffic. These studies resulted in a book chapter [14]. Also, taking advantage of the fact the current smartphones are equipped with multiple sensors and wireless technologies (e.g. accelerometer and bluetooth) and that the wireless context was processed in cloud, it was developed a mechanism where selective parts of the UE's context were moved to nearby devices, such as smart TVs or set-top-boxes [15].

The initial architecture was agnostic to the wireless access technology and able to dynamically instantiate virtual instances of both APs and UEs in the cloud, while (remotely and on-the-fly) redirecting data traffic among wireless access technologies. Nevertheless, beyond this flexibility and holistic vision, 5G networks aim the capability of being tailored to the emerging highly demand services (e.g., virtual reality). As such, concepts of network slicing were introduced to this architecture, and mechanisms where non-3GPP slices were dynamically instantiated for data (and video) offloading [16] were proposed. Also, such mechanisms were extended in order to enable the instantiation of network slices for allowing the access to corporation's internal services avoiding the use of VPN clients on user devices (i.e., the UE) [17]. Detecting that non-3GPP slices, specifically Wi-Fi slices, does not guarantee the pre-established bandwidth for "greedy" UDP upstream traffic, the use of SDN-enabled UEs allows the network (through the vUE) to dynamically set different service requirements directly in the UE via SDN SB interfaces [22]. Extending this work, the mechanism was tested under

scenarios where the 3GPP network adapts the slices depending of the current traffic of the UE, while jumping between 3GPP and non-3GPP networks [23]. In [21], the architecture framework was discussed aiming to provide insight on the challenges and impact associated with the deployment of an increasing amount of slices, using the same available infrastructural resources to instantiate sub-slices tailored to use cases and vertical tenants.

In parallel, the candidate developed architectures involving CPE and its virtualization (i.e., vCPE) [18]–[20]. Here, the candidate proposed SDN-based architectures for virtualizing the network functions of CPE, while seamless and transparently offloading the data traffic to the users. Also, this was supported by MANO-complaint mechanisms, not only the dynamic instantiation of the vCPE, but also its migration upon network triggers. Finally, the proposed network architecture was enhanced by the introduction of the slice management and orchestration entity (akin, SliMANO) [24]. Here, SliMANO is responsible for the life-cycle management of end-to-end slices, by instantiating, updating and deleting them NSIs. Also, SliMANO abstracts specificities of the network orchestrators (i.e., MANO, RAN and SDN controllers) from the slice requester.

As result, the thesis answers initial self-imposed questions as follows.

### 6.1.1 Fulfillment of research questions

*Q1. What will be the actual benefits of SDN flow-based mobility management control in 5G environments?*

The deployment of SDN flow-based mobility management allows a greater degree of flexibility and granularity, while contributing to a more homogenized network in terms of protocols. As presented in chapter 3, SDN-based mechanisms can be applied in wireless networks to enable handovers between different wireless interfaces, with the added benefit of providing remote management functionalities. Results showcased inter-technology flow-based handover with no packet loss and reduced delay. Also, in chapter 4, the same mechanisms were deployed over slice-based and 3GPP-complaint networks, demonstrating its feasibility.

*Q2. Which NFV architectural mechanisms should be deployed (and devised) to support the integration of SDN in wireless environments?*

NFV along with SDN is a key enabler for future slice-based networks. As presented in chapter 3, and further explored in chapter 4 and chapter 5, NFV allows to move network functions to the cloud, enabling network decisions to be performed therein, and facilitating the traffic redirection between access networks and slices. In this context,

NFV mechanisms such as infrastructure virtualization add a greater degree of flexibility to SDN in wireless environments by simplifying network devices hardware and offloading wireless control decisions (e.g., attachment and association) to the cloud.

*Q3. How will a composed SDN and NFV solution perform for wireless and mobile environments?*

While SDN enables inter-technology handovers, NFV allows a flexible anchoring of the datapath in different network points. Furthermore, the addition of network slicing allows the extension of logically isolated networks towards the end-devices avoiding the usage of tunnels over-the-air in certain use cases, resulting in seamless inter-technology and inter-slice handovers with no packet loss, showcasing its feasibility. In this context, different scenarios were implemented and experimentally evaluated, where: (i) in chapter 3, a NFV was used to virtualize the UE and SDN for data-path re-configuration on the fly, allowing the specific flows of the UE to be redirected to nearby devices; and (ii) in chapter 4, a corporate scenario leverages SDN and network slicing mechanisms for reducing overhead over-the-air.

*Q4. How would such a solution evolve into a pure distributed design, while still maintaining its cloud-based behavior?*

In virtualized network architectures, mechanisms can be developed in order to dynamically instantiate VNFs in different network locations (e.g., core, edge or fog), allowing to distribute (and redistribute) the network entities (or NFs) across the cloud network as necessary. In chapter 5, the migration of VNFs was experimentally evaluated in terms of impact on on-going traffic. Results showcased a greater degree of flexibility and modularization of virtualized equipments, by enabling their dynamic instantiation, reconfiguration and adding the VNFs to the function chain as needed.

*Q5. Which would be the key architectural aspects of a SDN/NFV-based wireless/mobile enhanced control infrastructure that would be transited into a network slice architecture, and which would be the requirements and benefits involved?*

The flexibility provided by SDN and NFV makes them the key enablers for the deployment of slice-based networks. While NFV enables the virtualization of NFs, SDN enables the dynamic reconfiguration of the datapath for chaining PNFs and VNFs, that ultimately creates a logically-isolated network for supporting a certain service type (i.e., a network slice). This allows MNOs to run their networks in IaaS environments and dynamically reconfigure their network to serve certain use cases.

*Q6. How should mobility management be implemented in a Network Slice architecture and which resources should be shared between slices?*

In this thesis, the UE's context was virtualized into the core network, and named as vUE. This vUE was capable of interacting with remaining network entities in order to perform handovers and request slices. In this line, the vUE is considered to be an isolated entity dealing with the control decisions of its physical counterpart (i.e., UE), with the UE's datapath being anchored in strategic points of the network slice (chapter 4). To retrieve information about available networks the vUEs interact with an SDN controller entity, which is shared among the different network slices.

*Q7. What level of isolation should mobility management have, should it be an independent building block?*

For 5G slice-based networks [52], the 3GPP states that shared and non-shared core network functions can be integrated into shared access network slices. Considering this, 5G architectures allow the configuration of network slices optimized to serve certain use cases, where an isolated mobility management function offers traffic isolation and granularity, while a shared function offers greater scalability. In this line, in this thesis, the vUE is considered an isolated entity responsible for the mobility management decisions of the physical UE (providing a greater degree of granularity of active data flows). Nevertheless, in the architecture of chapter 3, the EPC's MME was conceived as a control entity shared among access network slices of the same MNO serving multiple UEs. From 5G perspective, similarly to the architecture of chapter 3, the AMF can also be shared among slices of the same access network.

*Q8. How will a mobility management building block communicate with the remaining blocks of a network slice?*

As discussed in chapter 2, the 3GPP envisions 5G network systems with service-based interactions [25]. This approach simplifies the context exchange between entities, by providing a modular framework capable of accommodate multiple suppliers. As such, the mobility management building block (in this thesis, the vMN/vUE for inter-technology and inter-slice mobility) is able to interact with remaining network entities for slice related events (e.g., update and/or instantiation) through HTTP-based communications, such as REST (as implemented in chapter 4 and chapter 5).

## 6.2 Future Directions

Telecommunication environments have been evolving towards a SBA. In fact, in release 15 [25] the 3GPP already considers the service provisioning through HTTP-based micro-services the 5G system architecture. The adoption of such service-based models allows 5G core network operations to evolve towards cloud-native operations for interaction of among highly dynamic virtualized services instances.

As such, with the current tendency for virtualization, it is expected that future mobile systems will adopt SBA and micro-services for both radio and core networks, as well as for data-centres and its virtual services. Also, such architecture envisions a holistic vision of a combined fixed/wireless access over a highly functional Anything as a Service (XaaS) infrastructure, including terminal devices. As presented in chapter 3, section 3.2, this opens a path towards a deviceless communications for optimizing user experiences and where the execution of tasks are not coupled with a single device and may be moved to any suitable device which has exposed the corresponding micro-service identifiers.

Nevertheless, this decomposition into micro-services across the network brings a greater degree of complexity for ensuring execution correctness, while properly resolving runtime resource conflicts. Moreover, the devices (such as smartphones) become more volatile and operating in a multi-ownership by allowing resource sharing among users. In this context, the vision of a multi-access and multi-ownership SBA for fixed/wireless environments creates highly dynamic cloud-native network, but with great challenges in terms of execution correctness.

# References


[1] Cisco, *Cisco Visual Networking Index: Forecast and Trends, 2017-2022 White Paper*, Accessed on 27.03.2019, Feb. 2019. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html.

[2] ——, *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017–2022 White Paper*, Accessed on 27.03.2019, Feb. 2019. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-738429.html.

[3] NGMN, Next Generation Mobile Networks Alliance, *5G White Paper*, Fev 2015. [Online]. Available: https://www.ngmn.org/fileadmin/ngmn/content/images/news/ngmn_news/NGMN_5G_White_Paper_V1_0.pdf.

[4] Ericsson, *5G Systems White Paper*, Jan. 2015. [Online]. Available: https://www.ericsson.com/res/docs/whitepapers/what-is-a-5g-system.pdf.

[5] ——, *Ericsson Mobility Report: Mobile traffic Q1 2016*, Jun. 2016. [Online]. Available: https://www.ericsson.com/res/docs/2016/ericsson-mobility-report-2016.pdf.

[6] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, 'OpenFlow: Enabling Innovation in Campus Networks', *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008, ISSN: 0146-4833.

[7] B. Pfaff and B. Davie, 'The open vswitch database management protocol', 2013.

[8] A. Gudipati, D. Perry, L. E. Li, and S. Katti, 'SoftRAN: Software Defined Radio Access Network', HotSDN '13, pp. 25–30, 2013. DOI: 10.1145/2491185.2491207. [Online]. Available: http://doi.acm.org/10.1145/2491185.2491207.

[9] Y. Cai, F. Yu, and S. Bu, 'Cloud Radio Access Networks (C-RAN) in Mobile Cloud Computing Systems', in *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*, Apr. 2014, pp. 369–374. DOI: 10.1109/INFCOMW.2014.6849260.

[10] X. J. L. E. Li, L. Vanbever, and J. Rexford, 'CellSDN: Software-defined Cellular Core Networks', 2013.

[11] F. Meneses, C. Guimares, D. Corujo, and R. L. Aguiar, 'SDN-based Mobility Management: Handover Performance Impact in Constrained Devices', in *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, Feb. 2018, pp. 1–5. DOI: 10.1109/NTMS.2018.8328716.

[12] F. Meneses, D. Corujo, C. Guimaraes, and R. L. Aguiar, 'An abstraction framework for flow mobility in multi-technology 5G environments using virtualization and SDN', in *2017 IEEE Conference on Network Softwarization (NetSoft)*, Jul. 2017, pp. 1–5. DOI: 10.1109/NETSOFT.2017.8004217.

[13] F. Meneses, C. Guimarães, D. Corujo, and R. L. Aguiar, 'Handover Initiation Comparison in Virtualised SDN-based Flow Mobility Management', in *2018 IEEE Symposium on Computers*

*and Communications (ISCC)*, Jun. 2018, pp. 00 404–00 409. DOI: 10.1109/ISCC.2018.8538696.

[14] F. Meneses, C. Guimarães, D. Corujo, and R. L. Aguiar, 'Experimental Wireless Network Deployment of Software-Defined and Virtualized Networking in 5G Environments', in *Emerging Wireless Communication and Network Technologies: Principle, Paradigm and Performance*, K. V. Arya, R. S. Bhadoria, and N. S. Chaudhari, Eds. Singapore: Springer Singapore, 2018, pp. 335–360, ISBN: 978-981-13-0396-8. DOI: 10.1007/978-981-13-0396-8_17. [Online]. Available: https://doi.org/10.1007/978-981-13-0396-8_17.

[15] F. Meneses, C. Guimarães, T. Magalhães, D. Gomes, D. Corujo, and R. L. Aguiar, 'Deviceless Communications: Cloud-Based Communications for Heterogeneous Networks', *Wireless Personal Communications*, vol. 100, no. 1, pp. 25–46, May 2018, ISSN: 1572-834X. DOI: 10.1007/s11277-018-5621-9. [Online]. Available: https://doi.org/10.1007/s11277-018-5621-9.

[16] F. Meneses, R. Silva, D. Santos, D. Corujo, and R. L. Aguiar, 'Using SDN and Slicing for Data Offloading over Heterogeneous Networks Supporting non-3GPP Access', in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Sep. 2018, pp. 1–6. DOI: 10.1109/PIMRC.2018.8580969.

[17] ——, 'An Integration of Slicing, NFV and SDN for Mobility Management in Corporate Environments', *Transactions on Emerging Telecommunications Technologies*, May 2019.

[18] J. Filipe, F. Meneses, A. Rehman, D. Corujo, and R. L. Aguiar, 'A performance comparison of containers and unikernels for reliable 5g environments', in *2019 15th International Conference on the Design of Reliable Communication Networks (DRCN) - Special Session on Disaster Resilience of Communication Networks*, 2019, pp. 1–5.

[19] F. Meneses, M. Fernandes, T. Vieira, D. Corujo, S. Figueiredo, A. Neto, and R. L. Aguiar, 'Traffic-aware live migration in virtualized cpe scenarios', in *IEEE Workshop on Mobility Support in Slice-based network control for heterogeneous environments (MOBISLICE) (IEEE NFV-SDN workshops)*, 2019, pp. 1–6.

[20] F. Meneses, M. Fernandes, T. Vieira, D. Corujo, A. Neto, and R. L. Aguiar, 'Dynamic modular vcpe orchestration in platform as a service architectures', in *IEEE 8th International Conference on Cloud Networking (CloudNet)*, 2019, pp. 1–6.

[21] F. Meneses, R. Silva, D. Corujo, and R. L. Aguiar, 'Micro and Macro Network Slicing: an experimental assessment of the impact of increasing numbers of slices', *Wireless Personal Communications*, May 2019.

[22] F. Meneses, D. Corujo, A. Neto, and R. L. Aguiar, 'Sdn-based end-to-end flow control in mobile slice environments', in *IEEE Workshop on Mobility Support in Slice-based network control for heterogeneous environments (MOBISLICE) (IEEE NFV-SDN workshops)*, Nov. 2018, pp. 1–5.

[23] F. Meneses, R. Silva, D. Corujo, A. Neto, and R. L. Aguiar, 'Dynamic network slice resources reconfiguration in heterogeneous mobility environments', *Internet Technology Letters*, May 2019.

[24] F. Meneses, M. Fernandes, D. Corujo, and R. L. Aguiar, 'Slimano: An expandable framework for the management and orchestration of end-to-end network slices', in *IEEE 8th International Conference on Cloud Networking (CloudNet)*, 2019, pp. 1–6.

[25] 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects (3GPP-TS), *System Architecture for the 5G System*, 2018.

[26] R. Silva, 'Offloading mechanisms for mobile networks using SDN in virtualized environments', Master's thesis, Universidade de Aveiro, Portugal, 2017.

[27] 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects (3GPP-TS), *General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access (Release 16) (3GPP TS 23.401)*, 2018.

[28] O. N. Foundation, *SDN architecture [online]*, 2013. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_SDN_ARCH_1.0_06062014.pdf.

[29] ONF, Open Networking Foundation, *TR-256 Applying SDN Architecture to 5G Slicing*, Apr. 2016.

[30] R. Jain and S. Paul, 'Network Virtualization and Software Defined Networking for Cloud Computing: a Survey', *IEEE Communications Magazine*, vol. 51, no. 11, pp. 24–31, 2013.

[31] W. H. Chin, Z. Fan, and R. Haines, 'Emerging technologies and research challenges for 5G wireless networks', *Wireless Communications, IEEE*, vol. 21, no. 2, pp. 106–112, 2014.

[32] O. N. F. W. M̃. W. Group, *WMWG Charter Application [online]*, 2013. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/working-groups/charter-wireless-mobile.pdf.

[33] Y. Yiakoumis, J. Schulz-Zander, and J. Zhu, *Pantou : OpenFlow 1.0 for OpenWRT*, 2011. [Online]. Available: http://www.openflow.org/wk/index.php/OpenFlow%5C_1.0%5C_for%5C_OpenWRT.

[34] K. Pentikousis, Y. Wang, and W. Hu, 'Mobileflow: Toward software-defined mobile networks', *Communications Magazine, IEEE*, vol. 51, no. 7, pp. 44–53, Jul. 2013.

[35] X. Jin, L. E. Li, L. Vanbever, and J. Rexford, 'SoftCell: Scalable and Flexible Cellular Core Network Architecture', in *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '13, ACM, 2013.

[36] C. Guimaraes, D. Corujo, R. Aguiar, F. Silva, and P. Frosi, 'Empowering Software Defined Wireless Networks through Media Independent Handover Management', in *Global Communications Conference (GLOBECOM), 2013 IEEE*, Dec. 2013, pp. 2204–2209.

[37] C. Guimaraes, D. Corujo, and R. Aguiar, 'Enhancing OpenFlow with Media Independent Management capabilities', in *Communications (ICC), 2014 IEEE International Conference on*, Jun. 2014, pp. 2995–3000.

[38] C. Bernardos, A. De La Oliva, P. Serrano, A. Banchs, L. Contreras, H. Jin, and J. Zúniga, 'An Architecture for Software Defined Wireless Networking', *Wireless Communications, IEEE*, vol. 21, no. 3, pp. 52–61, Jun. 2014, ISSN: 1536-1284. DOI: 10.1109/MWC.2014.6845049.

[39] P. D. et al., 'BEST-AP: Non-intrusive estimation of available bandwidth and its application for dynamic access point selection', *Computer Communications*, vol. 39, pp. 78–91, 2014.

[40] J. Lee, M. Uddin, J. Tourrilhes, S. Sen, S. Banerjee, M. Arndt, K.-H. Kim, and T. Nadeem, 'meSDN: Mobile Extension of SDN', in *Proceedings of the Fifth International Workshop on Mobile Cloud Computing & Services*, ser. MCS '14, Bretton Woods, New Hampshire, USA: ACM, 2014, pp. 7–14, ISBN: 978-1-4503-2824-1.

[41] F. Meneses, D. Corujo, C. Guimarães, and R. L. Aguiar, 'Multiple Flow in Extended SDN Wireless Mobility', in *2015 Fourth European Workshop on Software Defined Networks*, Sep. 2015, pp. 1–6. DOI: 10.1109/EWSDN.2015.52.

[42] F. Meneses, D. Corujo, C. Guimaraes, and R. L. Aguiar, 'Extending SDN to End Nodes Towards Heterogeneous Wireless Mobility', in *2015 IEEE Globecom Workshops (GC Wkshps)*, Dec. 2015, pp. 1–6. DOI: 10.1109/GLOCOMW.2015.7414073.

[43] N. Makris, K. Choumas, C. Zarafetas, T. Korakis, and L. Tassiulas, 'Forging Client Mobility with OpenFlow: an experimental study', in *IEEE Wireless Communications and Networking Conference*, Apr. 2016.

[44] P. Bispo, 'A Software Defined Network Controller Quantitative and Qualitative Analysis', Master's thesis, Universidade de Aveiro, Portugal, 2017.

[45] B. A. A. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka, and T. Turletti, 'A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks', *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1617–1634, Third 2014, ISSN: 1553-877X. DOI: 10.1109/SURV.2014.012214.00180.

[46] V.-G. Nguyen, T.-X. Do, and Y. Kim, 'SDN and Virtualization-based LTE Mobile Network Architectures: A comprehensive survey', *Wireless Personal Communications*, vol. 86, no. 3, pp. 1401–1438, 2016.

[47] L. Suresh, J. Schulz-Zander, R. Merz, A. Feldmann, and T. Vazao, 'Towards Programmable Enterprise WLANS with Odin', in *Proceedings of the first workshop on Hot topics in software defined networks - HotSDN '12*, New York, New York, USA: ACM Press, Aug. 2012, p. 115, ISBN: 9781450314770. DOI: 10.1145/2342441.2342465.

[48] P. Dely, J. Vestin, A. Kassler, N. Bayer, H. Einsiedler, and C. Peylo, 'CloudMAC - An OpenFlow based Architecture for 802.11 MAC layer processing in the Cloud', in *2012 IEEE Globecom Workshops*, IEEE, Dec. 2012, pp. 186–191, ISBN: 978-1-4673-4941-3. DOI: 10.1109/GLOCOMW.2012.6477567.

[49] M. Bansal, J. Mehlman, S. Katti, and P. Levis, 'OpenRadio: A Programmable Wireless Dataplane', HotSDN '12, pp. 109–114, 2012. DOI: 10.1145/2342441.2342464. [Online]. Available: http://doi.acm.org/10.1145/2342441.2342464.

[50] R. Mijumbi, J. Serrat, J. Gorricho, S. Latre, M. Charalambides, and D. Lopez, 'Management and orchestration challenges in network functions virtualization', *IEEE Communications Magazine*, vol. 54, no. 1, pp. 98–105, Jan. 2016, ISSN: 0163-6804. DOI: 10.1109/MCOM.2016.7378433.

[51] G. N. ETSI, *ETSI GS NFV 002 V1.2.1 Network Functions Virtualisation (NFV); Architectural Framework*, 2014.

[52] 3GPP, *Study on management and orchestration of network slicing for next generation network*, spec.: 28.801 version: 15.1.0, Jan. 2018.

[53] R. Sherwood, M. Chan, A. Covington, G. Gibb, M. Flajslik, N. Handigol, T.-Y. Huang, P. Kazemian, M. Kobayashi, J. Naous, *et al.*, 'Carving research slices out of your production networks with OpenFlow', *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 1, pp. 129–130, 2010.

[54] 3GPP, *Study on Architecture for Next Generation System*, Jun. 2015.

[55] S. Wang, X. Wu, H. Chen, Y. Wang, and D. Li, 'An optimal slicing strategy for SDN based smart home network', in *Smart Computing (SMARTCOMP), 2014 International Conference on*, Nov. 2014, pp. 118–122. DOI: 10.1109/SMARTCOMP.2014.7043848.

[56] M. Richart, J. Baliosian, J. Serrat, and J.-L. Gorricho, 'Resource slicing in virtual wireless networks: A survey', *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 462–476, 2016.

[57] Y. Zaki, L. Zhao, C. Goerg, and A. Timm-Giel, 'LTE wireless virtualization and spectrum management', ser. Wireless and Mobile Networking Conference (WMNC), 2010 Third Joint IFIP, IEEE, 2010, pp. 1–6.

[58] ——, 'LTE mobile network virtualization', *Mobile Networks and Applications*, vol. 16, no. 4, pp. 424–432, 2011.

[59] M. Li, L. Zhao, X. Li, X. Li, Y. Zaki, A. Timm-Giel, and C. Gorg, 'Investigation of network virtualization and load balancing techniques in LTE networks', ser. Vehicular Technology Conference (VTC Spring), 2012 IEEE 75th, IEEE, 2012, pp. 1–5.

[60] A. Banchs, P. Serrano, P. Patras, and M. Natkaniec, 'Providing throughput and fairness guarantees in virtualized WLANs through control theory', *Mobile Networks and Applications*, vol. 17, no. 4, pp. 435–446, 2012.

[61] K. Nakauchi, Y. Shoji, and N. Nishinaga, 'Airtime-based resource control in wireless LANs for wireless network virtualization', ser. Ubiquitous and Future Networks (ICUFN), 2012 Fourth International Conference on, IEEE, 2012, pp. 166–169.

[62] D.-E. Meddour, T. Rasheed, and Y. Gourhant, 'On the role of infrastructure sharing for mobile network operators in emerging markets', *Computer Networks*, vol. 55, no. 7, pp. 1576–1591, 2011.

[63] M. Jiang, M. Condoluci, and T. Mahmoodi, 'Network slicing management amp; prioritization in 5G mobile systems', in *European Wireless 2016; 22th European Wireless Conference*, May 2016, pp. 1–6.

[64] K. Samdanis, X. Costa-Perez, and V. Sciancalepore, 'From network sharing to multi-tenancy: The 5G network slice broker', *IEEE Communications Magazine*, vol. 54, no. 7, pp. 32–39, Jul. 2016, ISSN: 0163-6804. DOI: 10.1109/MCOM.2016.7514161.

[65] X. Zhou, R. Li, T. Chen, and H. Zhang, 'Network slicing as a service: enabling enterprises' own software-defined cellular networks', *IEEE Communications Magazine*, vol. 54, no. 7, pp. 146–153, Jul. 2016, ISSN: 0163-6804. DOI: 10.1109/MCOM.2016.7509393.

[66] Q. Wang, J. Alcaraz-Calero, M. B. Weiss, A. Gavras, P. M. Neves, R. Cale, G. Bernini, G. Carrozzo, N. Ciulli, G. Celozzi, *et al.*, 'SliceNet: End-to-End Cognitive Network Slicing and Slice Management Framework in Virtualised Multi-Domain, Multi-Tenant 5G Networks', in *2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, IEEE, 2018, pp. 1–5.

[67] A. de la Oliva, X. Li, X. Costa-Perez, C. J. Bernardos, P. Bertin, P. Iovanna, T. Deiss, J. Mangues, A. Mourad, C. Casetti, *et al.*, '5G-TRANSFORMER: Slicing and Orchestrating Transport Networks for Industry Verticals', *IEEE Communications Magazine*, vol. 56, no. 8, pp. 78–84, 2018.

[68] N. Nikaein, R. Knopp, F. Kaltenberger, L. Gauthier, C. Bonnet, D. Nussbaum, and R. Ghaddab, 'OpenAirInterface: an open LTE network in a PC', ser. Proceedings of the 20th annual international conference on Mobile computing and networking, ACM, 2014, pp. 305–308.

[69] X. Foukas, N. Nikaein, M. M. Kassem, M. K. Marina, and K. Kontovasilis, 'FlexRAN: A flexible and programmable platform for software-defined radio access networks', in *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*, ACM, 2016, pp. 427–441.

[70] S. Sarkar, C. Becker, J. Kunz, A. Sarbhai, G. Annasamymani, S. K. Kasera, and J. V. d. Merwe, 'Enabling WiFi in Open Access Networks', in *Proceedings of the 4th ACM Workshop on Hot Topics in Wireless*, ACM, 2017, pp. 13–17.

[71] K. Koutlia, A. Umbert, S. Garcia, and F. Casadevall, 'RAN slicing for multi-tenancy support in a WLAN scenario', in *Network Softwarization (NetSoft), 2017 IEEE Conference on*, IEEE, 2017, pp. 1–2.

[72] J. Pérez-Romero, O. Sallent, R. Ferrús, and R. Agustí, 'On the configuration of radio resource management in a sliced RAN', in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, IEEE, 2018, pp. 1–6.

[73] C. E. Perkins, 'IP mobility support for IPv4, revised', 2010.

[74] A. Rasem, M. St-Hilaire, and C. Makaya, 'A comparative analysis of predictive and reactive mode of optimized PMIPv6', in *2012 8th International Wireless Communications and Mobile Computing Conference (IWCMC)*, IEEE, 2012, pp. 722–727.

[75] S. Gundavelli, K. Leung, V. Devarapalli, K. Chowdhury, B. Patil, *et al.*, 'Proxy mobile ipv6', 2008.

[76] J. C. Zúniga, C. J. Bernardos, A. de la Oliva, T. Melia, R. Costa, and A. Reznik, 'Distributed mobility management: a standards landscape', *IEEE Communications Magazine*, vol. 51, no. 3, pp. 80–87, 2013.

[77] H. Guo, L. Wang, B. Duan, C. Liu, and Z. Liu, 'LMA/HA Discovery Mechanism on the interaction between MIPv6 and PMIPv6', in *2009 5th International Conference on Wireless Communications, Networking and Mobile Computing*, IEEE, 2009, pp. 1–4.

[78] F. Giust, A. De la Oliva, and C. J. Bernardos, 'Mobility management in next generation mobile networks', in *2013 IEEE 14th International Symposium on' A World of Wireless, Mobile and Multimedia Networks'(WoWMoM)*, IEEE, 2013, pp. 1–3.

[79] F. Giust, L. Cominardi, and C. J. Bernardos, 'Distributed Mobility Management for Future 5G Networks: Overview and Analysis of Existing Approaches', no. January, pp. 142–149, 2015.

[80] J. Costa-Requena, 'SDN integration in LTE mobile backhaul networks', in *The International Conference on Information Networking 2014 (ICOIN2014)*, IEEE, 2014, pp. 264–269.

[81] L. Valtulina, M. Karimzadeh, G. Karagiannis, G. Heijenk, and A. Pras, 'Performance evaluation of a SDN/OpenFlow-based Distributed Mobility Management (DMM) approach in virtualized LTE systems', in *2014 IEEE Globecom Workshops (GC Wkshps)*, IEEE, 2014, pp. 18–23.

[82] K.-K. Yap, M. Kobayashi, R. Sherwood, T.-Y. Huang, M. Chan, N. Handigol, and N. McKeown, 'OpenRoads: Empowering research in mobile networks', *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 1, p. 125, Jan. 2010, ISSN: 01464833. DOI: 10.1145/1672308.1672331.

[83] Y. Grunenberger and F. Rousseau, 'Virtual access points for transparent mobility in wireless LANs', in *2010 IEEE Wireless Communication and Networking Conference*, IEEE, 2010, pp. 1–6.

[84] M. E. Berezin, F. Rousseau, and A. Duda, 'Multichannel virtual access points for seamless handoffs in IEEE 802.11 wireless networks', in *2011 IEEE 73rd Vehicular Technology Conference (VTC Spring)*, IEEE, 2011, pp. 1–5.

[85] Y.-E. Lin and T.-M. Tsai, 'Creation, management and migration of virtual access points in software defined WLAN', in *2015 International Conference on Cloud Computing and Big Data (CCBD)*, IEEE, 2015, pp. 313–320.

[86] Nokia, *Dynamic end-to-end network slicing for 5G: Addressing 5G requirements for diverse services, use cases, and business models. White Paper*, 2016. [Online]. Available: http://www.hit.bme.hu/~jakab/edu/litr/5G/NOKIA_dynamic_network_slicing_WP.pdf.

[87] O. N. F. Documentation, 'OpenFlow Switch Specification', Jun. 2012. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf.

[88] J. Barraca, D. Gomes, and R. L. Aguiar, 'Amazing – advanced mobile wireless playground', English, in *Testbeds and Research Infrastructures. Development of Networks and Communities*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, T. Magedanz, A. Gavras, N. Thanh, and J. Chase, Eds., vol. 46, Springer Berlin Heidelberg, 2011, pp. 219–230, ISBN: 978-3-642-17850-4.

[89] H. Li, G. Shou, Y. Hu, and Z. Guo, 'WiCloud: Innovative uses of network data on smart campus', in *2016 11th International Conference on Computer Science & Education (ICCSE)*, IEEE, 2016, pp. 461–466.

[90] 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects (3GPP-TS), *Architecture enhancements for non-3GPP accesses (Release 15) (3GPP TS 23.402)*, 2018.

[91] M. I. Sanchez, A. de la Oliva, and C. J. Bernardos, 'Experimental analysis of connectivity management in mobile operating systems', *Computer Networks*, vol. 94, pp. 41–61, 2016.

[92] A. Baheti, *Extensible Authentication Protocol Vulnerabilities and Improvements*, https://scholarworks.sjsu.edu/etd_projects/425, ; Online; accessed 1 Jun 2018, 2015.

[93] J. M. R. Castillo, H. Lundqvist, and C. Qvarfordt, 'Energy consumption impact from wi-fi traffic offload', ser. Wireless Communication Systems (ISWCS 2013), Proceedings of the Tenth International Symposium on, VDE, 2013, pp. 1–5.

[94] V. Nguyen, A. Brunstrom, K. Grinnemo, and J. Taheri, 'SDN/NFV-Based Mobile Packet Core Network Architectures: A Survey', *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1567–1602, thirdquarter 2017, ISSN: 1553-877X. DOI: 10.1109/COMST.2017.2690823.

[95] V. A. Cunha, I. D. Cardoso, J. P. Barraca, and R. L. Aguiar, 'Policy-driven vCPE through dynamic network service function chaining', in *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, Jun. 2016, pp. 156–160. DOI: 10.1109/NETSOFT.2016.7502463.

[96] J. Soares, M. Dias, J. Carapinha, B. Parreira, and S. Sargento, 'Cloud4NFV: A platform for Virtual Network Functions', in *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)*, Oct. 2014, pp. 288–293. DOI: 10.1109/CloudNet.2014.6969010.

[97] A. Kimura, S. Kawano, H. Tsuchiya, S. Homma, and A. Okada, 'Evaluation of Virtual Customer Premises Equipment Prototype System with Open Source Software', in *2018 IEEE 7th International Conference on Cloud Networking (CloudNet)*, Oct. 2018, pp. 1–3. DOI: 10.1109/CloudNet.2018.8549554.

[98] A. Mimidis, E. Ollora, J. Soler, S. Bessem, L. Roullet, S. Van Rossem, S. Pinneterre, M. Paolino, D. Raho, X. Du, J. Chesterfield, M. Flouris, L. Mariani, O. Riganelli, M. Mobilio, A. Ramos, I. Labrador, A. Broadbent, P. Veitch, and M. Zembra, 'The Next Generation Platform as a Service Cloudifying Service Deployments in Telco-Operators Infrastructure', in *2018 25th International Conference on Telecommunications (ICT)*, Jun. 2018, pp. 399–404. DOI: 10.1109/ICT.2018.8464838.

[99] N. ETSI, *GS NFV-MAN 001 V1. 1.1 Network Function Virtualisation (NFV); Management and Orchestration*, 2014.

[100] A. Devlic, A. Hamidian, D. Liang, M. Eriksson, A. Consoli, and J. Lundstedt, 'NESMO: Network slicing management and orchestration framework', in *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*, IEEE, 2017, pp. 1202–1208.

[101] A. G. Dalla-Costa, L. Bondan, J. A. Wickboldt, C. B. Both, and L. Z. Granville, 'Maestro: An NFV orchestrator for wireless environments aware of VNF internal compositions', in *2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*, IEEE, 2017, pp. 484–491.

[102] B. Sousa, L. Cordeiro, P. Simoes, A. Edmonds, S. Ruiz, G. A. Carella, M. Corici, N. Nikaein, A. S. Gomes, E. Schiller, *et al.*, 'Toward a fully cloudified mobile network infrastructure', *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 547–563, 2016.

[103] *OSM Release FIVE Technical Overview*, version: 1.0, ETSI, Jan. 2019. [Online]. Available: https://osm.etsi.org/images/OSM-Whitepaper-TechContent-ReleaseFIVE-FINAL.pdf.