Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

MASTER THESIS

TITLE: Development of an Orchestrator layer for a Wireless Cloud

MASTER DEGREE: Master in Science in Telecommunication Engineering & Management

AUTHOR: Martí Floriach Pigem

DIRECTOR: Antoni Gelonch Bosh

DATE: July, 22rd 2016

Títol: Development of an Orchestrator layer for a Wireless Cloud

Autor: Martí Floriach

Director: Antoni Gelonch Bosh

Data: 22 de juliol del 2016

Resum

La gran popularitat que han assolit els dispositius mòbils en els últims anys està fent que les infraestructures de telefonia mòbil actuals s'hagin quedat obsoletes. El Cloud-RAN es presenta com un candidat ideal per fer front a les limitacions del les infraestructures actuals fent ús d'una administració centralitzada. Tanmateix la seva implementació presenta un repte d'hagut als estrictes requeriments de temps real dels estàndards wireless i la implementació d'algoritmes d'administració eficients. Per aquesta raó en aquest projecte ens hem centrat en la implementació d'una infraestructura de cloud RAN per tal de troba solucions als problemes esmentats. Concretament ens em focalitzar en l'estudi d'algoritmes que permetin fer un millor ús dels recursos wireless. Posteriorment s'ha fet un estudi de la viabilitat de la infraestructura comparant-la amb les infraestructures actuals.

Els resultats obtinguts indican que el Cloud RAN permet fer un molt millor ús dels recursos wireless gràcies a la seva flexibilitat i al fet que és molt reconfigurable, a més a més la seva implementació redueix considerablement els costos de CAPEX/OPEX respecte les infraestructures actuals.

Title: Development of an Orchestrator layer for a Wireless Cloud

Author: Martí Floriach Pigem

Director: Antoni Gelonch Bosch

Date: July 22 rd 2016

Overview

In the last years mobile devices have gained much popularity, that's why the current mobile infrastructure is becoming obsolete. Cloud-RAN arises as an excellent candidat to overcome the strong limitations of the existing infrastructure as it uses a centralized administration of the resources. However, the implementation of such infrastructure presents a significant challenge due to the strict real time requirements of the wireless standards and the implementation of efficient administration algorithms. In order to provide solutions to the problems presented above, in this project we have focused on the implementation of a Cloud-RAN infrastructure. Specifically, we have focused on the study of algorithms that allows a better use of the wireless resources. Later, we have carried out a study to analyse the feasibility of the proposed infrastructure and we have compared it with the current infrastructures.

Results obtained show that Cloud-RAN allows a better management of the wireless resources thanks to its flexibility and the fact that it is totally reconfigurable. Furthermore, its implementation reduces the CAPEX/OPEX costs significantly with respect to the current infrastructure.

INDEX

ACRONYMS

API - Application Program Interface

AWS - Amazon Web Service

BBU - BaseBand Unit

BSP - Base Station Pool

BT - Base Station

CAPEX - Capital Expenditure

CoMP - Coordinated multi-point operation

CPU - Central Processing Unit

DSP - Digital Signal Processor

ERM - Entity Relationship Model

FFT - Fast Fourier Transform

FPGA - Field-Programmable Gate Array

GPP - General Purpose Processors

GPU - Graphics Processing Unit

GUI – Graphical User Interface

HVM - Hardware Virtual Machine

IaaS - Infrastructure as a Service

InP - Infrastructure Provider

LAA - License Assisted Access

LTE - Long Term Evolution

MCS - Modulation and Code Scheme

MIMO - Multiple Input Multiple Output

MPLS - Multiprotocol Label Switching

MVC - Model View Controller

NASA - National Aeronautics and Space Administration

NS - Network Service

NVF - Network Virtual Function

OOCRAN - Open Orchestrator Cloud Radio Access Network

OPEX - Operating Expenditure

PaaS - Platform as a Service

PC - Personal Computer

PCIe - Peripheral Component Interconnect Express

RRH - Remote Radio Head

SaaS - Software as a Service

SDR - Software Defined Radio

SDN - Software Defined Network

SP - Server Provider

TSP - Telecommunication Service Provider

UE - User Equipment

USRP - Universal Software Radio Peripheral

VIM - Virtualized Infrastructure Manager

VNF - Virtual Network Function

WCDMA - Wideband Code Division Multiple Access

List of Tables

List of Figures

INTRODUCTION

The continuing growth in demand from subscribers for better mobile broadband experiences are doing that the traditional mobile network architectures are in facing a serious situation. The traffic will be double every year, which will require more cost to build, operate and upgrade the current network infrastructure, while the revenue will be each year smaller. Meanwhile traditional networks are wasting several resources, due to the changes in a time-geometry pattern called “Tidal Effect” .

In addition, mobile operators are forced to maintain various wireless communication standards under the same infrastructure, increasing the capital expenditure (CAPEX) and operating expense (OPEX). Therefore it is necessary to find new alternatives to solve this problem, with the following features:

- Support of multiple air interface standards and flexible software upgrade. [1]
- Provision of reliable services with reduced cost, while maintaining healthy revenue. [1]
- Optimization among capacity, mobility, and coverage in broadband cellular wireless systems. [1]

Software Defined Radio

Software Defined Radio (SDR) refers to the technology wherein software modules running on a generic hardware platform consisting on DSPs, FPGA and general purpose microprocessors (GPPs) are used to implement radio functions such as mixers, filters, amplifiers, etc.
SDR can be used to implement military, commercial and civilian radio applications such as WCDMA, LTE, etc [2]

Current radio frequency (RF) frontends can convert radio frequency signals into a digital domain with a few steps, allowing make the signal processing into a digital domain using common language programming as C.

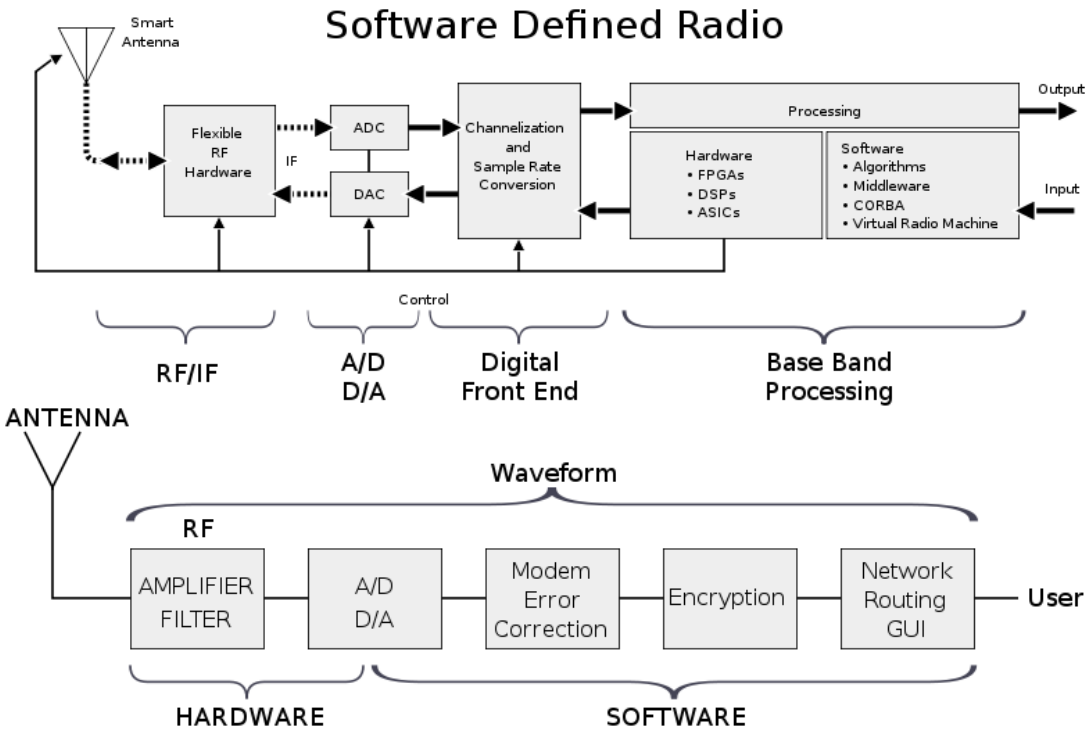

Fig. I.1 SDR implementation example

The implementation of radio functions in software take several advantages:

- Reconfigurability: allows to co-existence of multiple software modules implementing different standards on the same system and reconfigure the radio system selecting the appropriate software module [3].
- Ubiquitous Connectivity: the implementation in modules and multiples instances of such modules can co-exist in the infrastructure [3].
- Adaptive Radio: communication system have a means of monitoring their own performance and modifying their operating parameters to improve the performance of the system [3].
- Cognitive Radio: communication system are aware of their internal state(environment, localization, spectrum usage, etc ...) and take advantage of this modifying his own radio operating behavior. [3]

Cloud Computing

Cloud computing is a kind of Internet-based computing that provides shared processing resources and data on demand.
In this new paradigm, companies can avoid the upfront infrastructure costs and focus on the projects. At the same time, cloud infrastructures provide an improvement of the manageability with less maintenance, faster adjusting of the resources according to the demand, normally called scaling. With this model

companies pay for the resources that they are using, making the deployment and maintenance cheaper.

In cloud computing world we found two main characters:

- Infrastructure provider: It is the owner of the infrastructure and the one who rents resources.
- Tenants: They are the companies that are paying for the resources that they are using.

Depending on the grants that the infrastructure provider gives to the tenants, we can find three main types of cloud:

- Software as a Service (SaaS): In this model users have access to the applications and database. Infrastructure provider manages the infrastructure that run the applications. This model allows to avoid install ,run and maintain applications in his own infrastructure.
- Platform as a Service (PaaS): This model offers a developer environment to application developers. It is normally compost by an operation system with a toolkit to develop and test applications in a safety and controlled environment.
- Infrastructure as a Service (IaaS): This model gives to the tenants the maximum grants, allowing to design specific infrastructure in the cloud. This infrastructures are composed by computers interconnected with routers creating networks where each tenant has full control of all the components.

The following image summarize the three main models:

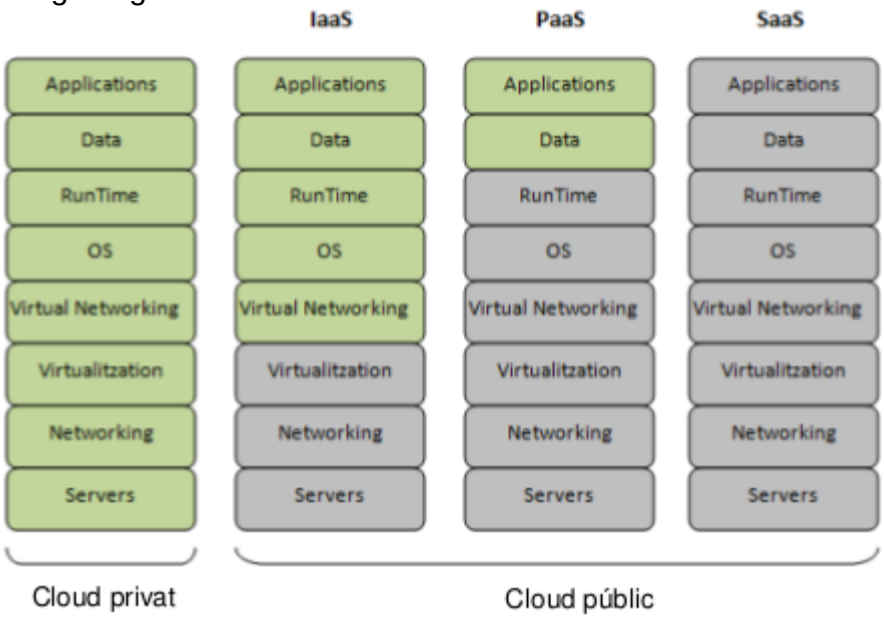

Fig. I.2 Cloud Computing types

In order to allow to share resources, it is necessary to ensure isolation between tenants. The most famous technique that achieve this purpose is the virtualization, that's why cloud computing is close associated to it.

Hypervisor and Virtual Machines

Computers run applications into the CPU but just can run one application in a specific time. For that reason, CPUs run applications according to an specific scheduler which decide what applications must run first. According to differences proposes, we find different kinds of schedulers.
Virtual machines at low level done something similar, each VM reserved a specific processing time of the physical CPU. Another VMs can not run its own application into that computation time. This allow that another VM can not degenerate the performance of another VM. The element that reserved computation time is the hypervisor, which reserve time and create virtual machines with a specific features [4].

MANO

Telecommunications Infrastructure Providers have shown interest in cloud features, therefore they desire to migrate their infrastructure in a virtual environment. In 2013's Mobile World Congress several institutions agreed to standardize the administration of these infrastructures in the cloud. The architecture is already defined but many challenges must be solved before its kickstart in 2016 and subsequent implementation in 2018.
The structure that ETSI designed is composed by three main blocks, where each one has a particular mission.

- Orchestrator: Responsible for onboarding of new services and virtual network function packages. This element manage the overall network.
- VNF Manager: Oversees life cycle management of VNF instances, coordinating and adapting his roles on the NFVI. This element manage each VNF instance according to the specifications provided by the orchestrator.
- Virtualized Infrastructure Manager (VIM): Controls and manages the NFVI compute, storage and network resources.

In the following graph we show the diagram block of the ETSI MANO architecture:

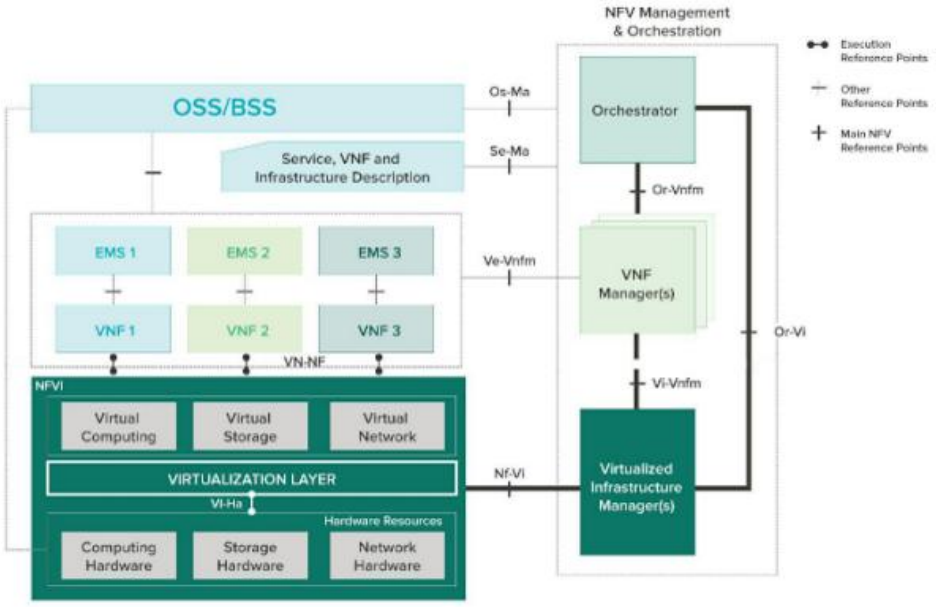

Fig. I.3 ETSI MANO architecture

In the following points we will describe with more detail each element.

Virtual Network Function and Network Function Virtualization

Virtual Network Function (VNF) is virtual machines that mimic the function of a specific hardware in the telecommunications field. And the Network Function Virtualization is the configuration of the VNF for a specific purpose. VNF is an object and the NVF is an instance of that object.

VNF Manager

This element manage the VNFs:

- Manage lifecycle of VNFs (deciding when create, maintains and terminates VNFs).
- It is responsible of handle: faults, configuration, accounting, performance and security of VNFs.
- It decide when scale up/down computation resources of the VNFs.

VIM

VIM manage NFVI resources in a domain, managing computing resources and network resources. Between its tasks we find:

- Manage the lifecycle of virtual resources (create, maintains and tear down VM from the physical resources).
- Keep inventory of VM associated with physical resources.
- Monitoring, performance and fault management of hardware and software.

Expose virtual and physical resources to another management systems through APIs.

Cloud Radio Access Network (C-RAN)

The union of the previous topics (SDR and Cloud Computing) allow to create network infrastructures with better performance than traditional ones, improving the spectrum efficiency exploiting the spectrum holes, deploy more cells with small size and take advantage of the frequency reuse and make the infrastructure flexible, green and more cheaper. This kind of infrastructures are called Cloud Radio Access Network (C-RAN).

Heterogeneous network supported by Cloud Radio Access Network (C-RAN) is expected to be the candidate for the new generation of access network techniques [5].This new set of techniques allow to decouple the baseband processing from the radio units and pooled in a centralized localization. This new network structure is composed by:

- Base Station Pool (BSP): a centralized pool of computing resources whose provide the signal processing and the orchestration of the overall network required by the cells of the area [6].
- Optical Fronthaul: the digitized samples will travel from the BSP to the next element ready to be sended to the user [6].
- Remote Radio Heads (RRH): the radio frequency signals from geographically distributed antennas are end/receive digitalized signals to/from BBU pool via optical fiber, and antennas are equipped with RRHs to transmit/receive radio frequency signals (RF) [6].

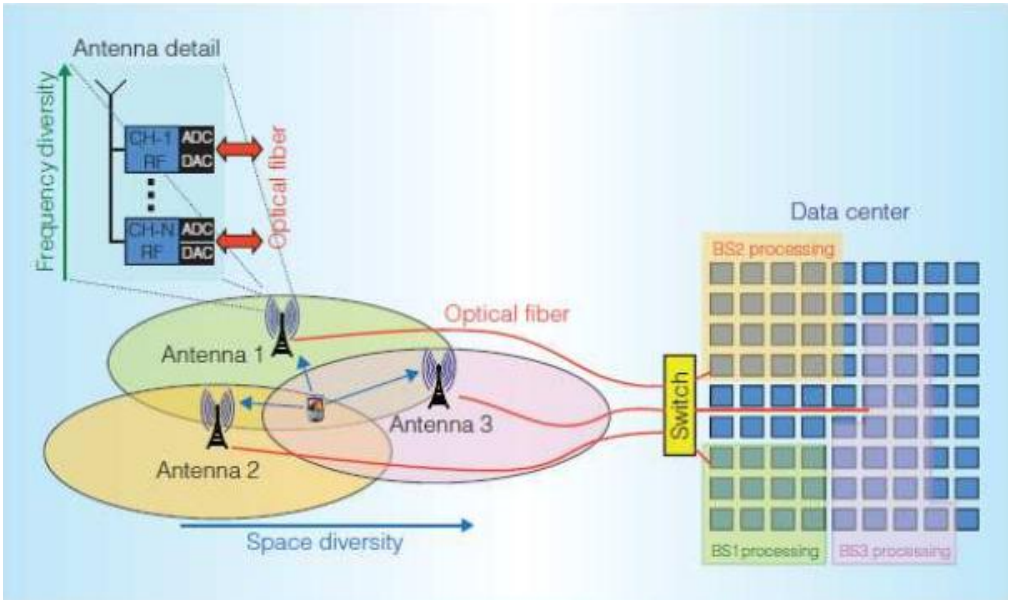

Fig. I.4 Cloud Radio Access Network architecture

With the resource of the BSP, C-RAN create BaseBand Units (BBU) whose act as a digital unit implementation of the base station traditional functions. This BBU is implemented using SDR, executed in a virtual environment (NVF), orchestrated for a cloud platform and running under a GPP with multiples acceleration tools as multicore, or multi threading, etc.

Common GPP (CPU and GPU) has a good processing capability with affordable price but some modules of the processing chain need more power as: Turbo decoder, FFT and MIMO decoders. For that modules hardware accelerators (DSP and FPGA) are required, interconnecting the board through the PCIe interface. By the other, hardware accelerator will be increase the price of the infrastructure.

CHAPTER 1. State of Art

This chapter introduces the state of the art of the previous topics, and the tools that we used throughout this thesis providing context. In addition, we will introduce the architecture and nomenclature of the used tools.

1.1 Software Defined Radio

Since the born of the SDR in 1991, there are multiples solutions to the market. Until a few years ago, all this platforms were commercial and in a lot of cases designed for military proposes. But currently, we can find open source platforms in the market such as: GNU radio [7], OpenLTE[8], etc…

In the course of this project, we will work with Long Term Evolution (LTE) base stations. For that reason, we decided to work with an open source platform designed specifically for the emulation of the LTE base station called srsLTE. Besides, srsLTE provides APIs to connect with several radio frequency frontends that we are using.

1.1.1 SrsLTE

SrsLTE [9] is a free and open-source LTE library for SDR User Equipment (UE) and eNodeB developed by Software Radio Systems. The library is highly modular with minimum inter-module or external dependencies. It is entirely written in C and, if available in the system, uses the acceleration library VOLK distributed GNURadio [10].

The library currently supports the Ettus Universal Hardware Driver (UHD) [11] and the bladeRF driver .

1.2. Cloud Computing

Nowadays Cloud Computing is a trending topic and several infrastructures are deployed in virtualized environment. There already exists multiple platforms available on the market. But we are just interested on the open source platforms and IaaS solutions for deployment of the cloud. With these conditions, OpenStack is the most suitable due to his reputation on the market.

1.2.1. OpenStack Architecture

OpenStack is a famous cloud computing platform designed for RackSpace Hosting and NASA on 2010. It is deployed on python and it is a IaaS solution. Several companies are using this platform to deploy their own private cloud computing infrastructure, one of the most famous companies are AWS (Amazon web service). Users manage their own infrastructure through a web-based dashboard, command-line or via a RESTful API. [12]

1.2. MANO

The infrastructure proposed by ETSI [13] for the deployment of the VNF appeared a few years ago and it is currently under deployment. The main architecture of the standard and the functions of the elements are decided, but connections and procedures are in development and have continuity modifications.

Even all this problems, we can find platforms available on the market. Normally, these platforms are focused on wire communication and they use OpenStack as VIM. The most famous one is OPNFV [14] by Linux Foundation but we can found others like OpenMANO [15], Gohan [16].

After investigating these platforms, we consider them too be difficult to install and maintain and moreover the learning curve is too high. By the other hand, they are mainly focused on wire communications, therefore, many features are not adapted to wireless systems.

1.3. Cloud RAN

China Mobile was the first one to introduce the cloud-RAN concept in the market a few years before. Using Intel processors of the Xeon family and single optical fiber and wavelength multiplexing division, they deployed the first cloud-RAN solution making a full virtualization of the fronthaul providing a LTE signal. In the recent implementation, they introduce CoMP algorithm in the uplink. [5]

Even this fact, Cloud RAN solutions are in development and most of them are private or they just provide a few functions of the overall potential. The only open source solution that we found is Open Air Interface [17]. This platform seems more focused on introducing 5G features on the cloud than providing collaborative features to the existing ones. Commercial and open source solutions are all implemented using OpenStack as VIM.

CHAPTER 2. Objectives

With the background of the previous chapter, we thought that it could be interesting to investigate in this topic and go in depth about some concepts.

We are interested on designing a cloud-RAN web-based platform following the ETSI MANO standard but focusing on wireless communications running under a cluster of GPPs, in particular on CPUs. We are also interested on creating a flexible infrastructure to reduce the intracell interference, and hence, improve the performance of the network while making a better use of the resources. Once the platform is completed, the next step will be analyze and improve the radio spectrum usage and reuse between several operators.

The currently ETSI MANO platform is not adequate for our purposes, because they are mainly focused on wire communication. For that reason, we decided to deploy a Cloud-RAN from scratch using OpenStack as a VIM and following the architecture of the ETSI MANO standard.

The objectives of the platform are:

- Create a real LTE processing chain and test it in the virtual environment.
- Share computational and radio resources between several operators.
- Make an intelligence reuse of the radio spectrum resources taking into account the interference with the operators and cells.
- Deploy cognitive infrastructures that can follow the subscribers bit rate demand depending of the time-geographic pattern.
- Deploy partial virtual infrastructures (BTS) and full virtualized infrastructures (BTs, channels and subscribers).
- Design a system to monitor all the elements of the infrastructures and actuate under fault conditions.

The results provided by the platform will be useful to make a comparison between traditional infrastructures and cloud RAN in terms of radio resources usage. Besides, partial and full virtualized infrastructures will be combined in order to make the simulation much more realistic.

Finally, we will make an analysis about the CAPEX/OPEX costs of the cloud-RAN infrastructure deployment versus the traditional ones.

CHAPTER 3. Open Orchestrator Cloud Radio Access Network (OOCRAN)

Regarding the objectives, we propose to develop a framework with the appropriate functionalities to achieve them. The platform will be called Open Orchestrator Cloud Radio Access Network (OOCRAN), and it will be designed in python using the popular web framework called Django release 1.9 as a programming platform and OpenStack as a VIM. The project code is open source and is availiable on this repository [18].

3.1 Django

Django [19] is a web framework based on Model-View-Controller (MVC) software architecture. MVC provide a clean, organized and agile programming methodology. This architecture is widely extended and used [20]. Thanks to that fact it will be easy to make modification to the code from third parties.

The MVC software architecture allows to divide:

- Views: are the front-end which is a graphical user interface (GUI) designed using HTML, CSS and Jquery.
- Model: are a methodology for capture and manipulate data from a database. Django allows to work with multiples kinds of database.
- Controller: are the back-end of the platform. They take decision according to the GUI requests and the model data. The controller are full programmed using python.

Django allows to work with multiples controller providing a clean and organized program architecture.

3.2 OpenStack Architecture

OpenStack [12] is a IaaS cloud computing platform that allows to administrate virtual machines and computational resources. OpenStack is composed by several elements called services. Each service has a specific intent. All services are interconnected to improve the management of the overall system. The following picture shows the interconnection between services:

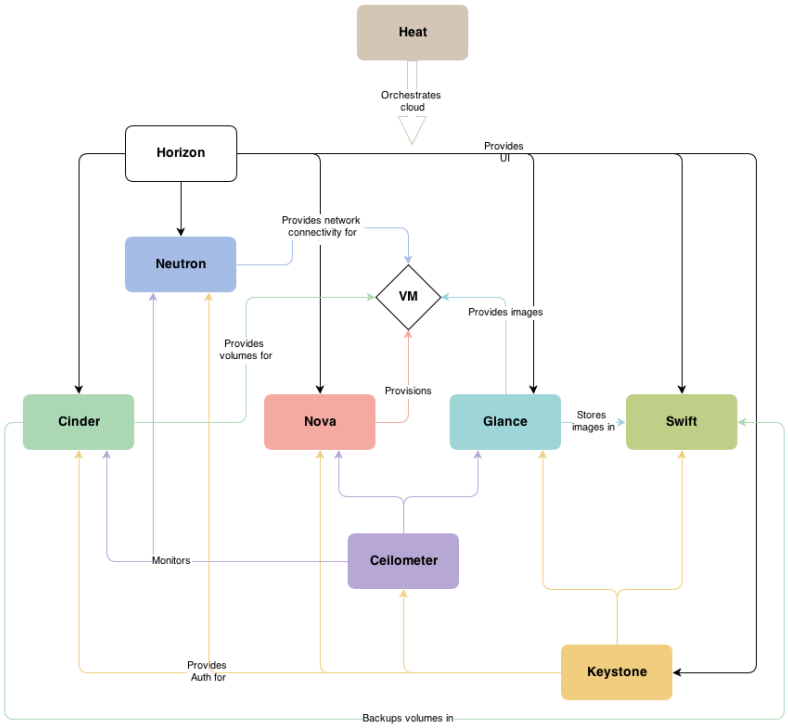

Fig. 3.1 OpenStack Architecture

We will explain the architecture from inside to outside. The core of the platform is the hypervisor, OpenStack works with several commercial hypervisor such as KVM, VMware, docker, etc. In order to configure the hypervisor skills for a particular VM, we found Nova service. Nova handles the VMs: manage the lifecycle, configure the parameters, etc. VMs run with a particular Operation System (OS) which is provided by Glance. This service is a repository of images that provides OS to the VMs.

Multiple VMs can be interconnected composing a network, in this network we find routers, switches, etc, all these elements are managed by Neutron service.

Networks and VMs are the elements that users pay for, therefore Keystone service manage the users permissions and allow or deny access to these elements. Their main function is as middleware.
Users need to manage their own infrastructures. Horizon provides a web server front end to handle users infrastructures.

Finally, we find Heat, this service is an infrastructure creator. Using a program language wrote in the template, users can deploy and save infrastructures.

There are more additional services in the OpenStack architecture, but we have just explained the most commons.
The main intents of each service is:

- Nova: manage the lifecycle of the VMs.
- Glance: provide images to run VMs.
- Neutron: manage the networks.

- Keystone: provides security and permissions in the VMs.
- Horizon: is a web front-end for to handle the infrastructure.
- Heat: manages the virtual infrastructures.

OpenStack works with a particular hypervisor, for this project we selected KVM because it is an open source project and provides good performance. However, it is necessary to introduce some optimization features in order to administrate the scheduler of the VM allocation on the CPUs.

3.3 OOCRAN Architecture

As we said in the chapter 2 (Objectives), we will follow the ETSI MANO standard but introducing a new element on the architecture. This element will be a pool of radio frequency resources (radio frequency spectrum), where various tenants will share it between them. The objective is to manage jointly and consistently all available resources to create the necessary infrastructure to support to a wireless operator.

Then the architecture of the OOCRAN will be:

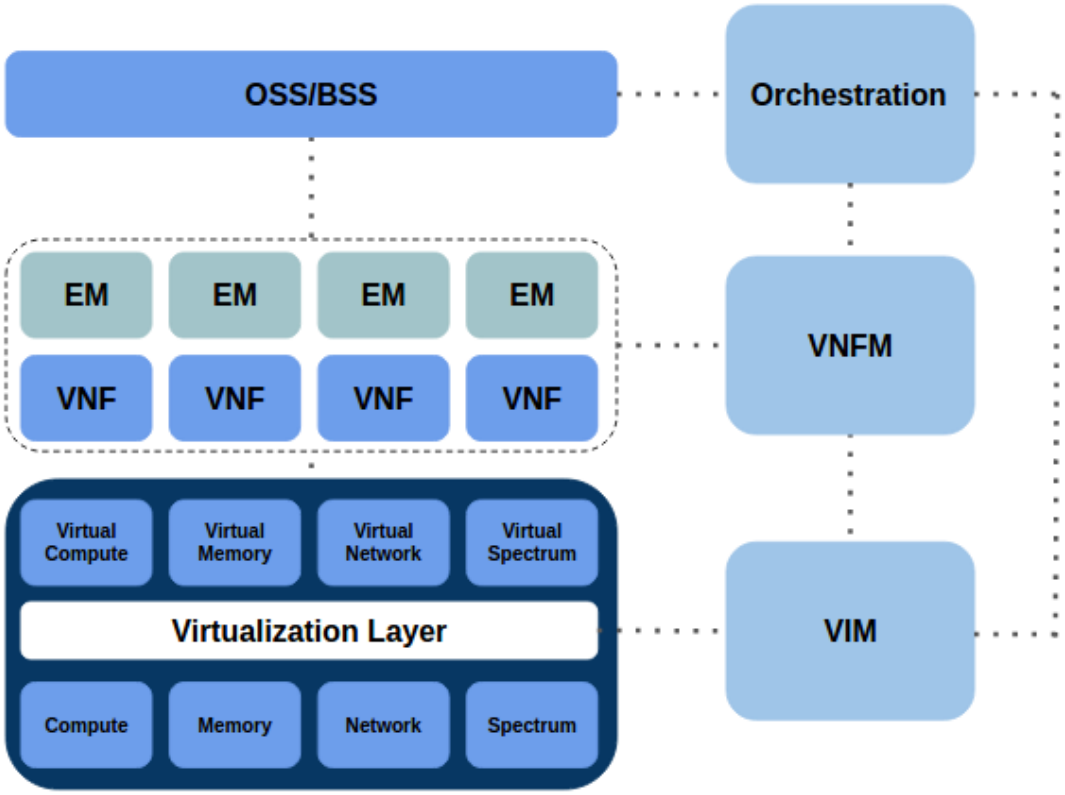

Fig. 3.2 OOCRAN Architecture

3.4 Tenants, roles and grants

With the implementation of ETSI MANO architecture also appears new business models with new players interacting between them. In the course of this thesis we will play one or multiples roles, thus it is time to define them and their associated privileges in the OOCRAN platform. These players can be defined as:

- Telecommunication Service provider (TSP): are the common mobile providers such as Orange, who provide connectivity user to user or user to Ethernet.
- Subscribers: are the clients who consume the TSP services.
- Infrastructure Provider (InP): are the entities who are renting the infrastructure, in this case RF front ends and the networks.
- Server Provider (SP): provides computing resources in order to run NVFs. Amazon web service (AWS) can be an example of SP.
- VNF Provider (VNFP): this entity provides VNF in order to emulate network functions in a virtualized environment.

TSPs will provide connectivity to the subscribers through the InP and SP. Moreover, TSP will provide a specific service (wireless standard) to the subscribers implemented by a VNFP.

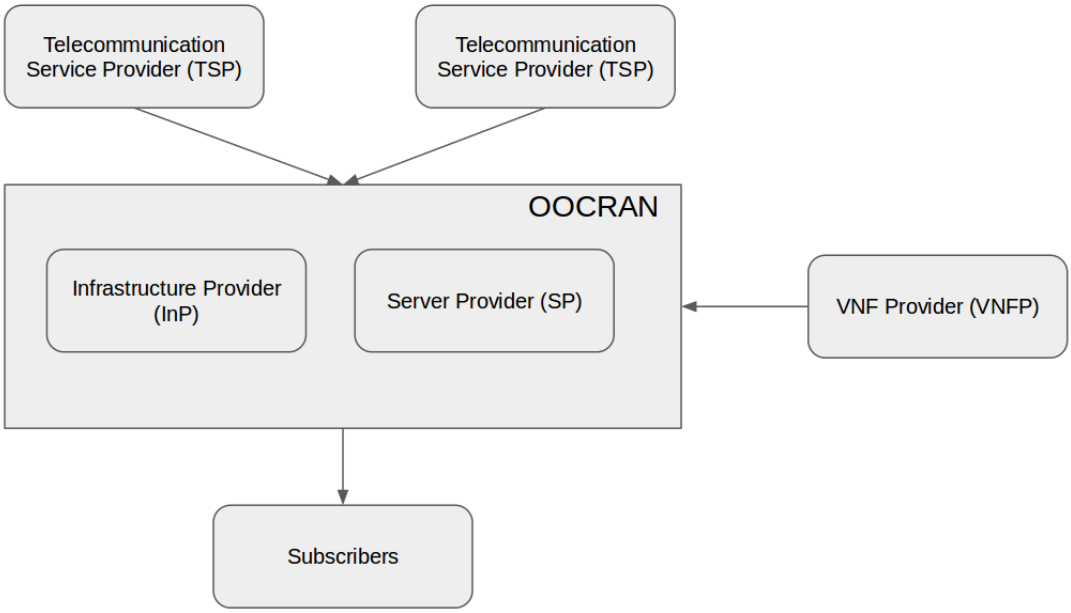

Fig. 3.3 Interaction between characters

In the course of this project, we will represent all the roles, but OOCRAN is designed for InP and SP. The investigation done in this project will benefit TSP, InP and SP. Finally, we will represent the VNFP role and the subscribers in order to make the testing of the platform.

3.5 Cluster Infrastructure

Once the concepts are explained, it is time to talk about the built infrastructure, the proprietary of this infrastructure is the SP and OOCRAN will configure the resources provided by this cluster. The infrastructure was created using the default installation of OpenStack release Mitaka, using a self-service network. Self-service allows each TSP to administrate their own network.

OpenStack in this release is composed by at least one controller node and one compute node. Controller node will manage the resources and virtual networks, by the other hand compute nodes are containers where the controller node attach and configure VM on it.

The hardware used by the implementation of the OpenStack have the following features:

	Controller (freq, cores, threads, cpus)	Compute 1 (freq, cores, threads, cpus)	Compute 2 (freq, cores, threads, cpus)
CPU (GHz)	2.37*4*2	2.37*12*2*2	2.37*12*2*2
RAM (GB)	8	52	52
Disk Space (GB)	120	240	240
Network (GBps)	1	10	10

Table. 3.1 Hardware specifications

The RRHs are USRP N210 which can recollect 25MS/s at 16 bits per sample, generating a maximum throughput of 25x32 = 800Mbps. By default we can connect with the USRP using a IP connection, attached in the 192.168.10.0/24 network.

The interconnection between all elements: USRP, OpenStack and OOCRAN is available on the next figure:

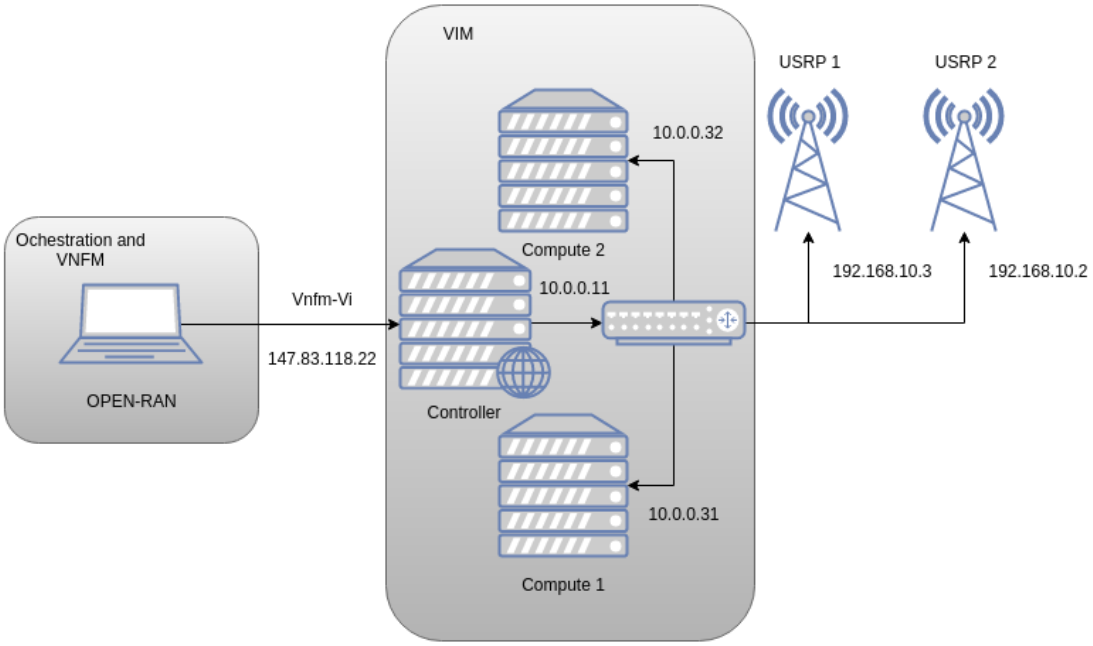

Fig. 3.4 OOCRAN hardware implemented

USRP are connected in the private network in order to provide connectivity to the NVF. By the other hand, OOCRAN will be connected with OpenStack through the controller in order to use the OpenStack APIs.

Once the infrastructure is created, the next step is to configure the computer resources management of the cluster. This configuration allows to do a better resources usage configuring the scheduler.

3.5.1. Software Layer Optimization

In this section we will describe the additional features activated on the OpenStack for the computational resources scheduler.

We though that the best configuration would be allocate each VM on a physical core, however, once we were doing the testing procedure, we did realize that we were wasting a lot of resources because the software modules (SDR) were not consuming all the available resources of the physical core. For that reason, we decided to configure the following parameters:

The first one is to force the CPU model of the NVF, avoiding the virtualization of CPU model. By doing that, we can use the optimization features of the CPU without emulation, increasing the performance. This features will be configurated on the nova configuration [21].

Parameter	Function
Cpu_model = custom	Allows to decide the family of the virtual CPU.
CPU_mode = host-passthrough	The virtual CPU family will be the same as the physical CPU

Table. 3.2 Configured nova parameters

Another feature is limiting the CPU usage of the NVF. Depending on the configuration of the NVF, the VM will waste more resources or less. By limiting the resources usage of each NVF, we can attach multiples NVF on the same core of the physical CPU. Besides, if one NFV fails it does not storb other NFVs. With the period and the quota we can limit the CPU usage. For example, if the quota is the half of the period, the CPU usage is 50%. By the other hand, it is necessary to activate the CPU pinning in order to apply these features [22] [23]. This features will be establish with the creation of the VM:

Parameter	Function
Hw:cpu_period = 5000	Establish what period of time will give resources to the VM.
Hw:cpu_quota = 2500	Establish the maximum number of resources that will be provided to the VM.
Hw: cpu_realtime = yes	Allows to run the VM under a real time scheduler (FIFO, Round Robin)

Table. 3.3 Virtual machine configuration

CPU pinning works running processes in different cores of the CPUs, providing resources to the particular process during a few period of time and then jump to another core. Thus, VM are reserving time processing of a particular core, not an entire core. This system allows not to force a particular core increasing the lifespan of all the physical cores.

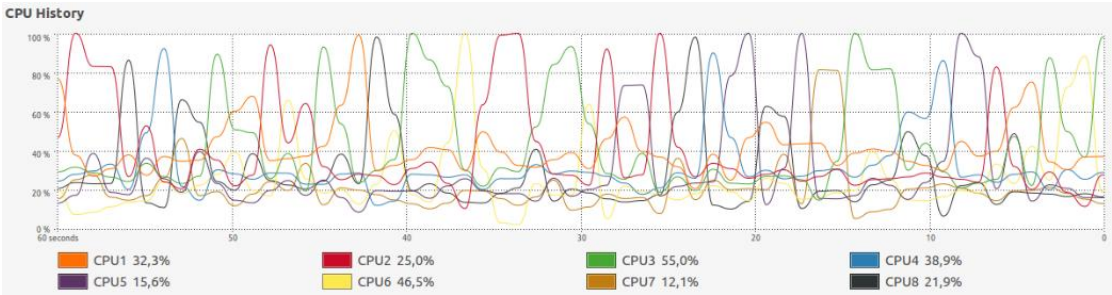

Fig. 3.5 Pinning effect

Once the hardware is configurated, it is time to move to the virtual environment where we will explain the virtual infrastructures implemented which are running under the implemented cluster.

3.6 VNF Implementation

All virtualized elements must be implemented in a VNF. We can find three types of them: base stations, channels and subscribers. We assume, all them are software modules running under a particular OS. We selected Ubuntu 14.04, due to the fact that we worked with them before.

The BTs and the subscribers are implemented using the srsLTE library (SDR code). But LTE standard has a strict requirements of real time (1ms), thus we need to used optimization features (VOLK libraries, SSE options, etc) in order to improve the performance of the code.

This software module allows to configure the following parameters: carrier frequency, bandwidth, power supply, modulation and code scheme (MCS), the message and the IP direction of the remote radio head. If these parameters are set in the boot command of the software module, OOCRAN will be able to configure the NFV without having to change the OOCRAN code. Therefore, it will be able to work with any sort of signal, bringing flexibility and agility when it comes to define a deployment.

```
./pdsch_enodeb_file -a addr={{ip}} -l 0.3 -g {{pt}} -f {{freC}} -p {{BW}} -i ../../../rfc793.txt -m 1 >> /home/nodea/run.log
```

This example allows booting a software module that generates LTE signals. Then, all the resources are given to a single user and the result is stored in .log file. The attributes in brackets “{{}}” indicate that it will be OOCRAN who will decide their value.

By the other hand, we will use a virtual channel that has a behavior of Additive white Gaussian noise, where we can configure the signal-to-noise ratio (SNR) level.

Finally, we introduce several interfaces to the all VNFs in order to reconfigure parameters and process data from other VNFs, creating a realistic communication system.

Once the VNFs are created, we can provide services using one or multiples VNFs. The provided service is a communication system.

3.7. System Communication Definition

We can characterize any communication system with a transmitter, channel and a receiver. As we explained in the objectives section the full chain can be virtualized or just parts of it. All the communications are using the User Datagram Protocol (UDP) protocol.

We will start describing the common configuration for the cloud-RAN, where we just virtualize the front-haul. The base station (NVF) will connect with RRH through an IP connection using the UDP protocol. In this case and in order not to have errors of real time we have configured the protocol to be not blocking, that means that the transmitter do not waits for an acknowledgment from the receiver.

In the following figure you can see the communication chain:

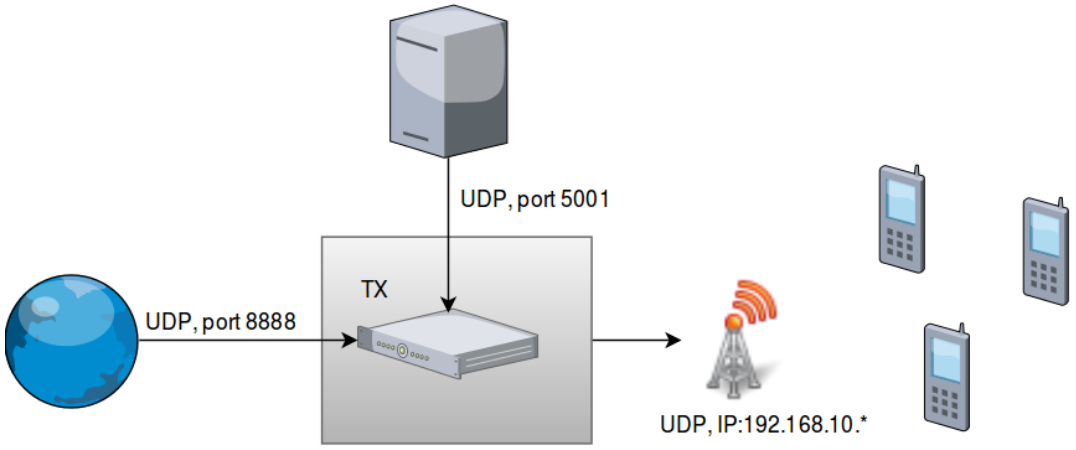

Fig. 3.6 Fronthaul virtualizing architecture

NVF and RRH will be inside the private networks, specifically, NVF will be in 20.0.0.0/24 and RRH in 192.168.10.0/24. When a NVF is launched the VIM will assign an IP to each VM using the DHCP protocol. It will be OOCRAN task to associate each NVF with its corresponding IP. This task will be carried out using the NOVA and Neutron APIs. The IP configuration of the RRHs must be done by hand.

Apart from the connection interface with the RRH, each BTs will have several services running in parallel.

- Reconfiguration service through the port 5001. The connection will be established with the OOCRAN to reconfigurate one or several attributes of the BTs.
- Data gathering service through the port 8888.

The full virtualized chain has more virtualized elements but all of them have the same interfaces than the previous architecture. The software modules can run in a separated NVF or in the same VM saving then more resources. In the following example all the elements will be virtualized, therefore, it will be necessary to create a circuit that connects all the NVF if we decide to launch each NVF separately. To do such circuit, we will place each NVF in a different subnet, BTS (20.0.0.0/24), channels (30.0.0.0/24) and subscribers (40.0.0.0/24), the IPs will be assigned using the DHCP protocol. Again, OOCRAN will configure the connections as well as the different networks.
In the following image, we can see the resulting circuit:

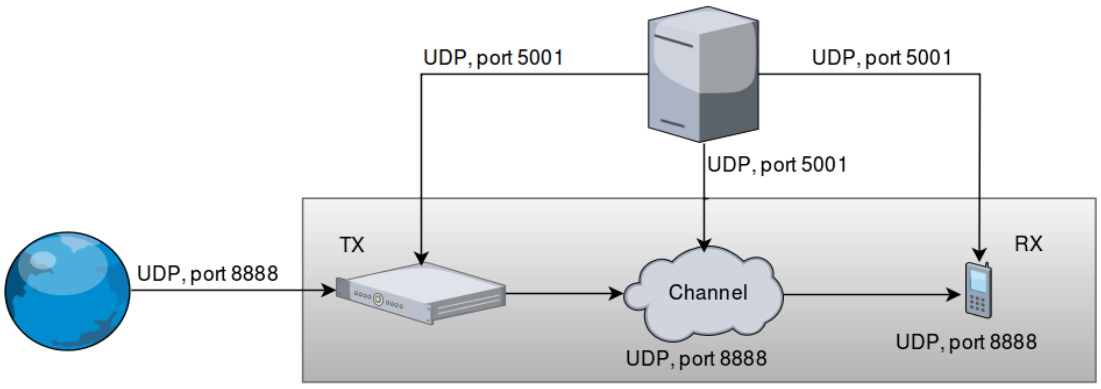

Fig. 3.7 Full communication system virtualization

Once the communication system was implemented, the next move was to design an infrastructure composed by multiple communication systems (mobile deployment) that work together, in this case these infrastructures provided a wireless mobile service.

3.8 Virtual Network Infrastructure

The communication systems are interconnected because the elements exchange data, thus we designed a network of base stations, virtual channels and virtual subscribers.These network is specific for each area and TSP, therefore, each area will have as many as networks as TSP. These networks will be built when the InP introduces a new area, and they will have the following configuration:

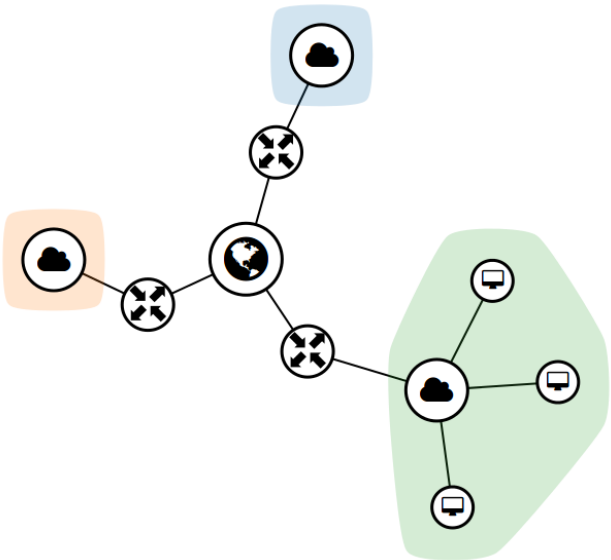

Fig. 3.8 Virtual network infrastructure architecture

Each network is composed by 3 subnetworks:

- BTs network where NVFs that will play the role of BTs will be assigned. The range of IPs that will be assigned is 20.0.0.0/24
- Channels network, the proposed IP range is 30.0.0.0/24
- Subscribers network where NVFs will be assigned IPs in the range 40.0.0.0/24.

As we can see in the previous figure, every subnetwork is connected to the interface that gives access to the outer network where there are the USRPs. We decided to implement this network because it is well organized, thus, it will allow us to look for the NVF in more efficient way and at the same it is easy to build. The template to generate this network can be found in the annex 1.

Over this infrastructure we will deploy one or multiple mobile infrastructures. We allow to deploy multiple mobile infrastructures because by doing that we can make a macro cellular deployment and a micro cellular deployment at the same time. Then, we can administrate them in a different manner and therefore, a change in a particular deployment does not interfere on the other infrastructures of the same TSP.

Mobile infrastructures combine complete and partial communications systems. Besides, it will be possible to use different wireless standards with different configurations for each standard. Virtual channels and subscribers of each base station will be defined in the same template. In annex 2 there is an example of these templates.

On the next section we will explain how we created the virtual network infrastructures and configured the NVFs.

3.9 Virtual Network Manager (VNFM)

The main function of the VNFM is to manage the VNFs and communicate the configuration to the VIM through the OpenStack API. In this section, we will explain how we will use the OpenStack API in order to configure the mobile deployment.

As we explain in the section 3.5. (Cluster infrastructure), there is a specific module of OpenStack deployed for the implementation of the infrastructures, this module is called Heat and works with templates which is a particular programming language. In these templates we will define the elements of the infrastructure [24].

OOCRANs' tasks are programmed and stored by using these templates and according the specifications from each TSP. Once the template is written, it will be sent to the VIM in order to create the expected infrastructure. The template

will be stored in a repository of infrastructures, then we will launch one or more templates when certain conditions are met.

In heat templates we will find several kinds of elements (VM, switches, networks, subnetworks, etc). Regarding the network elements, we just have to configure the input/output ports, IPs ranges.

In the case of VM, it will be necessary to configure several parameters to define the NVF properly. The structure to create a NVF using templates is the following:

```
Element_name:
   type: OS::Nova::Server
   properties:
     name: {{name}}
     image: {{image}}
     flavor: {{flavor}}
     networks:
     - network: {{network}}
     user_data_format: RAW
     user_data: |
       #cloud-config
       runcmd:
        - {{script}}
        - sh /home/nodea/start.sh
```

The command type: OS::Nova::Server indicates the sort of element his properties are explained in the list below:

- Name: NVF identifier, for instance, in the case of BTs we will identify them using a numerical value.
- Image: This is the software, by software we mean the whole operative system with its SDR modules that are going to be executed. As we have explained in the previous section, 3.6 VNF implementation, we have created these images.
- Flavor: hardware specifications (CPU time, amount of RAM and space on the disk). It is important to define the flavors so as to optimize the use of the hardware resource and do not waste them. That's why we designed an algorithm to define the flavors that we will explain later.
- Network: It indicates the network where the NVF will be connected. Depending on the task of the NVF it will be connected in one network or another.
- Script: Inside the VM there are several software modules and depending on the task, we should execute one or another. The execution of this software module will be done with a script. It is important to highlight that this script is programmed in bash as it allows to execute several modules in sequence or create complex execution structures.

The module that decides the values of the properties is the orchestrator, who depending on the state of the infrastructure take decisions. In the next section we will talk with more detail about the orchestrator.

3.10 Orchestrator

The main goal of the orchestrator is to administrate the virtual infrastructure, deciding the configuration parameters of the virtual network manager. Furthermore, this module contains the user interpreter that was designed using MVC software architecture explained on the section 3.1 Django.

Besides, for the orchestrator implementation we used Google maps APIs providing a visual location of the RRHs, and bootstrap to improve the visual interface experience.

As we said on section 3.1, Django works with multiples controllers and the architecture proposed has the following controllers:

- Vnfs: administrate the VNFs making a repository where the properties and the execution command of each one will be saved. BTs, channels and subscribers have the same defined properties explained on the section 3.4 VNFM. Furthermore when the TSP introduce a new VNF, this controller do a monitoring of the computational resources usage by the NVF and create a virtual hardware (flavor) with the minimum requirements for avoid resources wasting.
- Operators: administrate the TSPs and his own configuration (OpenStack credentials, MCS, settings, etc)
- Scenarios: Administrate the areas where the TSP allocate the RRHs and save the properties of each one (location, neighbors, frequency carriers, maximum power supply, maximum bandwidth allowed). Furthermore this controller create the virtual network infrastructure.
- Users: administrate the subscriber repository and the connections between subscriber and BTs.
- Deployments: administrate the NVFs, channels and deployments, working similar to the scenario controller using a CSV file, where TSP indicate the base stations, channels and subscribers. Betwen his tasks there are the frequency reusing and the creations of the mobile deployment.

For each area, we have a demanded bit rate forecasted according to the hour of the day. Then the TSP can select two deployment ways. Those ways will be administrated by the deploy controller, configuring the NVF parameters.

- Make a mobile deployment according to a forecast indication the period of time where the infrastructure will be active. Once this will be done the infrastructure will be deleted. The TSP is who decides when and what mobile infrastructure will be the most suitable.
- Let OOCRAN search for the best mobile infrastructure according to the demand. In order to work with this mode will be necessary to provide to OOCRAN a mobile infrastructure repository. From this repository OOCRAN

will search for the more suitable one (bit rate demand) depending of the hour of the day.

For each mobile infrastructure the deploy controller must assign a price according to the number of NVF launched and the bandwidth used by each one. The price will be specify on the section 4.5 cost implementation.

With the OOCRAN architecture being explained, we can move to the result chapter where we make the testing process of several OOCRAN functionalities

CHAPTER 4. Testbed

In this chapter we present the results obtained from defined case study where developed OOCRAN platform shown their capabilities. The case study, that include the deployment and management of the mobile infrastructure, provided the scenario where to perform the test defined in chapter 3 (objectives)

4.1. Scenario description

Case study defines a mobile deployment in the EETAC campus using femtocells. Femtocells have a cell radius around 30 meters. They are normally used in hotspot areas, however, in the new mobile generation 5G it is expected to make large deployments with this kind of base station. Thanks to its small coverage range it facilitates the application of spectrum resource management techniques, like frequency reuse to improve the bit rate. But it must be done under the cloud management premises.

Depending on the purpose of the test, we adapted the location, power supply and number of base station on the map. All the test has been done with the same assumptions:

- Femtocell deployment: the C-RAN will use this kind of base stations.
- Omnidirectional antenna pattern: more realistic pattern are difficult to implement and they do not introduce conceptual advantages on the test proposed.
- Free Space Propagation model: depending on the area the model will change.For that reason we used the most common and simple one.
- Overlook the city clutter (attenuation): introduce more complexity to the platform and its effect does not contribute so much to the proposed study.
- The RRHs can work with various frequency carriers: C-RAN was designed with these configuration in mind so it is strictly necessary to apply it.
- BTs allow sending data to multiple subscribers at the same time by assigning different bit rates to each one.
- All frequency carrier has a bandwidth of 1.4MHz to limit the scope of this study although any LTE bandwidth is available under current lab infrastructure.

4.1.1 Link Budget

Under a realistic case study it is mandatory to calculate the the link budget. Link budget is a power balance of the communication system, moreover, the link budget sets a maximum cell range available for the transmission [25].

Downlink		Uplink	
data rate	1 Mbps	data rate	64 Kbps
Transmitter - eNode B		Transmitter - UE	
HS_DSCH power (dBm)	23	Max. TX power (dBm)	24
TX antenna gain (dBi)	0	TX antenna gain (dBi)	0
cable losses (dB)	0	Body loss (dB)	0
EIRP (dBm)	23	EIRP (dBm)	24
Receiver- UE		Receiver - eNode B	
Noise figure (dB)	7	Noise figure (dB)	2
Thermal noise (dBm)	-104,5	Thermal noise (dBm)	-118,4
Receiver noise floor (dBm)	-97,5	Receiver noise floor (dBm)	-116,4
SINR (dB)	-10	SINR (dB)	-7
Receiver sensibility (dBm)	-107,5	Receiver sensibility (dBm)	-123,4
Interference Margin (dB)	0	Interference Margin (dB)	2
Control channel overhead (dB)	0	Cable Loss (dB)	2
RX antenna gain (dBi)	0	RX antenna gain (dBi)	0
Body loss (dB)	0	MHA gain (dB)	2
Maximum path Losses (FSL)	130,5	Maximum path Losses (FSL)	141,4

Table. 1.1 Linkbudget Calculus

As we have mentioned before, OOCRAN can work in two different modes:

a) Virtualization of the fronthaul.
b) Virtualization of the whole system.

The virtualization of the whole system provides a powerful simulation tool without wasting power resources in radio transmission. Moreover, it opens the way to scenarios, mixing simulations and real RF links that can provide interesting results at lower development/deployment cost.

Besides, the fact of creating a system chaining several NVFs allows us to decouple the implementation of each one of them. These changes in each NVF without it affecting the others. On the other hand, if we implement a new software module following the characteristics described in the section 3.7 we could send the signal to a RRH or a RRH of another manufacturer just by changing the drivers without modifying the SDR code.

Virtualizating the fronthaul ,we can send IQ samples and remove the processing chain of the RRH. If in the future we wanted to compress data between the BT and the RRH it would be as easy as add a new element that would do such operation.

Therefore, the fact of implementing the communications in several elements provides us additional flexibility.

4.2. LTE processing chain

The first step is to test a common communication system composed by transmitter, channel and receiver. As mentioned before, our goal is to implement two kind of virtualized components/subsystems: fronthaul virtualization and full chain virtualization. It is important to test and characterize the signal on both communication systems.

For this test, we transmitted a the data from a text file using a LTE downlink with a bandwidth of 1,4 MHz and a MCS of 1 (QPSK). We are just using only the physical layer of the downlink of the base station, because other layers do not contribute to improve the received wireless signal.

4.2.1. Real channel transmission

In this scenario the NVF sends IQ samples to the RRH using a a UDP protocol that requires up to a throughput of 80 Mbps. Our initial processing cluster infrastructure was designed with CAT 5 Ethernet cables, which support 1Gbps.

At this point we found a bottleneck on the cluster infrastructure. These bottleneck is produced because all traffic needs to pass through the controller node in order to reach the RRHs. This fact allows to create virtual networks between VM attached in different compute nodes but introduce an unnecessary loop on the cluster.

The signal transmitted by the RRH and captured by an spectrum analyzer at 5 meters from the source is shown in , the following figure:

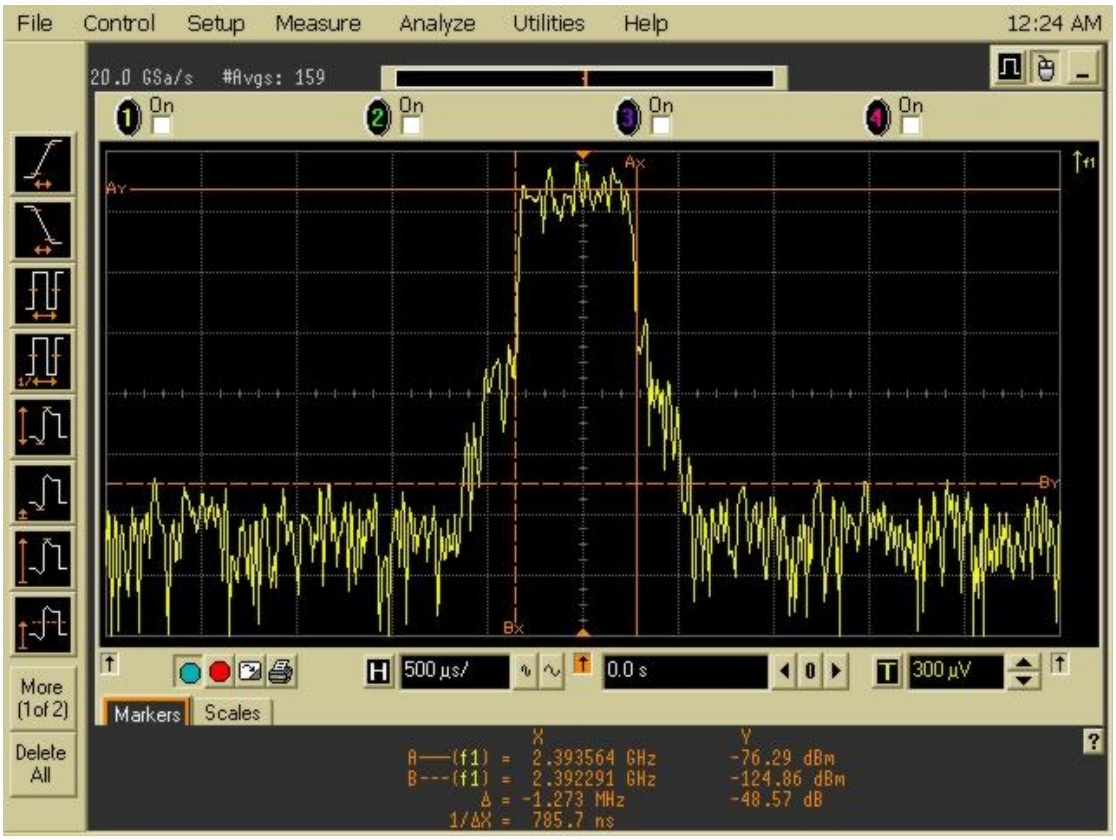

Fig. 4.1 LTE channel captured

We can see that the ACLR (adjacent channel leakage ratio) is around 49dB , the bandwidth close the expected value of 1,4 MHz and the noise floor around 49dBs [26] what can be considered as a good enough generated signal. The receiver can demodulate the signal with a resulting BLER (Block Error Rate) of 0.02 which indicate that the channel is good and the system is robust.

4.2.2. Full virtualized processing chain

Once we have seen the results of the front-haul virtualization, we experienced the same routing problems with the “Controller”. Thus in order to avoid the bottleneck installed all the software modules (transmitter, channel and receiver) on the same NVF. To execute this NVF has been necessary to reconfigure the ports and assign localhost to all IPs addresses involved. With this configuration the traffic flow is not routed outside the computing network avoiding the bottleneck.

The signal transmitted in this case has the same configuration than the previous real RRH system, but we emulate the channel with a SNR around 5dB. The channel introduce degradation on the signal producing errors on the demodulation process. Even assuming the channel effect the resultant BLER is around 0,02 good enough for the LTE communication system. The following

figure, illustrate the results, where the yellow window is the transmitter, the blue window is the channel and finally the green one the receiver.

Fig. 4.2 Full virtualized communication system test

In order to allow communication between all system modules, it was necessary to make a port reconfiguration without repeat ports.

4.3. Radio frequency resources Scenario

Once the communication system was tested is time to implement a mobile infrastructure with multiples base stations and applying techniques of mobile deployment. One of the important objective of this project is the analysis the frequency reuse under a cloud environment. This tested has been specially designed to study it.

In the section 4.1. (Scenario description), we supposed that all the antennas are omnidirectional. With this configuration we can approximate the cell effective range by an hexagon, where in each side there are a frequency overlapping with other base stations. Taking this in mind, the best way to implement a frequency reuse without waste radio resources is using three frequencies as like the following figure illustrate:

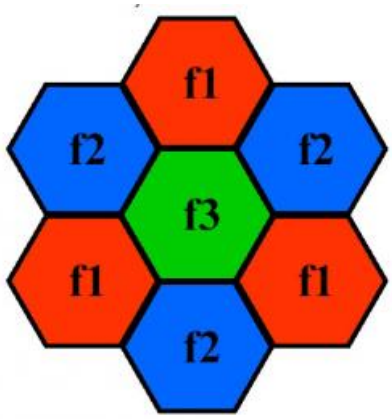

Fig. 4.3 Frequency reused applied [27]

By the other side, this frequencies reuse is particular for that TSP. If another TSP works on the same area, he will need three new frequencies as well.

4.3.1. Frequency bands reuse

In this case study we will deploy a mobile infrastructure like to the cluster showed on the figure 4.3. As we can observe in that figure each base station has at least three interference except the central base station who experience six interference. Thus our main goal is to provide LTE services in the area using just three frequency carriers.

In fact this problem is a coloring graph theory, therefore we search about this theory in order to implement the frequency reuse algorithm.

The orchestrator knows the base station location, his base station neighbors and the assigned frequencies. Thereby when a TSP request for frequencies carriers the orchestrator must do a overview of the network and decide the best frequency carrier for each base station.

Note that InP is who provides the frequency carrier and the TSP just request for a frequency carrier with 1.4 MHz of bandwidth.

```
Data: neighbors_assigned, bt_assigned,
Result: Assigned carrier frequency

freC = 2.39 GHz #starting available radio frequency band

While (True)
    If not carrier in  neighbors_assigned do #look for neighbor assigned
frecs
        If not carrier in bt_assigned do #look for actual BT assigned frecs
            assign_freC = freC
```

```
            break
        Else
            freC=freC+1.4MHz
    Else
        freC=freC+1.4MHz
end
```

The next figure indicates that the frequency reuse is correct, using three frequencies for the deployment. The resultant frequencies are 2.3907 GHz, 2.3921 GHz and 2.3935 GHz, the distance between carriers is 1.4 MHz and the useful spectrum starts on 2.39 GHz.

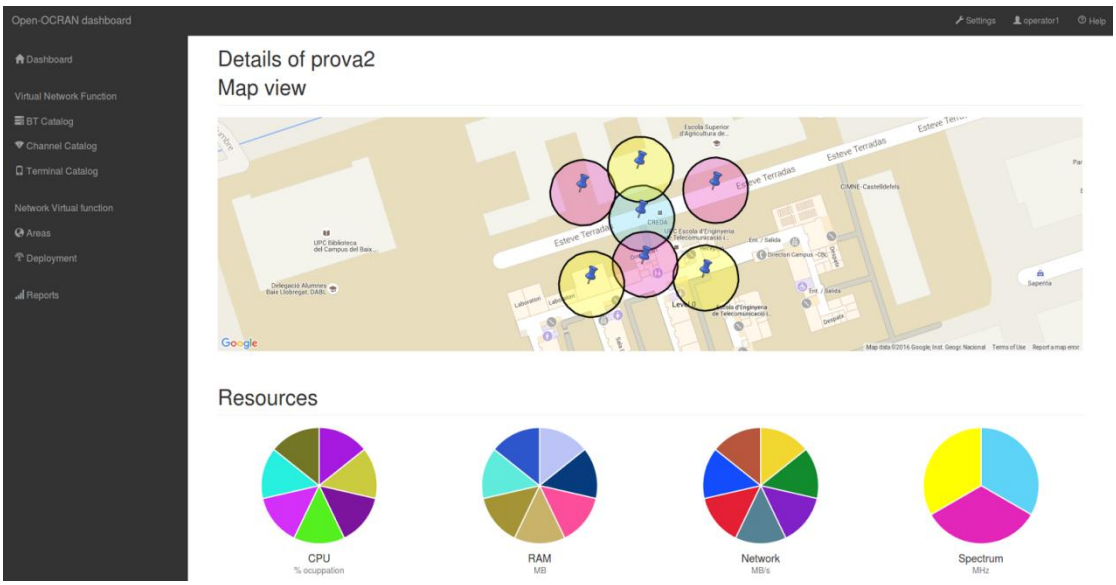

Fig. 4.4 Frequency reuse applied on OOCRAN

4.4. Dynamic deployment under demand

In the previous two sections, we tested the communication system and the infrastructure network. The next step has been to make periodical updates of the mobile infrastructure. The modifications can be done following two methodologies:

a) Reconfiguring the NVFs one by one.
b) Delete the entire mobile infrastructure and deploy a new one.

Selecting the second methodology it is necessary to create a new mobile infrastructure from scratch and this process takes some additional time. In this section we will focus on the advantages of making a new mobile infrastructure from scratch. This test will be useful to analyze the radio resource savings.

For this test study, we tried to follow two time-geographical traffic patterns: the residential and business ones. These two patterns are commonly observed in any mobile infrastructure.

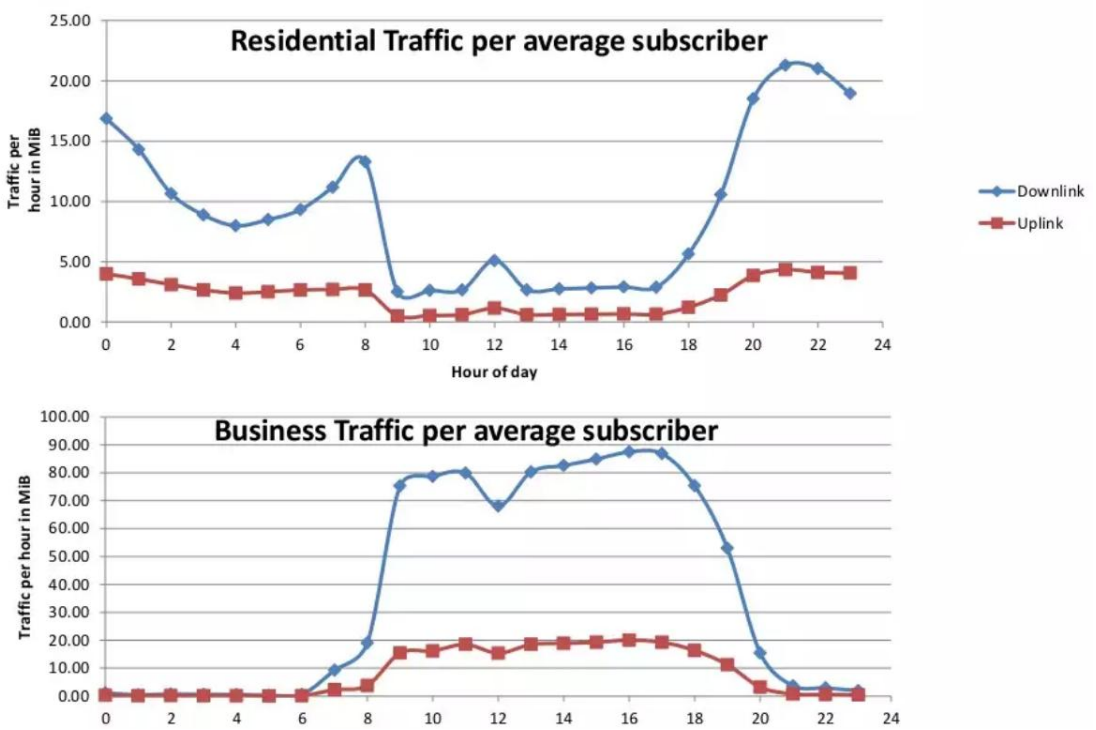

Fig. 4.5 Time pattern used on OOCRAN [28]

In the traditional deployment the best way to provide a good service for the customers is to design the infrastructure for endure the “busy hour” (the hour of the day when there are more connecting and more traffic demand). With this deployment is easy to see that in a lot of cases we are wasting a lot of resources. One of the options is to shut down base stations, saving power consumption but the TSP is still paying for the radio spectrum not being used at in that moment.

In C-RAN, we can shutdown the RRH and eliminate NVF for saving resources. The question is how faster we can react to the changes in the traffic pattern. It is well known that a good characterization makes possible to enhance energy and spectrum resource savings. Remember that in C-RAN the TSP paid for the use of resources not for buying them.

4.4.1. Out Rate study

The cell load can suffer modifications every 1ms (processing time of LTE), if we can reconfigure the base station at that speed the bit rate demanded will be equal to the offered bit rate. In that situation there are not radio resources wasting, but it is impossible to achieve this requirement if the mobile infrastructure requires to be expanded or reduced (create or delete NVFs).

Then, the proposed solution is using a moving average, normally making the average every hour. Therefore, normally we are covering the area, however, there are period of time where we are wasting resources and others where the cell is overload. The following graph illustrate this situation:

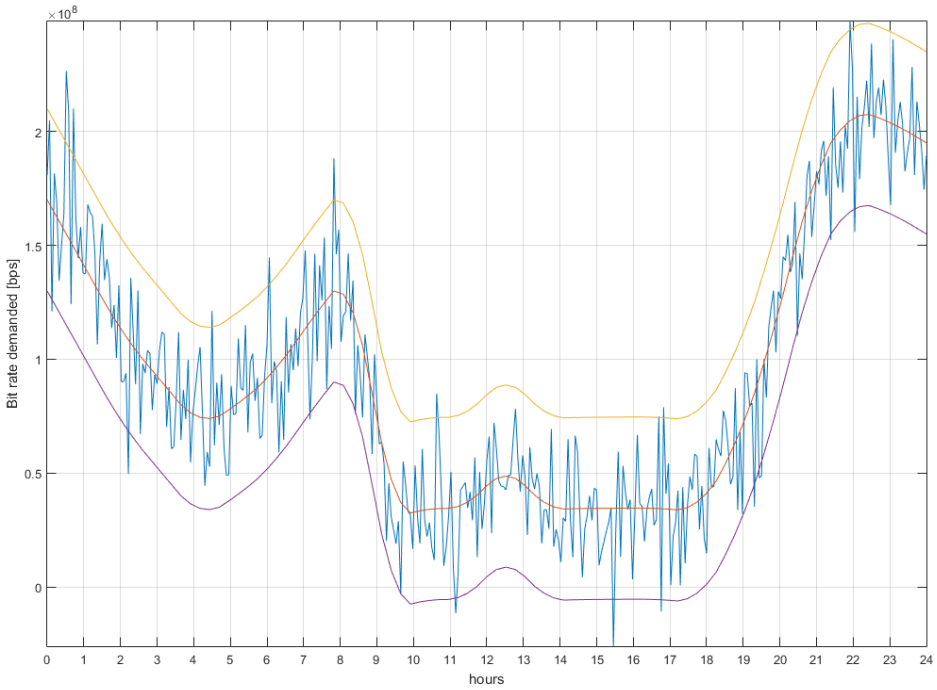

Fig. 4.6 Out rate example

Then we could build a mobile infrastructure above the average, avoiding the cell overload and wasting more resources or build the mobile infrastructure below the average saving resources but making the subscriber wait in order to get services. We would like to maximaze the following formula:

$$C(t) = R_u(t) - R_w(t) + US(t)$$

- C(t): cost-benefit function.
- $R_u(t)$: resources used.
- $R_w(t)$: resources wasted.
- US: unattended subscribers leads to unsatisfied clients and that generate losses.

The $R_u(t)$ generate profits, the $R_w(t)$ and unattended subscribers generate losses. Thus, it is necessary to found a suitable point on the C(t) function that allows save resources providing the maximum profits without unattended subscribers. It is easy to see that depending on the speed of the mobile infrastructure modifications this costs will be higher or lower.

With our own physical infrastructure we took VM creation speed measurements, and we saw that the launching and configuration of the base station, takes at

least 30s for one VM. And if we desire to reconfigure several of them this time increases following a polynomial function. The following graph illustrate this fact:

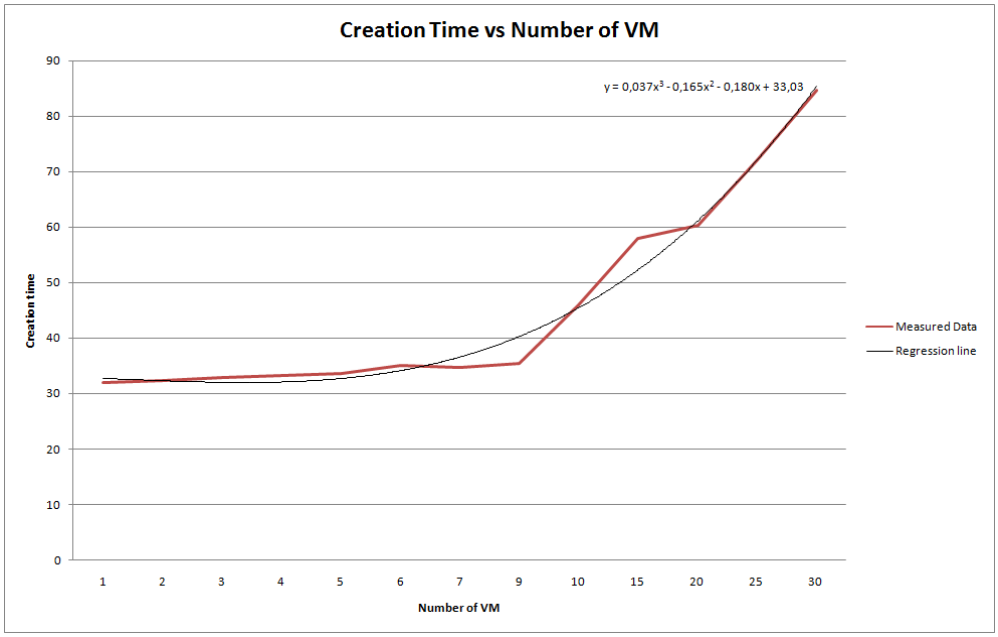

Fig. 4.7 VM time creation study

The VM time creation make impossible to follow the bit rate demand, but we can do a partial reconfiguration of the infrastructure in order to reduce the radio resources waste. In this study we delete the whole mobile infrastructure an launch again with the new reconfiguration.

In our scenario the deployment area is the EETAC campus with around 419m x 139m = 58241m², where making a deployment from scratch requires:

Number of BT	Area	Time
1	3.14*30²x1 = 2826 m²	30,12s
5	14130 m²	33,49s
10	28260 m²	45,87s
20	56520 m²	60,19
21	59346 m²	60,23s
30	84780 m²	84,63s

Table. 4.2 Time reconfiguration according to the area

In addition, we need one minute to implement the a new infrastructure from scratch. During this process all the subscribers attached to each base station

are released and the connection freed are not recovered until the creation of the new infrastructure finished.

As a result we should avoid updating the mobile infrastructure for small periods of times. The common time-geographic patterns are represented in periods of time of an hour, during this time the traffic demanded do not have a lot of variations. So we can assume that updating the infrastructure each hour is technologically possible.

OOCRAN allows to do this reconfiguration in automatic way, making a repository of infrastructure and according to the traffic demand forecasting launch or delete infrastructure depending on the hour of the day. OOCRAN calculates the traffic offered by each infrastructure and decide what is the most suitable according to the current demand.

The algorithm used are the following:

```
Data: forecast, catalog, select
Result: array of best deployments
for sample in forecast:
    Select = x
    for deploy in catalog:
        if deploy == sample:
            select = deploy
            continue
        if deploy > sample and deploy < select:
            select = deploy
            continue
```

With this algorithm the results provided with a repository with four deployments saved was:

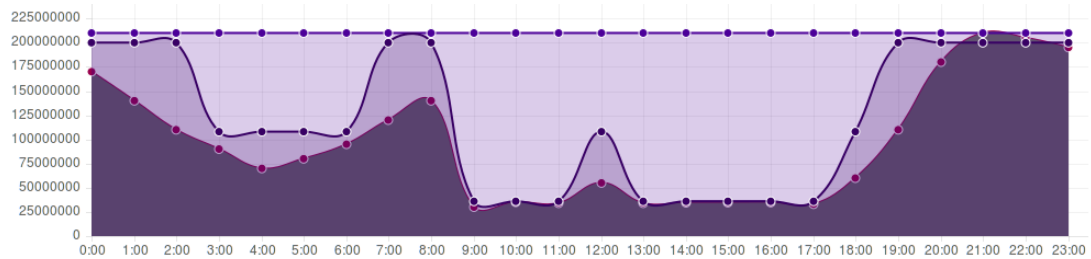

Fig. 4.8 Dynamic deployment using OOCRAN

A preliminary objective has been to create a repository with a number of deployments with enough granularity where to find the the best suitable one according to the requirements.

4.4.2. Infrastructure Management Strategies

On the previous section, the results obtained indicated that we can achieve a reconfiguration time of one minute in our scenario. However, this is not true because if we setup this configuration the mobile infrastructure will be updated every minute, hence there will be no time to provide service. Therefore, first it is necessary to apply more relaxed time scheduler and look for a softer reconfiguration algorithm that allows to reconfigure the mobile infrastructure without call faults.

An enhanced methodology for updating the infrastructure is doing a partial reconfiguration and force the attached subscriber to do a handover to new auxiliary cell. With this process we avoid the call failure making more profitable the mobile infrastructure. However we increase the reconfiguration time (subscriber must do two handovers in each reconfiguration) and increase the radio and compute resources using a auxiliary cell. Moreover the auxiliary cell must have enough capacity in order to provide services to the new subscribers, making this process more complicated (reduce bit rate by subscriber in order to attach more subscriber, etc)

4.4.3. Comparative dynamic vs traditional infrastructures

It is obvious that dynamic deployments introduce a lot of flexibility to the infrastructure but clearly become more difficult to administrate. This problems increase with big infrastructures where it is necessary to perform multiples partial reconfiguration.

The most advantage of the dynamic infrastructures are on the radio spectrum usage where for example in the figure 2, with a repository with 4 infrastructures we achieve a radio spectrum saving around the 40%.

In a traditional infrastructures, we can just reconfigure base station changing the bandwidth, power supply and frequency carrier but for example it is not possible to change the wireless standard or apply algorithms in order to improve the service. These are additional flexibility that must be taken into account in the cost equation. Moreover the business model of the traditional mobile infrastructure do not allow to do a good spectrum usage making the OPEX investments higher.

4.5. Implementation cost

The final case study is the cost of investment of the C-RAN infrastructure. In this section we analyze the cost of the CAPEX/OPEX and we propose a cost models for the use of the infrastructure. The calculation has done in eur/Mbps which allows to make a comparison between both infrastructures (traditional vs C-RAN).

4.5.1. CAPEX

In this cluster infrastructure two conventional computers has been used, one for management and the other for computing purposes. The code to process LTE is open source, so the derived development cost has not been taken into account. The price of the rental and cooling system are not taken into account because we rent a place then the price will be included on the OPEX. The equipment considered have the following features and price:

	Feature	Price
Intel i7 3 generation	2,67 GHz, 8 cores	1000 euros
Intel Xeon	2,37 GHz, 48 cores, 52GB RAM	10000 euros
USRP 210N	100MS/s at 8 bits with a BW of 20MHz	1810 euros
Ethernet wires	CAT 6 (10Gbps)	6 x 5 =35 euros
Total price		7845 euros

Table. 4.3 Hardware specification and price

The infrastructure can be created using multiples compute nodes reducing the price of the infrastructure. This asseveration is not totally true. The use of more compute nodes requires to occupy more space. Equipment footprint should also be taken into account.

Note that we are not taking into account the RRHs installation setup, which can be expensive in some cases.

	Channel Bandwidth	1.4 MHz	3 MHz	5 MHz	10 MHz	15 MHz	20 MHz
	Resource Blocks in the frequency domain	6	15	25	50	75	100
Normal Cyclic Prefix	OFDMA symbols per 1 ms	14					
	Modulation symbol rate (Msps)	1.0	2.5	4.2	8.4	12.6	16.8
	QPSK Bit Rate (Mbps)	2.0	5.0	8.4	16.8	25.2	33.6
	16QAM Bit Rate (Mbps)	4.0	10.1	16.8	33.6	50.4	67.2
	64QAM Bit Rate (Mbps)	6.1	15.1	25.2	50.4	75.6	100.8
	2×2 MIMO 64QAM Bit Rate (Mbps)	12.1	30.2	50.4	100.8	151.2	201.6
	4×4 MIMO 64QAM Bit Rate (Mbps)	24.2	60.5	100.8	201.6	302.4	403.2
Extended Cyclic Prefix	OFDMA symbols per 1 ms	12					
	Modulation symbol rate (Msps)	0.9	2.2	3.6	7.2	10.8	14.4
	QPSK Bit Rate (Mbps)	1.7	4.3	7.2	14.4	21.6	28.8
	16QAM Bit Rate (Mbps)	3.5	8.6	14.4	28.8	43.2	57.6
	64QAM Bit Rate (Mbps)	5.2	13.0	21.6	43.2	64.8	86.4
	2×2 MIMO 64QAM Bit Rate (Mbps)	10.4	25.9	43.2	86.4	129.6	172.8
	4×4 MIMO 64QAM Bit Rate (Mbps)	20.7	51.8	86.4	172.8	259.2	345.6

Fig. 4.9 LTE speed in function of the modulation and BW [29]

According to the table LTE with a bandwidth of 1,4MHz using a QPSK modulation total Mbps provided are 1,7 Mbps, therefore the total cost in eur/Mbps is:

Total cost = 7845/1,7 = 4614,70 eur/Mbps

At the end, the total price for MBps is quite lower if we compare with the traditional one, the main reason is the implementation in GPPs are more cheaper than ASICs.

4.5.2. OPEX

To calculate OPEX maintenance costs of computer and USRP are checked. Once this step is done, a cost will be assigned to each resource and according to the use given the user will pay a different price.

The costs are:

- Electricity tariff: 0,15 eur*KW.
- Useful life of the component around 2 years.
- Cooling electricity consumption: 2KWh.
- Rental around 400 eur*mouth.

Maintenance costs are:

Resource	Feature	Total price
Compute node	2 CPU with maximum usage 85 W	2*(85/1000)*0,15*3600= 91,8 eur*h
USRP N210	Maximum usage 35 W	(35/1000)*0,15*3600 = 18,9 eur*h
Repairment	2 years new compute	5000/(2*12*30*24) =0,2893 eur*h
Rent	Normal room with cooling provided	400/(30*24) = 0,56 eur*h
Total		111,4 eur*h

Table. 4.4 OPEX price by element

The total price of OPEX that the InP must paid is around 111,4 eur*h. With this infrastructure the InP can offer 44 cores, 50GB of RAM and 240GB of disk. Own VNF consume 20% of the one core, 1GB of RAM, 6GB of disk. With this configuration the InP can provide 40 NVF, the limitation is on the disk space.

Therefore the OPEX investment for deploy a base station is:

Cost per BT = 111,4/40 = 2,8 eur*h (1 core, 1GB RAM ,6GB disk , 20MHz of spectrum)

But the VNF is consuming about 20% of the core, 512 of the RAM, 6 GB of disk and 1,4 MHz of spectrum. The bottleneck in this case is the disk usage, if we are using 40 VM and each one occupies 6GB we are using 240GB of disk which is the entire disk.

In orther to do a better resource usage will be necessary to use a more high disk capacity, we can divide all cores in 5 (20%x5=100%) providing 44*5= 220 virtual CPUs. In this case we have a limitation with the radio spectrum.

Cost per BT = (111,4-18,9)/220 = 0,42 eur*h
Cost of BW = 1,4*18,9/20 = 1,32 eur*h
Total price = 1,74 eur*h

For this case study, we apply this price without profit, making two deployments during a small period of time.

- Macro: two base station running during two hours = 1,74x2x2= 6,96 eur
- Micro: three base station running during two hours = 1,74x3x2= 10,44 eur

Making a total price around 17,4 euros. The following picture show that deployment price is included on the OOCRAN platform.

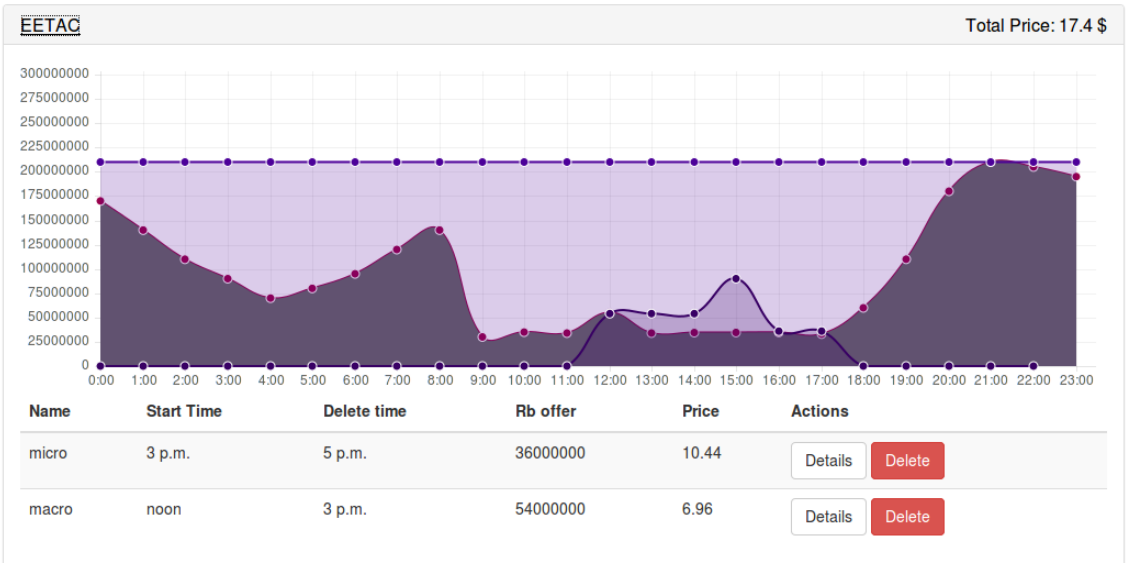

Fig. 4.10 Prices applied to the OOCRAN

The best way to make a comparison between traditional and C-RAN infrastructure is represent the CAPEX/OPEX in euros/Mbps.

Cost per MB (BT) = 0,42/1,7= 0,247 eur*h/Mbps

Cost per MB (USRP) = 1,32/1,7 = 0,776 eur*h/Mbps

Total cost = 1,023 eur*h/Mbps

The price changes according to the modulation and the used bandwidth, where for high modulation scheme the TSP achieve the maximum cost save because high modulations in the transmitter do not add extra processing time. The following table shows the costs in eur*h according to the bandwidth and modulation of the transmitter:

Modulation/BW	1,4MHz (20%)	3MHz (35%)	5MHz (58%)
QPSK	1,023	1,54	2,09
16QAM	0,511	1,21	1,76
64QAM	0,344	1,10	1,65

Table. 4.5 OPEX prize in fucntion of the modulation and BW

The calculations can been found in the annex 3.

As we increase the bandwidth the processing time increases as well. Then, for high bandwidth the price for a QPSK modulation is doubled.

Even all that, the optimal point is using high bandwidth and dense modulations. That’s because it allows to sent many more traffic and the price is lower than in the case we were using multiples NVF with bandwidths equal to 1.4MHz.

4.5.3. Comparative C-RAN vs traditional investments

The comparison just can be done in the same conditions, for that fact we can compare traditional deployment of LTE License Assisted Access (LAA) vs C-RAN. LTE LAA works on the free band 5GHz normally used by Wi-Fi. The following graph show that the price of the radio spectrum and the setup of the tower is the most expensive part of the infrastructure.

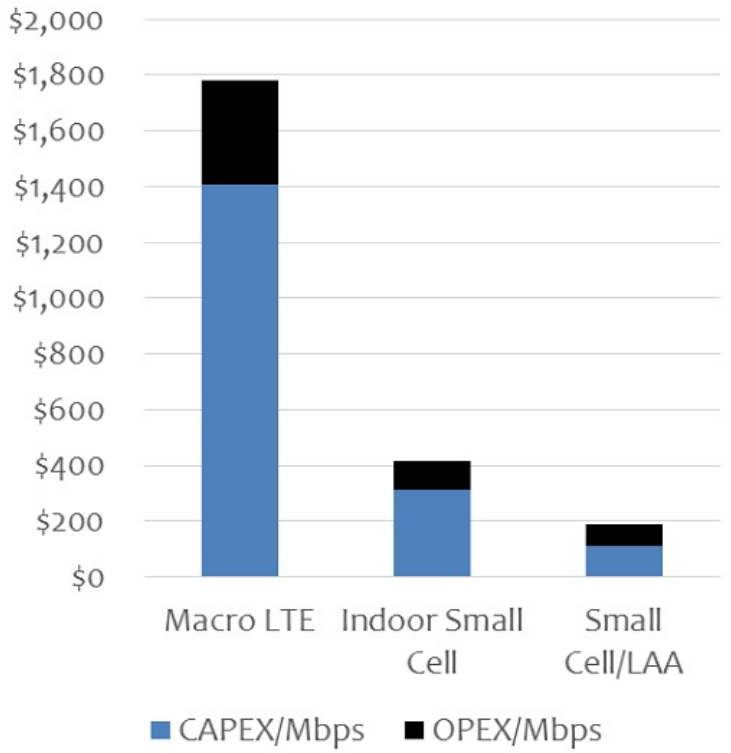

Fig. 4.11 CAPEX/OPEX price depending of the cell features [30]

The implementation of the C-RAN is the most cheaper one in both sense CAPEX/OPEX. The price reduction on macro cells will be cheaper as well but are not in condition to establish a number.

This price can change depending of the infrastructure requirements, own infrastructure is implemented using common hardware providing a cheaper deployment [31] [32]. For a more massive deployment and guarantee a minimum quality of service it is necessary to use more specific hardware as optical fiber, FPGA, DSP, etc The use of this devices increase a lot the final price of the infrastructure.

CONCLUSIONS

This thesis has presented a software platform that represents an attempt to approximate the ETSI MANO infrastructure to a wireless world.

This project shows that C-RAN can improve the network performance, implement several standards in one and manage it in a clean and easy way. Furthermore, new business model can be implemented under this infrastructure where TSPs benefits from it avoiding the maintenance cost of the infrastructure.

The results presented shows that, C-RAN is a new green network infrastructure where TSPs do a better use of the radio spectrum sharing it between them and just paying for the used time. Besides, sharing the infrastructure allows making a deployment with less base stations saving power consumption without reducing the quality of service.

The VIM implementation has problems with the throughput which can only be solved using optical fiber between the connections from data center to the RRHs. However, from the computation part of view OpenStack as a VIM present several features in order to improve the resource management, being a very versatile tool for several scenarios.

GPPs have a good performance and are enough flexible in order to run LTE signals but for the new mobile standard 5G will be necessary to introduce advanced hardware.

OOCRAN has proved that it is possible to make a better usage of the radio spectrum, but in this project we have just implemented a first step which is introduce dynamism to the infrastructure. The dynamism makes the infrastructure more flexible making it easier to manage and deploy cognitive infrastructures. Furthermore, with the design of the platform it will be easy to implement more advanced algorithm like massive MIMO or advance CoMP features as interface cancellation.

The cost implementation and maintenance of this kind of infrastructure are cheaper than the traditional ones if we use GPPs. Thereby it will be InP job to establish a competitive prize in the market. We make a proposal for LTE LAA small cells. In this project we can observe that the OPEX prize by the TSP is much lower compare with the traditional ones.

Finally we can say that we have accomplished all the objective proposed in this project, however, the OOCRAN platform can offer more features than the actual ones, thus there is room for improvement in this topic.

FUTURE WORK

As stated before, this project is merely a first attempt of implement a C-RAN platform, thus there are many open avenues exploration related with this thesis. In this section we will discuss some avenues that can be implemented in the OOCRAN platform that are currently under study.

The first one is the resource management, we have selected a configuration according to our own requirements but exist a lot of possible configurations. There are many studies about this topic that search for the best configuration, saving power consumption without waste resources using the configuration that yields best performance.

The implementation of the base station is the simplest one, for that reason, complex signal processing techniques will be difficult to implement without the use of multithreading and multicore features, the investigation on this topic are a trending topic. Nowadays, there are several studies about how to share the resources of the advanced hardware for example the FPGA in the cloud [32].

The dynamism of the mobile infrastructure present a good point of investigation because there a lot of investigation in that field searching for a better usage of the radio resources: virtual cells, CoMP algorithm, massive MIMO, etc All this fields will be more easy to implement in a C-RAN due to that centralized administration [33] [34] [35].

In our own infrastructure we have just implemented the physical layer of the communication system, which is the most import from the wireless point of view but it would be interesting to implement a complete base station. The implementation of the backhaul and fronthaul in the cloud could be problematic because of the strict real time requirements of the LTE, making a good point of investigation.

One important point that we have avoided in this thesis for his complexity is the communication between NVF. The management of this communications will be done using Software Defined Network (SDN) which allows a easy reconfiguration of the routing path which guarantes a minimum quality of service. This minimum quality of service could be achieved implementing advanced network administration such as Multiprotocol Label Switching (MPLS).

There are a lot of topics to study about because there are a lot of problematics that are not solved yet, for that reason, there are a few implementations of the C-RAN in a real work. Despite the problems, it is expected that for the new mobile communication technology 5G, C-RAN becomes a reality.

BIBLIOGRAPHIC

[1] Dawson, Alexander William, Mahesh K. Marina, and Francisco J. Garcia. "On the benefits of RAN virtualisation in C-RAN based mobile networks." *2014 Third European Workshop on Software Defined Networks*. IEEE, 2014

[2] Software-Defined Radio - Wipro Technologies

[3] What is Software Defined Radio - http://www.wirelessinnovation.org

[4] Kivity, Avi, et al. "kvm: the Linux virtual machine monitor." *Proceedings of the Linux symposium*. Vol. 1. 2007.

[5] Chen, Clark. "C-RAN: The road towards green radio access network." *White paper* (2011).

[6] Wu J, Zhang Z, Hong Y, Wen Y. Cloud radio access network (C-RAN): a primer. IEEE Network. 2015

[7] OpenLTE project - https://github.com/osh/openlte

[8] GNU Radio - http://gnuradio.org/

[9] srsLTE webpage - www.softwareradiosystems.com

[10] srsLTE code - https://github.com/srsLTE/srsLTE

[11] USRP N210 dataseet - https://www.ettus.com/product/details/UN210-KIT

[12] OpenStack project - https://www.openstack.org/

[13] ETSI GS NVF-MAN 001 v1.1.1 - ETSI

[14] OPNFV project - https://www.opnfv.org/

[15] OpenMANO project (telefonica) - http://www.tid.es/long-term-innovation/network-innovation/telefonica-nfv-reference-lab/openmano

[16] GOHAN - http://gohan.cloudwan.io/

[17] Open Air Interface - http://www.openairinterface.org/

[18] OOCRAN project - https://github.com/howls90/master-thesis

[18] Django project - https://www.djangoproject.com/

[20] MVC architecture - https://developer.chrome.com/apps/app_frameworks

[21] Nova configuration - http://docs.openstack.org/liberty/config-reference/content/list-of-compute-config-options.html

[22] Nova configuration - https://specs.openstack.org/openstack/nova-specs/specs/mitaka/implemented/libvirt-real-time.html

[23] Nova configuration - https://specs.openstack.org/openstack/nova-specs/specs/juno/approved/virt-driver-cpu-pinning.html

[24] Heat templates - http://docs.openstack.org/developer/heat/template_guide/hot_guide.html

[25] Link budget example - https://sites.google.com/site/lteencyclopedia/lte-radio-link-budgeting-and-rf-planning

[26] Measuring ACLR performance in LTE transmitters - Agilent Technologies

[27] FRequency reuse - https://www.assignmentexpert.com/blog/wave-interference-in-the-cellular-network/

[28] How to dimension user traffic in 4G networks (Leonhard korowajczuk) - CelPlan internacional, Inc

[29] Downlink bit rates - http://www.lte-bullets.com/LTE%20in%20Bullets%20-%20DL%20Bit%20Rates.pdf

[30] Madden: How Wi-Fi will save LTE ... And how LTE will save Wi-Fi - http://www.fiercewireless.com/tech/story/madden-how-wi-fi-will-save-lte-and-how-lte-will-save-wi-fi/2015-08-28

[31] 0.1 cent per MB: Ensuring future data profitability in emerging markets - Arne Jeroschewski, André Levisse, Alexandre Ménard

[32] Virtualizing FPGAs for Cloud Computing Applications - Stuart A. Byma]

[33] FutureWorks (looking ahead to 5G) - Nokia

[34] Chih-Lin I, Huang J, Duan R, Cui C, Jiang JX, Li L. Recent progress on C-RAN centralization and cloudification. IEEE Access. 2014;2:1030-9.

[35] Haque, Md Mejbaul, Mohammad Shaifur Rahman, and Ki-Doo Kim. "Performance Analysis of MIMO-OFDM for 4G wireless systems under Rayleigh Fading channel." *International Journal of Multimedia and Ubiquitous Engineering* 8.1 (2013): 29-40.

ANNEX

Annex 1. Network creation template

```
heat_template_version: 2013-05-23

description: Sample one mobile network deployed on the EETAC

resources:
  bar_canal_net:
    type: OS::Neutron::Net
    properties:
      name: bar_canal_net

  bar_canal_subnet:
    type: OS::Neutron::Subnet
    properties:
      network_id: { get_resource: bar_canal_net }
      cidr: 10.0.0.0/24
      gateway_ip: 10.0.0.1
      allocation_pools:
        - start: 10.0.0.3
          end: 10.0.0.254

  bar_router:
    type: OS::Neutron::Router
    properties:
      name: bar_router
      external_gateway_info:
        network: 0e67e979-6fb8-485a-923f-1c5d57351e76

  bar_canal_usrp_interface:
    type: OS::Neutron::RouterInterface
    properties:
      router_id: { get_resource: bar_router }
      subnet_id: { get_resource: bar_canal_subnet }

  bar_bts_net:
    type: OS::Neutron::Net
    properties:
      name: bar_bts_net

  bar_bts_subnet:
    type: OS::Neutron::Subnet
    properties:
      network_id: { get_resource: bar_bts_net }
      cidr: 20.0.0.0/24
      gateway_ip: 20.0.0.1
      allocation_pools:
        - start: 20.0.0.2
          end: 20.0.0.254
```

```
bar_canal_bts_interface:
  type: OS::Neutron::RouterInterface
  properties:
    router_id: { get_resource: bar_router }
    subnet_id: { get_resource: bar_bts_subnet }
```

Annex 2. NFV allocation and configuration

```
heat_template_version: 2015-10-15
description: None

resources:
  Bts1:
    type: OS::Nova::Server
    properties:
      name: <name>
      image: <VNF name>
      flavor: <flavor>
      networks:
      - network: <network>
      user_data_format: RAW
      user_data: |
        #cloud-config
        runcmd:
         - <script code>
         - sh /home/nodea/start.sh

  ...

  Channel1:
    type: OS::Nova::Server
    properties:
      name: <name>
      image: <VNF name>
      flavor: <flavor>
      networks:
      - network: <network>
      user_data_format: RAW
      user_data: |
        #cloud-config
        runcmd:
         - <script code>
         - sh /home/nodea/start.sh
   ...
```

Anex 3. Cost per bit calculs

All results are expresed in eur*h

BW = 3MHz
111,4/44 = 2,48 #euros per core
2,48*0,35 = 0,886 #euros per used core
3*18,9/20 =2,83 #euros by radio usage
0,886+2,83/4,3 =1.54 #QPSK
0,886+2,83/8,6 =1.21 #16QAM
0,886+2,83/13 =1.10 #64QAM

BW = 5MHz
111,4/44= 2,48
2,48*0,58 = 1,438
5*18,9/20 =4,72
1,438+4,72/7,2 = 2.09
1,438+4,72/14,4 =1.76
1,438+4,72/21,6 =1.65