1. **Introduction**
2. **State of Art and scenario**

2.1. SDR

2.1.1. ALOE

2.2. Cloud Computing

2.2.1. Virtual Machines

2.2.2. OpenStack Architecture

2.3. ETSI MANO

2.3.1. VNF

2.3.2. NVF

2.3.3. VNF Manager

2.3.5. VIM

2.4. Case Study

2.4.1. Cloud Random Access Network

2.4.2. Objectives

1. **Open Orchestration Cloud-RAN (Software Platform)**

3.1. Tenants, roles and grants

3.2. Infrastructures

3.2.1. Physical Infrastructure

3.2.2. Virtual Infrastructure

3.2.3. Virtual link

3.2.4. Physical link

3.2.5. Entity Relationship Model of database

3.3. Orchestration

3.4. VNF Manager

3.4.1. Heat template

3.4.2. Cloud Init

3.4.3. OpenStack API (VNFM-VIM)

1. **Test bed**

4.1. Scenario description

4.1.1. Infrastructure Provider

4.1.2. Telecommunication Provider

4.1.3. Downlink

4.1.4. Uplink

4.2. Frequency assignment on demand

4.3. Dynamic deployment

4.3.1. Study of the upload effect

4.3.2. Study of the reconfiguration procedure

4.4. Wireless elements emulation

4.5. Power Consumption analysis

1. **Conclusions**
2. **Future Work**

**Acronyms**

BT - Base station

SDR - software define radio

ALOE - Abstraction Layer and Operating Environment

PC - Personal Computer

LTE - Long Term Evolution

GSM - Global System for Mobile Communication

SaaS - Software as a Service

PaaS - Platform as a Service

IaaS - Infrastructure as a Service

CPU - Central Processing Unit

USRP - Universal Software Radio Peripheral

API - Application Program Interface

AWS - Amazon Web Service

NASA - Nation Aeronautics and Space Administration

VNF - Virtual Network Function

NVF - Network Virtual Function  
VIM - Virtualized Infrastructure Manager

NS - Network Service

ERM - Entity Relationship Model

TSP - Telecommunication Service Provider

SP - Server Provider

InP - Infrastructure Provider

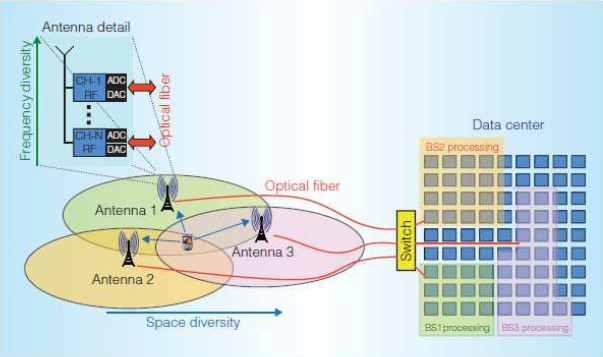
OOCRAN - Open Orchestration Cloud Radio Access Network

1. **Introduction**

Since the appearance of the mobile cellular networks, the infrastructure is designed in a distributed way. Where each server has at least one base station (BT) and signal processing is done over each base station.

This infrastructure provides several advantages such as robustness over failures, easy scaling, etc. But in order to provide a better utilization of the spectrum, increase bit rates, eliminate interference and reduce cost it is needed several techniques which require collaboration between BTs. These techniques require that each BTs has information of the network introducing a high overhead, delays and computational time. Therefore the new generation of cellular networks 5G will make changes on the infrastructure.

The main change will be move the entire or partial signal processing from the BTs into a same physical location. Therefore BTs network will be distributed, but we will move the intelligence (signal processing) from the distributed way to a centralized way. The following image illustrate this idea.



By making these changes, just one entity need to have knowledge about the whole network, allowing collaboration between BTs without introducing high overheads and delays.

1. **State of Art and Objectives**

**2.1. Software Define Radio**

The common radio communication systems are implemented on specific hardware, which are compost by: mixer,filters,amplifier... But Radio Frequency front end and analog-digital converters allows us to process samples into a digital domain.

Software define radio (SDR) is a radio communication system where the components are implemented by software, typically into a common personal computer (PC) or embedded system. That system has two important features. The first one, implementing systems on PC are more cheaper than specific hardware. And the second one, implements systems using software are more flexible than using hardware. For example if we want to do and update of the technology using SDN, it is as simple as just run a different program (one program for GSM and another for LTE). The same operation on hardware will mean buying a new system and configured it.

**2.1.1. ALOE**

ALOE is a framework whose main purpose is administrate the resources of a the PC in order to run SDR application. It works using modules which run in a strict real time requirements and it also is compatible with USRP API.

By the other hand, ALOE can run in multiprocessing and multicore mode. This two modes allow to run SDR systems into a cluster of CPUs without much configurations.

**2.2. Cloud Computing**

Cloud computing is a kind of Internet-based computing that provides shared processing resources (CPU and RAM) 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.

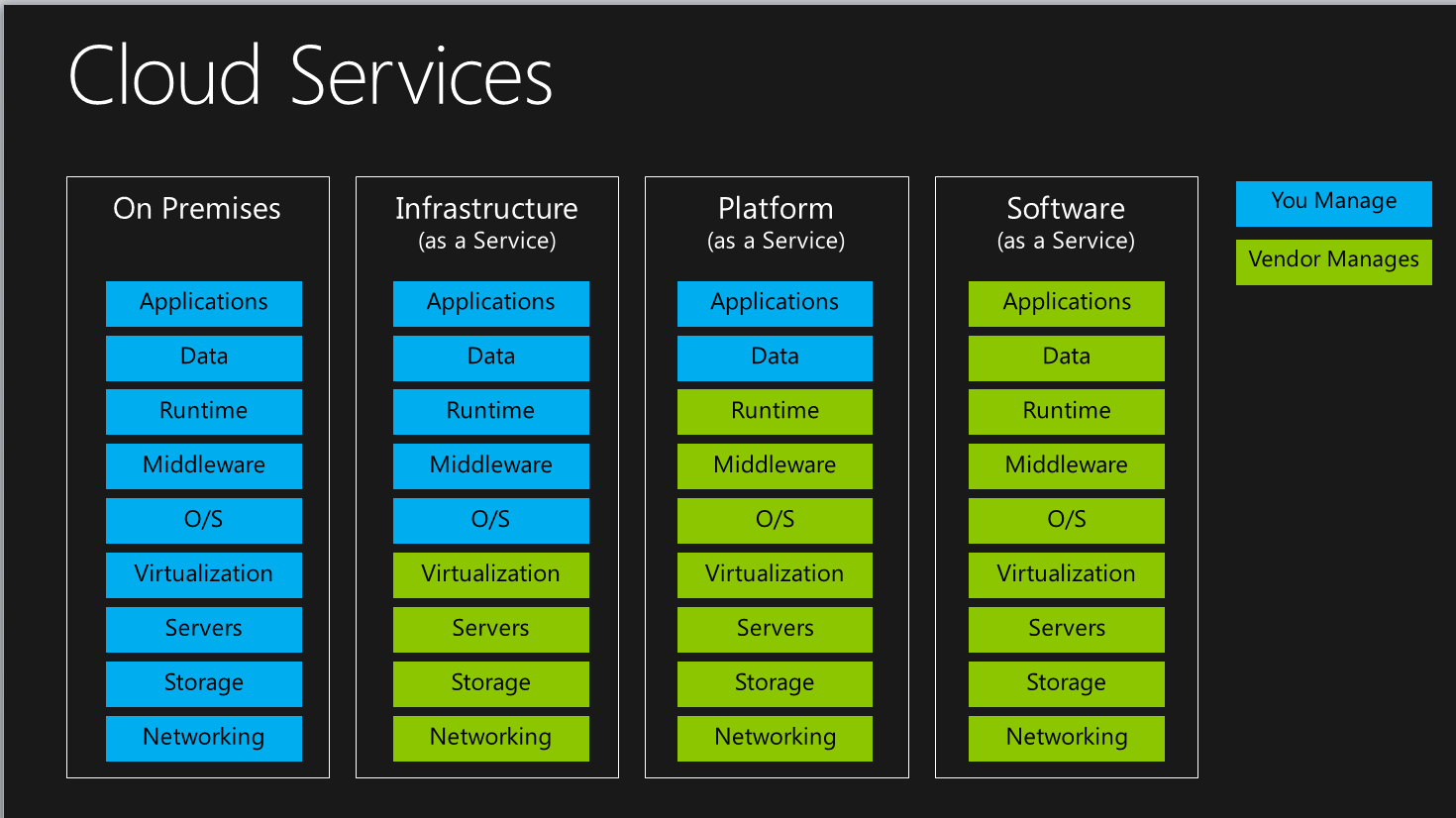
Depending of the grants that the infrastructure provider give to the tenants, we can find three main types of cloud:

- 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 compost 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:



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. But there exist another techniques such as linux containers or bare metal nodes. In the following section, we will talk about how virtualization is done.

**2.2.1. Hypervisor and Virtual Machines**

Computers run applications into the CPU but just can run one application in a specific time. Therefore CPUs run applications according to an specific scheduler which decide what applications must run first. According to differences proposes, we find difference kinds of schedulers. To sum up, scheduler decides the order of the applications that must run and how many time of processing time give to an specific application.

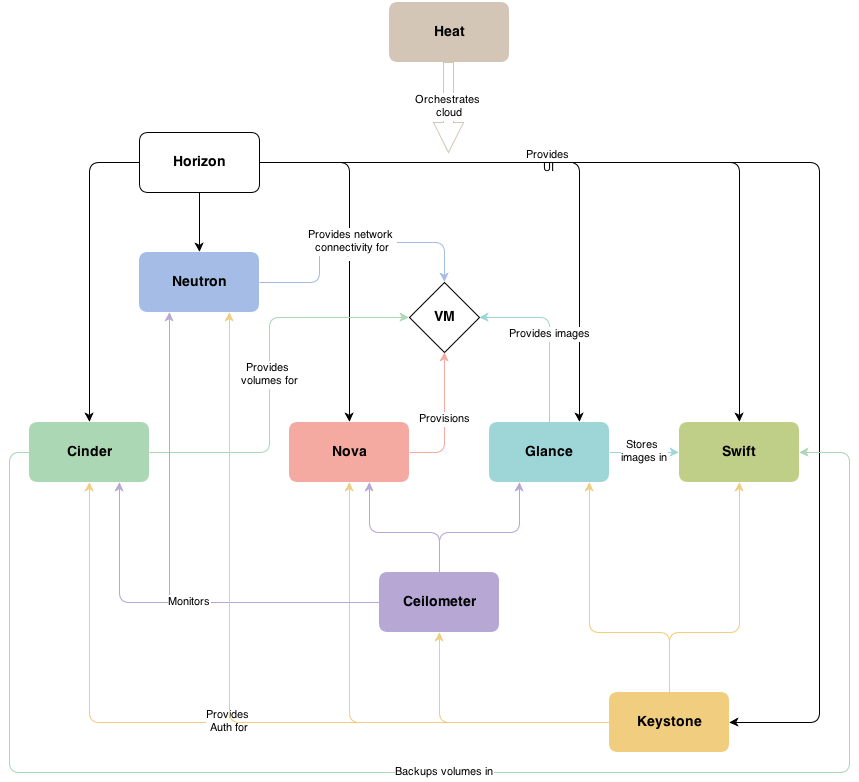
Virtual machines at low level done something similar, each VM reserved a specific processing time of the physical CPU. Another VMs can not run its own application into that computation time. This allow that another VM can not degenerate the performance of another VM. The element that reserved computation time is the hypervisor, which reserve time and create virtual machines with a specific features.

We found different kinds of hypervisors according to software level where they are running. The most common are the native or level 1 which have the best performance. For example VMware, KVM or Xen are the most famous platforms. This hypervisors run close to the hardware. Level 2 and level 3 hypervisors run ups to the several software layers, deteriorating the performance.

**2.2.2. OpenStack Architecture**

With an overview of the common elements of the cloud computing, it is time to enter in the commercial solution platforms. In this section, we will introduce OpenStack which it 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 his own private cloud computing infrastructure, one of the most famous companies are AWS (Amazon web service). Users manage its infrastructure through a web-based dashboard, through command-line or through a RESTful API.

OpenStack is composed by several elements called as services. Each service has a specific intent. An all services are interconnect for improve the managements features of the overall. The following picture show you the interconnection between services:



We will explain the architecture from inside to outside. The core of the platform are the hypervisor, Openstack can run with serveral commercial hypervisor such as KVM, VMware, docker, etc. In order to configure the hypervisor skills for a particular VM, we found Nova service. Nova handle the VMs, manage the life cycle, configure the parameters, etc. VM runs with a particular OS which is provide by Glance. This service is a repository of images to create VM.

The VM can be interconnected composing a network, in this network we found routers, switches, etc, all this elements are manage by Neutron service.

Networks and VM are the elements that users pay for, therefore Keystone service manage the users permissions and allow or deny access to this elements. Its functions are as a middleware.

Users need to manage his own infrastructures. Horizon provide a web server frond end to handle users infrastructures.

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

There more additional service in the OpenStack architecture, but we explain the most commons.

The main intents of each service is:

- Nova: manage the life cycle of the VMs.

- Glance: provide images for run VMs.

- Neutron: manage the networks.

- Keystone: provide security and permissions in the VM.

- Horizon: is a web front end for handle the infrastructure.

- Heat: manage infrastructures.

**2.3. ETSI MANO**

OpenStack is a platform that it has a high impact on the way of deploy servers or big data structures. Due to this, Telecommunications Infrastructure Providers have shown interest in migrating their infrastructure resources, for both wired and wireless services, to the virtual world. 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.

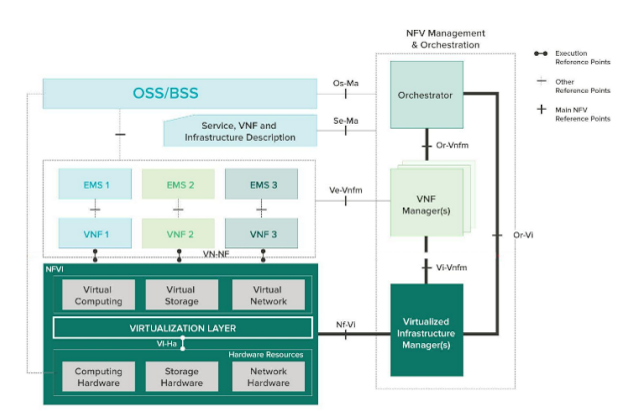
The structure that ETSI designed is composed by three main blocks, where each one has a particular mission.

- Orchestrator: Responsible for on-boarding 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:



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

**2.3.1. VNF**

VNF are virtual machines that mimic the function of an specific hardware in the telecommunications field. This entails a set of advantages and disadvantages. First, some advantages are presented:

- Cost reduction: it is no longer necessary to buy, transport or install new equipment. No physical space required. Migrations are easier and faster in the virtual world. Less management staff needed.

- Simplified deployment: less planning and time required. Multiple solutions proposed to enable reconfiguration or rescheduling.

- This new paradigm encourages innovation as it looks for new solutions to current problems.

- It brings agility to tasks and diminishes the effect of unforeseen inconveniences for the user.

- A much sought benefit, at least for administrators, is the removal of dependence on stock solutions (hardware). Software solutions enhance application personalization and scalability.

- Networks can be more dynamic as their behavior can be tailored.

-Possible operation at connectivity levels 2 and 3 (data and network) and is compatible with traditional networks allowing for a progressive implementation.

Disadvantages:

-Exposed to malicious attacks as a brand new technology.

-A high number of updates as it is yet in development phase.

-Increase in processing power requirements as well as operating with high data sizes. This will drive prices up as new processors are still in research phase.

-Only one point of failure as all the infrastructure is connected in a data center.

**2.3.2. NVF**

VNF is a peace of code that realize a network function into a VM. But this function must be configured for accomplish and specific case. Therefore two VNF can run the same code with a different configurations, in this case when a VNF is running with a specific configurations it is call NVF.

VNF is an object and the NVF is an instance of that object.

**2.3.3. VNF Manager**

It manage the VNF, VNFM does the following:

- Manage life cycle of VNFs (create, maintains and terminates VNF).

- It is responsible of handle: Faults, Configuration, Accounting, Performance and Security to VNFs.

- It scales up/down CPU usage to VNFs.

**2.3.5. VIM**

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

- Manage the life cycle 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 managements of hardware, software and VM.

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

**2.4. Case Study**

The ETSI MANO architecture explained in this documents is still in development, therefore there are not much more available platforms. Besides current platforms are focus on wire communications such as routers, firewalls, proxy ...

Due to that fact, we decides to create own platform using OpenStack as a VIM. Own platform is a simulator/real that orchestrate resources in order to create wireless connections. This kind of platforms are normally called as a Cloud RAN platform.

**2.4.1. Cloud Random Access Network**

Cloud RAN is a platform that orchestrate wireless communications for mobile networks, orchestrating resource in a centralized way allows to introduce new features that improve the performance of the network. C-RAN system, "Clean, Centralized processing, Collaborative radio, and a real-time Cloud Radio Access Network”. Among the many characteristics of the system we highlight the *Collaborative* word, of special importance in 5G networks.

Implement infrastructures on a virtual domain introduce several advantages in front of physical domains:

- Energy efficiency: when designing 5G mobile networks power and energy consumption are paramount factors to consider. As the amount of connected devices is growing more and more and the size of the cells in heterogeneous networks is decreasing (therefore, more Access Points are deployed) a green approach is not only desired, but mandatory. Sleeping modes and the ability to tailor the network’s capacity to the traffic load are vital.

- Cost reduction: obviously one, if not the main, driver for operators is minimizing both CAPEX and OPEX. C-RAN can contribute to OPEX reduction through centralized management and operation, avoiding new unnecessary base station deployments. Also, a reduction in power consumption and, therefore, cooling will benefit operational costs.

- Capacity improvement: Cloud RAN has an enormous potential to enhance the global performance of the network by easing cooperation between the different BSs. The main limiting factor of capacity is usually inter-cell interference and this can be coped with joint processing and scheduling. Virtual BSs would be able now to share signaling, data or Channel State Information (CSI) with the BBU pool. This paves the way to optimally implementing techniques like Coordination Multipoint (CoMP).

- Adaptability to non-uniform traffic: C-RAN can also bring in new improvements in terms of serving non-uniform traffic demands due to the enhancements in the load-balancing capabilities.

- Smart Internet Traffic Offload: another factor that has to be considered is the continuously increasing web traffic generated by portable devices like smartphones. This can collapse the core network of operators and degrade the quality of other services provided. C-RAN allows for offloading those connections and benefiting them (reduced latency) as well as the overlapping service (not collapsed) and operators (reduced back-haul and core network traffic and cost).

It should emulate the whole BTS or parts for give service to the users in a specific area, configuring each one in order to maximize the performance and reduce the interference of the overall network.

Though the platform, mobile operators rent the physical network for give coverage to his clients. Own platform is designed with this propose but at small scale and just using the physical layer as a BTS.

**2.4.2. Objectives**

Create a platform its allows as simulate or deploy an mobile infrastructure for give service to a wireless users or service providers. This deployment must economize resources such as radio electrical spectrum, power consumption and computational resources.

Platform must be connected BTS with RF front ends for transmit data through the channel. Or in other case, connect BTS to a virtual channel in order to simulate signal degradation of the channel.

In order to reduce interference, orchestrator must assign frequencies with an accurate bandwidth. Using a frequency reuse according to the scenario characteristics. Operators paid for use the spectrum, therefore we introduce cost function according to the spectrum use and computation resources. The scenarios must be deployed according to the bit rate demands, so infrastructure must supply that demands in time.

NVF must be created with a specific computational resource in order to avoid resource’s waste. Almost it necessary to monitor the performance of each one, it allows to prevent failures.

1. **Open Orchestration C-RAN (Software Platform)**

The main point of the project is create a platform that allows manage resources in order to create mobile infrastructure. For that propose we create a web-base dashboard where users can manage its own virtual infrastructures. This platform was deployed using a python framework called Django, release 1.9. We select python as a main programing language because its more easy to send querys to VIM (Vi-Vnfm) than using CuRL.

Before to talk about the architecture of the platform, we want to introduce the main characters, who will use the platform and what grants has each one.

**3.1. Tenants, roles and function**

In this section, we will introduce the client for who we are working and the roles and functions.

We will simulate four main groups:

- Telecommunication Service provider (TSP): are the common mobile providers such as: Orange. Who provide connectivity to user to user o user to Ethernet.

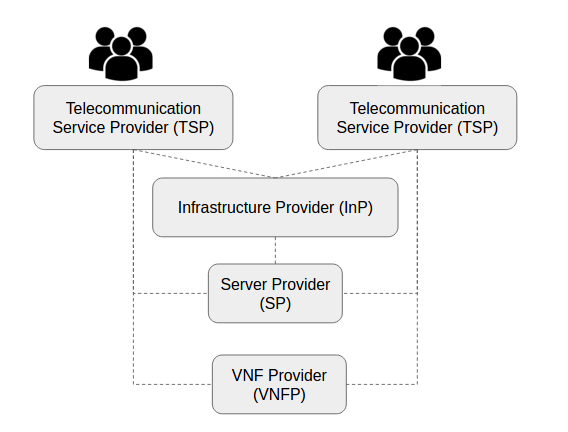
- Clients of the TSP: are the clients who are using the service of TSP (subscribers).

- Infrastructure Provider (InP): are the entity who are renting the infrastructure, in this case RF front ends and the networks.

- Server Provider (SP): provide computing resources in order to run NVFs. Amazon can be an example of SP.

- VNF Provider (VNFP): this entity provide VNF that emulate the all BTS or just parts. This BTS can process LTE, UMTS signals, etc

The following graph illustrate the relationship between the providers.



In this project, we will focus on the tasks of TSP and InP. Despite for emulate the all infrastructure, we take the role of clients, SP and VNFP as well.

**3.2. Infrastructures**

In this section, we will introduce the infrastructure where the platform run.

- Physical Infrastructure: the infrastructure where the platform runs with the specification of each element.

- Virtual Infrastructure: the infrastructure that TSP create for provide services to its own clients.

**3.2.1. Physical Infrastructure**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Controller** | **Compute 1** | **Compute 2** |
| **CPU (GHz)** | 2.37\*8 | 2.37\*24\*2 | 2.37\*24\*2 |
| **RAM (GB)** | 8 | 52 | 52 |
| **Network (GBps)** | 10 | 10 | 10 |

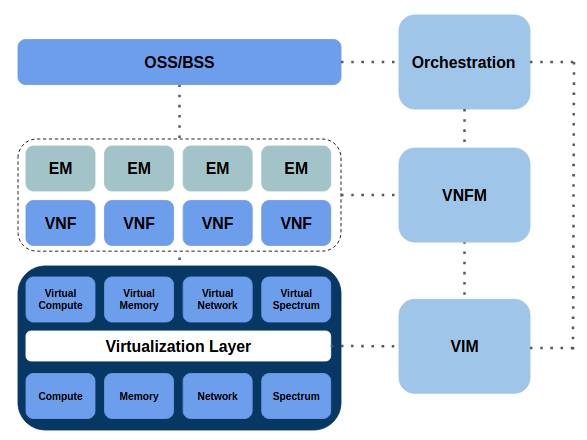
Notes: Base stations with BW of 10MHz generate 10GBps of traffic. For that reason, we will work with BW of 1.4MHz or 3MHz avoiding the bottle neck.

The physical laboratory are interconnected as the image follow:

**physical**

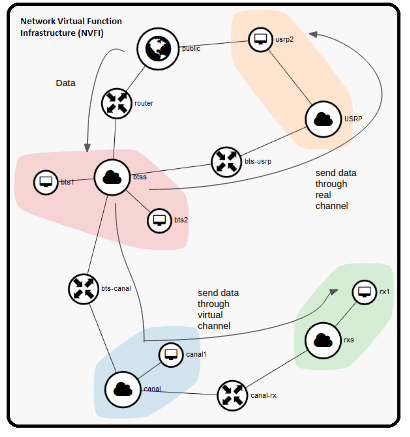
**3.2.2. Virtual Infrastructure**

The program infrastructure is deployed according to the ETSI MANO standard adding a new element for radio managing. This element is a pool of frequencies that VIM will use for administrate, assign and drop frequencies to each NVF. For that reason the most work is done on that element. On the next graph, we are put these concept:



Under this infrastructure multiple TSP will work. For make this explanation more easy, we will make and example for one TSP.

TSP wants to make a deployment on specific area where the InP set up radio frontends. Any frontend is associate with one or multiple NVF (depending of the TSP that are using that specific frontend). In order to send signal to the subcribers of the TSP we will generate a virtual infrastructure with NVF, routers and network according with the figure:



All the elements of the graph are virtual except the public icon (earth) which represent a network NIC with access to the USRPS and Ethernet.

There are fourth networks (clouds) interconnect though routers or switches for each deployment: BTSs, USRPs, Canals, RXs. And multiple NVF (computers)

Depending of the NVF function, the platform attach the NVF in one network or another. As we can see, there are two main roads

- send data though the real channel: that we will use when we want test the platform as a Cloud RAN platform.

- Send data though the virtual channel: that we will use for testing where will will virtualize all elements of the transmission chain (BTs, channel and subscriber)

All this elements together form the network virtual function infrastructure (NVFI) of the deployment.

On the following section, we will describe with more detail each branch of the NVFI.

**3.2.3 Virtual link**

We will start with the virtual one where all elements are virtualized.

The branch is composed by a chain of NVF where each one send information the next one using a UDP connection. We select that protocol because it does not use confirmation for each packet making the transmission more realistic and forced to run on real-time.

Physical layer will send the information to the channel in IQ (phase and quadrat) format.

Channel will receive that samples on the port 8888 and can introduce SNR and delay effect according to the TSP desires. Once it is done channel will send the updated IQ samples to the terminal at port 8888 again. Terminal will demodulate the signal and restore the signal.

**trans virtual**

**3.2.4 Physical link**

On the physical link channel and terminal are real, therefore physical layer must send the information thought a frontend. For that reason we are introduce a new NVF for make the transmission though the front end. The connection with the front end is a dedicated one for each manufacture for that reason we need a specific code for implement the transmission between the NVF and the frontend.

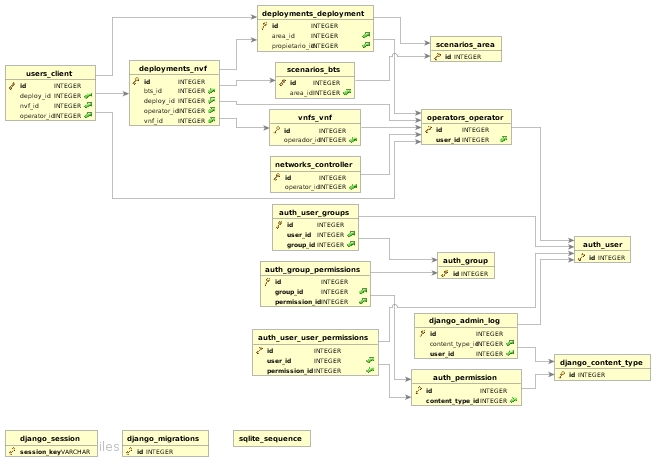
We decided to implement the frontend connection in a new NFV because making that it is possible send signal to one frontend or another with different manufactures without do changes on the physical layer code.

**trasnmitit physc**

**3.2.5 Entity Relationship Model of database**

In order to do an effective administration of the network is necessary to save data in a database. Django provide an easy configuration of several common databases, we decided to work with a SQL database because is a common one and it has a good performance for own proposes.

The tables and his relationships are illustrate in the the figure:

****

Django is a framework that works though “apps”. Each one carry out an specific function and is compose by at least one table but can has several.

We can found 5 app with this tables:

- deployments: Deployments, Nvfs

- operators: Operators

- scenarios: Areas, Btss, O\_Area

- Users: Clients

- VNF: Vnfs

The relationship between them are the following. “Operators” create a “Deployment” in a specific “O\_Area”. This “O\_Area” i composed by several RF frontends (“Btss”). The signal this frontends capture must be process in a “Nvfs” which it is an instance of a “Vnfs”. Finally in each “Btss” there are several subscriber (“clients”) attached.

The “O\_Area” table is particular for each “Operator” but the location of each BT is common for each operator. For that reason, we deployed an auxiliary table “Area” for save the physical configuration.

**3.3. Orchestration**

With the infrastructure defined, we will move to explain the services that the platform (OOCRAN) should be provide with more detail.

First of all, OOCRAN must be able to create and configure virtual and real transmission using LTE. And create a mobile network infrastructure.

TSP will make his own deployments under the physical one, but as a normal mobile planification it is necessary to do an radio frequency reuse for reduce the interference between cells. As the TSP are using the same physical infrastructure interference between TSP must be take into account too. Therefore OOCRAN should provide this frequency reuse between cells and TSP, TSP just will configure the the bandwidth of each BT.

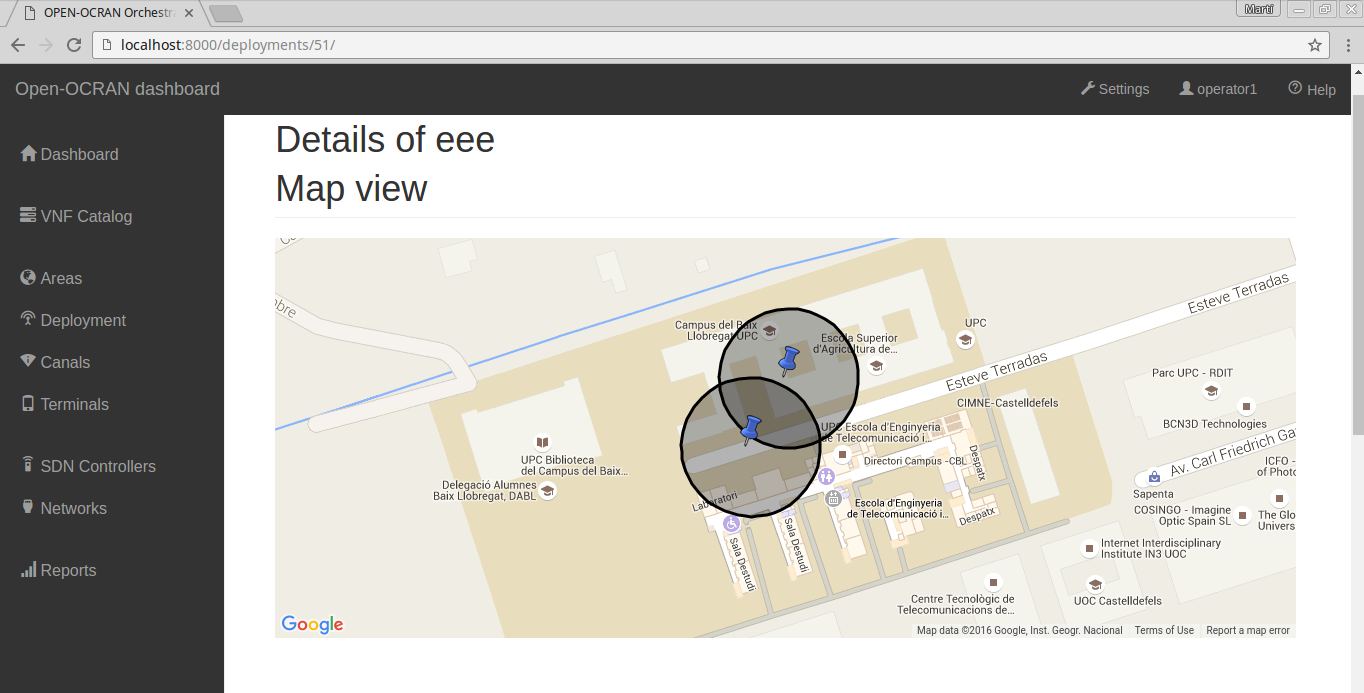
Cloud provide more flexibility to the network, for that OOCRAN will be provide flexibility as well. OOCRAN provide flexibility with tow modes:

- TSP can mutate the infrastructure by his own hand, adding, deleting BTs.

- TSP has a prediction of the bits required, therefore it can define a catalog of infrastructures and OOCRAN will deploy the more suitable one for the conditions of that time.

Finally OOCRAN must provide a monitoring of the all elements of the network such as configuring parameters, resources usage...

All this configurations will be done through a web-base interface, which is more easy than using a terminal.



**3.4. VNF Manager**

The main goal of the VNFM is to configure the infrastructure according to the orchestration desires. Therefore VNFM must define the infrastructure according to the VIM rules. For that propose, we use to applications:

**3.4.1 Heat Templates**

Heat is an easy way to save an infrastructure. This service create an infrastructure according to a yaml file, in this file we can define networks, routers and instance. This file is composed by forth main groups:

- version: indicate the version of the heat service.

- parameters: provide flexibility to the template, tenants can change some parameter without make big changes on the file.

- resources: are the representation of the elements: routers, networks, instances... Each one is a resources with a specific configuration that we will specify in the next steps.

- outputs: it is possible to show parameters values as like a log mode.

We use heat for the creation of the NFVI (figure virtual and real) which represent the networks and the connections between instances. Each NFVI correspond an O\_Area in the database and is unique. In the networks tenants will create instance according to the mobile network requirements. The following template illustrate the structure of the file:

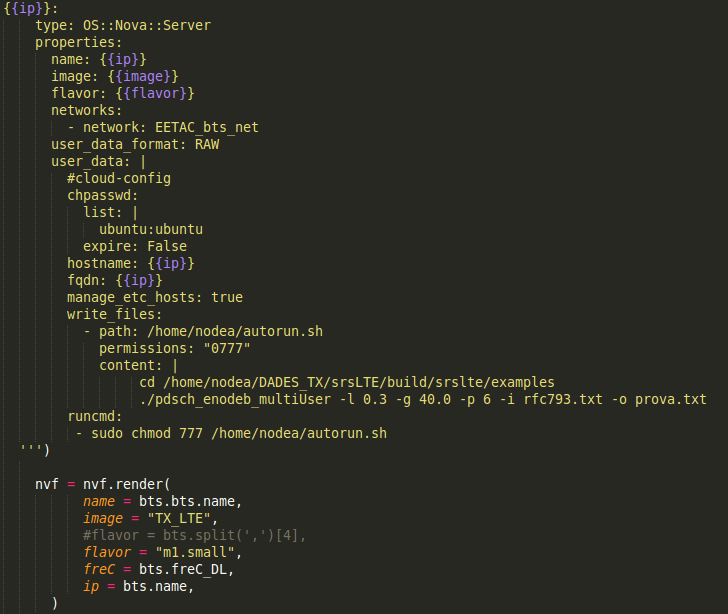
|  |
| --- |
| Heat\_template\_version: 2013-05-23  Description: OOCRAN  Parameters:  Value:  Type:  Description:  Default:  ...  ...  Resources:  Value:  Type  Properties:  Name:  ...  ...  Outputs:  Value:  Description:  Value: |

The HOT template created for the creation of the network is availiable on the appendix 1.

**3.4.2 Cloud Init**

Once the infrastructure is created, it is time to allocate instances (NVF) in the corresponding network according to the configuration for the operator point of view. Each NVF will execute a specific function with a particular configuration, for that reason it is necessary to configure one by one.

For the task, we used Cloud Init. This program allows to us to insert files and run programs in the VM once we launch the instance. The following figure show how the configuration looks like:



In this case, once the instance is in launch cloud init will execute a SDR code that send a txt file through the USRP.

**3.4.3. VNFM-VIM (OpenStack API)**

The VIM is an external platform (OpenStack) therefore it was necessary to configure it using APIs.

As we explain in the section 2.2.2 (OpenStack service) this platform was build in multiples service. In this project, we are work with the following services:

- Nova

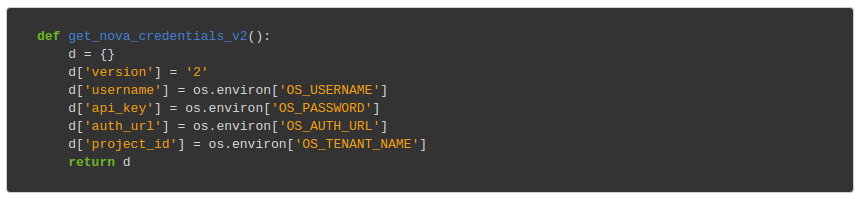
- Ceilometer

- Neutron

- Heat

- Keystone

All services require a authentication process:



In the authentication process, we need to indicate: user name, password, URL of the controller (http://controller:5000) and the name of the project (normally the same name as the user name).

Once the authentication is done, we can send querys to the VIM in order to configuration parameters or launch instance. You can find more information about the APIs use methodology on the OpenStack wiki:

[http://docs.openstack.org/user-guide/sdk\_compute\_apis.html]

1. **Test bed**

**4.1. Scenario description**

In orther to test the platform, we need to play two roles at the same time:

- Infrastructure provider: making a realistic mobile planification allocation RF front around the area.

- Telecomunication provider: renting this compute resources and RF front in that area.

**4.1.1. Infrastructure provider**

We made a plannification for two areas, the university buildings and the bar.

For the RF front end allocation, we decided to set up antennas every 60m. Each antenna has a coverage range around 30m (femtocell).

For make this testing as simple as possible, we made the following approximations:

- Omnidirectional antenna pattern.

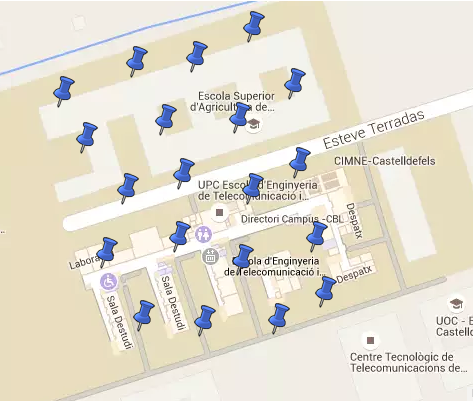
- RF front ends can work with multiple carrier.

- Base station can works as a microcell, femtocell o picocell.

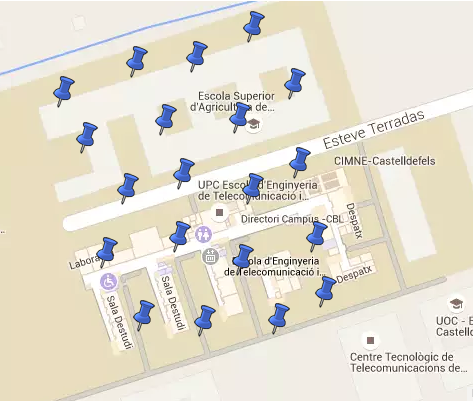
- All subcribers are demanding LTE services with at least 1Mbps.

- Multiples telecommunication providers are deploy infrastructures on the same area and using the same RF front ends.

This images show the antenna allocation of university building and bar:



In the bar scenario the distribution of antennas is the following:



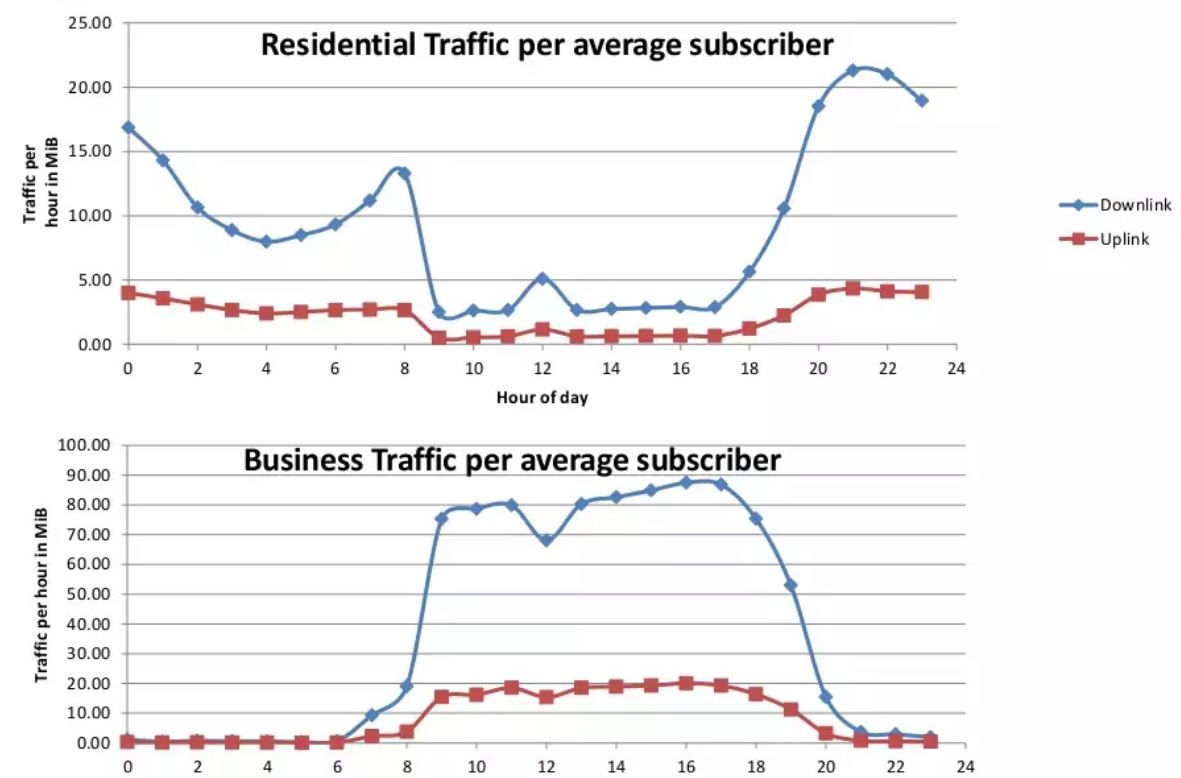
**4.1.2. Telecomunication provider**

Telecom provider will select a group of antennas in the area. Once they decide the most useful antennas for deploy his mobile network, the platform must be create and configure each NVF.

Each TSP will make a deployment on the university and the bar.

The university has a traffic model of a residential area, by the other hand the bar has a traffic model of the business area.

The following graph illustrate the traffic demand in the two cases according to the hour:



For make this test more realist we make a link budget. The planification was made using femtocells.

**4.1.3. Downlink**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| data rate (Mbps) | 1 |  |  |  |
| Transmitter - eNode B |  |  |  |  |
| HS-DSCH power (dBm) | 23 |  | B=360kHz |  |
| TX antenna gain (dBi) | 0 |  |  |  |
| cable loss (dB) | 0 |  |  |  |
| EIRP (dBm) | 23 |  |  |  |
|  |  |  |  |  |
| Receiver - UE |  |  |  |  |
| UE noise figure (dB) | 7 |  |  |  |
| Thermal noise (dBm) | -104,5 |  |  |  |
| Receiver noise floor (dBm) | -97,5 |  |  |  |
| SINR (dB) | -10 |  |  |  |
| Receiver sensibility (dBm) | -107 |  |  |  |
| Interference Margin (dB) | 0 |  |  |  |
| Control channel overhead (dB) | 0 |  |  |  |
| RX antenna gain (dBi) | 0 |  |  |  |
| Body loss (dB) | 0 |  |  |  |
| maximum path loss (FSL) | 130 |  |  |  |
|  |  |  |  |  |
| distance (m) | 31,46217615 |  |  |  |

**4.1.4. Uplink**

|  |  |  |  |
| --- | --- | --- | --- |
| data rate (Kbps) | 64 |  |  |
| Transmitter - UE |  |  |  |
| max TX power (dBm) | 24 |  | B=360kHz |
| TX antenna gain (dBi) | 0 |  |  |
| cable loss (dB) | 0 |  |  |
| EIRP (dBm) | 24 |  |  |
|  |  |  |  |
| Receiver - eNode B |  |  |  |
| Node B noise figure (dB) | 2 |  |  |
| Thermal noise (dBm) | -118,4 |  |  |
| Receiver noise floor (dBm) | -116,4 |  |  |
| SINR (dB) | -7 |  |  |
| Receiver sensibility (dBm) | -123,4 |  |  |
| Interference Margin (dB) | 0 |  |  |
| Control channel overhead (dB) | 0 |  |  |
| RX antenna gain (dBi) | 0 |  |  |
| Body loss (dB) | 0 |  |  |
| maximum path loss (FSL) | 147,4 |  |  |
|  |  |  |  |
| distance (m) | 233,232334 |  |  |

**4.1. Frequency assignment on demand**

Algoritme d’assignacio, com s’assignen i em quin criteri

**4.1.1. Comparative spectrum usage**

Comparativa entre planificacio tradicional y una sota demanda

**4.2. Dynamic deployment**

Com configura les bts perque gastin al minim i el algorime

**4.2.1. Comparative between static and dynamic deployment**

Lo rapid que ens podem adaptar els recursos demanats pels clients

**4.3. Wireless elements emulation**

Enviar atraves del medi virtualitzat i del real a la vegada

**4.3.1. Comparative between emulation and real elements**

Fer una comparacio de lo realistic que pot arribar a ser

**4.4. Power Consumption analysis**

fer presupostos de com fespelegar la infrastructura i del preu comparat am un desplegament tradicional

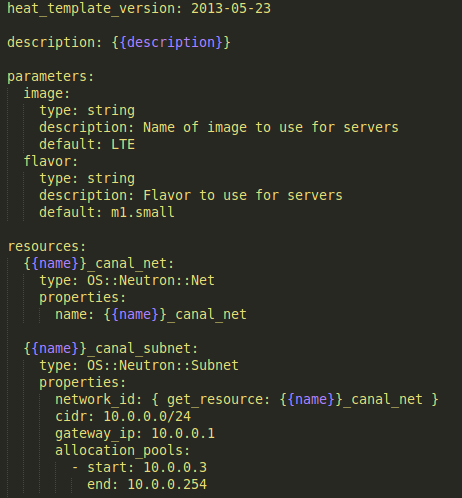
**4.4.1. Model of power consumption**

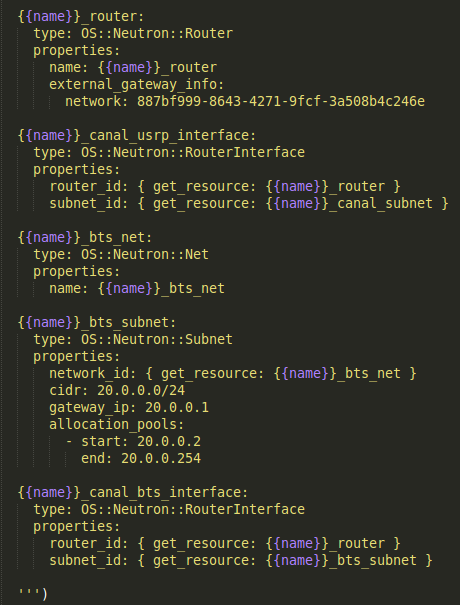
Capex i opex de la infrastructura

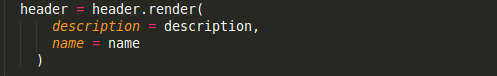
**5. Conclusions**

**Future Work**

**Appendix**







**Reference**

<https://en.wikipedia.org/wiki/Cellular_network>

<https://en.wikipedia.org/wiki/Software-defined_radio>

<https://github.com/flexnets/aloe/wiki/ALOE-Project>

<https://en.wikipedia.org/wiki/Cloud_computing>

<http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_nfv-man001v010101p.pdf>

Service network Systems - Jesus Alcober