University of Catania

Department of Electric, Electronic and Computer Engineering

Ph.D. Course in Systems, Energy, Computer and Telecommunication
Engineering– XXXII Cycle

A SDN/NFV BASED C-RAN ARCHITECTURE FOR 5G MOBILE NETWORKS

Ph.D. Student:
Ing. Gianluca Camillo Valastro

Coordinator: Prof. Paolo Arena Tutor: Prof. Daniela Panno

To my family

ABSTRACT

In these years, various standardization entities are defining the features regarding the future 5G Radio Access Network (RAN) architecture. A considerable innovation concerning the design of 5G mobile networks will be a concrete step towards enabling effective high throughput and low latency services.

A novel architecture design for RANs to address critical elements in resources management and to achieve the 5G mobile networks performances is needed. In this context, Cloud-Radio Access Network (C-RAN), a new mobile network architecture characterized by a functionalities redistribution, is considered as the most suitable solution.

The principle of C-RAN is related to a functional splitting turning the monolithic 4G evolved NodeB (eNB) into a two or three tier Node B, namely Next Generation NodeB (gNB), comprising two or three types of entities, the remote unit (RU), the distributed unit (DU) and the centralized unit (CU). Software Defined Networking (SDN), Network Function Virtualization (NFV) and Self Organized Network (SON) are considered the enabling technologies to achieve these goals.

SDN focuses on decoupling control and data plane, NFV performs the functionalities abstraction from the underlying hardware, whereas SON permits to maximize automation of various the aspects.

In this dissertation, a hierarchical layered software-defined architecture for future 5G networks is proposed. The architecture relies on SDN, NFV and SON principles. As further contribution, our solution aims to provide new features in order to facilitate SDN/NFV/SON application in mobile networks.

The research project related to my PhD has been conducted in the context of an international academic collaboration between the University of Glasgow and the University of Catania. I have technically coordinated this research activity and the relative participation of our team, in representation of the University of Catania, to the "Mosaic5G" research project, as contributor member.

In order to implement the aforementioned software defined architecture, we design and implement a flexible SDN/NFV-based SON testbed for future 5G mobile networks. The main contribute of our work is to cover the need for a SDN/NFV-based SON testbed, enabling the investigation of the potential of these paradigms for practical implementations. We exploit, FlexRAN, as a Software Defined RAN (SD-RAN) platform enabling the SDN concept and OpenAirInterface, as an open source software-based implementation of the 3GPP full protocol stack.

Research work led to two different Testbed deployments. A first Testbed in Catania (TestbedCT), based on NFV to set-up an entirely emulated environment. A subsequent Testbed in Glasgow (TestbedGLA), as a real environment deployment, using Software

Defined Radios (SDRs) and real user equipments.

Both the testbed deployments enable novel research and provides teaching opportunities in next generation RAN architectures and several other areas of system-level research. Furthermore, they act as benchmark for many use cases with a significant proximity to real network deployment criticalities.

The research activities during the PhD involve another research collaboration with Bristol Is Open, a joint venture between the University of Bristol and Bristol City Council, with the aim of deploying a SDN/NFV based testbed to prove the application of this two paradigms in the context of mobile core networks, exploiting OpenDaylight as SDN controller.

Finally, a research collaboration with the telco research team in Milan of Altran, engineering consulting company, is conducted, focusing on the virtualization of the IP Multimedia Subsystem (IMS) network architecture in the context of the Voice over 5G (Vo5G).

Acknowledgments

First of all, I want to thank my supervisor, Prof. Daniela Panno, for her support during these three years of PhD Course. Her precious advices, both regarding work and life aspects, helped me to grow up as man and as researcher. It is a pleasure and a privilege to work under her guidance.

Then I want to thank all my family. The last three years were not easy, but you always believed in me. You supported me with your love and care.

This last paragraph is dedicated to my wife Mariangela, I love you, no words could explain what you mean to me.

CONTENTS

LIST OF ABBREVIATIONS

Abbreviation	Description
3GPP	Third Generation Partnership Project
5GC	5G core network
5G NG-RAN	5G Next Generation RAN
5G NR	5G new radio interface
5GPPP	5G Public Private Partnership
BS	Base Stations
BBU	Base Band Unit
CBS	Control Base Station
CP	Control Plane
CPRI	Common Public Radio Interface
CDSA	Control Data Separated Architecture
C-RAN	Cloud or Centralized Radio Access Network
C-SON	Centralized-SON
CU	Centralized Unit
DBS	Data Base Station

DP	Data Plane
D-SON	Distributed-SON
DU	Distributed Unit
EE	Energy Efficiency
eMBB	Enhanced Mobile Broadband
eNB	evolved NodeB
EPC	Evolved Packet Core
gNB	Next generation NodeB
HetNets	Heterogeneous Networks
H-SON	Hybrid-SON
IMS	IP Multimedia Subsystem
IoT	Internet of Things
KPI	Key Performance Indicator
LTE	Long Term Evolution
MC	Macro Cell
mMTC	Massive Machine Type Communications
ng-eNB	Next generation eNodeB
NGFI	Next Generation fronthaul interface

NGMN	Next Generation Mobile Network
NG-RAN	Next Generation RAN
NG-SON	Next Generation-SON
NFV	Network Function Virtualization
NSA	Non-Standalone
OAI	OpenAirInterface
O&M	Operation and Maintenance
QoS	Quality of Service
RAN	Radio Access Network
RAT	Radio Access Technology
RAU	Remote Aggregation Unit
RCC	Radio Cloud Center
RRH	Remote Radio Head
RRS	Remote Radio System
RRU	Remote Radio Unit
SA	Standalone
SC	Small Cell
SCF	Small Cell Forum

SDN	Software Defined Networking
SDR	Software Defined Radio
SD-RAN	Software Defined RAN
SON	Self Organized Network
UE	User Equipment
uRLLC	Ultra-reliable and Low Latency Communications
USRP	Universal Software Radio Peripheral
Vo5G	Voice over 5G
VM	Virtual Machine
V-RAN	Virtual-RAN

CHAPTER
ONE

1 INTRODUCTION

1.1 Introduction

5G networks will be innovative networks able to provide revolutionary services, functionalities and capabilities hard to obtain by exploiting the actual network technologies. Therefore, 5G networks should be re-designed in order to accomplish the new requirements.

Recently, the progress regarding various aspects of the IT world has contributed to the creation and the evolution of an open ecosystem. The users will leverage on various types of devices, which will allow the connection to different types of networks in order to access different kind of services.

Actually, this progress and the relative flexibility mainly regards services and applications, whereas the network architectures are not suitable for these new technologies, leading to the need of a next generation mobile networks.

Therefore, the next generation mobile networks will be characterized by the presence of different kind of physical networks. The coexistence of these networks, by overcoming economical and juridical constraints, can be exploited to provide to the end-users an enhanced experience related to the requested service.

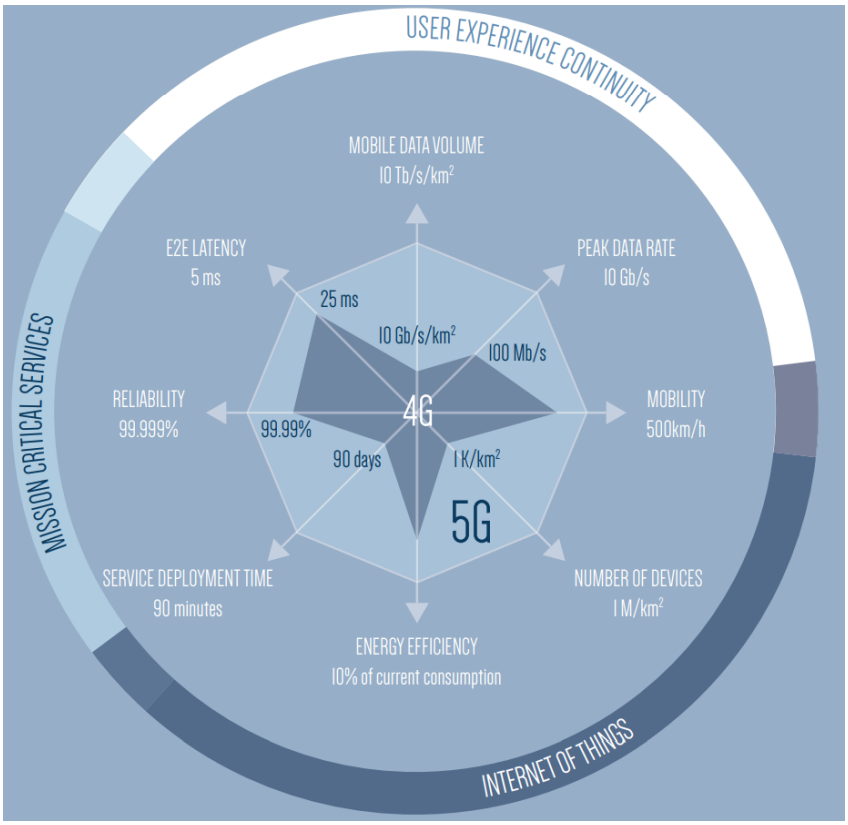

Figure 1.1: *5G requirements*

As can be seen in Fig. 1.1, strict requirements are envisioned for the 5G networks:

- Ensuring a wireless capacity 1000 times greater than the actual
- Ensuring a services diversification
- Achieving an energy saving up to 90%, primarily as regards the RAN side, which is responsible for the 80% of the energy consumption in a mobile network.
- Decreasing the time to create services
- Creating a new network architecture based on the concepts of availability and reliability,
- Ensuring a perceived down time close to zero.

This focus on the services and their requirements is based on the assumption that, in contrast to the previous mobile networks generations, the 5G is not centered on the end-user but on the services, or more specifically, the verticals, i.e., the different vertical markets, such as automotive, energy, city management, government, healthcare, manufacturing, and intelligent transport systems. The definition of verticals remarks the definition introduced by IMT Vision recommendation ITU-R M.2083-0, regarding three principal use cases as:

- **Enhanced Mobile Broadband (eMBB)**, mainly regarding the human-devices involved in high data rate and large payload transmission with a moderate reliability, typical of high quality

streaming, augmented and virtual reality.

- **Ultra-reliable and Low Latency Communications (uRLLC)**, mainly regarding devices involved in tactile internet or remote control. These communications are characterized by small payload transmissions with low latency and high reliability, since concern mission critical communications such as medical or emergency items.

- **Massive Machine Type Communications (mMTC)**, mainly regarding communication among a high number of devices with small payload and non-delay-sensitive transmissions, typical of Internet of Things (IoT).

In order to exploit this heterogeneity of services and relative requirements, changes and enhancements, regarding the management of the mobile networks, are needed. The major challenges in terms of research and innovation concerning the mobile networks in the context of 5G cover different aspects. In particular, the European Union funded 5G Public Private Partnership (5GPPP) supported different projects focusing on a wide range of aspects, from physical layer to the overall architecture, network management and software networks. The motivation relies on the assumption that 5G is not only an innovation related to a new radio but mainly focuses on the attempt to create a new framework allowing the integration of previous and new technologies in order to achieve the 5G requirements.

This process is conducted by a specific working group of the 5GPPP Initiative, the 5G Architecture Working Group. The work of the group has been organized in three phases. Each one of these phases led to the

production of a white paper in the versions 1.0, 2.0 and 3.0, respectively published in July 2016, January 2018 and July 2019 [1-3]. The view given by the 5G PPP contributors and the results coming from the 5G PPP projects deeply contributed to the standardization process of the 5G mobile networks conducted by various standardization entities, e.g., the Third Generation Partnership Project (3GPP), as shown in Fig. 1.2.

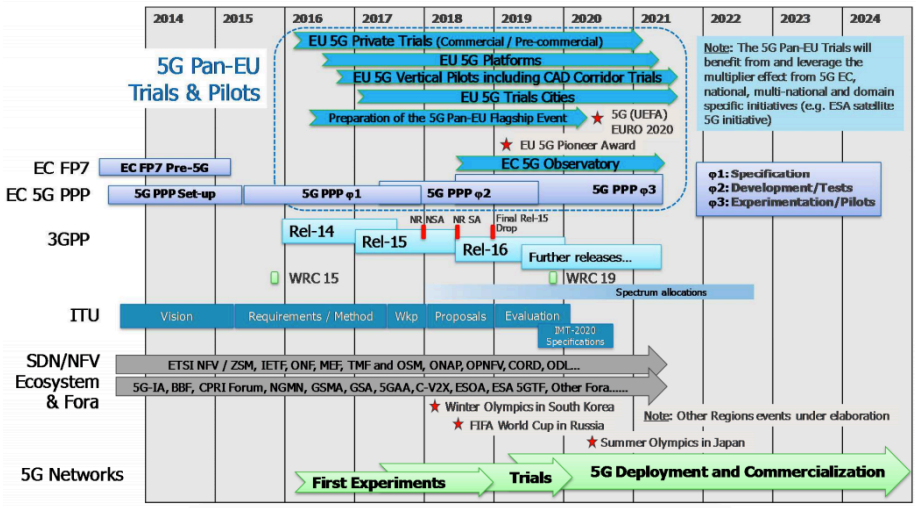

Figure 1.2: *The standardization process regarding the 5G networks*

The standardization process regards both the RAN and the CN. As shown in Fig. 1.3, 5G network is composed of a 5G access network, called 5G Next Generation RAN (NG-RAN) and a 5G core network (5GC) [4]. The NG-RAN consists of 5G new radio interface (NR) access nodes and/or of non-3GPP access network nodes, both connected to the 5GC.

On the basis of the user plane/control plane terminations, the 5G access network node can be defined as:

- **evolved NodeB (eNB)** - LTE access network node from 3GPP Rel-8 up to Rel-14.

- **Next generation eNodeB (ng-eNB)** – LTE access network node from 3GPP Rel-15, providing user plane and control plane protocol termination towards the user equipment (UE), connected via the NG interface to the 5CG.

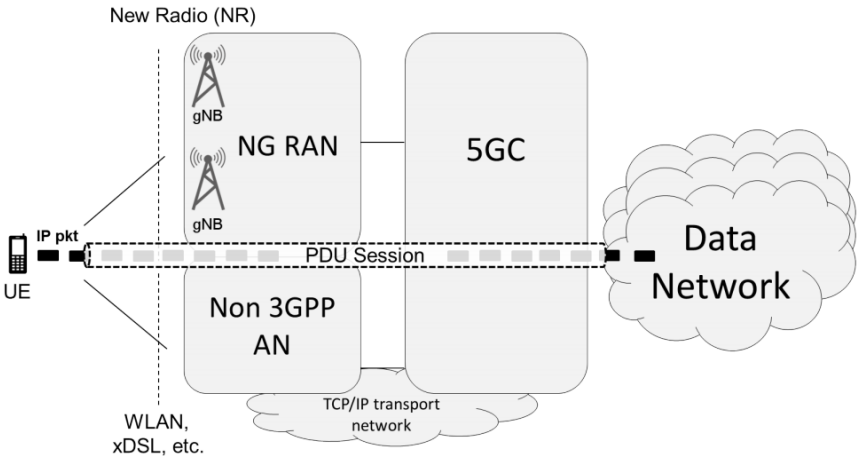

Figure 1.3: *5G network architecture*

- **Next generation NodeB (gNB)** – 5G access network from 3GPP Rel-15, providing NR user plane and control plane protocol terminations towards the UE, via the NG interface to the 5GC.

Furthermore, as shown in Fig. 1.4, different deployment options are

defined on the basis of the RAN-CN interconnections. In particular, the various options can be divided in standalone (SA) and non-standalone (NSA). In the SA options (1,2,5) only one radio access technology is deployed, 5G NR or the evolved Long Term Evolution (LTE) connected to the corresponding generation of core network. In the NSA options (3,4,7), NR radio access nodes are combined with LTE access nodes and the core network may be either Evolved Packet Core (EPC) or 5GC.

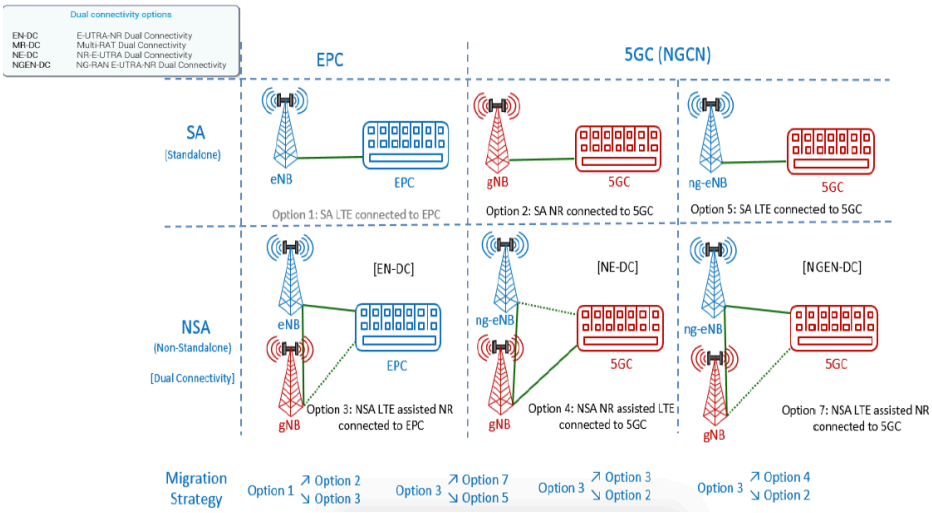

Figure 1.4: *5G Deployments and migration strategies*

Finally, different migration strategies are defined in order to move from a previous type of deployment to another. Considering a starting configuration, it is possible the migration only to a restricted number of available deployment options.

Finally, as regards the network functions, in 5G networks generally they will be exposed as services, so the 5G core is said to have a service-based architecture. More specifically, the procedures related to specific network functions are defined as services. Furthermore, in previous mobile networks generations a standardized real or logical point-to-point interface interconnecting two network entities is defined, and this interface uses a specific bit-oriented protocol. In the 5GC, service-based interfaces are exploited, supported by web-oriented tools such as HTTP/2, REST and JSON. Consequently, the 5G architecture has been designed to enable the softwarization of the network functions.

This led to an evolution regarding the network design, characterized by a new need of flexibility and programmability, different from the "ossified" hardware previous network architecture.

Software defined networking (SDN), network function virtualization (NFV) and cloud computing are recognized as the key enabling technologies to enable these enhancements. In Chapter 2, a detailed overview of the RAN evolution will permit to better understand how these paradigms are contributing to this software revolution.

1.2 Structure of this Dissertation

The dissertation is organized as follows. In Chapter 2 is presented a brief overview on 5G technology and a State-of-the-art overview.
In Chapter 3 we introduce the system model and the network model being considered and then present a detailed description of the proposed software defined architecture. In Chapter 4 the first version of the testbed, based on the ns-3 simulator. In Chapter 5 we present the software tools utilized in the second version of the testbed, OpenAirInterface and FlexRAN. In Chapter 6 we expose the testbed based on these two software tools, as well exposing the results and the relative considerations.

Finally, in Chapter 7 the conclusion remarks, where discuss also the works in progress. The list of related publications is presented as well.

CHAPTER TWO

2 BACKGROUND AND RELATED WORK

2.1 Background

In next decade, as exposed in [2], future mobile networks will fulfill multiple 5G features in terms of guaranteed user data rate, high throughput, low delay, number of UEs and mobility support at high speed. Cell densification is considered as a key solution in order to realize these enhancements. In this context, the basic idea is to deploy the access nodes as close as possible to the end users, in order to satisfy the required Quality of Service (QoS) and maximize system throughput. Due to the increasing user density, small cells become smaller and denser, leading to the ultra-dense networks concept. In [5] a quantitative measure of the cell density for which a network can be considered ultra-dense (10^3 cells / km^2) is provided.

In this line, future networks will be realized as a set of multiple base stations with different features, in terms of connected users, transmitted power and coverage areas, i.e., small cell and macro cell deployment.

Among the main aspects available in literature concerning densification, we focus on dual connectivity and centralization. Based on

Dual Connectivity scheme standardized by 3GPP for Release 12 small cell enhancements [6], DOCOMO proposes in [7] the Phantom Cell Concept, as a solution for the next ultra-dense networks.

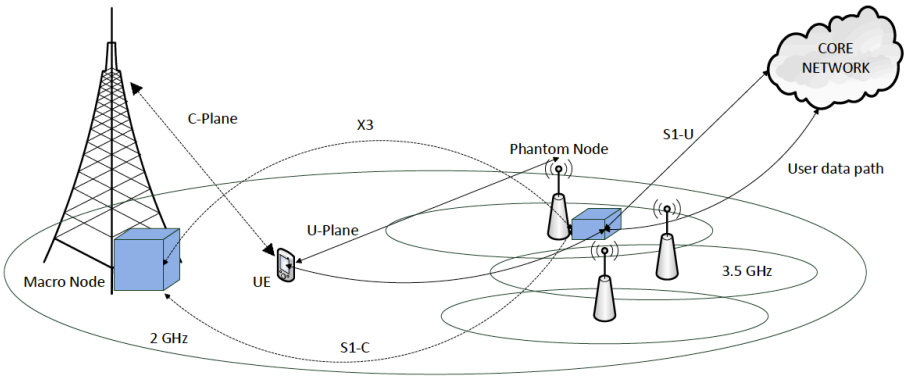

Figure 2.1: *Docomo splitting solution*

As shown in Fig. 2.1, this solution is mainly focused on a splitting between control plane (C-Plane) and user data plane (U-Plane). Moreover, different frequency band allocation in Macro cell and small cell coverage areas is considered. In particular, for a UE in a phantom cell coverage area, C-Plane is provided by the Macro eNodeB at low frequency band (2 GHz), in order to improve connectivity and mobility management, whereas U-Plane is provided by Phantom base station at higher frequency band (3.5 GHz), in order to boost user data rate.

As regards the centralization aspect, we observe that in the traditional LTE RAN, see Fig.2.2a, the eNodeB is composed of a radio

front-end entity, referred as Remote Radio Head (RRH) and a base band computational unit, called Base Band Unit (BBU). In this context, the BBUs are located in a distributed mode (i.e., each BBU is co-located with the related RRH). As shown in Fig.2.2b, in the last few years, mobile RAN is slightly evolving to a more centralized architecture, known as Centralized-Radio Access Network (C-RAN) [8].

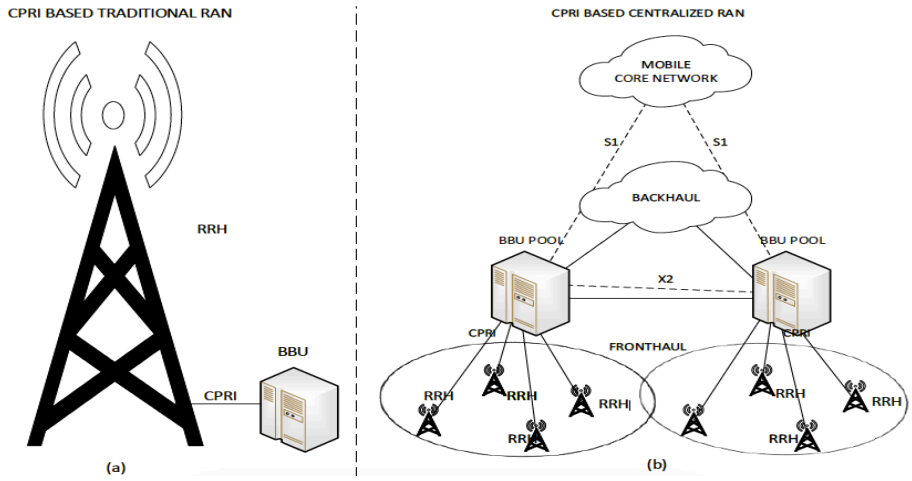

Figure 2.2: *(a) CPRI based Distributed RAN - (b) CPRI based Centralized RAN*

In C-RAN architecture, base band computational resources are pooled in remote locations, still hardware-based, known as BBU Pools, whereas RRHs are placed in edge locations. Centralization feature provides notable advantages in terms of CAPEX/OPEX, since enables operators to centralize hardware, significantly reducing energy consumption and maintenance costs. Exploiting the positive aspects of dual connectivity

and centralization in an ultra-dense scenario enables to perform a centralized radio resource management capable of addressing significant issues as severe interference between co-tier small cells, support and co-deployment of LTE-Advanced and 5G RAN entities (inter-technology compatibility). Furthermore, a RAN evolution is also required in order to support Multi-Radio Access Technology (RAT) feature and seamlessly integrate new radio access technologies, e.g. millimeter waves, with existing ones.

These significant requirements cannot be achieved without new key features as softwarization, programmability, virtualization, fronthaul and radio interface redesigning and resource coordination. The control and data plane splitting proposed by DOCOMO could be realized through SDN. In its native wired nature, SDN focuses on decoupling control and data plane of network forwarding elements as switches. The communication between the forwarding entity and the control entity, i.e. the controller, is performed by the OpenFlow protocol. Accordingly, the data plane is performed by OpenFlow enabled switches, whereas the control plane is performed by logically centralized but physically distributed SDN controllers.

In order to take full advantage of SDN capabilities in mobile and wireless environments, a first critical issue is related to its possible integration. NFV, focusing on virtualization of hardware based functionalities, can be take in account in virtualization process of the BBU entities forming the BBU Pool. NFV permits to realize in a software fashion the hardware based baseband functionalities. These enhancements make RAN architecture more scalable and reliable, since a programmable management of virtualized functionalities is achievable. At this aim, as depicted in [9], SDN and NFV are considered as the enabling technologies to realize these enhancements.

Following to the above assumptions, in a further evolution of the Centralized-RAN, BBUs can be not only centralized but also virtualized and coordinated by a centralized entity, in order to optimize resource allocation, leading to Cloud-RAN or Virtual-RAN. Accordingly, Cloud-RAN, SDN and NFV are recognized as the key enabling solutions for future mobile networks.

2.2 Overview on related works

As mentioned above and depicted in [10], in terms of centralization and architecture redesigning, in the last few years mobile RAN has already slightly evolved to a more centralized and coordinated fashion, C-RAN. As shown in Fig. 2.3, the transport network connecting RRHs to the related BBU Pool is called Fronthaul. Fronthaul links could be implemented using different technologies like Ethernet, optical fiber or millimeter wave communication.

On the one hand, due to high bandwidth and low delay requirements optical fiber communication would be the proper solution for C-RAN fronthaul, but its deployment is usually high costly and not flexible. On the other hand, wireless solutions, such as millimeter wave communication, are cheaper and easy to deploy, but characterized by lower bandwidth and higher delay.

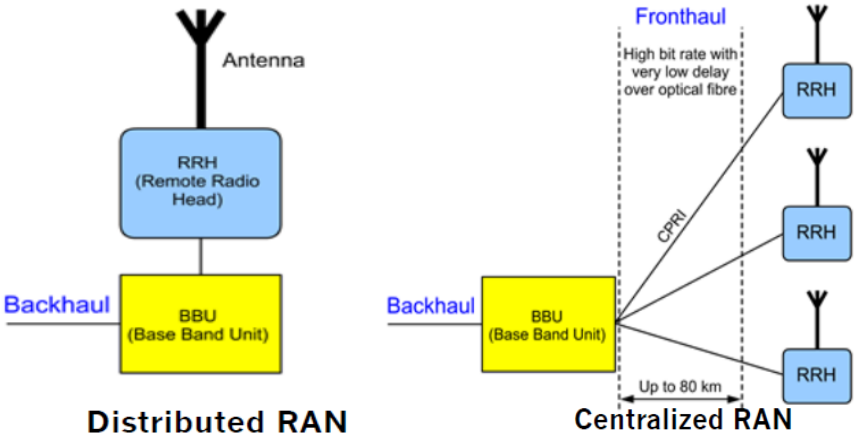

Figure 2.3: *RAN evolution*

The selection of the appropriate frounthaul link option is strictly related to a further significant aspect, i.e., centralization option. In fact, centralization could be implemented in different patterns, i.e., full centralization, partial centralization and hybrid centralization. These patterns are related to various functional splits proposed by different organizations, e.g., SCF [11], NGMN [12] and 3GPP [13], as shown in Fig. 2.4.

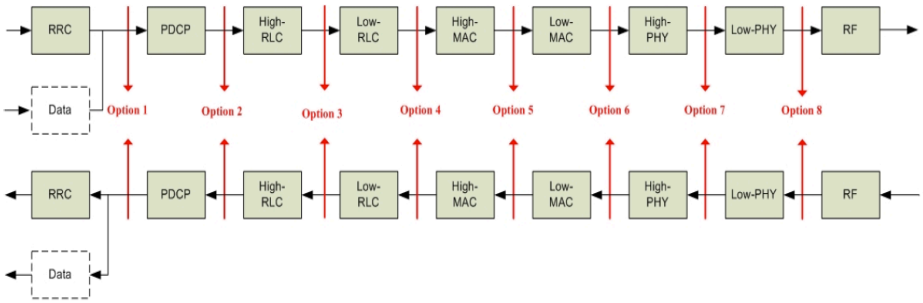

Figure 2.4: *3GGP Functional splitting options - Source: 3GPP TR 38.801 V.14.0.0.*

In full centralization, coinciding with option 8, only RF functionalities are deployed in RRH site, PHY Layer and upper layers in BBU Pool site, in order to maximize management capacity, making easier operation and maintenance. On the other hand, as exposed in [14], due to high bandwidth and overhead related to IQ data transmission, this solution suffers from performance limitations.

In partial centralization, coinciding with option 6, PHY layer functions are implemented in RRHs, upper layers in the central unit (BBU). As opposed to fully centralized option, bandwidth and overhead requirements are reduced, since the demodulated signal is carried rather than the modulated one, but pooling gain is limited. In hybrid centralization, coinciding with option 7, also known as LLS (Lower Layer Split), part of the physical layer functions are implemented in RRHs, remaining functions in BBU.

In conclusion, partially and hybrid centralized solutions are characterized by lower requirements than fully centralized, in terms of bandwidth and overhead, allowing to deploy an Ethernet fronthaul, rather than fiber. Furthermore, fully centralized option is better in terms of pooling gain, since aggregating computational resources related to the entire protocol stack permits to implement more advanced processing algorithms. Thus, an appropriate functional splitting should take into account a trade-off between these solutions.

As regards the interface redesigning, we observe that in the traditional LTE RAN, as shown in Fig.2.2a, the interface between RRHs and BBUs is a Common Public Radio Interface (CPRI). CPRI was developed for local link between BBU and RRH in antenna sites and to support lower bandwidth transmissions. Moreover, as shown in Fig.2.2b, first implementations of C-RAN fronthaul are still CPRI-based. As depicted in [15], different solutions are present in literature. In particular,

two examples of Centralized RAN are considered.

The first example is China Mobile Research Institute C-RAN proposal [8], deploying a centralized control, in particular virtualized BBU pool are connected to RRHs through fiber connections. The second example is proposed by DOCOMO and is considered as an Advanced C-RAN, since implements Phantom Cell Concept, based on carrier aggregation and small cell technologies [16]. These two solutions are based on a full centralization, thus suffer from the IQ data transport related issues, which causes a bottleneck in Fronthaul Network.

In order to resolve it, in [17] is proposed by Akyildiz et al. a Software Defined architecture based on hybrid centralization, but still implementing the CPRI as Fronthaul Interface. Nevertheless, in next few years, increasing the distance between RRHs and BBUs, the fronthaul link will be extended in the range of Kms, so traditional CPRI will not be the proper solution. In fact, one of the most significant issues will be the high data rate requirements related to the increasing number of antennas, as effect of massive MIMO deployments [2]. In addition, strict 5G requirements in terms of higher bandwidth and lower delay, leads to a further evolution of fronthaul and interface designing.

In this line, in recent times IEEE NGFI working group defines a new transport fronthaul interface for future mobile networks, called Next Generation fronthaul interface (NGFI). This solution aims to resolve the abovementioned issues related to CPRI, thanks to features as adaptive bandwidth, statistical multiplexing, support for high-gain coordinated algorithms, data rate decoupled from the number of antennas at RRUs, support for different air interface technology.

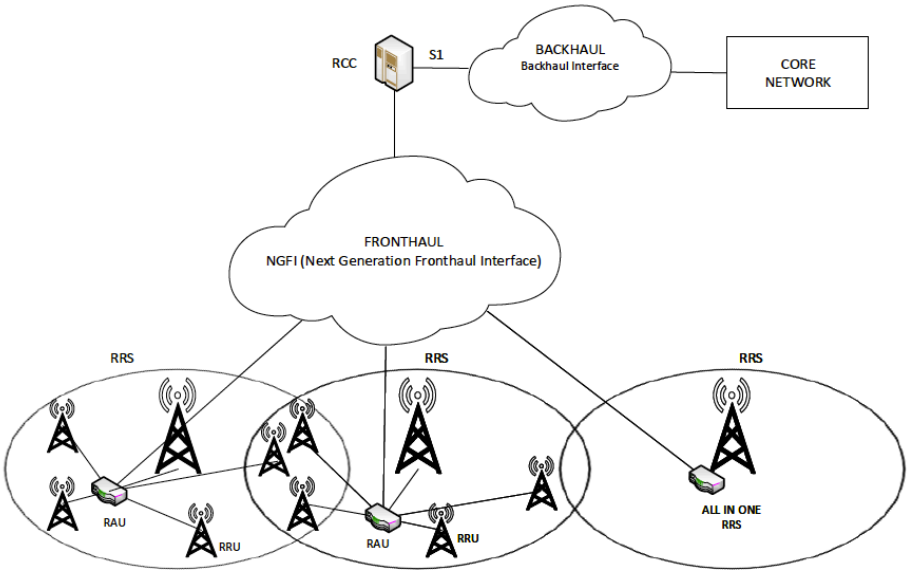

Figure 2.5: *NGFI based Centralized RAN*

As shown in Fig. 2.5, according to [18], BBU and RRH functionalities are redefined, since part of BBU functionalities are moved to the edge location. Consequently, Remote Radio System (RRS) concept is introduced. In particular, the RSS concerns network elements as antennas, Remote Radio Units (RRUs) and Remote Aggregation Unit (RAU). RRUs will be an enhanced radio frontend performing part of base band processing, the RAU will be a logical entity performing local aggregation functionality and part of BBU processing.

Accordingly, BBU will be redesigned as Radio Cloud Center (RCC), since will exploit softwarization and virtualization features. The RCC will perform the remaining BBU processing. The RRU coverage will be equivalent to a metrocell. The fronthaul network will be redesigned, since fronthaul links will evolve from point-to-point to multipoint-to-

multipoint, exploiting the aggregation feature of the RAU. According to this redesign, since bandwidth requirements will be relaxed, the fronthaul transport technology would be packet based, i.e. Ethernet, in order to facilitate NGFI standardization and real deployment.

CHAPTER
THREE

3 PROPOSED ARCHITECTURE

3.1 SYSTEM MODEL

In this thesis, a hierarchical layered architecture for future 5G networks is proposed. The solution exploits SDN/NFV features, in order to deploy a programmable and virtualized architecture. We aim to propose new features that could facilitate SDN/NFV integration in wireless/mobile environment, concretizing Docomo proposed solution in an ultra-dense scenario.

The hierarchical architecture scheme includes different logical layers, as shown in Fig. 3.1. The considered scenario concerns a group of macro cells. Exploiting a NGFI based architecture, each macro and the RRUs deployed in the related coverage area form the infrastructure layer.

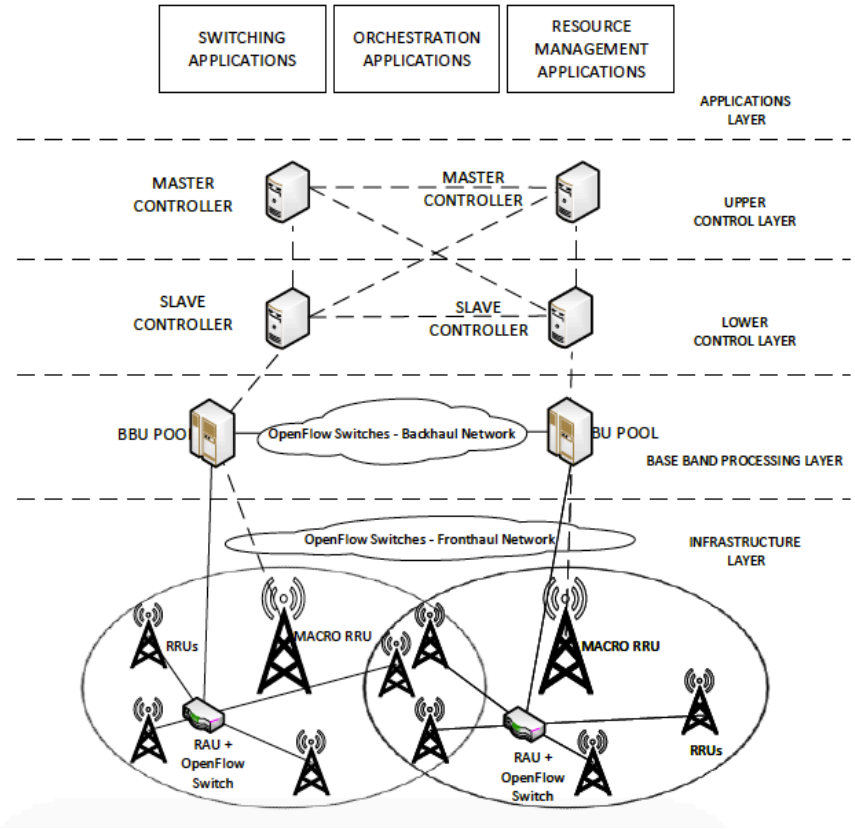

Figure 3.1: *Proposed SDN/NFV based C-RAN architecture*

The baseband processing layer is composed of BBU Pools, concerning the entire computational capacity, connected through fronthaul links and RAU to RRUs. The control layer is performed by a set of SDN controllers, each controller takes care of several management aspects, performing various functionalities. The application layer as the various applications deployed on top of the SDN controllers.

A detailed description of the proposed architecture is provided, the proposed SDN/NFV based C-RAN architecture comprises the following layers.

3.1.1 Infrastructure Layer

As opposed to CPRI-based SoftAir proposal, inspired by NGFI-based architecture, in particular by the RAU, we consider NGFI as fronthaul interface. This permits to overcome the CPRI related issues still present in SoftAir proposal and to deploy an Ethernet based fronthaul network. Inspired by [17], in order to softwarize mobile networks, as novelty we propose to deploy an OpenFlow agent upon the RAU entity, which in conjunction with OpenFlow-enabled forwarding devices, performs a programmable forwarding on the basis of SDN Controller rules. Furthermore, in the fronthaul and backhaul network, OpenFlow enabled forwarding devices are deployed. In this line, control and data plane splitting is achieved and the data plane becomes programmable.

3.1.2 Baseband processing layer

The baseband processing layer concerns a set of BBU Pools, representing the entire computational capacity. In particular, as regards functional splitting options, we design virtualized base band functionalities also in RAU, in order to dynamically implement a particular functional splitting option. Furthermore, as opposed to a CPRI based architecture, the flexibility of the designed virtualized base band processing in RAU, permits to relax the fronthaul link requirements, enabling the deploy of an Ethernet fronthaul network. This enhancement is appropriate, e.g., when a fronthaul link suffers from a fault or the capacity of a BBU Pool is not sufficient in relation to the experienced data traffic.

3.1.3 Low and upper control Layer

Furthermore, as opposed to a CPRI based architecture, the flexibility of the designed virtualized base band processing in RAU, permits to relax the fronthaul link requirements, enabling the deploy of an Ethernet fronthaul network. This enhancement is appropriate, e.g., when a fronthaul link suffers from a fault or the capacity of a BBU Pool is not sufficient in relation to the experienced data traffic.

The control layer is composed of a set of SDN controllers, each controller takes care of several management aspects, performing various functionalities. The architectural control layer is logically centralized but physically distributed, as a set of slave controllers and master controllers.

Master controllers, forming the upper control layer and located in remote sites, manage a group of macro cells, keeps in account long time scale and less fine grained parameters, acting as reference entities for slave controllers.

Slave controllers, forming the lower control layer, located in edge sites, as opposed to master controllers, keep in account short time scale and more fine grained parameters, acting as management entity for a group of small cells.

In order to realize these enhancements, is necessary to customize northbound and southbound interfaces. In particular, the southbound interface is performed by the OpenFlow interface, enabling programmable management of underlying network elements. Northbound interfaces allow applications to interact with controller. On the basis of the reports sent by RAN entities, the algorithms running in applications computes the related output.

3.2 Logical controller areas and new proposed functionalities

As shown in Fig. 3.2, the proposed slave controller, focusing on different features, can be seen as composed of different logical areas, corresponding to the related northbound applications. These proposed functionalities could allow to fulfill strict 5G requirements, since elaborating and combining reports from network entities, will be able to perform optimum rules in terms of switching, resource allocation and virtual function instantiation/migration.

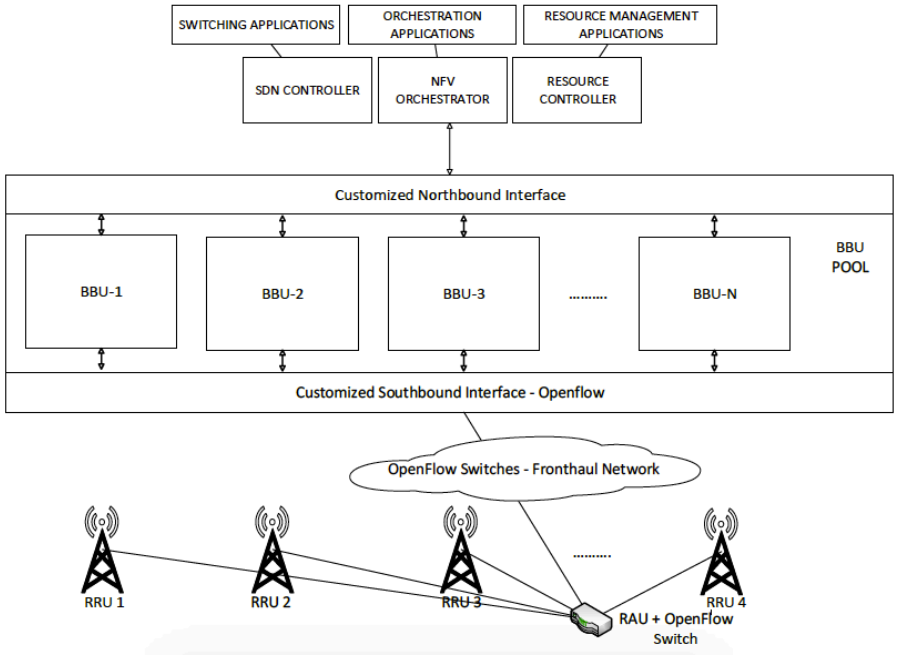

Figure 3.2: *Proposed Slave Controller*

3.2.1 SDN Controller logical area

We design this logical area as strictly related to the particular designed fronthaul network. In fact, in our design, fronthaul network will evolve from a point-to-point CPRI-based network to multipoint-to-multipoint NGFI/Ethernet based network. Since fronthaul topology will be more complex, manage forwarding in a fast, programmable and dynamic fashion exploiting OpenFlow will be necessary, realizing control and data plane splitting related to forwarding functionalities. This SDN enhancement in the proposed mobile RAN architecture is deployed in OpenFlow native scope, i.e. wired network forwarding, but could be very useful to achieve strict 5G requirements. The proposed functionalities related to this area are:

- **Forwarding functionalities in fronthaul network.**

As exposed above since fronthaul network will be more complex, this controller area will allow to manage the mobile
network in a centralized way, coordinating forwarding behavior among OpenFlow enabled network entities, like RAU and switches. This is achievable only through a centralized entity which perform an overall and continuously updated view of the network state.

An Ethernet-like fronthaul will permit to exploit actual OpenFlow capabilities. Moreover, will be possible, through an extended OpenFlow protocol, adding new matching fields, realize splitting based on different rules (simple control and data plane splitting, UE-related and cell-related, different types of splitting in downlink and uplink). In this case, in a dynamic fashion, would be created a Virtual cluster, as a set of RRU

dynamically associated with a BBU Pool and consequently to a slave controller.

- **Forwarding functionalities in Multi-RAT RAN.**

Since 5G environment will be a Multi-RAT environment, this new functionality will concern Multi-RAT support. Interconnecting different wired and wireless involved subsystems, should be possible only deploying an intelligent and fast forwarding, in order to comply with 5G delay requirements. In fact, will be necessary a convergence entity which will be able to select optimum options in terms of switching, security issues and reliability. These enhancements could be realized in our slave controller, in association to the aggregation feature performed by the RAU. As opposed to the abovementioned functionality, in which slave controllers would manage a group of small cells based on the same technology, in this case our slave controller would manage a group of access point based on different technologies. This would be achievable exploiting a Virtual Multi-RAT cluster, dynamically related to a particular BBU POOL and consequently to a particular slave controller.

3.2.2 NFV Orchestrator logical area

We design this logical area in order to realize a NFV Orchestrator, which would be able to allocate dynamically computational resources related to BBU Pool.

The novelty introduced with this logical area comes with these enhancements:

- **Dynamic instantiation of BBU/RAU/RRU related functionalities**

Since part of the protocol stack processing will be implemented in a virtualized fashion, would be possible to dynamically instantiate virtualized base band processing functionalities based on fronthaul link requirements, activating or dis-activating the related virtual entities, in order to dynamically deploy different functional splitting options. These variations could be related to the capacity required in a particular time interval or period of the day.

- **Dynamic instantiation/migration of slave controllers**

In order to follow capacity demands or the need for a convergence controller in multi-rat environment, would be possible to instantiate or migrate slave controller, implemented in a virtualized fashion. In this case could be useful realize a “lighter virtualization”, in a container fashion.

3.2.3 Resource Management Controller logical area

This logical area is designed to instantiate rules based on related resource management algorithms, in order to allocate resources on the

basis of instantaneous requirements in terms of capacity, rate and link state. Moreover, would be possible to deploy a mobility management in a single or multi RAT environment.

- **Resource management in single or Multi RAT RAN**

This functionality concerns algorithms implementation based on a programmable resource management. Slave controller, on the basis of parameters regarding the actual state of the network, elaborates the optimum resource allocation. This logical area would permit to realize an efficient allocation through cooperative concepts, exploiting the overall point of view of the slave controller, in the above mentioned Virtual Cluster.

- **Resource management for single or Multi RAT handover.**

Handover algorithms implementation and resource reallocation based on the related requested service. Thanks to the concept of Virtual Multi-RAT cluster, would be possible to associate the best Multi-RAT neighbor, in order to manage a transparent handover between related technologies.

3.3 Docomo use case

On the basis of the above assumptions, we design an architectural solution for DOCOMO proposed control and data plane splitting. This proposed architecture realizes the abovementioned splitting and exploit SDN/NFV features described in the previous section. Furthermore, as

opposed to Docomo Advanced C-RAN, exploiting the flexibility of the proposed virtual base band processing in RAU. We are able to overcome IQ data transmission issues and to dynamically implement a specific functional splitting option. As regards the control and data plane splitting proposed by Docomo, two different cases are considered in the proposed architecture:

- **UEs in the coverage area of a macrocell**

In this case control and data plane traffic would be routed by OpenFlow enabled RAN entities, on the basis of no splitting option rules specified by the SDN Controller, since Macro RRU provide both control and data plane.

- **UEs in the coverage area of a phantomcell**

In this case is necessary a splitting between control and data plane. In particular, data plane traffic is routed to Phantom RRU, control plane traffic is routed to Macro RRU, through OpenFlow enabled RAN entities, according to the rules specified by the SDN Controller.

In addition, as designed by DOCOMO in a further evolution of the Phantom Cell Concept, the interface between Macro RRU and BBU Pool is designed as the X3 Interface, the interface between phantom RRU and BBU Pool is designed as the X4 interface. In this specific considered scenario, we are focusing on slave controller case. On the basis of different considerations regarding the strict throughput and delay requirements, the slave controller could be located in the same site of the macro RRU.

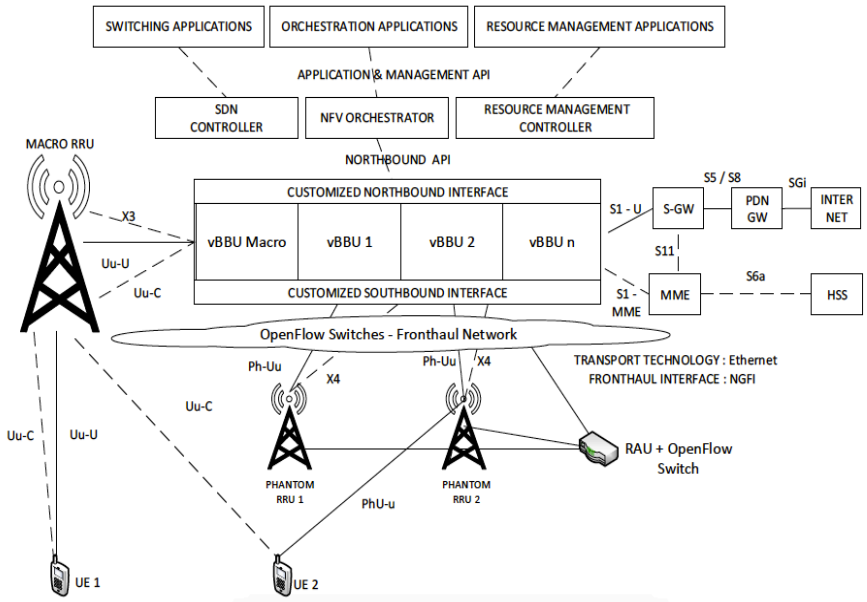

Figure 3.3: *Docomo Use Case*

In this case, the X3 interface would be an internal interface used to manage the communication between Macro RRU and the related BBU Pool. In case of slave controller is located in a remote site, different from Macro RRU site, the X3 interface would also accomplish the related requirements in terms of delay and synchronization. Following the above considerations would be more suitable to locate the slave controller in the same location of Macro RRU, in order to exploit as well as possible the enhancements related to the proposed architecture. The application running on top of the slave controller would be able to communicate through customized northbound interface API. These applications would

be strictly related to the above mentioned logical areas, but would work on a cooperative fashion in order to enhance the overall architecture performances. Through a customized southbound interface, the OpenFlow interface, would be possible to exploit SDN capabilities, thanks to an OpenFlow agent running on the specified RAN entities.

CHAPTER
FOUR

4 NS-3 BASED TESTBED

4.1 PRELIMINARY STEPS

In order to implement the proposed architecture, among the designed logical areas, we design and implement a first deployment of the testbed focusing on the functionalities related to the SDN Controller logical area of the SDN/NFV based C-RAN architecture proposed in the Chapter 3. We decide to deploy and to prove firstly this proposed feature, as a control/data plane splitting aspect, because we consider it as more straightforward than the other proposed functionalities, being an enhancement in OpenFlow native scope, i.e. wired network forwarding in the RAN and in the interconnections with the core network.

Following these assumptions, the topology considered in this first testbed implementation reflects the fronthaul network proposed in our C-RAN architecture, as a multipoint-to-multipoint NGFI/Ethernet based network. Since fronthaul topology will be more complex, manage forwarding in a fast, programmable and dynamic fashion exploiting OpenFlow will be necessary, realizing control and data plane splitting related to forwarding functionalities.

At this aim, we realize, in a simulated environment, a first software defined deployment of the proposed LTE RAN architecture trough the ns-3 simulator. In particular, we configured two different ns-3 modules, known in literature as LENA ns-3 LTE Module [19] and the OFSwitch13 module [20].

The environment has been implemented on top of a SuperMicro 828-14 server. We decided to install and configure Ubuntu 16.04.2 LTS Xenial Xerus as operating system, in order to respect the requirements and the compatibility of the used ns-3 software modules with the specific operating system. We decided to assign a public class B IP address to permit the remote interconnection with the server and with the aim of conducting remote testing sessions. Furthermore, a Virtual Private Network has been implemented in order to secure the aforementioned remote interconnection through the open-vpn and the easy-rsa packages.

The ns-3 environment has been installed and configured firstly in a virtualized entity and subsequently in a real environment, i.e., the Linux Ubuntu 16.04 operating system installed on top of the server. At this aim the VirtualBox software tool has been installed to deploy virtual machines. On top of these virtual machines the ns-3 environment has been deployed. This choice permits us to conduct the installation and configuration phases in isolated environments, in order not to cause potential issues or instability on top of the operating system of the server. Only when a good confidence with the software tools has been achieved and the correct functioning of the environments has been proved the aforementioned steps have been reproduced on top of the real operating system.

In the next sections a brief description of these software tools is given.

4.2 ns-3 simulator

The ns-3 simulator is a discrete-event network simulator, targeted primarily for research and educational use. ns-3 is free software, licensed under the GNU GPLv2 license, and is publicly available for research, development, and use.

The software is written in C++ and Python languages. In this simulation environment are defined different types of classes whose permit to conduct tests regarding different types of networks (fixed, wireless and mobile).

For our purpose, i.e., the simulation of the LTE architecture, we leverage on the LENA ns-3 LTE module. The LENA ns-3 LTE module permits to simulate a LTE RAN / LTE-EPC standard architecture.

In particular, the installation and configuration phases regarded the release 3.26 of the ns-3 simulator. The two aforementioned phases have been conducted on top of a Linux Ubuntu 16.04 virtual machine, with the aim of reproducing later the same procedures in the real operating system, as previously underlined. The Fig. 4.1 depicts the LTE-EPC simulation model used in ns-3 simulator. The model is composed of two different components:

- **the LTE Model**, including the LTE Radio Protocol stack (RRC, PDCP, RLC, MAC, PHY), exploiting the UE and eNB nodes.

- **the EPC Model**, including the network interfaces, protocols and entities regarding the core network, exploiting the SGW, PGW, MME and the eNB nodes.

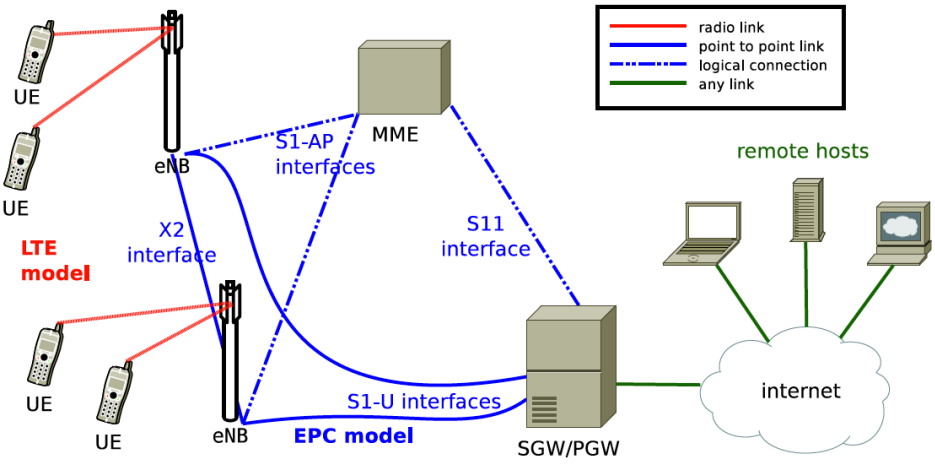

Figure 4.1: *LTE and EPC models*

The setup and configuration processes are conducted through the helper objects. The main helper objects are related to the two aforementioned models. In this line, we can distinguish:

- **the LteHelper object**, which permits to configure the LTE radio access network entities, and to manage the setup and release of EPS bearers. In the LteHelper class both the API definition and its implementation are provided.

- **the EpcHelper object**, which permits to configure the Evolved Packet Core entities. Unlike the LteHelper class, the EpcHelper class is an abstract base class, providing only the API definition. Thus, the relative implementation is realized through the child classes in order to allow for different EPC network models.

The simulation environment permits to conduct LTE-only simulations, if only the LteHelper is used, or complete LTE-EPC simulations if both LteHelper and EpcHelper are used. In this case, a master-slave relation between the two helper is created. Furthermore, the LteHelper acts as the Master that interacting directly with the user program, whereas the EpcHelper acts as the slave, configuring the EPC models through explicit methods called by LteHelper.

4.3 OpenFlow 1.3 Module

The OpenFlow 1.3 module, called also OFSwitch13 module, has been designed to integrate the updated SDN capabilities in ns-3 simulator. In fact, in the ns-3 simulator a module supporting the OpenFlow functionalities was already implemented, but this module was based on a very outdated OpenFlow protocol, the 0.8.9 version (2008). Thus, the integration of the OFSwitch13 permitted to use the features of the version 1.3 of the OpenFlow protocol, both in terms of switch nodes and controller application interface.

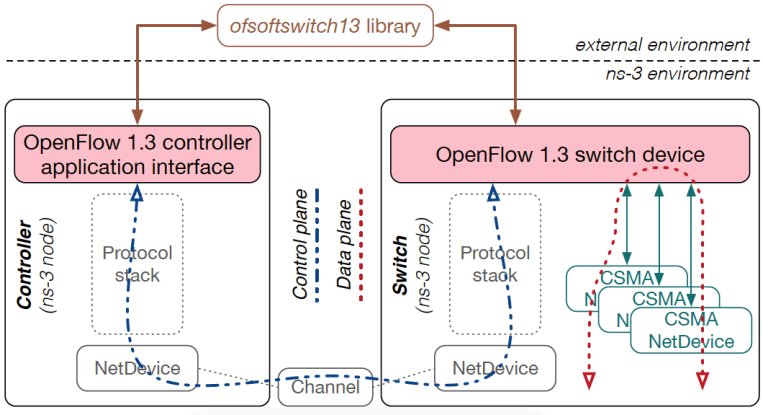

Figure 4.2: *OFSwitch 1.3 Module*

As shown in Fig, 4.2, the OFSwitch13 permits to interconnect the ns-3 nodes, orchestrating the network through the controller application interface in order to implement the desired control logic. The interconnection between the controller and the switch entities is conducted through the standard ns-3 protocol stack, devices and channels. Furthermore, the external OpenFlow 1.3 Software Switch for ns-3 is compiled as a library, the ofsoftswitch13 library, in order to implement the switch data path and to make possible the conversion of the OpenFlow messages to and from wire format.

Finally, the dpctl utility tool for configuring the switch from the command line is deployed. The source code that permits the implementation of the OFSwitch13 module is located in the directory src/ofswitch13. The version of the OFSwitch 1.3 used in the testbed is the 3.0.0, compatible with the 3.26 version of ns-3. In particular, we made these choices because the version 3.0.0 has been developed and tested together with ns-3 versions 3.26.

4.4 Script-based simulations: Fixed networks and OFSwitch 1.3

The first objective has been the practical demonstration, for our purpose, of an efficient OpenFlow integration in the LTE simulation environment in order to prove the SDN capabilities underlined in the proposed software defined architecture.

At this aim, a C++ script has been used to simulate a scenario in which two host are interconnected to an OpenFlow switch through a CSMA link. The proper functioning of the test is conducted by the ping process between the two hosts, passing through the OpenFlow switch.

It follows a brief description of the used C++ scripts and an overview of the first results:

```
#include <ns3/core-module.h>
#include <ns3/network-module.h>
#include <ns3/csma-module.h>
#include <ns3/internet-module.h>
#include <ns3/ofswitch13-module.h>
#include <ns3/internet-apps-module.h>

using namespace ns3;

int
main (int argc, char *argv[])
{
// Enable checksum computations (required by OFSwitch13 module)
GlobalValue::Bind("ChecksumEnabled",BooleanValue (true));
```

In ns-3 it is necessary to define the objects acting as container for the nodes of the simulated networks. It is possible through the NodeContainer class, which permits to instantiate multiple nodes. The nodes have been created through the smart pointer Ptr<Node>. In our case, we created two different hosts nodes and an OpenFlow switch node:

```
// Create two host nodes
  NodeContainer hosts;
  hosts.Create (2);

  // Create the switch node
  Ptr<Node> switchNode = CreateObject<Node> ();
```

Subsequently, through the use of the helper class CsmaHelper, it has been possible to model and to configure the networks. Before the use of this helper, it has been necessary to instantiate the objects of the NetDeviceContainer class, whose represents an abstraction of the network interfaces connecting the nodes. In particular, we created the interfaces of the hosts and the switch ports through two different NetDeviceContainer objects. In the for loop, through the Install() method and the related to the CSMA helper it has been possible to create the links between the hosts and the switch.

Finally, through the smart pointer Ptr<Node>, the node controllerNode has been created.

```
// Use the CsmaHelper to connect the host nodes to the switch.
CsmaHelper csmaHelper;
NetDeviceContainer hostDevices;
NetDeviceContainer switchPorts;

for (size_t i = 0; i < hosts.GetN (); i++)
{
  NodeContainer pair (hosts.Get (i), switchNode);
  NetDeviceContainer link = csmaHelper.Install (pair);
  hostDevices.Add (link.Get (0));
  switchPorts.Add (link.Get (1));
}


// Create the controller node
Ptr<Node> controllerNode = CreateObject<Node> ();
```

The OFSwitch13InternalHelper helper class permits to represent and to configure an OpenFlow network domain, installing the controllerNode previously created through the InstallController() method. Furthermore, this permits to enable the switch node to support the OpenFlow Datapath on the relative ports.

```
// Configure the OpenFlow network domain
  Ptr<OFSwitch13InternalHelper> of13Helper =
  CreateObject<OFSwitch13InternalHelper> ();
  of13Helper->InstallController (controllerNode);
  of13Helper->InstallSwitch (switchNode, switchPorts);
  of13Helper->CreateOpenFlowChannels ();
```

The InternetStackHelper helper permits to create a default TCP/IP protocol stack and to install it on the hosts nodes of the network.

```
// Install the TCP/IP stack into hosts nodes
InternetStackHelper internet;
internet.Install (hosts);
```

The definition of the pool of IP addresses and the relative assignment to the associate nodes has been possible through Ipv4AddressHelper class.

```
// Set IPv4 host addresses
Ipv4AddressHelper ipv4helpr;
Ipv4InterfaceContainer hostIpIfaces;
ipv4helpr.SetBase ("10.1.1.0", "255.255.255.0");
hostIpIfaces = ipv4helpr.Assign (hostDevices);
```

Among the available Helpers we have chosen to use the V4PingHelper, which permits to configure a ping session between two hosts. In our case, the packet flow is forwarded by the OpenFlow Switch, that is managed by the previously configured controller, managing the data path regarding the two hosts.

```
// Configure ping application between hosts
V4PingHelper pingHelper = V4PingHelper (hostIpIfaces.GetAddress (1));
pingHelper.SetAttribute ("Verbose", BooleanValue (true));
ApplicationContainer pingApps = pingHelper.Install (hosts.Get (0));
pingApps.Start (Seconds (1));
```

Finally, through the methods of the Simulator class, we defined the duration of the simulation, managing the life cycle of the simulation through the Run() and the Destroy() methods.

```
// Run the simulation
Simulator::Stop (Seconds (10));
Simulator::Run ();
Simulator::Destroy ();
}
```

This script permits us to demonstrate a proper functioning of the OFSwitch 1.3 module in simulation concerning a generic fixed network.

4.5 Script-based simulations: LTE networks and OFSwitch 1.3

In this section, we prove the proper integration and functioning of the OFSwitch 1.3 module in the case of simulation concerning LTE networks. The simulation is script-based and the script is similar to the one used for the fixed network simulation. For this reason, the overview of the script regards only the new entities deployed in the simulation.

In this case, the script permitted us to run a simulation demonstrating the effective connection managed by the configured controller through a single Openflow Switch, between a UE and a remote fixed host.

The first step has been the creation of the LteHelper and of the EpcHelper through the relative methods in order to properly create both the RAN and the EPC architecture models.

```
// Create LTE/EPC helper

Ptr<LteHelper> lteHelper = CreateObject<LteHelper> ();
Ptr<PointToPointEpcHelper>epcHelper =
CreateObject<PointToPointEpcHelper> ();
lteHelper->SetEpcHelper (epcHelper);
```

The second step has been the creation of the remotehost node through the relative container, installing the internet protocol stack.

```
// Create a single RemoteHost
NodeContainer remoteHostContainer;
remoteHostContainer.Create ();
Ptr<Node> remoteHost = remoteHostContainer.Get (0);
InternetStackHelper internet;
internet.Install (remoteHostContainer);
```

Then through the relative NodeContainer and the method the node for the Openflow switch is created.

```
// Create switch nodes
NodeContainer switches;
switches.Create (1);
```

The following lines permitted us to create, through the appropriate NodeContainer, the nodes for the UE and for the enb.

```
// Create ueNodes/enbNodes
NodeContainer ueNodes;
NodeContainer enbNodes;
enbNodes.Create ();
ueNodes.Create ();
```

The creation of the nodes for the controller and for the pgw have been realized through the smart pointer Ptr<Node>.

```
// Create the controller node
 Ptr<Node> controllerNode = CreateObject<Node> ();

 // Create the pgw node
 Ptr<Node> pgw = epcHelper->GetPgwNode ();
```

The following lines permitted us to realize the configuration of the Internet connection through the PointToPointHelper, setting different values.

```
 // Create the Internet
 PointToPointHelper p2ph;
 p2ph.SetDeviceAttribute ("DataRate", DataRateValue
(DataRate ("100Gb/s")));
 p2ph.SetDeviceAttribute ("Mtu", UintegerValue (1500));
 p2ph.SetChannelAttribute ("Delay", TimeValue (Seconds
(0.010)));
```

The following lines configures the interconnection between the fixed remote host and the pgw, setting the chosen ip address pools.

```
 NetDeviceContainer internetDevices = p2ph.Install (pgw,
remoteHost);
 Ipv4AddressHelper ipv4h;
 ipv4h.SetBase ("1.0.0.0", "255.0.0.0");
 Ipv4InterfaceContainer internetIpIfaces = ipv4h.Assign
(internetDevices);
 Ipv4Address remoteHostAddr = internetIpIfaces.GetAddress ();
```

```
 Ipv4StaticRoutingHelper ipv4RoutingHelper;
 Ptr<Ipv4StaticRouting> remoteHostStaticRouting=
ipv4RoutingHelper.GetStaticRouting (remoteHost-
>GetObject<Ipv4> ());
 remoteHostStaticRouting->AddNetworkRouteTo (Ipv4Address
("7.0.0.0"), Ipv4Mask ("255.0.0.0"), 1);
 remoteHostStaticRouting->SetDefaultRoute (Ipv4Address
("10.0.0.5"), 1);
```

Through the mobilityHelper it has been possible to configure the mobility model of the nodes involved in the simulation. In our case, we decided to implement a constant position mobility model, i.e., all the nodes persist in the same position during the simulation.

```
// Install Mobility Model
MobilityHelper mobility;

mobility.SetMobilityModel("ns3::ConstantPositionMobilityMod
el");
 mobility.Install(enbNodes);
 mobility.Install(ueNodes);
 mobility.Install(remoteHostContainer);
 mobility.Install(switches);
 mobility.Install(controllerNode);
 mobility.Install(pgw);
```

We created the devices on top of previously created nodes, and we configured the mac scheduler, in this case a proportional fair MAC scheduler.

```
// Create Devices and install them in the Nodes
(eNB and UE)
NetDeviceContainer enbDevs = lteHelper->InstallEnbDevice
(enbNodes);
 NetDeviceContainer ueDevs = lteHelper->InstallUeDevice
(ueNodes);
 // Default scheduler is PF, uncomment to use RR
 //lteHelper->SetSchedulerType ("ns3::RrFfMacScheduler");
```

The following permits to install the IP stack on top of the UE devices, to set the relative interfaces and to assign to each UE device one IP address from those previously specified. Finally, the default gateway for the UEs is configured.

```
// Install the IP stack on the UEs

internet.Install (ueNodes);
 Ipv4InterfaceContainer ueIpIface = epcHelper-
>AssignUeIpv4Address (NetDeviceContainer (ueDevs));

 // Assign IP address to UEs, and install applications
 for (uint32_t u = 0; u < ueNodes.GetN (); ++u)
  {
    Ptr<Node> ueNode = ueNodes.Get (u);

    // Set the default gateway for the UE
    Ptr<Ipv4StaticRouting> ueStaticRouting =
ipv4RoutingHelper.GetStaticRouting (ueNode-
>GetObject<Ipv4> ());
    ueStaticRouting->SetDefaultRoute (epcHelper-
>GetUeDefaultGatewayAddress (), 1);
   }
```

The next lines are necessary to set the attach of the specified UE to the chosen enb.

```
// Attach one UE per eNodeB
 for (uint16_t i = 0; i < numberOfNodes; i++)
    {
     lteHelper->Attach (ueDevs.Get(i), enbDevs.Get(0));
    }
```

These lines are used to configure the application that causes the traffic generation and the relative intercommunication between the Ue node and the remote host node, specifying the necessary parameters.

```
// Configure applications to generate traffic
UdpEchoServerHelper echoServer (9);
ApplicationContainer serverApps = echoServer.Install
(remoteHost1);
 serverApps.Start (Seconds (1.0));
 serverApps.Stop (Seconds (20.0));
UdpEchoClientHelper echoClient (remoteHostAddr1, 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue (5));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
 echoClient.SetAttribute ("PacketSize", UintegerValue (512));
ApplicationContainer clientApps = echoClient.Install
(ueNodes.Get(0));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (20.0));
serverApps.Start (Seconds (2));
clientApps.Start (Seconds (4));
```

Finally, the simulation is configured and the NetAnim tool is used in order to visualize the different information during the execution of the simulation. The NetAnim tool is an offline animator based on the Qt toolkit. Through this tool, we animated the simulation using the XML trace file collected during simulation. Furthermore, the following lines permit to set the necessary parameters in order to visualize the node position statistics with node trajectory plotting (path of a mobile node), and to show IP and MAC information, including peer IP and MAC for point-to-point links.

```
// Run the simulation

Simulator::Stop(Seconds(simTime));

ns3::AnimationInterface anim (animFile);
anim.SetConstantPosition(ueNodes.Get(0), 10.0, 20.0);
anim.UpdateNodeDescription(ueNodes.Get(0), "Ue0");
anim.SetConstantPosition (enbNodes.Get(0), 10.0, 30.0);
anim.UpdateNodeDescription(enbNodes.Get(0), "EnodeB 0");
anim.SetConstantPosition (switches.Get(0), 20.0, 20.0);
anim.UpdateNodeDescription(switches.Get(0), "Switch");
anim.SetConstantPosition (controllerNode, 30.0, 20.0);
anim.UpdateNodeDescription(controllerNode, "Controller");
anim.SetConstantPosition (remoteHostContainer.Get(0), 40.0,
40.0);
anim.UpdateNodeDescription(remoteHostContainer.Get(0),
"Remote Host 0");
anim.SetConstantPosition (pgw, 30.0, 30.0);
anim.UpdateNodeDescription(pgw, "P-GW");
Simulator::Run();
Simulator::Destroy();
}
```

4.6 Results and considerations

As can be seen in Fig. 4.3, the build and compile processes related to the C++ script described in the previous section have been successfully executed. Furthermore, after the run process, thanks to the NetAnim tool, we visualized the configured positions of all the specified nodes. As set in the script, the simulation generated six different nodes: the UE, the OpenFlow Switch, the controller, the P-GW, the enb and the remote host.

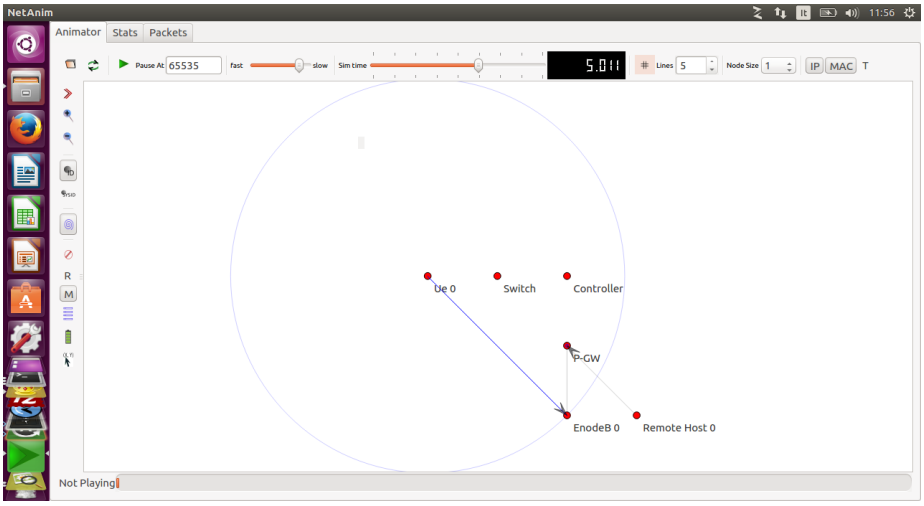

Figure 4.3: *ns-3 simulations, node positons*

The Fig.4.4 shows the proper communication between the Ue and the remote host. The relative packets exchange flows through the OpenFlow switch and it is managed by the configured controller.

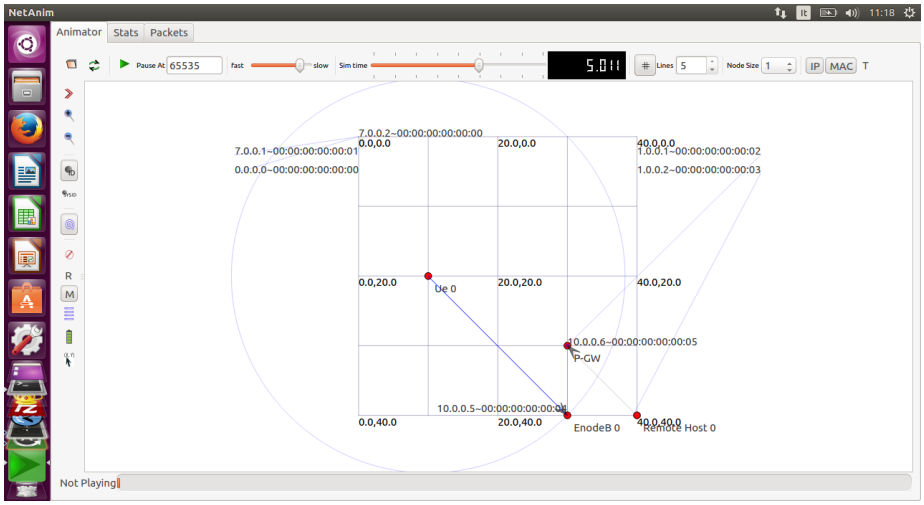

Figure 4.4: *ns-3 simulations, node interactions*

The considerations based this first version of the testbed, based on the ns-3 simulator are this simulation environment permitted us, through the execution of the two aforementioned C++ scripts, to test a proper use, for our objectives, of the OpenFlow protocol when it is embedded in a mobile network context. It has been possible thanks to the configuration work conducted in order to properly use ns-3 and the OFSwitch 1.3 module. Despite these positive aspects, the use of ns-3 simulator and the OFSwitch 1.3 module, permitted us only to prove a proper integration of the OpenFlow protocol in its nature context, i.e. the wired side of the mobile networks. Thus, utilizing a testbed based on these tools it has not been possible to prove the remaining innovative aspects of the proposed software defined architecture. In order to prove these concepts in a wireless/mobile context we have decided to base our testbed on another solution, exploiting other types of software tools.

CHAPTER
FIVE

5 MOSAIC5G RESEARCH PROJECT

5.1 MOTIVATIONS

In order to overcome the negative aspects of the ns-3 based testbed, with the aim of implementing an emulated environment, closer to the real networks than the ns-3 environment, that permits to deploy only a simulated environment, we decided to design and to implement a new flexible SDN/NFV-based SON testbed for future 5G mobile networks based on distinct software tools. The main contribute of our work is to cover the need for a SDN/NFV-based testbed, enabling the investigation of the potential of these paradigms for practical implementations. The research project related to my PhD has been conducted in the context of an international academic collaboration between the University of Glasgow and the University of Catania.

I have technically coordinated this research activity and the relative

participation of our team, in representation of the University of Catania, to the “Mosaic5G” research project, as contributor member.

5.2 Mosaic5G

As specified in [21], the Mosaic5G initiative is created in order to provide an open, flexible and agile 4G/5G experimentation platform. The main objective is to share an ecosystem of open-source platforms and use cases for 5G system research exploiting SDN, NFV and MEC as key technology enablers. Mosaic-5G Ecosystem consists of the following elements:

- **OpenAirInterface (OAI)** [22], composed of OAI-RAN and OAI-CN as 3GPP compatible implementations of a subset of RAN (Release 14) and CN (Release 12) features, respectively.

- **FlexRAN** [23], as a flexible and programmable platform developed to implement the SDN concepts at the RAN domain, enabling a SD-RAN.

- **JOX** is an event-driven Juju-based service orchestrator core with several plugins to interact with different network domains, e.g., RAN and CN.

- **Store** includes a constellation of platform packages, software development kits, network control applications and datasets.

- **LL-MEC** is an ETSI-aligned MEC platform that can act as a software-defined core network controller.

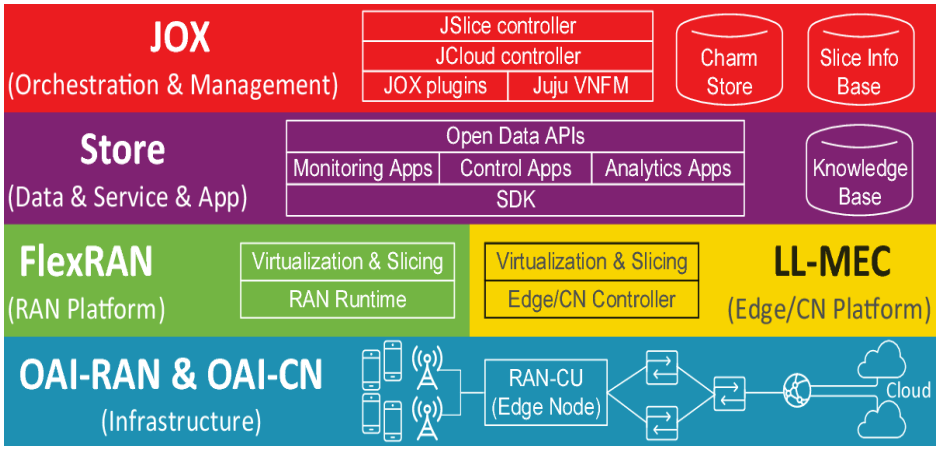

Figure 5.1: *MOSAIC5G platform*

In the research activity only OpenAirInterface RAN, OpenAirInterface CN and FlexRAN controller entities are used, so in the next section only these will be described in details.

5.3 OpenAirInterface

As described in [22] OpenAirInterface is an open experimentation and prototyping platform created by the Mobile Communications Department at EURECOM to enable innovation in the area of mobile/wireless networking and communications, as an open-source software-based implementation of the 3GPP LTE protocol stack. As stated in the previous section, the specific release referenced by the research activity is the LTE Release 14 (OAI RAN), because actually a

full 5G release of the platform is still under development. Since 2018, it has been included in the Mosaic5G initiative.

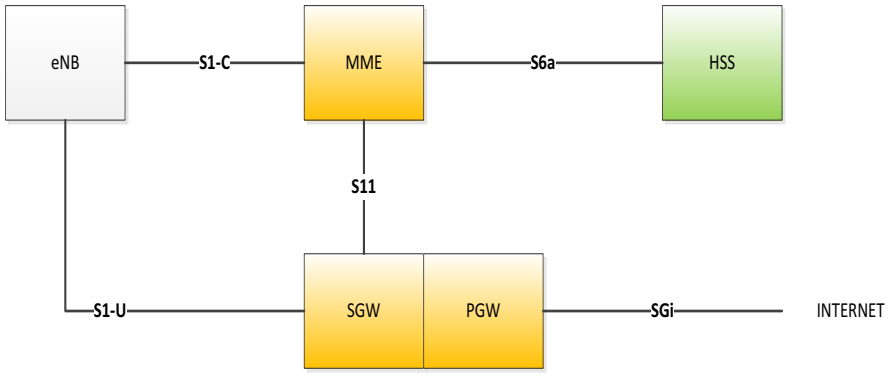

Figure 5.2: *OpenAirInterface LTE entities*

The RAN and CORE sides of the 3GPP protocol stack are developed through two different branches:

- **openairinterface5g** as a software implementation of the RAN side.

- **openair-cn** as a software implementation of the EPC side.

In order to realize a better comprehension of the functionalities created by leveraging on these software platforms, it is very useful to have a brief overview of Network Elements and Source Code architecture of OpenAirInterface. Indeed, specific OAI files and functions have been identified, properly editing the related code in order to achieve the research objectives, as will be described in the next chapters.

Different deployment scenarios can be considered with the EURECOM eNB and UE as follows:

- Commercial UE <-> OAI eNB + Commercial EPC
- Commercial UE <-> OAI eNB + OAI EPC
- Commercial UE <-> Commercial eNB + OAI EPC
- OAI UE <-> Commercial eNB + OAI EPC (experimental)
- OAI UE <-> Commercial eNB + EPC (experimental)
- OAI UE <-> OAI eNB + Commercial EPC (experimental)
- OAI UE <-> OAI eNB + OAI EPC
- OAI UE <-> OAI eNB

In particular, each entity can be deployed on the same host, in an all-in-one deployment, or in a dedicated host. As regards, the code, each repository is available through two different branches, the Master Branch and Develop Branch. The develop branch is used to prove the stability of the new features, which after are embedded in the develop branch. These two branches are the main ones, but several additional branches called tags are available and permits to test specific use cases or research aspects. The OAI source code in a release directory or in the trunk directory is organized as follows:

- **cmake_targets**, Openair build system (latest)
- **common**, Common code to all layers
- **openair1**, Physical layer source code
- **openair2**, Layer 2 (MAC, RLC, RRC, PDCP) source code

- **openair3**, Middleware code (mainly unused)
- **targets**, Specific code for executables

The OAI code needs a powerful x86 system to be properly executed. The following are the requirements regarding the type of supported CPU:

- Intel Core i5, i7 Generation 3/4/5/6
- Intel Xeon
- Intel Atom
- At least 4 cores > 3GHz

In order to realize the RF layers' functionalities, it is necessary to connect via a wired link these systems to a Software Defined Radio (SDR) platform. SDR is a radio communication system which permits to execute in a software fashion (on top of the systems), traditionally implemented in hardware (e.g. mixers, filters, amplifiers, modulators/demodulators, detectors, etc.). Only a subset of the available SDR platforms are compatible with the OAI platform, others are still under testing:

- ExpressMIMO2
- USRP B2xx, X300 and X310
- Blade RF
- LMS-SDR
- Sidekiq (experimental)

In particular, during the research activity the USRP PC-hosted software radio platforms B2xx and X310 have been used, based on the USRP Hardware Driver software (UHD).

In addition to the real-time mode exploiting the aforementioned SDR platforms, the full protocol stack can be executed in the emulation mode for validation and performance evaluation from both system and link level perspectives. The emulation mode capability is developed in order to properly reproduce the behavior of the wireless access technology in a real network setting while respecting the temporal frame timing of the air-interface. In particular, two distinct emulation modes are available:

- **PHY Abstraction mode:** This mode leverages on the PHY abstraction unit which simulates error events in the channel decoder;

- **Full PHY Layer mode:** This is a more detailed and computationally intensive mode, which exploits convolution of the real PHY signal with an emulated channel in real-time.

The emulator permits to exploit the available the 3GPP channel models, with specific path loss, shadow fading and stochastic small scale fading parameters. The emulation modes can be a proper way of testing of new functionalities prior to its deployment in real RF environment exploiting the SDR platforms. Thanks to these features OAI is a more suitable solution compared to system-level simulations through, e.g., MATLAB, which are based on analytical approach with no notion of time. Discrete-event simulator as ns-3 models the protocol layers abstracting them or executes the relative functionalities in an abstracted mode, whereas the OAI implements the full protocol stack to run on a real execution environment respecting frame timing constraints. These

aspects permit to OAI to be a more realistic platform (even in emulation mode) compared to the aforementioned alternatives.

These are the positive aspects and the reasons that pushed us to use the OpenAirInterface platform instead of the ns-3 simulator.

5.4 FlexRAN

In order to implement the Control Data Separation concept in our proposed architecture, we have exploited the FlexRAN software platform. In fact, as previously exposed, our proposal envisages a centralized entity that is in charge of managing various control decisions related to a set of cells. The FlexRAN platform is designed to accomplish these requirements, as an open-source SD-RAN platform able to flexibly separate control and user plane operations. Furthermore, due to its specific design it permits to centralize RAN domain control logics among multiple base stations (either monolithic or disaggregated RAN) or to delegate control decisions in a distributed fashion.

In this line, FlexRAN allows to exploit different control functions in a hierarchical control framework performing a “real-time” monitoring, control delegation and reconfiguration in the RAN domain. The FlexRAN software platform is a good solution thanks to peculiar characteristics as flexibility and programmability. These features enable a productive deployment of different types of functionalities. Finally, its open source nature permits to edit the code and create new functionalities.

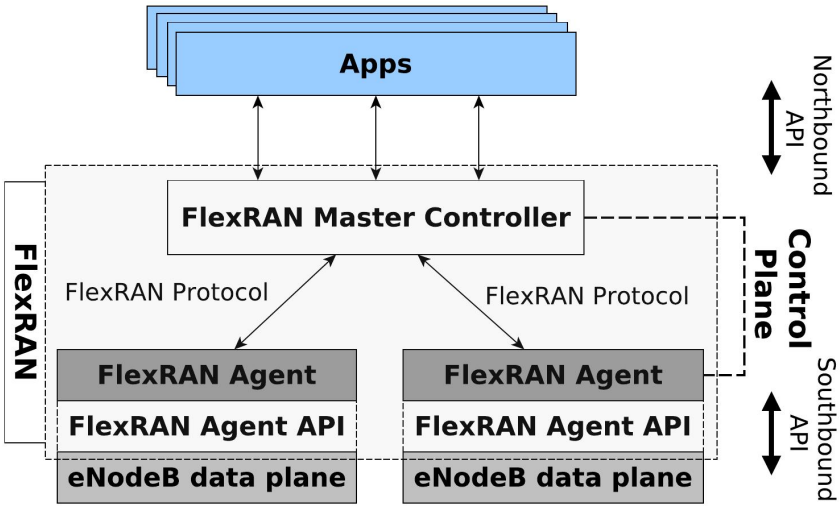

Figure 5.3: *FlexRAN*

As shown in Fig. 5.3, the two main entities composing the FlexRAN software platform are:

- **FlexRAN Master Controller**, enabling coordinated control over multiple RANs, providing high/low-level primitives and provision SDKs for control application.

- **FlexRAN Agent**, acting as a local control entity, virtualizing the underlying RAN radio resources, and enabling distributed control applications.

Control plane is composed of a Master Controller connected to a number of FlexRAN Agents, one for each eNodeB. Control and data plane separation is provided by the FlexRan Agent API, which act as the southbound API with FlexRAN control plane on one side and eNodeB.

Thanks to this architecture, FlexRAN can support various slice requirements (e.g., isolation) and also improve multiplexing benefits (e.g., sharing) in terms of radio resource abstractions. The communication between the FlexRAN Master Controller and the FlexRAN Agent is performed through the FlexRAN protocol, a protocol a la OpenFlow, providing capabilities as statistics, reconfiguration, triggering of events and control delegation. FlexRAN software platform is strictly related to the OpenAirInteface software platform. In fact, it is born as an extension to a modified version of the OpenAirInterface platform.

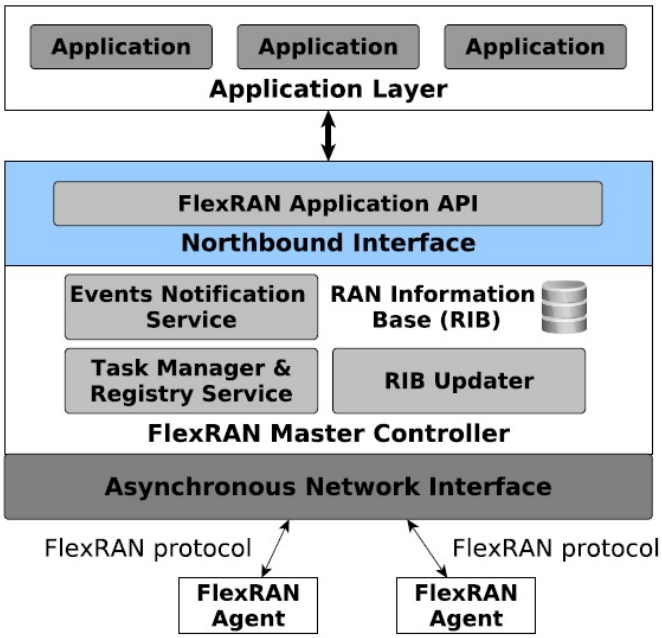

Figure 5.4: *FlexRAN entities*

FlexRAN Master controller leverages on the RIB (RAN Information Base), a database storing the statistics and configuration regarding UEs, eNodeBs and the relative FlexRAN agents, to manage the Network. In particular, the RIB is structured as a forest graph. The root note of each tree is an Agent and the nodes of the second level are the cells associated with a specific agent. Leaves are UEs associated to a specific cell. Each RIB modification is managed by the RIB updater. More specifically, the applications are not allowed to modify the RIB but send information about modifications to the agents through the northbound interface. Modifications are delivered back to the Master through the Statistics reports and event notifications sent by the agent.

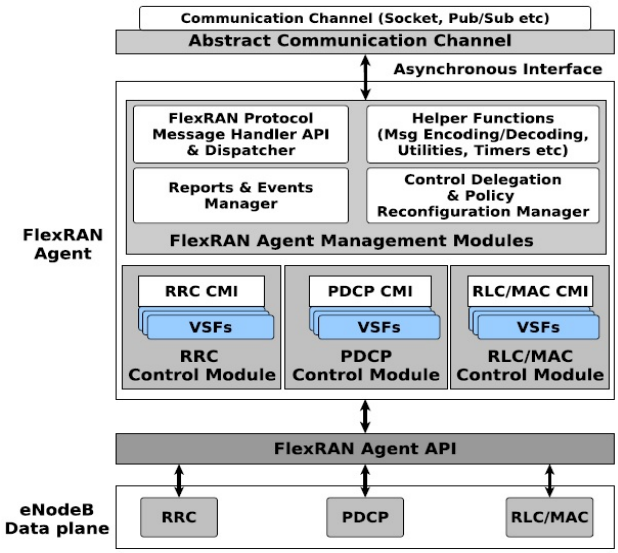

Figure 5.5: *FlexRAN VSFs*

FlexRAN Agent provides a number of eNodeB Control Modules, e.g., the RRC Control Module for the Radio Resource Management and the RLC/MAC Control Module for scheduling. Each Control Module is executed by one or more Virtual Subsystem Function (implemented in C), for the actual action of the specific functionality performed by the Agent. The Number and Type of VSFs per each control module is defined through a CMI (Control Module Interface). CMI is placed at higher level, allowing the Agent to abstract the set of operations of the Control Module. It is worth noting that new operations can be introduced by extending the Control Module Interface. In order to reduce signalling overhead, Master is able to decide to perform Control Delegation functionalities (FlexRAN Agent Management module). This can be a very useful (and sometimes essential) solution when considering centralized time-critical applications (for example remote scheduling of more eNodeBs). Indeed, when operations require low-latency decision-making, Master Controller could be not quite fast enough. Consequently, individual agents get the delegation for time critical decision.

Finally, as said Master and Agent sides interact each other through the FlexRAN Protocol and more specifically through FlexRAN protocol messages. A message handler and dispatcher entity residing at the agent side is responsible to receive FlexRAN protocol messages from the FlexRAN master controller and forward them to be handled by the appropriate VSF of the corresponding control module, using the FlexRAN Agent API. TCP is used for the communication of the agents with the master and the exchange of protocol messages.

CHAPTER
SIX

6 OPENAIRINTERFACE/FLEXRAN BASED TESTBED

6.1 MOTIVATIONS

As underlined in [24], the enabling aspects of the envisioned 5G RAN architecture will be the Network Function Virtualization (NFV) paradigm and the control plane/data plane (CP/DP) splitting concept. The CP/DP splitting is explicated through the Control Data Separated Architecture (CDSA) in the RAN context and the SDN paradigm in the core network context. CDSA is considered a necessary step towards the developing of the fifth generation of mobile networks. CDSA approach allows a logical separation between signaling and data traffic in RAN, also guaranteeing traffic off-loading.

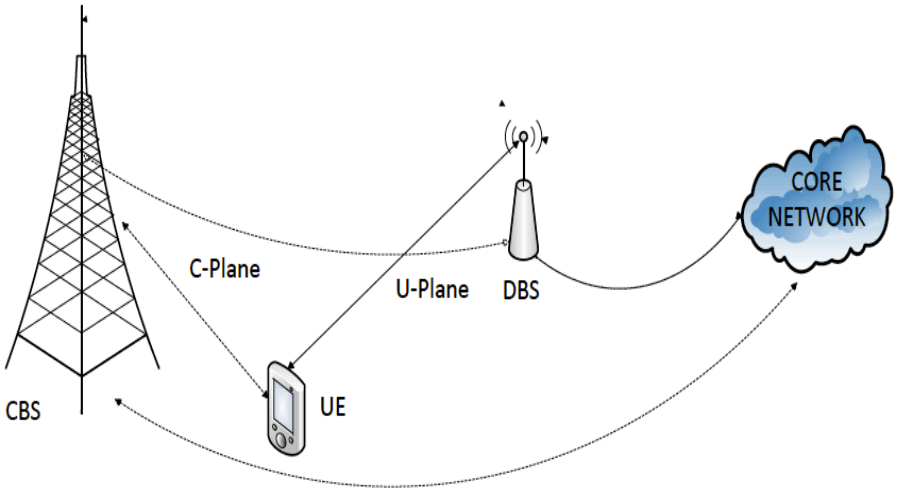

Figure 6.1: *CDSA*

As shown in Fig. 6.1, Macro cells, also known as Control Base Stations (CBSs) are required to provide the control plane (signaling traffic), while Small cells, also known as Data Base Stations (DBSs), take care of data transmission. CBSs are conceived to provide continuous and reliable coverage at low frequency bands, while DBSs offer high data rate traffic by using higher frequency bands. Thus, the CDSA offers several advantages, e.g., improved energy efficiency, signaling efficiency, and mobility management. A further aspect which will play an integral role in future mobile networks is the SON concept. As underlined in [25], concerning the previous generations of mobile networks, the SON concept was introduced to automate the Operation and Maintenance (O&M) processes in order to improve network efficiency and performance. As regards 5G mobile networks, the SON concept will cover not only the O&M context, but a set of different aspects. Thus, it is required an optimization of the original view, leading to a so-called Next

Generation-SON (NG-SON). NG-SON will permit to achieve a full awareness of the current RAN and core network status, enabling a proper management through the determination of optimal network parameter values, thus minimizing the human intervention.

As regards the small cells, the SCF has already predicted that there would be tens of millions of new SC deployed around within next couple of years [26], so centralized and complex methods would be no more feasible to implement. Thus, it will be necessary to choose appropriately the degree of centralization in terms of control plane functionalities and NGSON capabilities, depending on the considered scenario. SDN, NFV and NG-SON are considered the enabling technologies to achieve these goals.

6.2 OpenAirInterface/FlexRAN based testbed

As regards CDSA and SON, in literature most of the works that explore these concepts are based on simulations, which cannot model all the peculiarities of real networks such as signaling delays, effect of implementation and measurement inaccuracies, and delays in channel and measurement reports. At the best of our knowledge, one of the few works which considers a CDSA testbed is [27], where the authors validate their spectrum management application.

Following the above assumptions, in order to deploy a flexible, programmable and virtualized RAN, we design a second implementation of our testbed, based on our proposed hierarchical layered software defined architecture, aiming to provide new features that facilitate

SDN/NFV integration in future mobile networks.

The main contribute of our work is to cover the need for a CDSA based testbed, enabling the investigation of the NG-SON capabilities for practical implementations. The SDN Controller logical area, the NFV Orchestrator logical area and the Resource Management Controller logical area functionalities, proposed in our SDN/NV C-RAN architecture, can be considered as perfect examples of NG-SON capabilities.

We deploy these NG-SON capabilities through OpenAirInterface and FlexRAN, because after an in-depth understanding of the inherent features of these software tools, we realized that these features would have enabled us to implement in a SDN/NFV based real-time testbed the designed logical areas.

In particular, in the following we describe two implemented use cases, presented in two different papers, regarding features of the NFV Orchestrator logical area and the Resource Management Controller logical area functionalities. An implementation of the functionalities related to the SDN Controller logical area, based on the OpenAirInterface and FlexRAN software tools, is a work in progress and is described in the next chapter.

Furthermore, aside from the implementation of these proposed functionalities, the testbed aims to address certain open research problems, e.g., optimal location of the DBSs when the static 3D channel mapping of the environment is available, DBS switching on/off for energy saving and impact of user and traffic distribution on specific KPIs. We focus our proposal on two experimental setups, a real one and a virtualized one, both based on the FlexRAN and OpenAirInterface software tools.

The first Testbed in Catania (TestbedCT), based on Network Function Virtualization to set-up an entirely emulated environment. A subsequent Testbed in Glasgow (TestbedGLA), as a real environment deployment, using SDRs and real user equipments. Both the testbed deployments enable novel research and provides teaching opportunities in next generation RAN architectures and several other areas of system-level research. Furthermore, they act as benchmark for many use cases with a significant proximity to real network deployment criticalities.

6.2.1 Virtualized testbed

The virtualized setup runs on the 5G-SDN/NFV testbed at the University of Catania and is performed through virtualized entities. In particular, the FlexRAN controller entity, as for the OAI entities, are deployed trough the virtual machine (VM) technology. In particular, as shown in Fig. 6.2, we deploy the OAI and FlexRAN entities each one on a dedicated VM.

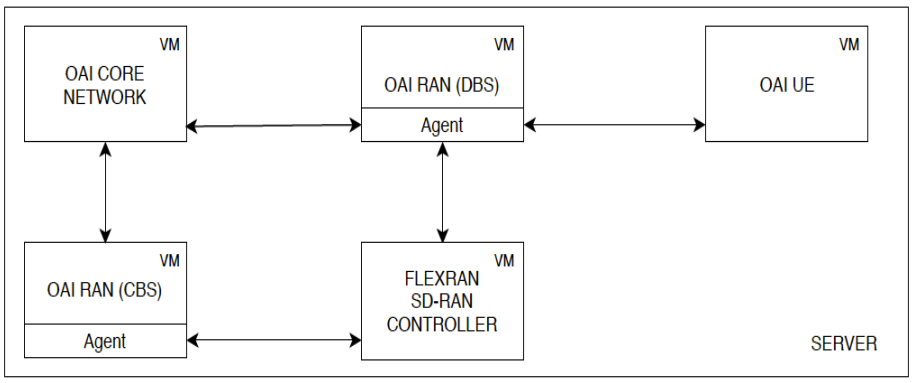

Figure 6.2: *Virtualized Testbed*

The deployment can be performed by changing the considered parameters in terms of Operating system, kernel version, vCPU and vRAM. In particular, as regards the releases for each involved entity, we utilize OAI RAN tag 2018.w41 (with agent support), OAI CN v 0.5 and FlexRAN Real Time Controller v2.0. The wireless medium is obtained using PHY abstraction unit feature of OAI, causing the emulation of the RAN, RFs and UEs entities. As regards the hardware machines the Testbed exploit one SuperMicro Server Intel Xeon CPU E5-4610 v2 @ 2.30GHz, RAM 256 GB.

6.2.2 Real testbed

The real setup runs on the 5G-SON testbed at the University of Glasgow. As shown in Fig. 6.3, the real testbed is performed through servers and real RF, i.e., Software Defined Radio (SDR) Platforms. As regards the hardware machines the Testbed is composed of:

- one Mini PC Kit NUC7i7 DNHE, for OAI RAN 2018.w36 tag
- one Intel Server System R1304SPOSHBNR Intel Xeon E3-1220 v6 Quad-core @3 GHz, for OAI Core Network v.0.5
- one Tower Desktop with the same processor model as the Server machine, for FlexRAN Controller v 2.0
- Universal Software Radio Peripheral (USRP) B205mini for small cell acting as RF, connected to the RAN machine through a USB 3.0 connection
- USRP X310 for macrocell, acting as RF. The X310 could be connected to a different RAN machine through a 10Gigabit Ethernet connection with SFP+ termination

The said machines are connected each other to a Netgear Prosafe XS716E 10Gigabit switch, with 10Gigabit Ethernet cables. The University holds an experimental license in the Band 7 frequency range. The UE is performed with two different solutions, a COTS phone and a USB dongle. In the COTS phone case, we exploit a real smartphone as UE, properly programmed the USIM, in order to be able to perform the attach to the network triggered with the OAI emulation platform. In the USB dongle, we deploy the UE through a NUC mini-pc, connecting it to a USB dongle, with a properly programmed embedded USIM performing the UE.

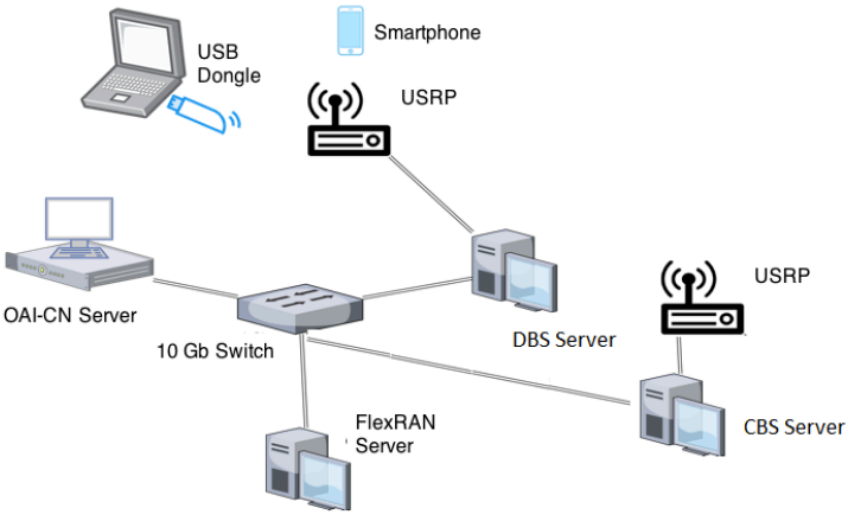

Figure 6.3: *Real Testbed*

6.3 First case study: RAN entities life-cycle management

In this paper, first we describe the implementation of a specific case study, i.e., the RAN entities activation/deactivation procedures. We consider three steps.

First, we create an application, in order to dynamically trigger the instantiation of a base station in terms of virtualized or real entities. Second, editing the native code, we implement specific commands, running on top of FlexRAN/OpenAirInterface, to dynamically manage the RAN entities life-cycle. Finally, we performed time measurements, concerning the RAN entities activation/deactivation procedures, to prove the proper Testbed functioning.

6.3.1 The application

The application is designed on the basis of the native FlexRAN/OpenAirInterface instructions. As shown in Fig. 6.4, after the initialization step, properly issuing the specified instructions as input parameters, the application is able to dynamically trigger the creation of RAN and core network entities, exploiting the output parameters computed in the computation step. The instantiation, depending on the considered testbed setup, can be executed by the creation of the related real or virtualized entities. So, the specific instructions issued in the computation and the creation steps are strictly related to the specific testbed setup.

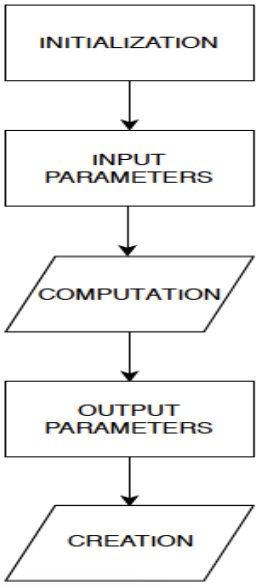

Figure 6.4: *Flowchart*

Generically, the OAI instruction which permits to trigger the instantiation of a base station is:

sudo -E ./lte-softmodem -O
/home/user/openairinterface5g/targets/
PROJECT/GENERIC-LTE_EPC/CONF/file.conf

In the performance evaluations section, we will contextualize the specific instructions issued in the computation and creation steps.

6.3.2 Start and stop commands

In order to dynamically issue through the application, the activation and the deactivation of a given base station, we exploit a native FlexRAN/OpenAirInterface command and its related API endpoint, properly editing it to perform the considered tasks. In the vanilla FlexRAN/OpenAirInterface code, the so-called Cell Reconfiguration command permits to change the cell configuration of the specified base station. The issue of this command causes the so-called soft-restart of the base station. The soft-restart is intended as the restart of the L1/L2/L3 protocol stack layers. The cell reconfiguration command is performed through a command like:

curl -X POST http://FLEXRAN-URL:PORT
/conf/enb/:id? --data-binary "@file.json"

The parameters which can be modified are specified in “file.json” file. At the time of writing it is possible to change the associated value only for these parameters: dlBandwidth, ulBandwidth, dlFreq, ulFreq end eutraBand. The file has the following format:

“dlBandwidth” : value ,
“ ulBandwidth” : value ,
“dlFreq” : value ,
“ulFreq” : value ,
“eutraBand” : value

The design of the application and the command leverages on an appropriate code editing conducted thanks to a in depth understanding of

the FlexRAN/OpenAirInterface files and functions involved in the cell reconfiguration command workflow.

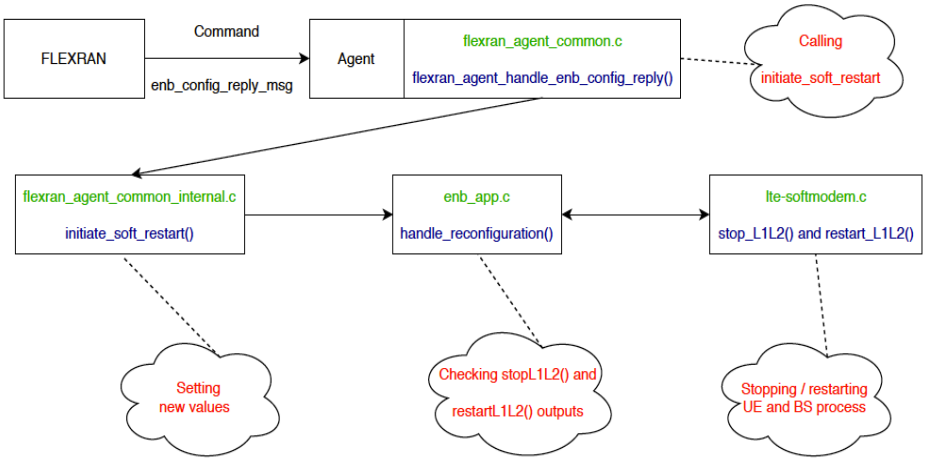

Figure 6.5: *Command workflow*

As shown in Fig. 6.5, the command, issued through the command line, causes the creation of the enb-config-reply message. This message is managed by the agent through the flexran-agent-handle-enb-config-reply function defined in the flexran-agent-common.c. This function has a double task. In particular, checking the message type field, if the specific message is a enb-reconfiguration-message it is charge to call the initiate-soft-restart function, which is defined in the flexran-agent-common-internal.c. This function has the scope of setting the values specified in the json file for the given parameters. Furthermore, it performs a check in order to accept only the allowed values. The effective restart of the base station is performed through the stop-L1L2 and the restart-L1L2 functions defined in the lte-softmodem.c.

The functions perform respectively the stop and the restart of a given base stations by deactivating and reactivating the L1/L2/L3 protocol stack layers. The interconnection with the core network and FlexRAN controller entities are not interested by the procedure. This is the reason for naming this procedure softrestart. Finally, the handle-reconfiguration function, which is defined in the enb-app.c checks the correctness of the cell reconfiguration procedure.

Considering the native soft-restart as the track command, our contribution comes with the creation of three different commands: stop, start and Stop&Restart. The stop command permits to stop a given base station specifying its agent id. In a dual way, the start command triggers the start of a particular base station given the specific agent id. Finally, the Stop&Restart command, as a mix of the two aforementioned commands, permits to dynamically control the life-cycle of a given base station. More specifically, the start and the stop commands have been implemented by isolating the related code entities and by disabling them.

In order to realize the stop command, the start related code entities have been disabled, whereas the stop part of the code is disabled for the start command. The in depth understanding of the considered FlexRAN/OpenAirInterface code permits us to implement a more complex command, the Stop&Restart command.

By editing the related code, we implement a dynamic control of the state of the base station. In particular, we can manage the life-cycle of the base station in terms of activation and de-activation time, respectively called Ton and Toff.

Furthermore, we can manage the activation/deactivation of a specific base station performing, at the end of the Ton time interval, a check regarding the attached UEs. If one or more UEs are attached, in order not to cause a discontinuity of the service for the considered UEs, the base

station is maintained active. If no UE is attached we can decide to deactivated the base station or let it in the active state.

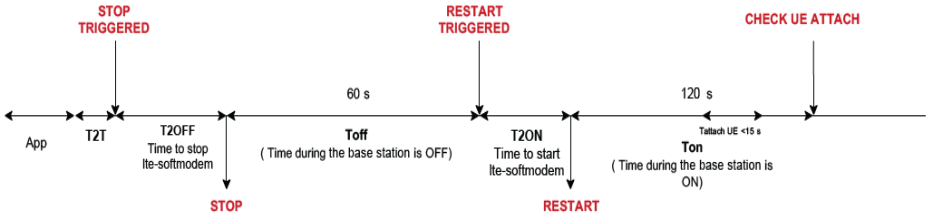

Figure 6.6: *Stop&Restart*

The Fig. 6.6 shows an example of a specific execution of the Stop&Restart command. By changing the considered values, we are able to manage the life-cycle of the base station. In this particular case, Toff is set to 60 seconds and Ton to 120 seconds.

6.3.3 Timing evaluations

In order to prove the effectiveness of our proposal, we conduct timing evaluations, concerning the RAN entities activation/deactivation procedures, both on the real and the virtualized testbed. In the case of the real testbed setup, the deployment computation step for a base station is performed through the issue of the following instruction:

sudo -E ./lte-softmodem -O
/home/user/openairinterface5g/targets/
PROJECT/GENERIC-LTE_EPC/CONF/
enb.band7.tm1.50PRB.usrpb205.conf

The lte-softmodem is the software process which permits to perform all the functionalities related to the considered protocol stack concerning the base station. The lte-uesoftmodem is the counterpart concerning the UE. The issue of this command permits us to deploy a base station performing the RF through an USRP b205-mini, exploiting the transmission mode 1, 50 physical resource blocks (PRBs) and band 7 working mode. As regards the real testbed setup, the lte-uesoftmodem process is automatically triggered by in the case of the USB dongle, whereas it is not needed in the case of the COTS phone.

In the case of the virtualized setup, the deployment computation step for a base station is performed through the issue of two different instructions, one for the RAN entity and one for the emulated UE.

More specifically these two instructions, respectively for the RAN entity and the UE entity, are:

sudo -E ./lte-softmodem O
/oai-ran/targets/PROJECTS/PROJECTS/
GENERIC-LTE-EPC/CONF/
rcc.band7.tm1.if4p5.50PRB.lo.conf

sudo ./lte-uesoftmodem O
/oai-ran/targets/PROJECTS/PROJECTS/
GENERIC-LTE-EPC/CONF/rru.oaisim.conf -r
50 --siml1

For the virtualized testbed setup, the issue of two different instructions is needed since in the OAI environment the virtualization of a base station and the associated UEs can be performed only considering a two-tier base station, composed of the base band processing unit and

the RF (respectively specified as rcc and rru in the instruction, considering the IF4P5 functional splitting between these entities). This approach causes the creation of three different software entities, two for the lte-softmodem process (rcc and rru), one for the lte-uesoftmodem (UE). In this case, we deploy an emulated RF which works considering 50 physical resource blocks (PRBs), the band 7, the transmission mode 1.

As regards the emulated UE, with the issue of the related command, we consider 50 PRBs and the L1 emulation (the lower physical layer and the over the air medium). Once the deployment creation step is completed for both the testbed setups, we perform the time measurements by editing the OAI source code using the CPU clock signal. In order to perform the time measurements considering an overall time synchronization, we sync all the involved entities through the Network Time Protocol (NTP).

The first considered parameter is the so-called Time-to- Trigger (T2T), defined as the time between the FlexRAN command triggering and its effective reception performed by the specified agent. As underlined in [27], this parameter depends on the collocation of application, controller and agent (e.g., locally or remotely). Even if in [27] the authors define this parameter, they do not provide any related time measurement. At this aim, we conduct the time measurements, as regards the virtualized testbed setup, both in local and remote case, whereas as regards the real testbed only in the remote case.

The other two parameters are the aforementioned T2ON and T2OFF. As regards the T2ON parameter the time measurement is conducted only in the real testbed setup, since it is not possible, due to code constraints to deploy the restart functionality in the case of two-tier base station, as that required for the virtualization in the OAI environment.

6.3.4 Results and considerations

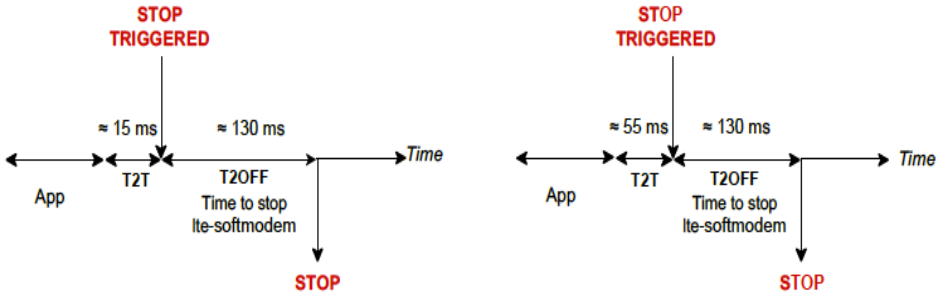

Figure 6.7: *T2T and T2OFF - Virtualized Testbed*

With regards to the virtualized testbed, as shown in Fig. 7, we on average measured:

- T2T of 15 ms, with a minimum value of 11 ms and a maximum value of 19 ms, for the local case (Fig. 7a)
- T2T of 55 ms, with a minimum value of 50 ms and a maximum value of 60 ms, for the remote case (Fig. 7b)
- T2OFF of 130 ms, with a minimum value of 140 ms and a maximum value of 150 ms (Fig. 7a / Fig. 7b)

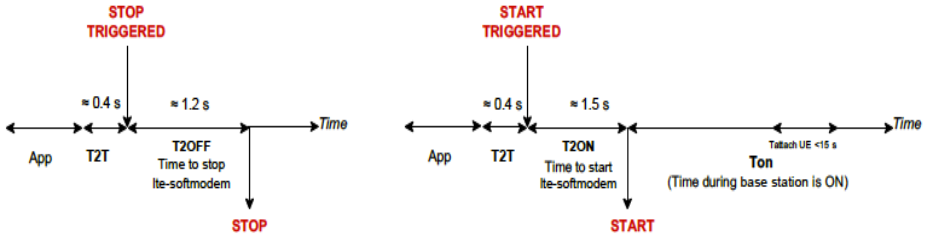

Figure 6.8: *T2T, T2OFF and T2ON - Real Testbed*

Regarding the real testbed, as shown in Fig. 8, we averagely measured:

- T2T of 0.1 seconds (Fig. 8a / Fig. 8b)
- T2OFF of 1.2 seconds (Fig. 8a)
- T2ON of 1.5 seconds (Fig. 8b)

Furthermore, the time measurements permit us to establish a minimum value of 15 seconds for the Ton parameter, defined as the time interval needed to check the attach for a UE on the basis of the messages exchange between the involved entities. Finally, it is useful noting that the values of T2OFF and T2ON as regards, e.g., the virtualized testbed setup do not vary. The explanation is that the implemented Stop&Restart is an example of Hybrid-SON approach, since thanks to the particular architecture design and inherent features of FlexRAN, it mixes centralized-SON (C-SON) and distributed-SON (DSON) approaches. In fact, if the application runs on top of the Master controller it is possible to follow a C-SON approach, whereas in the case of the Slave controller a D-SON approach. Furthermore, if the control delegation feature of FlexRAN controller (the application is performed directly in the context of the agent) is considered, it is possible to implement a fine-grained D-SON approach. So, these considerations are validated by the values of the T2OFF and T2ON parameters.

In fact, as can be seen in the Fig. 6.5, the first step involves FlexRAN (remote or local controller), the subsequent steps involve only the specified agent. So, the D-SON nature of these steps makes the T2OFF and T2ON values independent of any possible external parameter with respect to the context of the agent.

6.4 Second case study: Motion Sensor based Small Cell Sleep Scheduling

The fifth-generation mobile communication network, 5G, promises to stretch the limits of the Key Performance Indicators (KPIs) of current systems by taking into account several criteria such as latency, resilience, connection density and coverage area, alongside the traditional spectral efficiency and Energy Efficiency (EE) criteria in its design. 5G has performance targets of sub-millisecond end-to-end latency, 100-fold increase in typical user data rates, 100 times increase in connection density and 10 times increase in EE, compared to current systems [28]. These targets will have a different importance in the several usage scenarios introduced by IMT Vision recommendation ITU-R M.2083-0 as eMBB, URLLC and mMTC.

Network densification is one of the keys to meet the demand for higher data rates. Network densification refers to adding more cells by deploying new base stations (BSs) in order to achieve higher capacity within an area of interest. However, including more BSs requires consuming more energy to run the network, as the energy consumption of a network is directly proportional to the number of BSs it contains. Since BSs constitute the main part of the energy consumption in mobile cellular networks [29], network densification would make the case severer in terms of EE. An effective and efficient network densification can be accomplished by adopting the CDSA concept as authors underline in [24]. CDSA approach allows a logical separation between signaling and data traffic in RAN. In this architecture, intelligence is partially or completely removed from most of the nodes in the network to be concentrated in fewer central nodes. This results in cost saving, higher performance and resource efficiency [24].

The promotion of the CDSA is closely linked to the concept of the SON which contributes to the flexibility required for 5G. Initially, SON was identified as a key design principle for LTE, focusing on its distributed declination [30]. However, as shown in [25], a NG-SON for future 5G networks, designed in order to maximize automation of all the aspects at all the possible levels, depending on the specific use cases is required. In fact, NGSON provides optimization based on a higher level (cell cluster scale) scenario, Centralized-SON (C-SON), or on a smaller scale, Distributed-SON (D-SON). Each solution has its advantages and a Hybrid-SON (H-SON) architecture brings together all the advantages of D-SON and C-SON. It will be necessary to choose appropriately the degree of centralization in terms of control plane functionalities and SON capabilities, depending on the considered scenario. SDN, NFV and C-RAN are considered the enabling technologies to realize these enhancements. In this context, keeping in mind the paradigms and technologies that will enable ultra-densification, some issues need to be addressed.

As predicted by the SCF [26] over 70 million Small Cells (SCs) will be deployed by 2025, with a corresponding annual energy consumption of more than 3 TWh, special care should be given to EE to avoid enormous energy consumption within the networks. Spatio-temporal changes in data traffics of BSs pave the way for EE by allowing switching unused (or lightly used) BSs off to reduce the energy consumption of the whole network [24]. In other words, since the traffic loads of cells often present various patterns over different days of a week and/or time of a day, BSs could be switched off during low traffic (or no traffic) periods.

Cell switching strategies based on traffic conditions and/or proximity of SCs to a macro cell (MC) have been widely studied in the literature

[31-34]. In [31], the authors propose a traffic load based cell on/off switching mechanism using actor-critic reinforcement learning. Both centralized and distributed solutions are proposed in [32] by considering BS on/off switching, user association, and power control jointly in order to enhance EE of the system.

A mixed integer programming formulation was used for the centralized approach and near optimal solution was obtained. A proximity based SC sleeping technique for Heterogeneous Networks (HetNets) was presented in [33]. SCs, which are far from the MC with a certain threshold, are opportunistically switched off and covered by the neighboring SCs using cell range extension, while the traffic in the SCs closer to the MC is offloaded to the MC in case of switching off. The authors in [34] proposed a joint user association and cell switching algorithm, where activation states are first determined, and then EE is further improved by associating the users accordingly.

In [35], the authors propose a centralized solution which aims to compute the optimum number of base stations to switch off in order to maximize the energy saving, while maintaining coverage, capacity and Quality of Service. It adopts tools in order to include multiple criteria with different priorities in the switch off decision making process. The introduction of multiple decision inputs allows to capture efficiently spatial and temporal traffic fluctuation and, as a consequence, to optimize the set of switched off stations.

Nonetheless, most of the existing techniques related to cell switching rely on complex algorithms, since they often employ centralized approach in order to determine which SCs to switch on/off and/or when to do so. These proposals could be classified as C-SON solutions. To cope with all the envisaged scenarios, the C-SON and D-SON solutions will have to coexist and be suitably adopted. However, in the D-SON

case, complex and expensive methods would be no more feasible to implement due to the large number of SCs.

6.4.1 Proposed algorithm

In this dissertation, a low-cost, low-complexity SC scheduling algorithm is proposed. More particularly, a motion sensor is used in the system in order to detect user presence in an environment of interest. The SC goes into sleep mode if there is no user detected, while it wakes up when the presence of a user is detected by the motion sensor. After waking up, the SC waits for a certain amount of time to check if the detected user is associated with it. If the user is not associated with the SC, then it goes back into sleep mode; however, in case the SC has attachment with the detected user, then it keeps being active and checking whether the attachment is maintained at certain intervals.

Therefore, the proposed cell sleep scheduling is a good example of D-SON, as it introduces a distributed (i.e. the decisions are made locally and there is no need for a central entity) and low-complexity (i.e. it is merely a binary decision process, which is triggered by a motion sensor, for an individual SC) algorithm.

The experimental setup runs on the 5G Self-Organized Network (5GSON) testbed at the University of Glasgow, which is based on OAI. OAI is a software tool enabling an open-source implementation of both the CN and RAN based on 3GPP standards. OAI entities can be either deployed on separate machines or in an all-in-one setting.

With inherent features of FlexRAN, it is therefore possible to enable C-SON and D-SON solutions, performing both centralized and distributed control. In a higher-level control (many macro cells) the FlexRAN performs complex controls with a wider overview: traffic

loads, user density, etc. While, in a local control (distributed), we can delegate the control logic to the Agents activated only in the first instance by FlexRAN. In the latter case, it needs to apply control strategies with low computational complexity. As shown in Fig. 6.9, both the OAI software entities run on an Intel NUC minicomputer. The Radio Frontend (RF) is provided by an Ettus B205mini USRP SDR platform.

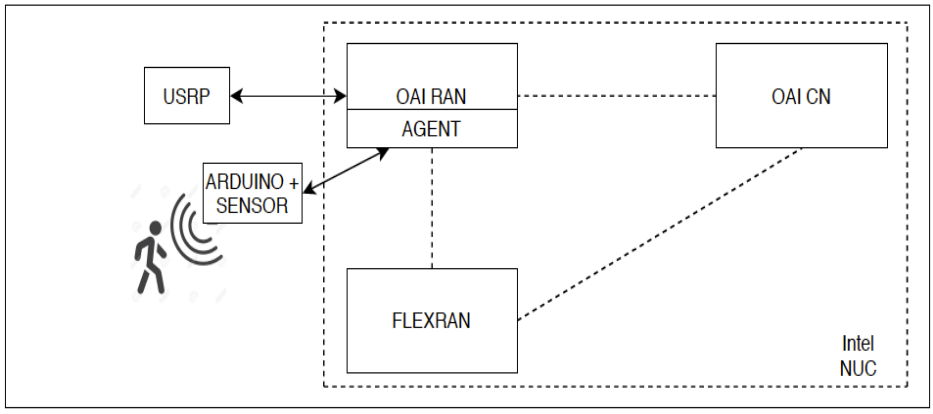

Figure 6.9: *Testbed network architecture*

The proposed D-SON algorithm is designed to minimize the energy consumption of SCs by utilizing a motion detection circuit. We utilize an Arduino Uno board and an HC-SR501 PIR motion sensor. The motion sensor is connected to and powered by the Arduino board, which draws its power from the Intel NUC computer via USB. The USRP is also powered by the computer via a USB 3.0 port, with a USB power meter connected between them to measure the energy consumption of the USRP. On the other hand, the Intel NUC computer draws its power via a power meter that is connected to the mains to measure the energy consumption of the whole system.

6.4.2 Algorithm design

In this section, we present our proposed D-SON algorithm to minimize the energy consumption of SCs by utilizing a motion detection circuit. The proposed algorithm has been designed using OAI and FlexRAN to switch off the RF frontend of the ng-eNB SDR without affecting the ng-eNB connection with the CN. The ng-eNB goes into sleep mode by leveraging on FlexRAN functionalities to switch off layers 1 and 2, and Radio Resource Control (RRC) sub-layer of the 3GPP protocol stack and free all resources. Subsequently, the RF front-end of the USRP is turned off. This implies that the ng-eNB always maintains connection with the CN.

While in sleep mode, the ng-eNB machine periodically scans its serial port every y seconds for updates from the motion detection circuit to wake up. The moment motion is detected, the Arduino board writes the binary value “1”, denoting “ON”, to the serial port of the ng-eNB machine for w seconds before writing binary “0” to reset the serial port and then waiting for the motion to be detected again for the process to repeat. The length of w is set such that the ng-eNB does not miss any event between successive probes of the serial port, that is $w > y$. Once the ng-eNB reads this ON value, it wakes up for x seconds and waits for users to connect to it. If no users connect within this time, the ng-eNB goes back to sleep and starts scanning the serial port again. However, if there is a user connected, the ng-eNB will not go to sleep and will periodically check every z seconds until no user is connected before going back to sleep. Fig. 6.10 gives the flowchart of the proposed algorithm.

Given that the goal of this paper is to reduce energy consumption of SC networks, it is important to minimize the energy consumption of the motion detection circuit so as not to undo the gains of this algorithm. Accordingly, we put the Arduino board in sleep mode and utilize its interrupt pins to power it up whenever motion is detected. Once motion is detected, the Arduino board is powered on and it writes to the serial port of the ng-eNB machine and goes back to sleep. This results in about 45% reduction in power consumption of the motion detection circuit compared to when the Arduino board is always on.

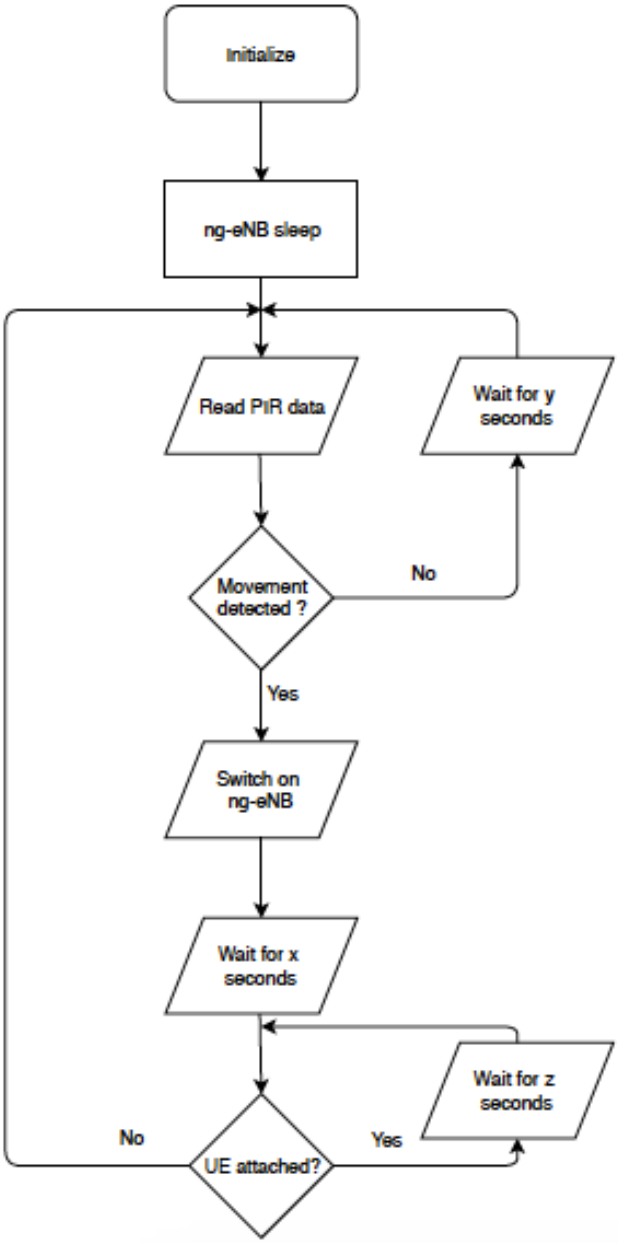

Figure 6.10: *Proposed motion sensor-based sleep scheduling flowchart*

6.4.3 Power consumption model

According to [29], the power consumption of a base station with a single transceiver chain can be modeled as:

$$P_{in} = P_0 + \Delta p\, P_{out};\ 0 < P_{out} \leq P_{max} \quad (1)$$

where P_0 and P_{out} denote the power consumption of the base station at the minimum non-zero output power and the RF output power radiated at the antenna elements, respectively. The parameters p and Pmax represent the slope of the load dependent power consumption and the average transmit power of the base station, respectively. It can be deduced from (1) that the power consumption model consists of a fixed part (P_0) and a load-dependent part ($\Delta p\ P_{out}$). With regards to the SDR platform, the B205mini USRP consists of an FPGA and RF front end. The RF front-end comprises of one transceiver chain that is made up of a power amplifier and an RF small-signal transceiver module. Focusing on active and sleep modes, the USRP has been measured to draw roughly 2 W when the ng-eNB is in sleep mode, while it consumes about 2.5 W when in active mode. Hence, we assume that the FPGA module and the RF front-end denote the fixed and variable parts of the USRP power consumption, respectively. Accordingly, in this paper, P0 comprises of the power consumptions of the USRP FPGA and the intel NUC minicomputer that relates to the CN, FlexRAN and maintaining a connection between the ng-eNB and the CN, whereas the RF front-end, motion sensing circuit and the power requirement for running ng-eNB layers 1-3 on the Intel NUC minicomputer make up the variable part. In the remainder of this dissertation, we will refer to the Intel NUC minicomputer as the ng-eNB machine, unless stated otherwise.

6.4.4 Performance evaluation

In this section, we present the performance results of our motion sensor-based SC sleep scheduling algorithm against the no sleep scheduling approach, whereby the SC does not employ any form of energy saving and is always active. The experimental setup was deployed in a 16 m^2 lab that hosts the 5GSON testbed at the University of Glasgow and the ng-eNB was toggled between active and sleep modes based on movements in the lab. No user equipment was allowed to connect to the network during the experiments in order to mimic a deployment location that experiences low-to-medium user activity such as staircases, areas around toilets or even office rooms. In this paper, we assume w = y = 1 second, x = 60 seconds and z = 30 seconds.

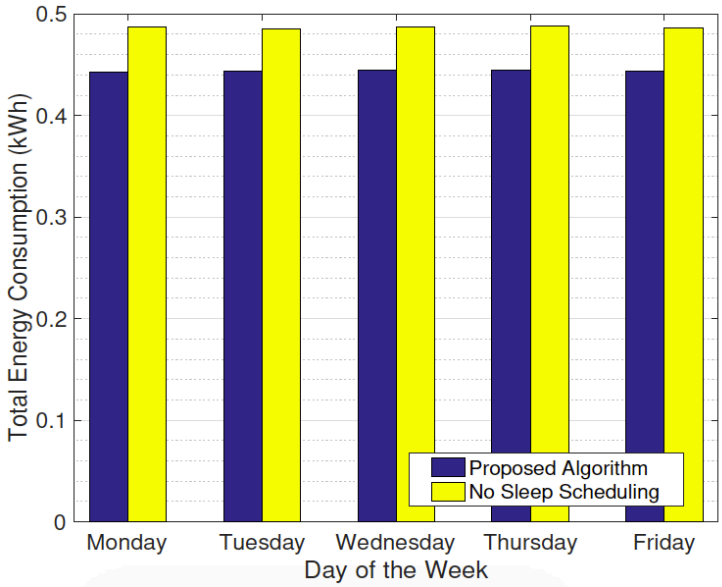

Figure 6.11: *Total energy consumption comparison of our proposed SC sleep scheduling algorithm versus the no sleep scheduling approach*

Fig. 6.11 shows the total energy consumption comparison of our SC sleep scheduling algorithm versus the benchmark no sleep scheduling approach. For both approaches, the energy consumption of the full setup, including the Intel NUC minicomputer, motion detection circuit and USRP was measured at 24 hour intervals over 5 days, from Monday to Friday. It can be seen that our approach achieves an average of 8% energy saving compared to the no sleep scheduling approach due to no or low user activity at certain times in the lab which results in the ng-eNB going into sleep. Note that this energy saving performance is highly dependent on the power rating of the computer used as a computer with high power consumption rating would overshadow the energy gains of using our proposed algorithm.

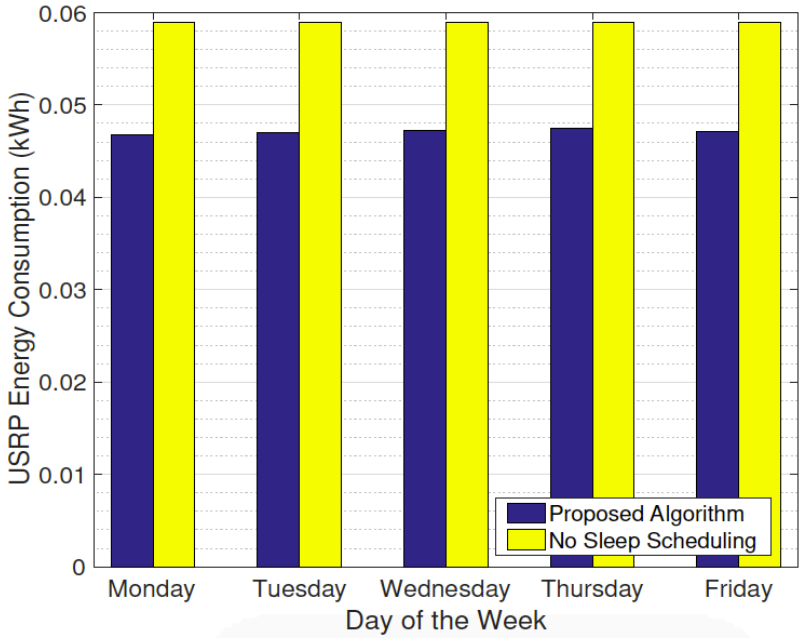

Figure 6.12: *USRP energy consumption comparison of our proposed SC sleep scheduling algorithm versus the no sleep scheduling approach*

Fig. 6.12 shows the USRP energy consumption comparison of our algorithm against the no sleep scheduling approach. It can be seen that the no sleep scheduling approach has a constant USRP energy consumption across the days of the week as it is always in the active mode irrespective of user presence. On the other hand, there is a slight variation in energy consumption of our proposed algorithm across the days of the week. This is because the lab sees different levels of user presence and activity across different days of the week and the effect of switching off the RF front-end is more pronounced when only the USRP is considered as it constitutes a fifth of the USRP power consumption. Accordingly, our algorithm achieves about 20% reduction in energy consumption when compared to the no sleep scheduling approach. It is worthy of note that the energy consumption of the USRP does not reach zero as the FPGA consumes about 80% of the power consumed by the USRP, even when the USRP is not transmitting.

CHAPTER
SEVEN

7 CONCLUSIONS

7.1 CONCLUDING REMARKS

We propose new functionalities, on top of a SDN/NFV architecture inspired by work in [17]. As opposed to the SoftAir proposal, we deploy a NGFI based architecture, in order to overcome CPRI issues related to massive MIMO deployment. Furthermore, in order to implement dynamically different functional splitting options, a virtualized RAU entity is proposed. On the other hand, in order to realize the proposed SDN enhancements, optimized controller is designed. The architectural controller is logically centralized but physically distributed, as a set of slave controllers and master controllers. Master controllers, located in remote sites and managing a group of macro cells, keeps in account long time scale and less fine grained parameters, acting as reference entities for slave controllers, located in edge sites, which as opposed to master controllers, keep in account short time scale and more fine grained parameters.

In particular, the proposed slave controller, focusing on different features, can be considered as different logical controller areas, corresponding to related northbound applications. These proposed functionalities could allow to fulfill strict 5G requirements, since elaborating and combining reports from network entities, will be able to perform optimum rules in terms of switching, resource allocation and virtual function instantiation/migration.

On the basis of the above assumptions, we designed an architectural solution for ultra-dense scenarios, in order to concretize, trough SDN/NFV features, the Phantom cell concept proposed by Docomo. Furthermore, as opposed to Docomo Advanced C-RAN, exploiting the flexibility of the proposed virtual base band processing in RAU, we are able to overcome IQ data transmission issues.

We designed a simulation testbed, in order to evaluate a first subset of the proposed functionalities. A programmable and virtualized RAN architecture has been implemented, in order to deploy SDN/NFV enhancements and to test radio resource management algorithms, in a real-time like environment.

In our simulations, we will firstly focus on the control and data plane splitting aspect of the proposed architectural design. Regarding the OpenFlow integration in the simulation scenario, we implemented a first software defined implementation of the proposed LTE RAN architecture trough ns-3 simulator. In particular, we configured two different ns-3 modules, known in literature as LENA ns-3 LTE Module [19] and the OFSwitch13 module [20]. The LENA ns-3 LTE Module permits to simulate a LTE RAN / LTE-EPC standard architecture. The OFSwitch13 module is an OpenFlow based module, deploying the OpenFlow 1.3 protocol in ns-3 simulation model. First results demonstrated, for our purpose, an efficient OpenFlow integration in our LTE simulation

environment. Evaluating the negative aspects of the ns-3 simulator based testbed, we have chosen to deploy the testbed based on software tools which permitted us to exploit an emulation environment in contrast to the previously deployed simulation environment.

At this aim, a flexible SDN/NFV-based SON testbed for future 5G mobile networks. The main contribute of our work is to cover the need for a CDSA based testbed, enabling the investigation of the potential of CDSA for practical implementations. We focus our proposal on two experimental setups, a real one and a virtualized one, both based on the FlexRAN and OpenAirInterface software tools. SDR, VM, FlexRAN controller and OpenAirInterface offer us the flexibility, programmability and high performance capabilities needed, making the testbed an ideal enabler for the development of various 5G use cases.

We implement a first case study, i.e., the RAN entities activation/deactivation procedures, performing time measurements, concerning the aforementioned procedures, in order to prove proper Testbed functioning. Furthermore, we validate the C-SON and D-SON capabilities of our testbed, considering the features of the obtained results.

As second case study, we have proposed and implemented on the real testbed, a low-cost, low-complexity SC sleep scheduling algorithm to minimize the energy consumption of SCs in 5G and beyond networks. Our algorithm is based on the DSON approach and it leverages on a motion detection circuit to instantaneously toggle the SC between sleep and active modes based the presence of a mobile user, without the need for complex traffic prediction algorithms. Experimental results show that our algorithm can achieve up to 20% USRP energy consumption saving when compared to the no sleep scheduling approach.

Both the testbed deployments enable novel research and provides teaching opportunities in next generation RAN architectures and several other areas of system-level research. Furthermore, they act as benchmark for many use cases with a significant proximity to real network deployment criticalities.

7.2 Work in progress

7.2.1 RAN Slicing Application

A RAN Slicing application capable of dynamically performing RAN slicing, based on a real-time evaluation of the RAN capabilities. The module operates in a learning fashion in a first training phase and in a self-adaptive fashion in a second routine functioning. In particular, we focus our proposal on a flexible instantiation of virtualized functionalities implementing the different layers of the protocol stack. The flexibility is related to the given traffic classification and to the architectural design. We are designing and implementing the application in order to prove its effectiveness on top of our testbed deploying the RAN entities by means of Docker container and/or virtual machines, triggering the dynamic instantiation through a script.

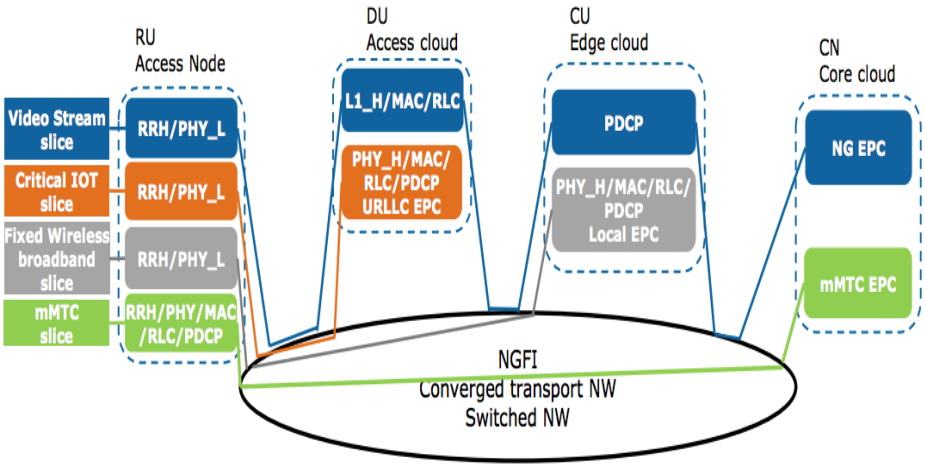

Figure 7.1: *End-to-end slicing*

The figure 7.1 shows a potential implementation of the end-to-end network slicing concept, through which logical networks, comprising different type of resources in terms of RAN, transport network and core network entities, can be created and associated to particular users or to serve specific type of service requests. A particular functional split is executed for each type of service, thus implementing the related protocol stack layers in the RU, DU and CU instances. Furthermore, the core network entities can be executed on top of DU or CU instances. Following these assumptions, we are deploying the aforementioned conceptual view on top of our testbed. Certain specific functional splits are executed through the related OpenAirInterface code, permitting the deployment of the related RU, DU and CU instances.

The Figure 7.2 focuses on the RAN, showing a potential RAN slicing executed on top of our testbed. We are realizing performance evaluations in terms of delay and throughput for each type of service request served through a particular functional split deployed in our RAN entities. The same performance evaluations are made for the core network and transport networks, permitting us to realize and to test an end-to-end slicing.

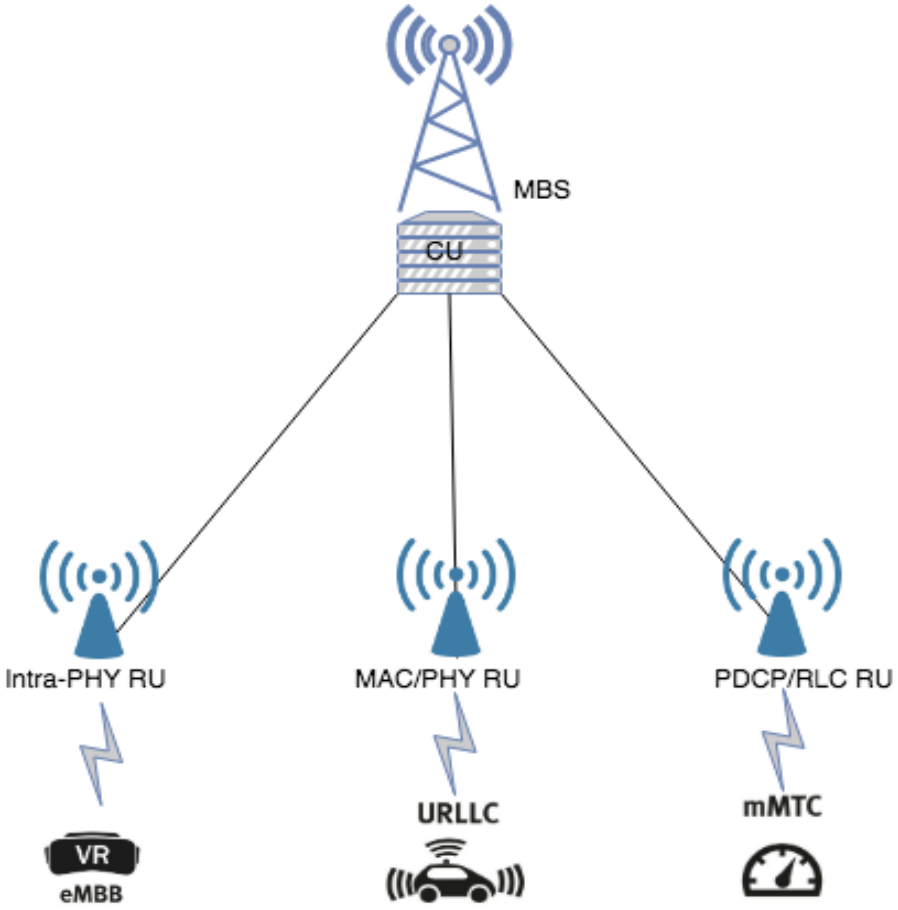

Figure 7.2: *RAN Slicing in our testbed*

7.2.2 Machine Learning based Switch ON/OFF

We are developing an improved and more efficient version of the algorithm exploiting Machine-learning techniques, in the context of the research collaboration with the University of Glasgow. We are exploiting network traffic pattern prediction to determine small cell switching on-off (URRP B205mini) based on datasets processed by machine learning. To this end, the EE algorithm code will be optimized and input from machine learning included. An initial edit at the code oriented to support machine learning in the algorithm, has been already implemented and consequent additional UK plug power meter measurement showed a significant improving on the OAI RAN machine energy saving.

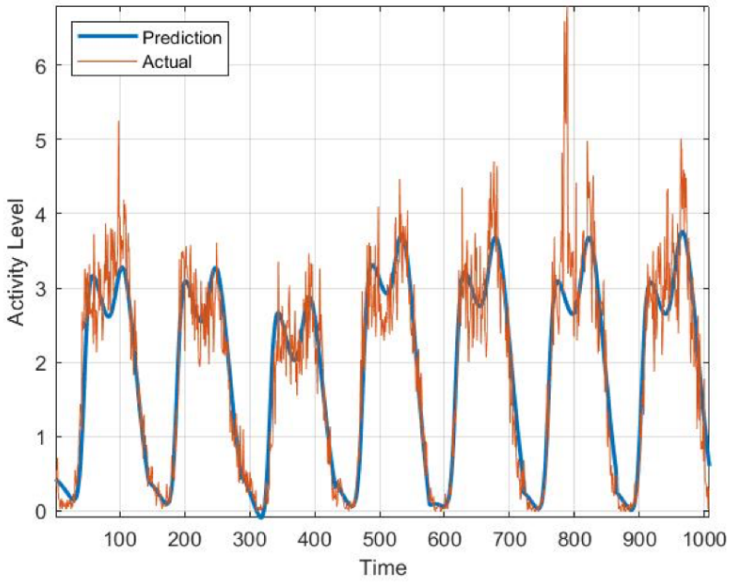

Figure 7.3: *Activity levels, predicted and actual values*

Energy consumption of the full setup, including the Intel NUC mini computer and USRP has been measured at 24 hour intervals over 7 days. The Figure 7.3 shows the matching between the effective activity levels of the considered days and the predictions realized with the machine learning application. These preliminary results show an appropriate matching, only few differences occur for each considered level. Additional work is in progress in order to improve the prediction performances. The objective is to control the activation and the deactivation through the predicted values in order to reduce the energy consumption of the full setup. In these terms, preliminary results, as can be seen in Figure 7.4, show that when eNB is always on (no EE enabled, blue curve), energy consumption is higher compared to the Energy consumption achieved while the improved version of Stop&Restart EE algorithm is running (orange curve).

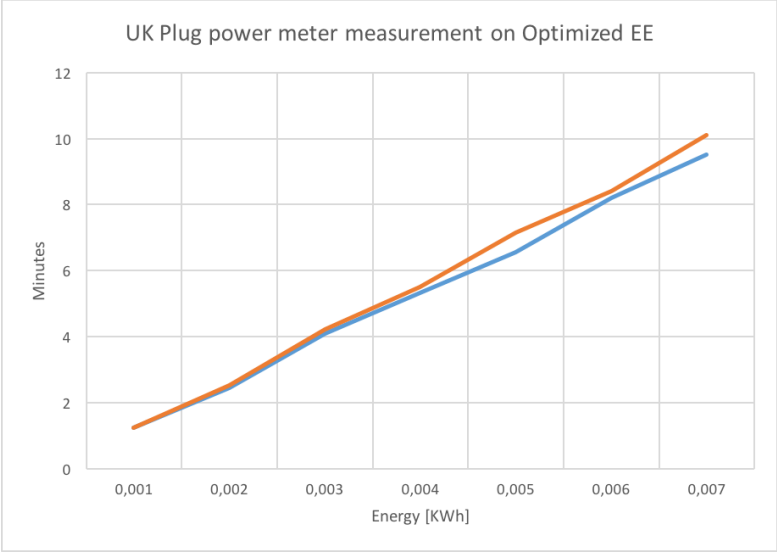

Figure 7.4: *Energy consumption comparison, no EE algorithm (blue curve), Stop&Restart EE algorithm (orange curve)*

7.2.3 Fronthaul/Core networks and OpenStack/OpenDaylight

The research activities during the PhD involve another research collaboration, with Bristol Is Open, a joint venture between the University of Bristol and Bristol City Council, with the aim of deploying a SDN/NFV based testbed to prove the application of this two paradigms in the context of fronthaul/core networks of future 5G networks, exploiting OpenDaylight as SDN controller.

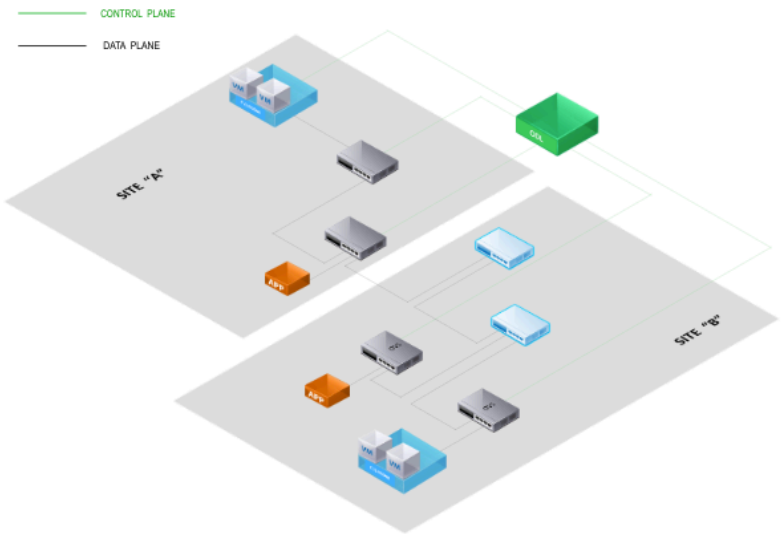

Figure 7.5: *Network topology, control plane/data plane spliting*

The research activity aims to focus on the network performances of fronthaul/core networks in which OpenFlow switches are deployed. A proper testbed has been designed in order to prove the feasibility of the proposal and to test the aforementioned network performances. We exploit the hardware and software resources on top of the Bristol Is Open research network. In particular, the testbed consists of an OpenStack deployment with different virtual machines. On top of certain virtual machines the OpenAirInterface RAN and Core network instances are executed, whereas on top of other virtual machines OpenvSwich instances are deployed. This implementation permits us to test the network performances in the context of different functional splitting use cases, both in fronthaul and core networks. Each use case is characterized by the deployment of a different number of OpenVSwitch instances in the considered links. These OpenVSwitches form the data plane, directly controlled by the OpenDayLight SDN Controller, forming the control plane.

The figure 7.5 shows a potential topology in which two different sites are deployed. On the basis of the considered fronthaul scenario, different entities of the OpenAirInterface RAN are executed on top of the related virtual machines. The considered virtual machines can be located on the same site or on a different site, permitting the performance evaluation both in local or remote scenarios. The same performance evaluations can be made in the case of the core network. The RAN and the core network deployments can be concurrently evaluated, so considering an end-to-end slicing in which the further aspect influencing the network performance is the presence, the number and the locations of OpenvSwich instances in the fronthaul or core network links.

Many features can be directly managed through the Administration GUI provided by the OpenDaylight SDN Controller software tool, as can be seen in the Figure 7.6. At the time of writing different performance evaluation are conducted on top of the described SDN/NFV testbed. The related results will be presented on dedicated papers.

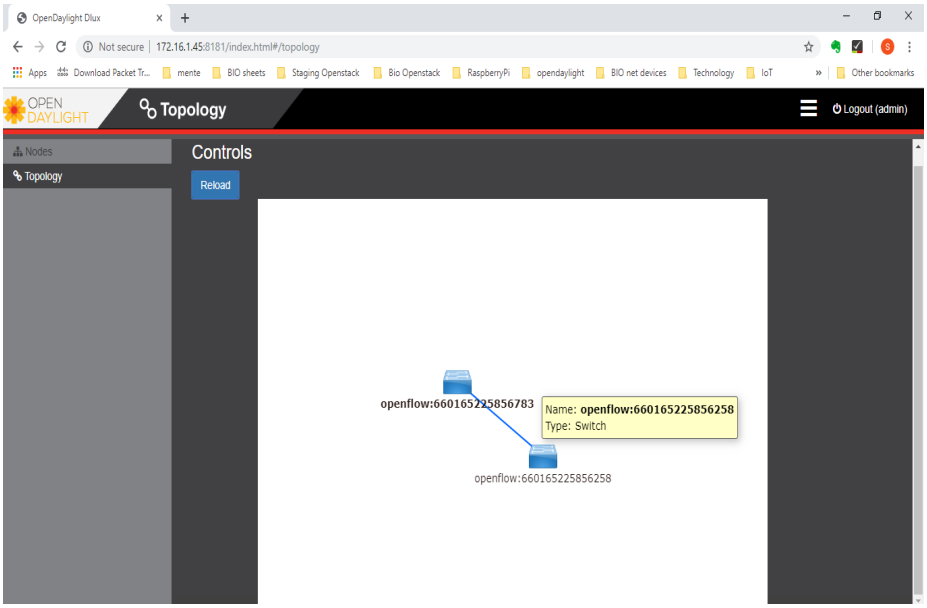

Figure 7.6: *OpenDaylight SDN Controller GUI*

7.2.4 IMS Virtualization

Finally, a research collaboration with the telco research team in Milan of Altran, engineering consulting company, is conducted focusing on the virtualization of the IMS network architecture in the context of the Voice over 5G. This research collaboration leads to the design and the deployment of a IMS testbed in order to couple it with our OpenAirInterface/FlexRAN based testbed. We are exploiting the Clearwater IMS software tool. Clearwater is an open source software implementation of the IMS mainly designed for Cloud environment, capable of providing voice, video and messaging services to users.

The figure 7.7 shows the Clearwater architecture and the software-based IMS entities.

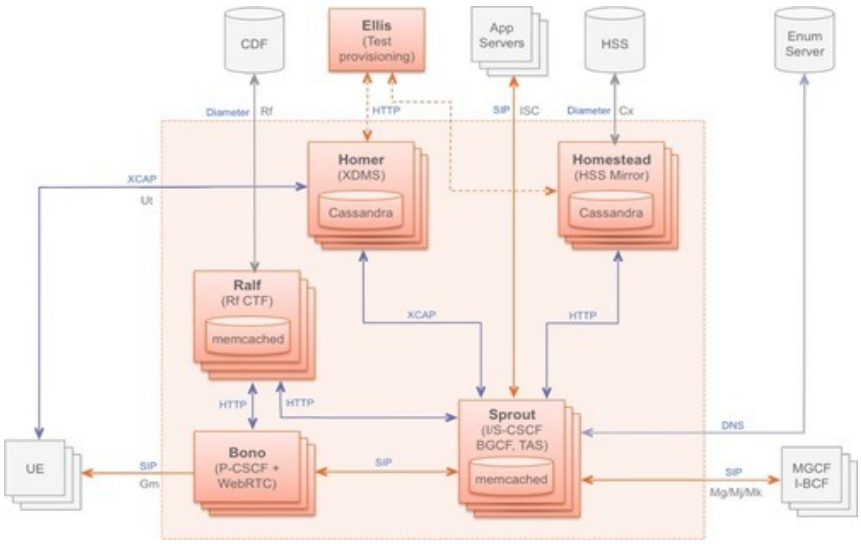

Figure 7.7: *Clearwater IMS software-based architecture*

The Figure 7.8 shows a preliminary deployment based on virtual machine instances. The OpenAirInterface RAN and core network entities, the FlexRAN controller entity and the Clearwater IMS entity are deployed on top of dedicated virtual machines instances.

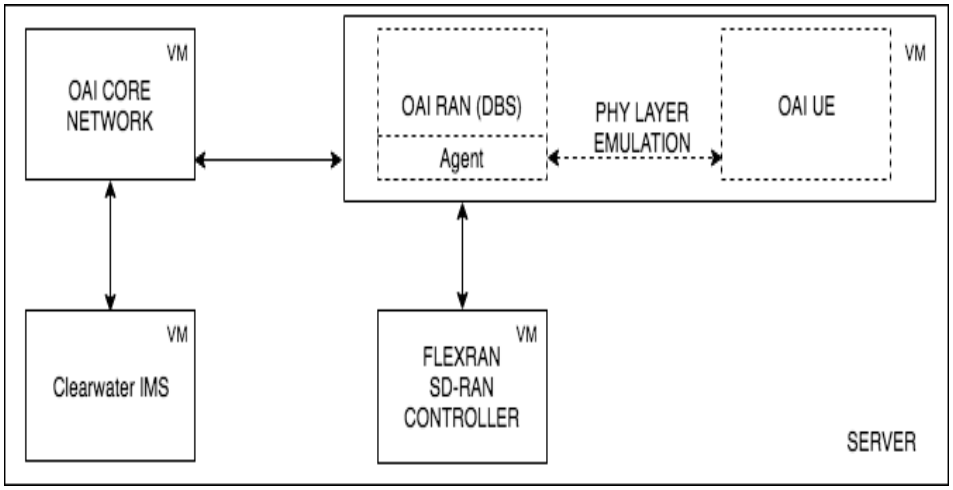

Figure 7.8: *Clearwater IMS coupled with our testbed.*

7.3 List of publications

[1] G. C. Valastro, D. Panno, and S. Riolo, “A SDN/NFV based C-RAN architecture for 5G Mobile Networks,” in 2018 International Conference on Selected Topics in Mobile and Wireless Networking (MoWNeT), June 2018, pp. 1–8.

[2] Y. A. Sambo, G. C. Valastro, G. M. M. Patanè, M. Ozturk, S. Hussain, M. A. Imran and D. Panno “Motion Sensor-based Small Cell Sleep Scheduling for 5G Networks,” in 2019 IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD 2019), September 2019.

[3] G. M. M. Patanè, G. C. Valastro, Y. A. Sambo, M. Ozturk, S. Hussain, D. Panno and M. A. Imran, “Flexible SDN/NFV-based SON testbed for 5G mobile networks,” in 2019 The 23rd International Symposium on Distributed Simulation and Real Time Applications (IEEE/ACM DS-RT 2019), October 2019.

[4] G. C. Valastro, Y. A. Sambo, G. M. M. Patanè, M. Ozturk, S. Hussain, D. Panno and M. A. Imran, “Machine Learning based Predictive Cell Switching,” in IEEE Communication Letters Journal, IEEE, (Under Submission).

[5] G. C. Valastro, Y. A. Sambo, G. M. M. Patanè, M. Ozturk, S. Hussain, D. Panno and M. A. Imran, “SDN/NFV based RAN Slicing Application for 5G mobile networks,” (Under Submission).

[6] E. Catania, A. La Corte, D. Panno, G. C. Valastro, “IoT Privacy in Ultra-Dense Networks,” (Under Submission).

BIBLIOGRAPHY

[1] 5G PPP, “View on 5G Architecture,” 5G PPP Architecture Working Group, Version 1.0, July 2016

[2] 5G PPP, “View on 5G Architecture,” 5G PPP Architecture Working Group, Version 2.0, December 2017

[3] 5G PPP, “View on 5G Architecture,” 5G PPP Architecture Working Group, Version 3.0, June 2019

[4] A. Detti, “Functional Architecture”, 5G Italy White Book: From Research to Market, CNIT

[5] M. Ding, D. Lopez-Perez, G. Mao, P. Wang, and Z. Lin, “Will the area spectral efficiency monotonically grow as small cells go dense?” in 2015 IEEE Global Communications Conference (GLOBECOM), Dec 2015, pp. 1–7.

[6] 3GPP, “Study on Small Cell enhancements for E-UTRA and E-UTRAN; Higher layer aspects,” 3rd Generation Partnership Project (3GPP), Technical Report (TR) 36.842, 01 2014, version 12.0.0.

[7] H. Ishii, Y. Kishiyama, and H. Takahashi, “A novel architecture for LTE-B: C-plane / U-plane split and Phantom Cell concept,” in 2012 IEEE Globecom Workshops, Dec 2012, pp. 624–630.

[8] C. M. R. Institute, “C-RAN - The Road Towards Green RAN,” China Mobile Research Institute, Whitepaper, 12 2013, version 3.0.

[9] I. F. Akyildiz, S. Nie, S.-C. Lin, and M. Chandrasekaran, “5G roadmap: 10 key enabling technologies,” Computer Networks, vol. 106, pp. 17 – 48, 2016.

[10] A. Checko, H. L. Christiansen, Y. Yan, L. Scolari, G. Kardaras, M. S. Berger, and L. Dittmann, “Cloud ran for mobile networks - a technology overview,” IEEE Communications Surveys Tutorials, vol. 17, no. 1, pp. 405–426, Firstquarter 2015.

[11] SCF, “Small cell virtualization functional splits and use cases,” Small Cell Forum (SCF), Technical Report (TR), 06 2015, version SCF159.05.1.01.

[12] NGMN, “Further study on critical C-RAN technologies,” next generation mobile networks (NGMN), Technical Report (TR), 03 2015, version 1.0.

[13] 3GPP, “Study on new radio access technology: Radio access architecture and interfaces,” 3rd Generation Partnership Project (3GPP), Technical Report (TR) 38.801, 04 2017, version 14.0.0.

[14] A. de la Oliva, J. A. Hernandez, D. Larrabeiti, and A. Azcorra, “An overview of the cpri specification and its application to c-ran-based lte scenarios,” IEEE Communications Magazine, vol. 54, no. 2, pp. 152–159, February 2016.

[15] I. F. Akyildiz, S.-C. Lin, and P. Wang, “Wireless software-defined networks (W-SDNs) and network function virtualization (NFV) for 5G cellular systems: An overview and qualitative evaluation,” Computer Networks, vol. 93, pp. 66 – 79, 2015

[16] Docomo, “DOCOMO 5G White Paper 5G Radio Access: Requirements, Concept and Technologies,” Docomo, Whitepaper, 2014.

[17] I. F. Akyildiz, P. Wang, and S.-C. Lin, “Softair: A software defined networking architecture for 5g wireless systems,” Computer Networks, vol. 85, pp. 1 – 18, 2015.

[18] C. M. R. Institute, “White Paper of Next Generation Fronthaul Interface,” China Mobile Research Institute, Whitepaper, 2014.

[19] N. Baldo, M. Miozzo, M. Requena-Esteso, and J. Nin-Guerrero, “An open source product-oriented lte network simulator based on ns-3,” in Proceedings of the 14th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, ser. MSWiM ’11. New York, NY, USA: ACM, 2011, pp. 293–298.

[20] L. J. Chaves, I. C. Garcia, and E. R. M. Madeira, “Ofswitch13: Enhancing ns-3 with openflow 1.3 support,” in Proceedings of the Workshop on Ns-3, ser. WNS3 ’16. New York, NY, USA: ACM, 2016, pp. 33–40.

[21] N. Nikaein, C.-Y. Chang, and K. Alexandris, “Mosaic5G: Agile and flexible service platforms for 5G research,” ACM SIGCOMM Comp. Com. Rev., vol. 47, no. 3, Jul. 2018.

[22] N. Nikaein, R. Knopp, F. Kaltenberger, L. Gauthier, C. Bonnet, D. Nussbaum, and R. Ghaddab, OpenAirInterface 4G: an open LTE network in a PC, International Conference on Mobile Computing and Networking, 2014.

[23] X. Foukas et al., “FlexRAN: A flexible and programmable platform for software-defined radio access networks,” in Proc. of the 12th International on Conference on Emerging Networking Experiments and Technologies (CoNEXT ’16), Dec. 2016, pp. 427–441.

[24] A. Mohamed, O. Onireti, M. A. Imran, A. Imran, and R. Tafazolli, Control-data separation architecture for cellular radio access networks: A survey and outlook, IEEE Communications Surveys Tutorials, vol. 18, no. 1, pp. 446/465, Firstquarter 2016.

[25] Huawei, Next Generation SON for 5G, Huawei Technologies Co., Tech. Rep., 2016.

[26] SCF, Small cells market status report, Small Cell Forum, Release 10.0 SCF050.10.03, Dec. 2018.

[27] C.-Y. Chang, L. Kuacz, R. Schmidt, A. Kliks, N. Nikaein, “Spectrum Management Application - A Tool for Flexible and Efficient Resource Utilization”, GLOBECOM 2018, IEEE Global Communications Conference, December 2018.

[28] 3GPP, “5G; Study on scenarios and requirements for next generation access technologies,” 3rd Generation Partnership Project (3GPP), TS 38.913, Sept. 2018.

[29] G. Auer, V. Giannini, C. Desset, I. Godor, P. Skillermark, M. Olsson, M. A. Imran, D. Sabella, M. J. Gonzalez, O. Blume, and A. Fehske, “How much energy is needed to run a wireless network?” IEEE Wireless Communications, vol. 18, no. 5, pp. 40–49, October 2011.

[30] O. G. Aliu, A. Imran, M. A. Imran, and B. Evans, “A survey of self organisation in future cellular networks,” IEEE Communications Surveys Tutorials, vol. 15, no. 1, pp. 336–361, First 2013.

[31] S. Sharma, S. J. Darak, and A. Srivastava, “Energy saving in heterogeneous cellular network via transfer reinforcement learning based policy,” in 2017 9th International Conference on Communication Systems and Networks (COMSNETS), Jan 2017, pp. 397–398.

[32] M. Feng, S. Mao, and T. Jiang, “Boost: Base station on-off switching strategy for green massive mimo hetnets,” IEEE Transactions on Wireless Communications, vol. 16, no. 11, pp. 7319–7332, Nov 2017.

[33] R. Tao, W. Liu, X. Chu, and J. Zhang, “An energy saving small cell

sleeping mechanism with cell range expansion in heterogeneous networks," IEEE Transactions on Wireless Communications, vol. 18, no. 5, pp. 2451–2463, May 2019.

[34] W. Ur Rehman, A. Hussain, and M. M. Butt, "Joint user association and bs switching scheme for green heterogeneous cellular network," in 2018 IEEE Globecom Workshops (GC Wkshps), Dec 2018, pp. 1–6.

[35] A. Dudnikova, P. Dini, L. Giupponi, and D. Panno, "Multi-criteria decision for small cell switch off in ultra-dense lte networks," in 2015 13th International Conference on Telecommunications (ConTEL), July 2015, pp. 1–8.