



## D2.1 - 5GinFIRE Experimental Infrastructure Architecture and 5G Automotive Use Case

Editor:	Riwal Kerherve, B-COM	
Deliverable nature:	Report (R)	
Dissemination level:	Public (PU)	
Date: planned   actual	30 April 2017	26 May 2017
Version   No. of pages	1.0	113
Keywords:	State of the art, use cases, architecture, 5G, MANO, Automotive, Smart cities	

---

### **Abstract**

This deliverable is the first technical deliverable of the project. It is the result of the project work to establish the baseline of the state of the art in cloud computing, Software Defined Networking (SDN), Network Function Virtualisation (NFV) and Service Function Chaining (SFC). It captures a snapshot of available technologies and components. In addition this deliverable describes the use cases that will be demonstrated later in the project. Finally it describes the first draft 5GinFIRE architecture that will be deployed by the project. This deliverable is the basis for the future work of the project on Experimentation Architecture Tooling, Core MANO Service Management and Orchestration, as well as 5G Infrastructure Integration and Experimentation Enablement.

---

---

## Disclaimer

---

This document contains material, which is the copyright of certain 5GINFIRE consortium parties, and may not be reproduced or copied without permission.

All 5GINFIRE consortium parties have agreed to full publication of this document.

Neither the 5GINFIRE consortium as a whole, nor a certain part of the 5GINFIRE consortium, warrant that the information contained in this document is capable of use, nor that use of the information is free from risk, accepting no liability for loss or damage suffered by any person using this information.

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732497. This publication reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.*



---

## Impressum

---

Full project title: Evolving FIRE into a 5G-Oriented Experimental Playground for Vertical Industries

Short project title: 5GINFIRE

Number and title of work-package: WP2 - Reference Architecture & Internal Use Case

Number and title of task:

- Task 2.1 - SDN, NFV, SFC Components and Technologies: Adopting State of the Art
- Task 2.2 - 5G Automotive Use Case and Beyond: Detailed Requirements and Design Specifications
- Task 2.3 - Experimental Infrastructure Architecture, Model Entities Specification and Design

Document title: 5GinFIRE Experimental Infrastructure Architecture and 5G Automotive Use Case

Editor: Riwal Kerherve, B-COM

Work-package leader: Diogo Gomes, INSTITUTO DE TELECOMUNICACOES

---

## Copyright notice

---

© 2017 B-COM and members of the 5GINFIRE consortium

---

## Executive summary

---

In 5G, virtualisation is a key concept recognised and adopted by the standardisation bodies such as ETSI and 3GPP. Following the high industry interest to exploit the promised benefits of virtualisation at all levels, i.e. compute, storage and networking, a plethora of initiatives have started worldwide to deliver specifications, open source and commercial products that enable the construction, deployment and operation of general virtualised infrastructures.

In the first part of this document we identify and briefly assess the technologies that are available for building the 5GINFIRE experimental infrastructure. We do this for a selected number of popular technologies, which based on the community engagement, are likely to become pillars for the commercial deployment of 5G networks by 2020 and beyond. Obviously an experimental infrastructure should be as close as possible to an anticipated 5G network, which is a moving target today.

The assessment of the technologies for consideration in building the 5GINFIRE experimental infrastructure is based on a set of criteria that are important for a research and innovation project. These criteria are: (i) open source availability that provides us the benefit of a wider community engagement and gives us the flexibility to try out alternative options and features by changing the code, (ii) maturity of the product that provides us confidence to master the technology due to ease of installation and good documentation, (iii) standard compliance that increases the likelihood of the deployed infrastructure is as close as possible to the adopted standards in the area, (iv) available feature set to increase the richness of features of our own infrastructure and finally (v) experience of the partners in the project with certain open source products in order to better manage the learning curve that sometimes can be very steep.

Following these criteria we assess technologies in the areas of general cloud computing, software defined networking (SDN) and network function virtualisation (NFV) which is a wide area and includes NFV management and orchestration, NFV infrastructure and network service descriptors.

The Future Internet Research and Experimentation community has also started initiatives to deploy virtualised network testbeds, as for example in the cases of FED4FIRE and FUTEBOL. We briefly analyse the status of these initiatives.

In the second part of this document we elaborate of the 5GINFIRE use cases that are used as source of requirements for building the infrastructure and to showcase its capabilities. The use cases are positioned in the areas of the automotive vertical sector and smart cities.

In the automotive case the requirements are extracted from scenarios such as sensing-based and video-camera-based assisted driving, which use a multitude of information sources (intra-vehicle, as well as inter-vehicle and vehicle-to-infrastructure) to enable assisted driving. A second scenario is providing event recording services in virtualised multi-stakeholder network environments to showcase function virtualisation of a service that collects important vehicle data and stores them in the network in a tamper-proof way, mimicking the functionality of a “black-box”.

In the smart cities case the requirements are derived from the scenarios to facilitate the use and exploitation of available open data provided by existing sensor deployments in cities, as well as interfacing with the capabilities of the existing deployments of sensor functionality in the testbeds of the project partners in Bristol, Aveiro in Europe and São Paulo and Uberlândia in Brazil.

In the third part of this document the 5GinFIRE architectural approach is described that is based on model entities specification. 5GINFIRE identifies four main actor roles, namely the experimenter, the virtualised function (VxF) developer, the testbed operator and the service administrator. It specifies the experimentation workflow that illustrates the interactions among the four actor roles and exemplifies the interactions through usage scenarios.

In the sequel it specifies the architecture by identifying the internal and external components and their interfaces that are based on existing standards to the extent that such standards exist. A portal is described that supports the experimenter in obtaining access to testbed resources and available functionality, e.g. available VxFs that can be orchestrated and deployed. Finally a number of support services for experimenters are identified and will be deployed in order to increase ease of use of the experimental infrastructure, e.g. helpdesk, tools repository, wiki-based documentation etc.

This document is the basis for the future work of the project on Experimentation Architecture Tooling, Core MANO Service Management and Orchestration, as well as 5G Infrastructure Integration and Experimentation Enablement.

## List of authors and reviewers

---

### List of authors

Company	Authors
B-COM	Riwal Kerherve, Olivier Choisy
INSTITUTO DE TELECOMUNICACOES	Diogo Gomes, Susana Sargento
UNIVERSIDAD CARLOS III DE MADRID	Ivan Vidal, Borja Nogales
UNIVERSITY OF BRISTOL	Reza Nejabati
UNIVERSITY OF PATRAS	Spyros Denazis, Christos Tranoris
UNIVERSIDADE FEDERAL DE UBERLANDIA	Flavio de Oliveira Silva
UNIVERSIDADE DE SAO PAULO	Rubens Mendonça
TELEFONICA INVESTIGACION Y DESARROLLO SA	Diego R. Lopez
EURESCOM	Halid Hrasnica, Anastasius Gavras

### List of reviewers

Company	Reviewers
EURESCOM	Anastasius Gavras
EASY GLOBAL MARKET SAS	Franck le Gall

# Table of Contents

---

- Executive summary ..... 3
- List of authors and reviewers ..... 5
- Table of Contents ..... 6
- List of figures ..... 8
- List of tables ..... 10
- Abbreviations ..... 11
- 1 Introduction..... 13
- 2 Adopting State of the Art on Cloud, SDN and NFV ..... 14
  - 2.1 Objectives ..... 14
  - 2.2 Methodology ..... 16
  - 2.3 Cloud..... 17
    - 2.3.1 VIM Solutions ..... 18
    - 2.3.2 5GinFIRE Candidates..... 20
  - 2.4 SDN ..... 21
    - 2.4.1 SDN Controllers ..... 21
    - 2.4.2 5GinFIRE Candidate(s) ..... 29
  - 2.5 NFV ..... 30
    - 2.5.1 NFV management and orchestration ..... 30
    - 2.5.2 NFV Infrastructure..... 45
    - 2.5.3 Network Service Descriptors: Models and Catalogue solutions for Describing and Configuring service topologies ..... 49
    - 2.5.4 5GinFIRE Candidate(s) ..... 51
  - 2.6 FIRE ..... 55
    - 2.6.1 FED4FIRE and Authentication..... 55
    - 2.6.2 Concepts slice, sliver and Rspecs..... 55
  - 2.7 Conclusions..... 56
- 3 5GinFIRE Use Cases ..... 58
  - 3.1 Introduction..... 58
  - 3.2 Use Cases..... 58
    - 3.2.1 Automotive Domain ..... 58
    - 3.2.2 Smart Cities Domain ..... 67
  - 3.3 Testbeds in 5GinFIRE Use Cases ..... 73
    - 3.3.1 Automotive..... 73
    - 3.3.2 Smart Cities..... 75

- 3.3.3 Additional Testbeds Required in 5GinFire ..... 77
- 4 Experimental Infrastructure Architecture, Model Entities Specification and Design ..... 83
  - 4.1 Introduction..... 83
  - 4.2 5GinFIRE actors and terminology ..... 83
    - 4.2.1 Actors..... 83
    - 4.2.2 Terminology..... 83
  - 4.3 5GinFIRE Experimentation Workflow..... 84
  - 4.4 5GinFIRE experimentation platform usage scenarios and requirements ..... 85
  - 4.5 5GinFIRE architecture..... 92
    - 4.5.1 Internal and external components and their interfaces ..... 92
  - 4.6 Exposed Vxfs and APIs to the experimenters ..... 98
    - 4.6.1 5G-In-A-Box ..... 98
  - 4.7 Testbed components and extensions to support 5G experimentation ..... 99
    - 4.7.1 7.1 5G-In-A-Box ..... 99
    - 4.7.2 Testbed components in the 5TONIC laboratory..... 102
  - 4.8 Resources reservation ..... 103
  - 4.9 Integration and deployment strategy ..... 104
    - Integration..... 104
    - Deployment..... 104
  - 4.10 Experimentation Support components and services ..... 105
    - 4.10.1 Helpdesk ..... 105
    - 4.10.2 Wiki..... 106
- 5 Conclusion ..... 107
- References..... 108

# List of figures

---

- Figure 1 5GinFIRE Experimentation Workflow, Technologies and Infrastructures..... 14
- Figure 2: ETSI's NFV Reference model..... 15
- Figure 3: OP-NFV building blocks ..... 16
- Figure 4: OpenDaylight - Operational view ..... 21
- Figure 5: ONOS subsystem ..... 24
- Figure 6: OpenContrail System Overview [22] ..... 26
- Figure 7: Ryu Architecture..... 28
- Figure 8: NFV-MANO Architecture ..... 31
- Figure 9: OSM Mapping to ETSI NFV MANO ..... 32
- Figure 10: OpenMANO Architecture ..... 34
- Figure 11: Open Baton Architecture ..... 36
- Figure 12: ECOMP Platform..... 38
- Figure 13: Comparison of ETSI MANO and AT&T ECOMP Architectures ..... 39
- Figure 14: Management Layers..... 40
- Figure 15: OPEN-O Architecture..... 40
- Figure 16: Juju in ETSI-NFV MANO Structure ..... 44
- Figure 17: OPNFV Platform Diagram (Release Colorado) ..... 46
- Figure 18: Logical architecture of Ionic ..... 48
- Figure 22: Concepts of slices, slivers [49]..... 56
- Figure 32: Interactions between sensors, vehicles and access points/road side units..... 59
- Figure 33: 5GINFIRE architecture in the ITS and IoT infrastructure ..... 60
- Figure 34: Scenario for assisted driving..... 61
- Figure 35: Emulation of traffic signal devices ..... 62
- Figure 36: Scenario for video-camera-based assisted driving..... 63
- Figure 37: 5GINFIRE architecture in the Smart City and IoT infrastructure ..... 69
- Figure 38: 5GINFIRE IoT Integration Components ..... 70
- Figure 39: Networks and Service requirement for smart cities ..... 71
- Figure 40: Vehicular Network Architecture..... 73
- Figure 41: OBU ..... 74
- Figure 42: RSUs mounted in the municipality building and in a traffic light..... 75
- Figure 43: 5GINFIRE Testbeds for Smart Cities ..... 76
- Figure 44: BiO Infrastructure..... 79
- Figure 45: FIWARE IoT Platform ..... 81
- Figure 46: FIWARE IoT General Infrastructure ..... 81



Figure 47 : 5GinFIRE experimentation workflow overview..... 84

Figure 48: Use cases diagram ..... 86

Figure 49: Architectural approach of 5GinFIRE ..... 92

Figure 50: b<>com \* Unifier GW \* functional Architecture Overview ..... 99

Figure 51: 5G-In-A-Box deployed as a PNF locally to testbed ..... 101

Figure 52: b<>com \* Unifier GW \* deployed as a set of VNF locally to testbed..... 102

Figure 53: 5GinFIRE environment support organization ..... 105

Figure 54: Support basic process flow..... 106

## List of tables

---

Table 1: SDN Controllers Comparative Analysis..... 29

Table 2: NFV Comparative Analysis..... 52

Table 3: Testbed VIM version..... 104

## Abbreviations

---

5G : *5th Generation*

AGPL : *GNU Affero General Public License*

API : *Application Programming Interface*

BSS : *Business Support System*

CI/CD : *continuous integration/continuous deployment*

DCU *Data Collection Unit*

DSRC : *Dedicated Short Range Communication*

ETSI : *European Telecommunications Standards Institute*

EVI : *Experimental Vertical Instance*

GPL : *GNU General Public License*

HA : *High Availability*

HOT : *Heat Orchestration Template*

IF-MAP : *Interface for Metadata Access Points*

ISG NFV : *Industry Specification Group for Network Functions Virtualization*

KVM : *Kernel-based Virtual Machine*

MAAS : *Metal as A Service*

MANO : *management and orchestration*

NBI : *Northbound Interface*

NETCONF : *Network Configuration Protocol*

NFV : *Network Function Virtualization*

NFVI : *NFV Infrastructure*

NFVO : *NFV Orchestrator*

NoSQL : *not only SQL*

NSO : *Network Service Orchestrator*

OBU : *On Board Unit*

OP-NFV : *Open Platform for NFV*

OS : *Operating System*

OSGi : *Open Services Gateway initiative*

OSM *Open Source Mano*

OSS : *Operations Support System*

OVSDB : *Open vSwitch Database Management Protocol*

PNF : *Physical Network Function*

PXE : *Preboot Execution Environment*

REST : *Representational state transfer*

RO : *resource orchestrator*

RSU : *Road Side Unit*

SBI : *Southbound interfaces*

SDN : *Software Defined Network*

SFC : *Service Function Chaining*

TOSCA : *Topology and Orchestration Specification for Cloud Applications*

VIM : *Virtual Infrastructure Management*

VNF : *Virtual network function*

VNFM : *VNF Manager*

VXLAN : *Virtual Extensible LAN*

---

# 1 Introduction

---

5GinFIRE is an EU H2020 project that aims to build and operate an Open 5G NFV based reference ecosystem of Experimental facilities. It intends to allow 5G NFV-based architectures for verticals industries and enables experimentations through Open Calls. Their objective will be to deploy applications on top of the provided ecosystem.

To do this, one of the 5GinFIRE ambitions is to interact with other related projects from EU H2020 like FIRE projects where some of the partners are already involved. It is the case also for the FI-PPP with the FIWARE platform and 5G-PPP.

Indeed, synergies are quite straightforward with FIRE, which is an initiative that offers the possibility to experiment with networks, infrastructures and tools in a multidisciplinary test environment. Interactions with FIRE during the 5GinFIRE Open Calls will be a real asset. Regarding the FIWARE platform, it is already integrated in some of the 5GinFIRE testbeds.

At the same time, 5GinFIRE aims to design a 5G experimentation reference platform that can support vertical industries across the same platform. In this context, 5GinFIRE project will target the 5G Automotive Vertical sector as pilot as well as environmental monitoring in a SmartCity context. Nevertheless the platform will be engineered as generic as possible in order to support any vertical EVIs.

To answer that challenge, this deliverable describes the foundation stones that will drive other work packages until the end of the project.

Indeed in the section 2, it is provided the state of the art concerning Cloud, SDN, NFV and SFC components and technologies. Keeping the objective to build an architecture that is aligned with on-going standardization process, this section highlights software and technologies that will be adopted within 5GinFIRE. Two FIRE projects, which are candidate projects to interact with, are detailed in this section.

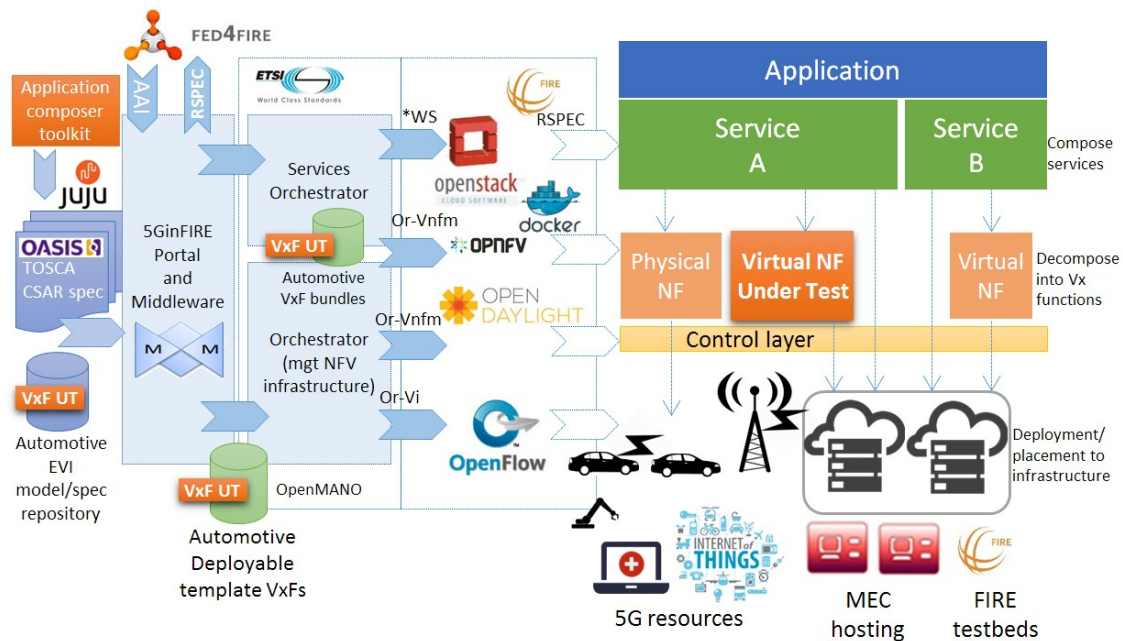
The section 3 focuses on use cases and on testbeds. Two use cases that can use 5GinFIRE testbeds facilities have been identified. First one is an automotive use case and the second one is an environmental use case. Both use cases leads us to identify the requirements for the 5GinFIRE experimentations hosting.

Based on two previous sections, the section 4 defines the architectural choice, the integration and deployment strategies that the project will follow.

## 2 Adopting State of the Art on Cloud, SDN and NFV

### 2.1 Objectives

The objectives of this section are to identify the State of the Art technologies that will implement the methodology and experimentation workflow as defined in 5GinFIRE DoW displayed in Figure 1



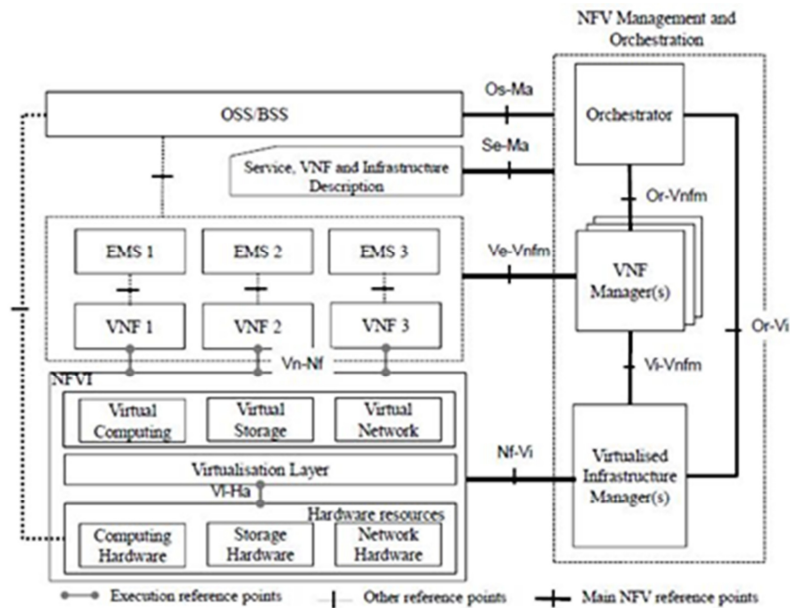
**Figure 1 5GinFIRE Experimentation Workflow, Technologies and Infrastructures**

Our approach takes into account the following high-level requirements and ambitions:

- Adopt and being interoperable with current Cloud/SDN/NFV/MEC standards
- Use open standards and integrate technologically mature, and widespread open source toolsets
- Put effort to enable experimentation and make it effortless for experimenters to deploy experimentation scenarios
- Being interoperable with FIRE standards and facilities

5GinFIRE objective was to adopt the ETSI architectural recommendation for services that are being considered for NFV and being implemented under the NFV model, augmented for application/service composition and experimentation capabilities. In order to instantiate experimentation scenarios end-users will use the 5GinFIRE portal as well as tools to design/deploy the experiment

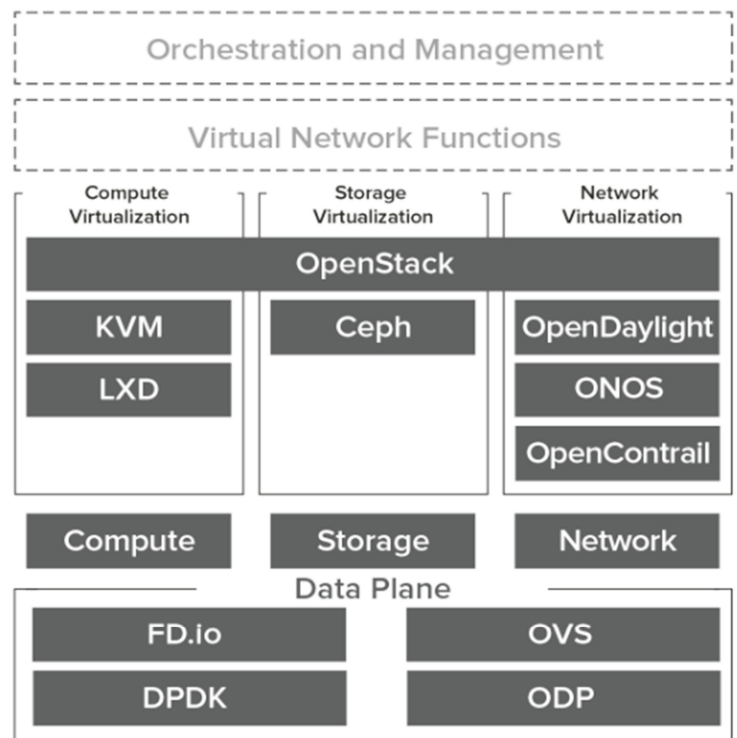
5GinFire testbed shall reflect what a 5G network could be, in accordance with ETSI's NFV reference architecture (Figure 2).



**Figure 2: ETSI's NFV Reference model**

5GinFire target is to provide an implementation of this architecture and one objective of this document is to provide a state of the art of the candidate building blocks. Such building blocks shall be selected in a list of carrier grade and state-of-the-art open source solutions. A possible reference can be OP-NFV recommendations which are providing the following set in its third "Colorado" release:

- Operating Systems: Linux (Ubuntu & CentOS)
- Hypervisor: KVM, tuned for VNFs support
- Virtual Infrastructure Manager: OpenStack
- SDN Controller: OpenDaylight, ONOS, OpenContrail
- VNF Orchestrator: not yet addressed



**Figure 3: OP-NFV building blocks**

OP-NFV recommendations are very valuable; those building blocks (Figure 3) shall as well fit a set of criteria that matter to us. We will take care on defining and using them when providing the state of the art.

The Objective chapter will provide the set of criteria and the state of the art will be split in 4 main chapters:

- A Cloud chapter where different VIMs will be evaluated,
- A SDN chapter on which we will describe 4 different SDN controllers,
- A NFV chapter that will describe the set of tools that is used to do and manage the service chaining like some orchestrators, some deployments tools, etc.
- A FIRE chapter that describe and compare the projects FED4Fire and Futebol as testbed.

## 2.2 Methodology

We targeted some technologies from the most popular that can be found nowadays and we will compare them with each other. To that end, we identified some criteria the technologies we want to use in 5GinFIRE should meet.

### Open source:

Open source is one key aspect of the technologies we want to choose, shall be. But we must also pay attention to the license type (Apache, GPL, AGPL, etc.) because some of the included conditions can determinate how our contributions will be protected or not and many other matters. The frequency of contributions is a good signal to judge if software is in good health and well maintained. Measuring the number of contributors and looking at strong contributors coming from industries are other major aspects that we will consider when making our choice.



**Maturity:**

We want to minimize our efforts on maintenance or on development and this can be carrying out by looking at the maturity of each controller. We want to be aware if they are carrier class, if software is well packaged to ease its installation, but also if the documentation is available, well described and up to date.

**API and standard compliance:**

Commonly speaking, every equipment or technology is interfaced with other equipment's on its northbound or southbound interfaces then we should inspect the supported APIs or standards for each of them.

For instance, on SDN state of the Art, we will ensure on the southbound interface, it is OpenFlow compatible and check the version it is supporting. Due to the disparity in switch equipment's, it would also help if it supports other standard like Netconf, Yang or TOSCA. On its northbound interface, we will confirm it is "Open Source MANO compliant" and examine other orchestrators like Open-O.

**Feature set:**

Every Equipment or technology has its own supported features or plugins and some can be interesting for the project and other less. We will check them out and enumerate the ones which fit the most to our needs; it will help on making our choice.

In the case of the SDN, we can also check the complexity of developing and integrating a plugin in the SDN controller we evaluate.

**Capacity to be mastered by partners:**

In 5GinFIRE project, every partner has their own fields of expertise, we are coming from different areas: research, education or the industries. For instance, some of us might have already a lot of experience with one component and wish not to spend too much effort on learning a new one doing the same thing. We will deem taking this into account in the choice we will make.

## **2.3**      *Cloud*

---

Cloud Computing takes a very important role in 5G due to the move from dedicated hardware to software based network elements. Cloud Computing is therefore a vital technology that supports the 5G infrastructure and functionalities and that can ultimately define the success of 5G deployments.

Cloud Computing solutions range from public offering such as Amazon Web Services (AWS), Google Compute Engine, Microsoft Azure, to name a few, to private clouds instantiated in-house through software packages provided by leading software developers such as VMWare, Mirantis, Ericsson. Supporting these offerings are many times Open Source software solutions that can be customized by companies such as OpenStack, OpenNebula and Eucalyptus or built upon such as Docker, Kubernetes, XenServer.

The previous solutions provide either an ecosystem of resources (Processing, Storage, Network, IAM, etc.) or a single resource (Processing in the case of Docker, Kubernetes and XenServer). In order to deploy a service on top of these resources it is necessary to orchestrate those resources. In some

cases it is even necessary to connect resources belonging to different service providers and/or software providers.

Virtual Infrastructure Management (VIM) is therefore a requirement for a successful and scalable Cloud resource usage.

### **2.3.1 VIM Solutions**

---

VIM (Virtual Infrastructure Management) Solutions enable the network operator to make use of the Cloud Resources (Processing, Storage and Network) to build and/or instantiate their own network elements. The VIM solution has a strong impact in the overall 5G architecture as it will enable or limit many of the foreseen 5G functionalities.

In this section we analyse 3 solutions, chosen due to their popularity in the VNF community

#### **2.3.1.1 OpenStack**

##### **Excellence**

OpenStack is the most prominent Open Source project on Cloud Computing. OpenStack consists of an ecosystem of several smaller projects that manages compute, storage, networking and various support functions such as Identity, Orchestration, and Monitoring.

OpenStack adopts Apache 2.0 license, which allows for the use the software for any purpose, to distribute it, to modify it, and to distribute modified versions of the software, under the terms of the license, without concern for royalties. This licensing enables commercial companies to profit from OpenStack results and to include OpenStack on its products and services.

##### **Evaluation**

OpenStack is a very mature project with a governance model laid out on top of the OpenStack Foundation. This governance model is very important as there are more than 60k contributors many of which financed by various IT companies worldwide. With such many contributors, the vitality of the project is undeniable.

OpenStack is currently on its 14th release and 7 years have gone by since the initial release codenamed “Austin” in 2010. In these years, OpenStack has kept steady biannual release cycles which receive support from the community for periods ranging one year. The project claims to be carrier grade, with many carriers advertising the use of OpenStack in their portfolios, nonetheless several the burden of running OpenStack is very high and requires dedicated and highly trained technicians. Documentation is extensive but spread around various locations and versions, which in turn may difficult the adoption.

OpenStack has come to dominate the Private Cloud much the same way as AWS dominates the Public Cloud. In an area where defacto standards dominate the market, is widely accepted that OpenStack API is the most supported API in the industry.

OpenStack is extremely feature rich, which means that the number of features supported by OpenStack far exceed the needs of 5GinFire.

5GInFire requires a VIM solution that can be responsible for controlling and managing the NFV infrastructure (NFVI) compute, storage, and network resources, these easily map to OpenStack components Nova, Cinder and Neutron.

In 5GInFire, several partners are involved in OpenStack which makes it a very good candidate to be adopted by the project, as these partners have the right expertise to support OpenStack in the scope of the project.

## Summary

We enumerate the main features of OpenStack:

- Prominent open-source project with Apache 2.0 license.
- Mature project with strong community
- Well documented
- Feature rich

### 2.3.1.2 *OpenVIM*

#### Excellence

OpenVIM is the VIM solution of Open Source MANO (OSM), an open source project that implements ETSI NFV architecture. OpenVIM enables all-in one installation and is fully integrated with the other OSM components such as VNF Configuration & Abstraction and the Resource Orchestrator. OpenVIM is therefore the reference VIM in OSM and provides Enhanced Platform Awareness (EPA).

OpenVIM is released with Apache 2.0 license, similarly to OpenStack. OSM (which OpenVIM is part of) governance is help by ETSI and has the support of the European Telecommunication Industry.

#### Evaluation

The project is currently in its first release (Release ONE), although it had existed before being contributed to ETSI. This means that there are various previous versions, but that the project is mostly very recent.

OpenVIM is mostly a shoehorn VIM solution to demonstrate MANO concepts. Even in the scope of OSM, OpenVIM is just one of the possible VIM solutions, being OpenStack the other most prominent VIM. Nevertheless, OpenVIM integrates perfectly with the remaining OSM components and makes it the obvious solution when all-in-one MANO solutions are desired (such as for development purposes).

Feature wise, OpenVIM provides just about what the minimum requirements of a VIM in the scope of OSM. Choosing OpenVIM as a VIM solution should be tightly coupled with choosing OSM as the overall MANO solution, and even then, one should consider the limitations of the all-in-one approach of OpenVIM.

Since in 5GInFire at least one partner is involved in OSM, OpenVIM can be fully supported and made an adequate candidate for VIM solution.

## Summary

We enumerate the main features of OpenVIM:

- Fully compatible with Open Source MANO (OSM) .
- Open Source License
- Easy to use

### **2.3.1.3 VMWare**

#### **Excellence**

VMWare is a commercial virtualization solution broadly used by telecommunication operators. VMWare has a broad portfolio of products targeting various areas, which include the telecommunication market, and more specifically creating NFV Platform.

VMWare vCloud NFV platform is aligned with ETSI NFV Architectural Framework and provides components for compute, storage and network devices. In the scope of VMWare vCloud NFV platform VMWare vCenter Server Appliance, VMWare vCloud Director and VMWare NSX Manager play the role of the VIM as specified in ETSI MANO.

#### **Evaluation**

VMWare is a closed source solution that provides carrier grade services. It is very stable and has a very good commercial support. VMWare vCloud NFV platform adheres to ETSI standards and easily integrates with existing telecommunication systems providing business continuity for telecom operators.

VMWare's solution clearly matches the requirements of 5GinFire, but its intrinsic costs hinder the adoption by partners and researchers with limited budgets. Furthermore, customization of the solution is very limited, as there is no access to the source code.

#### **Summary**

VMWare provides all the necessary features, but lacks an Open Source License.

### **2.3.2 5GinFIRE Candidates**

---

5GinFire is only targeting carrier grade open source solutions so VMWare will not be considered in the scope of the project as closed source.

OpenVIM is targeted at all in one installation, since the projects intends to run a broad infrastructure it will not be considered.

OpenStack is the defacto Open Source VIM solution, most developers are aware of OpenStack API and capabilities and several partners have knowledge running OpenStack deployments.

No development work is going to be required inside 5GinFIRE, only integration work as well as maintenance is to be considered. Since testbeds already dispose of some OpenStack nodes (some coming from the FIWARE project) as well as on 5TONIC infrastructure, integration efforts should be a reasonable task.

## 2.4 SDN

SDN (Software-defined networking) is a novel architecture that could be briefly described by providing separation of the control plane and forwarding plane and allowing through open APIs to program network dynamically. It proposes a centrally managed architecture via SDN controllers and the role of this chapter is to strive providing the SDN controller for the 5GinFIRE project.

### 2.4.1 SDN Controllers

We decided to focus on 4 of the most emerging SDN controllers that are OpenDaylight, ONOS, OpenContrail and Ryu. We studied them by following the aspects describe in section 2.2.

#### 2.4.1.1 OpenDaylight

##### Excellence

OpenDaylight is an open source SDN controller first released on February 2014. The OpenDaylight Foundation is part of the Linux Foundation, a large open source community pushing several projects in different fields.

The architecture of OpenDaylight (Figure 4) reflects the general concepts of SDN.

- On north are located the applications that control the network. They use the controller to gather information about the network and push new rules.
- The central control platform implements a set of pluggable modules to perform all the required actions.

On south are located the different routing elements of the network, either real or virtual. OpenDaylight southbound API implements a set of protocols to communicate with those devices, such as OpenFlow or Netconf.

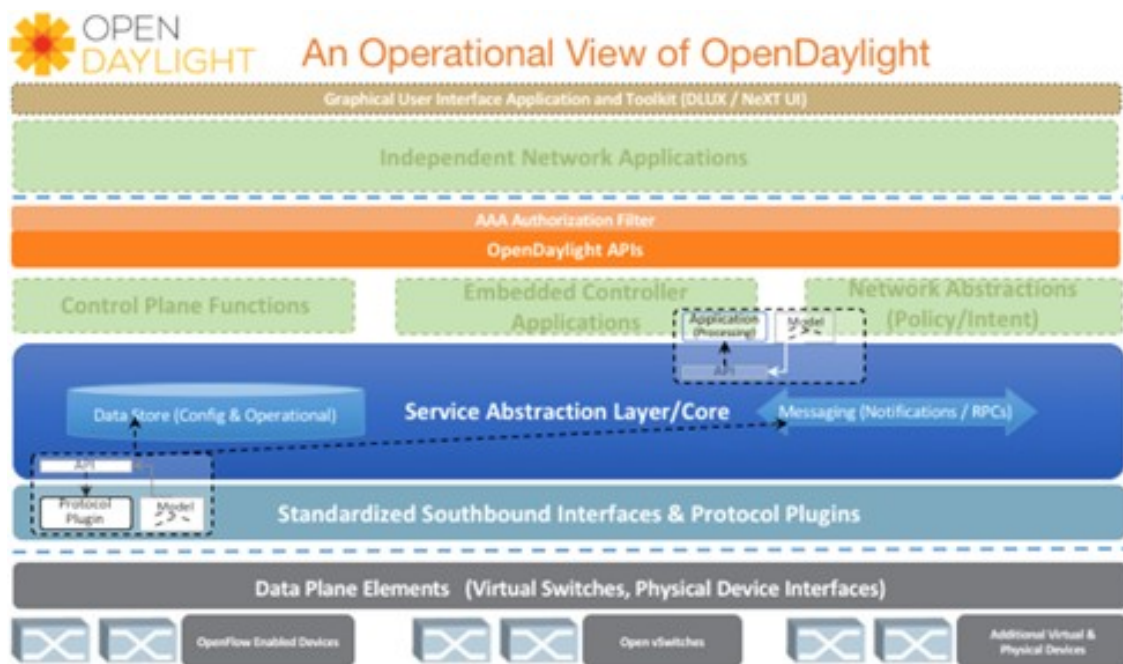


Figure 4: OpenDaylight - Operational view

OpenDaylight is backed by a very large community. In 2016, 524 developers contributed to the project, for a total of 920 developers since the beginning of OpenDaylight, ranking OpenDaylight project team among the top teams of the open source world. Moreover, the evolution of the project is keeping the same frequency compared to 2015. There are about 2000 commits per month [2]. The community can gather on a regular basis during the OpenDaylight summits that take place once a year, or for more specific events such as developer's forum. Beside the developers that participate to the project as volunteers, the companies can join the project as members. Today many companies are part of the project, with different levels of involvement. Among the most involved and most influent members (called platinum members) we can quote Cisco, Intel or Red Hat [3]. This demonstrates a real interest of the communities of developers and enterprises for the OpenDaylight project. Thanks to this active community the 5th version of the controller, Carbon, is already under development.

OpenDaylight is not only a demonstration product, as some SDN controllers have been in the past. It includes characteristics of carrier grades solutions. OpenDaylight controllers can be clustered [4]. Clustering is a key feature since it allows the enforcement of High Availability and the fast scaling of resources, especially in the cloud context. OpenDaylight also uses microservices architecture to deal with complex problems with simple functions. This architecture is an asset for large, complex systems. Indeed, it makes each component lighter, simpler, easier to upgrade and more efficient in its work.

## Evaluation

As part of the Linux foundation OpenDaylight is fully open source, published under the eclipse public license v1.0 [2].

Thanks to the large community, which activities have been presented in the previous section, the project has quickly evolved to reach a high level of maturity, with about one new version every 8 months. First released on February 2014 with the Hydrogen version, the project is now on its 4th stable version, Boron, released on December 2016. The next version, Carbon, is under development [5]. In addition to this very fast evolution OpenDaylight has been proven to be stable and mature enough to be used in the industry. Per a 2016 study [6] 61% of the enterprises that have deployed an SDN solution have chosen OpenDaylight, and most companies that consider deploying SDN solutions in the future also want to start with this controller. Among the solutions using (or based on) OpenDaylight we can find HPE Carrier SDN, Huawei Agile controller, Brocade SDN Controller, Extreme networks oneController, Inocybe Open Networking Platform or Virtuora Network Controller (Fujitsu). This adoption by industry tends to prove that OpenDaylight is a carrier grade controller, and this is confirmed by the carrier grade features that are implemented, such as clustering or microservices architecture.

Such popularity is probably due to the vast number of features displayed by OpenDaylight. On its southbound interface the controller can handle various standardized protocols, including OpenFlow (all versions), Netconf [7] and OVSDB [8]. The northbound APIs are not standardized, as in any other SDN controllers, but they still offer a lot of possibilities. Among all the available features we can highlight NEMO [9], an intent NBI developed by Huawei that could represent a kind of standardized NBI.

If the available northbound applications or plugins do not fit the needs of the project OpenDaylight offers a rich documentation to help developers to code their own features. This documentation is updated with each new release, which represents a high frequency.

Beside its efficiency as a standalone controller OpenDaylight can also be integrated in a more general architecture such as the NFV architectural framework defined by ETSI [10]. In this framework, the controller is tightly linked with the MANagement Organization (MANO) component and the Virtual

Infrastructure Manager (VIM). As part of the Linux Foundation, MANO Open-O is specifically designed to be able to work with a controller, and especially with the two controllers supported by the foundation: OpenDaylight and ONOS [11]. However other MANO components, such as Open Source MANO (ex OpenMANO) can be used through some manipulations [12].

OpenDaylight profits from a strong popularity, leads the SDN controllers' community and 5GinFIRE's partners naturally work with it on their internal projects and acquire then significant competences. After having gone through all the partners, a good number of them claim to have a high expertise with OpenDaylight: B-COM, ITav, UnivBris, UFU and TID, who is a Contributor of the NetIDE project and a member of the ODL Advisory Board.

## Summary

As we saw in the previous section OpenDaylight:

- Is open source, with a very large and active community
- Is mature
- Offers a rich standardized southbound API
- Offers a rich northbound API
- Possesses a lot of features and applications
- Can be integrated in an NFV architectural framework using Open-O or OSM
- Benefits of a high partners' expertise

### 2.4.1.2 *ONOS*

#### Excellence

With its first version released on December 2014 [10] ONOS (Open Network Operating System) is the most recent mainstream open source controller. ONOS presents itself as an OS for networks [11], as opposed to the traditional SDN controller seen as experimental devices. As a network, OS ONOS has the same place in the architecture as a traditional SDN controller, but is supposed to have much more responsibilities, such as:

- Manage the limited resources and divide them between users
- Isolate different users from each other
- Provide abstraction to hide resource complexity
- Provide security
- Supply useful and basic features, so that developers do not have to code them again and again

Those OS-oriented features tend to make ONOS more complete and useful than a traditional SDN controller, whose role is much more limited. In fact, ONOS supporters consider that traditional SDN controllers are basically rule-pushers, with not a lot of added values.

Besides the attributes listed below a very important aspect of ONOS - the most important perhaps - is the distributed aspect. ONOS is thought, from architecture to implementation, to be distributed. Distribution allows High Availability (HA) and easy scaling of resources by addition of new servers.

The Figure 5 describes the ONOS subsystem distribution.

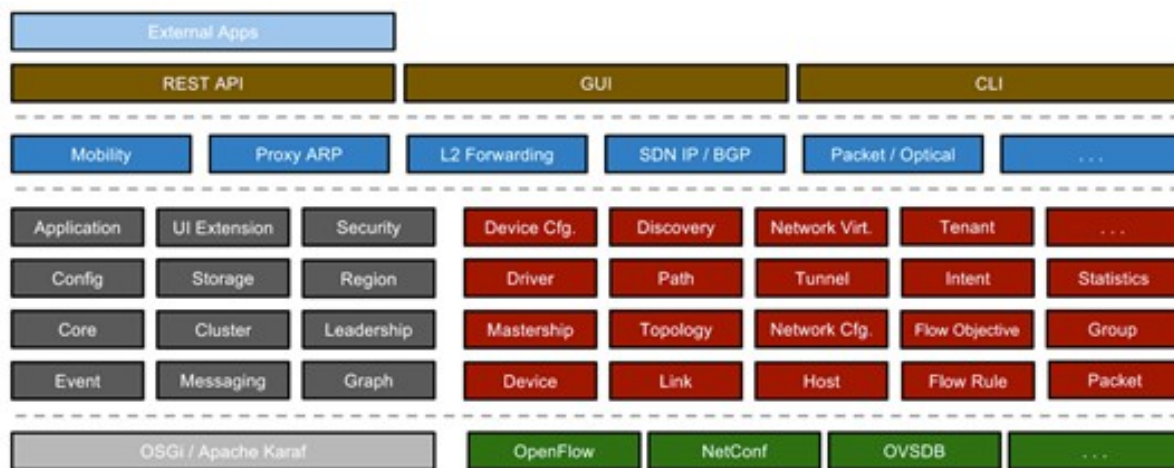


Figure 5: ONOS subsystem

## Evaluation

As part of the Linux Foundation ONOS is fully open source.

Although it is a recent project, ONOS is supported by a large community and has already many releases. The first one, Advocet, was delivered on December 2014. The 9th and last one, Ibis, was delivered on December 2016 [13]. ONOS is adopted by many professionals, which demonstrate its maturity. 23% of the enterprises that have deployed an SDN solution have chosen OpenDaylight, and 21% of the companies that consider deploying SDN solutions in the future also wants to start with this controller [3]. Those figures show that ONOS is not as popular as OpenDaylight, by far. However, this might change in the future, for two reasons. Firstly, as explained in the previous section ONOS is more recent than OpenDaylight, which gives OpenDaylight an advantage, but in the same time it is supposed to be more carefully designed, more mature, which could give it an important advantage in the long run. Secondly OpenDaylight and ONOS are not designed to play the same role: while OpenDaylight is more data centre-oriented ONOS is more adapted for the WAN management. Since SDN is more deployed in the data centers today it seems logical that OpenDaylight is more adopted, but this might change in the future [14].

Among the solutions using ONOS, or a controller based on ONOS, we can find: Huawei, ECI, Virtuora Network Controller (Fujitsu) or Atrium (ONF).

ONOS architecture pays a lot of attention to its NBI and SBI. As an OS, ONOS is designed to accept any new southbound protocol to communicate with any network device, and hide the complexity and diversity of the network to higher levels. Among the southbound protocols already supported we can quote OpenFlow (all versions), Netconf and OVSDB. The north API is not standardized, but ONOS provide a REST API for northbound applications. Beside this generic API ONOS provides an intent framework to simplify application developers' work, and a general view of the network [11].

If the available northbound applications or plugins do not fit the needs of the project ONOS offers a rich documentation to help developers to code their own features. This documentation is updated with each new release, which represents a high frequency.

Regarding the MANO architecture ONOS is equivalent to OpenDaylight. As part of the Linux Foundation, Controllers Open-O is designed to integrate it [8]. It is also possible to use it in Open Source MANO [9].

Even if its notoriety cannot be compared to OpenDaylight, ONOS is a serious alternative and very promising. By polling partners of the project, few of them declared having a strong expertise with ONOS, but for most of them they already started considering it and improving their skills.



## Summary

As we saw in the previous section ONOS:

- Is open source, with a larger and larger community
- Is mature, although it is not as much adopted as OpenDaylight
- Offers a rich standardized southbound API
- Offers a rich northbound API, including a homemade intent API
- Possesses features and application, maybe less than OpenDaylight
- Can be integrated in an NFV architectural framework using Open-O or OSM
- Benefits of a medium expertise from the partners

### **2.4.1.3**      *OpenContrail*

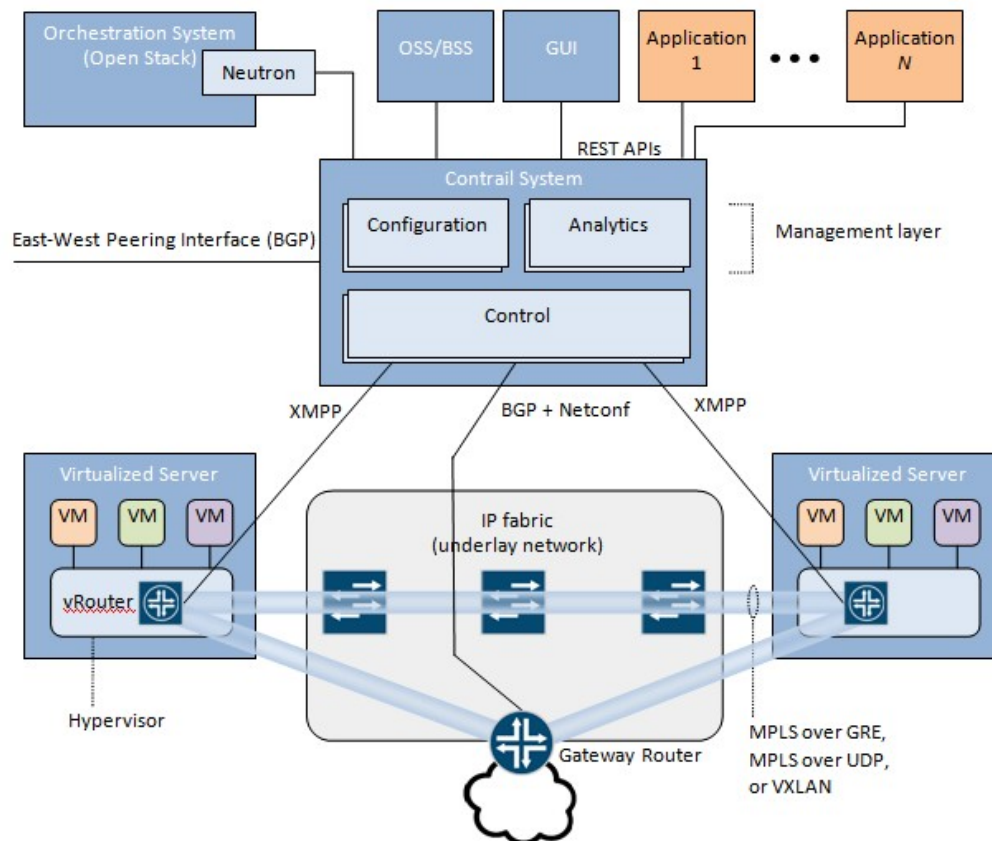
#### **Excellence**

OpenContrail [21] defines itself as an open-source network virtualization platform for the cloud. The OpenContrail system is a platform for Software Defined Networking [22]. This platform has two main components: the OpenContrail controller and the OpenContrail Virtual Router (vRouter).

Although the literature has several quantitative and qualitative [15], [18] controller comparisons, most of them do not consider OpenContrail. In [23] provides a high level quantitative comparison that includes OpenContrail.

The Controller is a logically centralized and physically distributed SDN controller. It is important to notice that it not an OpenFlow controller, but, the responsible for providing management, control and analytics functions for the virtualized network.

The vRouter is a forwarding plane of the virtualized networks. It is conceptually like an open source vSwitch, such Open vSwitch (OVS), but it also provides routing and higher layer services. The Controller provides the logically centralized control plane and management plane and is responsible for the vRouter orchestration.



**Figure 6: OpenContrail System Overview [22]**

The Contrail System presented in Figure 6 is composed by a set of nodes with the roles of Configuration, Analytics and Control. The instances of these nodes can be physical servers or virtual machines.

The Configuration Nodes are responsible for the management layer and provides the northbound REST API that is used by a web-based GUI of the Contrail System and can be used by OSS/BSS systems or other applications [22]. Each Configuration Node has a persistent storage of the configuration state that is that is synchronized with other configuration instances. Using the Interface for Metadata Access Points (IF-MAP) protocol the configuration high-level data model is pushed down in a low-level configuration to the Control Nodes.

The Analytics Nodes collect, store and correlate information from network elements such as statistics, logs, events and errors [22]. These nodes also provide a northbound REST API to allow client applications to submit queries. The analytics information is stores in a NoSQL database and is synchronized with other Analytics Nodes instances of the Contrail System. The analytics information is gathered from the Control Nodes using a XML over TCP protocol called Sandesh.

The Control Nodes receive the configuration state from the Configuration Nodes and exchange this information with the vRouters agents that are running inside Compute Nodes using the XMPP protocol [22]. These nodes are also responsible to exchange routes with the gateway nodes (routers and switches) using BGP and send configuration state to the gateway nodes using NETCONF. Control nodes interact with each other and with network elements to ensure that network state is eventually consistent.

The other component of the Contrail system is the vRouter, as presented in Figure 5. The vRouter runs inside the Compute Nodes, a general-purpose x86 server that hosts VMs. The vRouter has two building blocks: the vRouter Agent and the vRouter Forwarding Plane.

The vRouter Agent is a linux user space process that is peer point of the Control Nodes [22]. It receives the low-level configuration state and installs it in the forwarding plane. Also, it reports analytics information. Each vRouter agent is connected to at least two control nodes for redundancy in an active-active redundancy model.

The vRouter Forwarding Plane is a Linux kernel mode process that is responsible to handle the data plane [22]. The forwarding plane is a set of routing instances. Each routing instance is assigned to a tenant and has internally a Forwarding Information Base (FIB) and a Flow Table. This flow table is used to apply forwarding policies to packets. This component of the vRouter is also responsible to support the Overlay tunnels based on MPLS over GRE, MPLS over UDP or VXLAN to transport packets between different compute nodes. The Flow Table has the same concept of an OpenFlow table but there is no relation to it.

## Evaluation

The controller is licensed under an Apache 2.0 License [19] and the vRouter is distributed under the terms BSD 2-Clause License [20]. By analysing the GitHub repository is possible to show that Contrail has a strong commitment from Juniper and consistent number of commits in the last four years (2013 to 2017). The controller has a group of 116 different contributors [19] while the vRouter a group of 44 unique contributors [20].

The system has a documentation that guides the installation, configuration, monitoring and troubleshooting [17] and also detailed description of the Configuration API [16]. The northbound REST API is proprietary and it is not “Open Source MANO compliant”.

The 5GINFIRE partners reported limited expertise regarding Open Contrail platform.

## Summary

As we saw in this section regarding Open Contrail:

- Is open source, with an active community
- Is mature
- Offers a REST based northbound API
- It is not based on OpenFlow
- Provides a comprehensive documentation
- 5GINFIRE has limited expertise with the platform

### 2.4.1.4 *Ryu*

#### Excellence

Ryu is a component-based software defined networking framework [24]. The first release of the Ryu controller was in September 14 2013 [24]. Many improved updates since then have been released from the Ryu developers. Ryu provides software components with well-defined Application Program Interfaces (API) that make it easy for developers to create new network management and control

applications. By using Southbound APIs [25] Ryu communicates information to the network switches and routers, also implementing Northbound APIs for the application layer communication.

## Ryu architecture

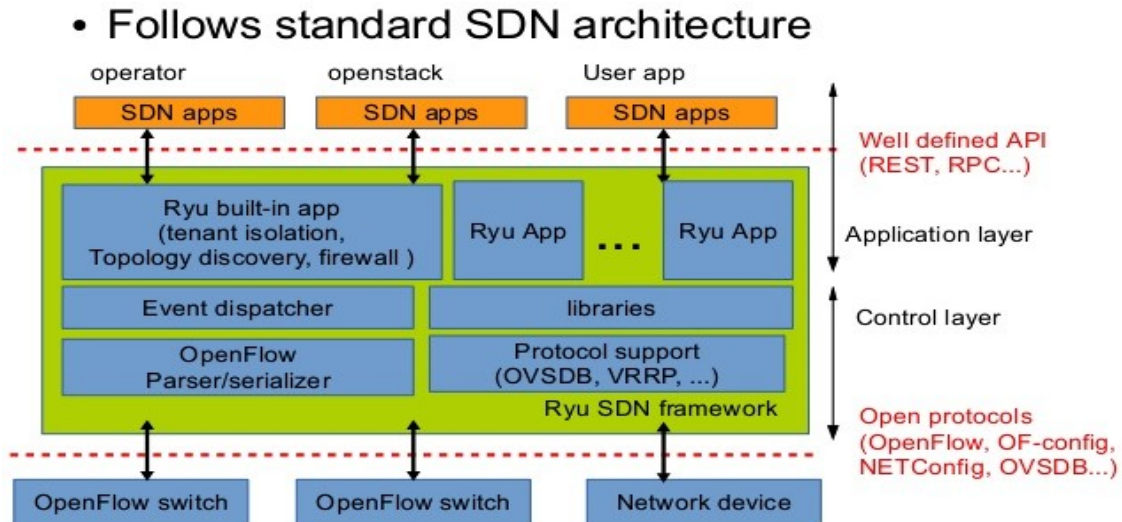


Figure 7: Ryu Architecture

Ryu supports various protocols for managing network devices [26], such as OpenFlow, Netconf, OF-config, etc. Written entirely in Python all the code is open source and freely available under the Apache 2.0 license [24]. Ryu is managed and maintained by the open Ryu community. Ryu implements the general SDN architecture as depicted in Figure 7. The communication between the SDN applications and the OpenFlow switches is possible through the Southbound and Northbound APIs. Ryu supports all the versions of OpenFlow [24].

### Evaluation

Ryu controller is entirely written in python, with many example applications and tutorials making Ryu easy to use [25]. Often researchers choose Ryu since it is easier to implement their methodologies and evaluate them, also Ryu controller is preferred for data center virtualization. Although Ryu comes with a good documentation and it is easy configurable other controllers like OpenDaylight (ODL) and ONOS are selected for industrial purposes [27] since they provide more features and have larger communities. Moreover, ODL uses the java framework OSGi [28] which allows ODL to be preferred for professional use since maintainability and scalability is a high priority. Unfortunately, Ryu does not come with this advantage.

### Summary

Ryu controller in a nutshell:

- Fully open source with available tutorial for applications.
- Supported by the Ruy Community.
- Licensed under the Apache 2.0 license.
- Often selected for research purposes.
- Provides Northbound and Southbound APIs.
- Supports all OpenFlow Versions.

## 2.4.2 5GinFIRE Candidate(s)

This section concludes the SDN Controllers comparative analysis and provides thus the SDN controller candidates to the 5GinFIRE infrastructure architecture.

**Table 1: SDN Controllers Comparative Analysis**

CRITERIA		ONOS	OpenDaylight	OpenContrail	Ryu
<b>Open Source</b>	Type of License	Apache 2.0	Eclipse Public Licence - Version 1.0	Apache-2.0 (Controller) and BSD-2 (vRouter)	Apache 2.0 license
	Number of Contributors	162 (GitHub)	994 (GitHub)	116 (controller) and 44 (vRouter)	74 (GitHub)
	Frequency of Contributions	Weekly	Weekly	Weekly	Weekly
<b>Maturity</b>	First Release Date	Dec. 5, 2014 Avocet	February 4, 2014 Hydrogen	September, 2013	September 14 2013
	Latest Release Date	Dec 2016, Ibis	November 3 2016 Boron	3.2 (December, 2016)	February 11, 2017
	Medium Release Interval	New release every 3 months	New release every 18 months	New release or update every 4 moths	New release every 5 months
	Carrier Grade		Not open-source	Open Source	Not open-source
	Ease to Install	easy		easy	
	Documentation Available	Official Website	Official website	Official website	Official website
	Documentation Up to Date	Up to date for latest release	Up to date for latest release	Up to date for latest release	Up to date for latest release
<b>API and Standard compliance</b>	Northbound APIs	REST	REST	REST	REST
	OpenMANO Compliant	yes	v0.4 release	no	
	Southbound APIs	OpenFlow, Netconf, OVSDB	NetConf, CLI, SNMP (via MIBs), XMPP, or OpenFlow, OVSDB	OVSDB, BGP, Netconf, XMPP	OF-Config, NETCONF, OpenFlow
	OpenFlow Versions	Supports version 1.0 and 1.3.	Handling version 1.0 or 1.3 on the fly.	NA	Supports fully 1.0, 1.2, 1.3,

		Implementati on of 1.5 is ongoing			1.4, 1.5
<b>Capacity to be mastered by partners</b>		medium expertise	strong expertise	limited expertise	medium expertise

This section described the Software Defined Networking (SDN) technology, more precisely the SDN controllers were analysed in detail in order to come to a conclusion and select a controller as the 5GinFIRE candidate. The controllers evaluated are the OpenDaylight, ONOS, OpenContrail and Ryu. For the evaluation the comparison included the licence of each controller the maturity of the software and the adaptability, the APIs quality and expendability and finally the feature set provided by the controllers, this information aided us into our assessment to select the controller that fits the standards that the project demands.

By summarising the architecture and the capabilities of all the controllers the project arrives to the choice of the OpenDaylight controller as the candidate for the project experiments and architecture. This choice is based on the comparison between the controllers’ evaluations, OpenDaylight is the most commonly used tool and the best fit for the project needs. Also it is the most supported controller among all of them as well as the one with the larger community and the most mastered by partners. Finally OpenDaylight provides the software that can assess all the requirements of the project and can scale to accomplish future project goals.

Besides, with OpenDaylight the integration shall be very limited in time as this controller is already present inside 5TONIC infrastructure and ready to be used. No particular development is necessary, the project is much more focused for stability and efficiency. Some maintenance efforts are considered to maintain the SDN stack in a production-level.

## **2.5 NFV**

This section presents an overview of the main projects and technologies that have been identified by the 5GinFIRE consortium as relevant in the field of Network Functions Virtualization (NFV). In the following, we analyse the diverse functionalities provided by these solutions in the context of the NFV reference architecture defined by the ETSI Industry Specification Group for Network Functions Virtualization (ISG NFV). We conclude this section presenting a summary of the main features and functionalities of the analysed solutions, along with a set of considerations to motivate the selection of technologies to support NFV functionalities within 5GinFIRE.

### **2.5.1 NFV management and orchestration**

This section covers the most relevant projects and open-source initiatives that focus on the development of specific solutions to support the management and orchestration of NFV services and resources. To serve as a reference, Figure 8 presents the structure of the NFV management and orchestration (MANO) system defined by ETSI, and its interrelation with the other components of the NFV reference architectural framework. This subsection does not cover Virtualized Information Manager (VIM) solutions, as these are specifically addressed in Section 2.

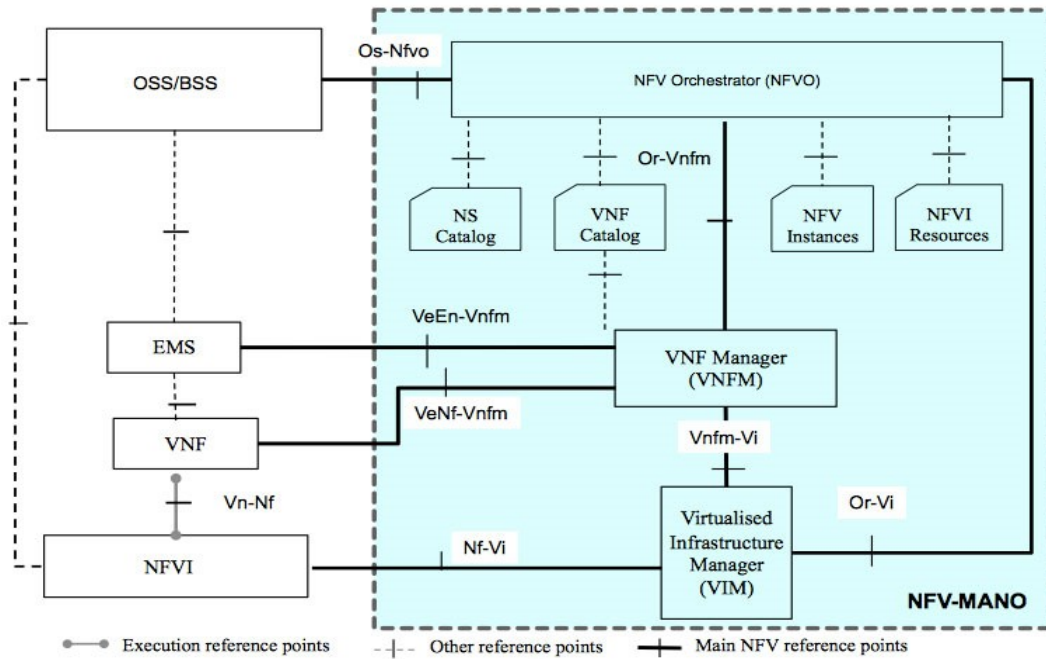
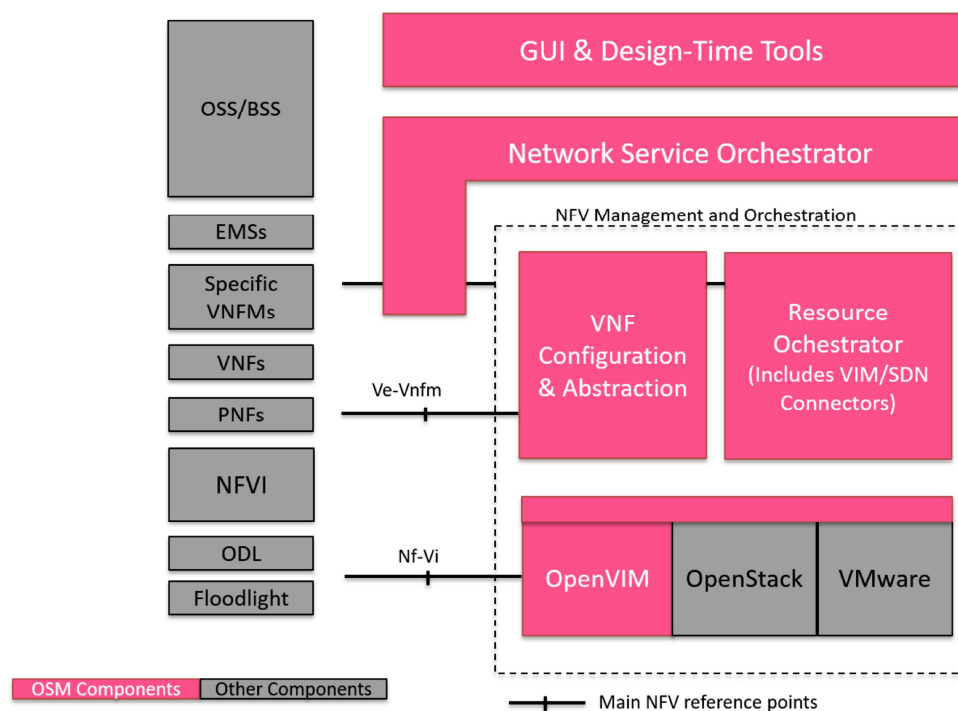


Figure 8: NFV-MANO Architecture

2.5.1.1 *Open Source MANO*

**Excellence**

Open Source MANO (OSM) [52] is an ETSI-hosted open-source project involving leading network operators, NFV cloud providers and research and academic centers, including Telefónica, British Telecom, Telenor, Telecom Austria Group, Intel, Canonical, RIFT.io, Mirantis, VMware, 6WIND, IMDEA Networks and DELL, among others (the up-to-date list of members and participants is provided in the OSM website, [53]). The project aims at providing a practical open source implementation of a NFV Management and Orchestration (MANO) stack aligned with the NFV reference architectural framework defined by the ETSI NFV ISG [29].



**Figure 9: OSM Mapping to ETSI NFV MANO**

Figure 9 (extracted from Whitepaper: OSM Release One, a Technical Overview [30]) illustrates the different architectural components that comprise the OSM stack, and their relation with the other components defined by the ETSI framework.

The Network Service Orchestrator (NSO), based on RIFT.ware [54] from RIFT.io/Intel, takes care of the delivery of end-to-end network services, interacting with the Resource Orchestrator and the VNF Configuration & Abstraction components of the OSM architecture. It provides the point of contact in the OSM architecture to support the lifecycle management of network services, catalogue management and on-boarding/configuration of network services and VNFs, among others. The current OSM software stack implementation includes a Graphical User Interface (GUI), which provides an intuitive easy-to-use mechanism to interact with the NSO, providing the necessary tools for VNF on-boarding and for the creation and instantiation of network services.

The resource orchestrator (RO), which is based on OpenMANO [55], coordinates the allocation and setup of the computing, storage and network resources that are necessary for the instantiation and interconnection of VNFs. For this purpose, the RO may interact with multiple Virtualized Infrastructure Managers (VIMs), which may be of different types (see the Evaluation section). This component provides most of the functions associated with the NFV Orchestrator defined by the ETSI NFV framework, and follows a plug-in model to support the addition of other types of VIM and SDN Controllers.

Finally, the VNF Configuration and Abstraction layer (based on Juju Charms from Canonical) [56] is aligned with the VNF Manager defined by the ETSI NFV reference architectural framework, overseeing VNF configuration per the corresponding VNF descriptors, following a model-driven approach.

This architecture of the OSM stack has been developed following four guiding principles:

- Layering, that allows the plug-in replacement of the various layers (NSO, RO, VIM, etc.) with additional or alternative components
- Abstraction, helping to clarify what is required by the different layers of the system



- Modularity, even within layers, that enables a plug-in model to facilitate module replacements as the OSM develops and evolves.
- Simplicity, trying to have the minimal complexity necessary to implement the governing information models in the solution.

## Evaluation

OSM source code is available under the Apache License, Version 2.0 [57]. The implementation includes contributions at the different architectural levels from outstanding organizations in the field of NFV, particularly:

- VIM: Telefonica's OpenMANO's OpenVIM.
- VNFM: Juju from Canonical.
- NFVO: Rift.ware, OpenMANO, Murano (by Mirantis).

In addition, the current release of the OSM software stack (OSM ONE), available from October 2016, includes a one-step installing process based on containers and Juju modelling, aiming at simplifying testing, customization and deployment processes with respect to the previous release (OSM ZERO). This version supports a plug-in model framework that facilitates maintenance operations and future extensions of the platform, while improving the interoperability with other components like VNFs, VIMs or SDN controllers. OSM plug-in model allows integrating multiple VIMs (the current release supports OpenStack, OpenVIM, and VMware vCloud Director). Release ONE allows the deployment of multi-site network services that span across multiple datacentres, and includes OpenVIM as part of the OSM run-time environment to provide a reference VIM for all-in-one installations with support of Enhanced Platform Awareness (EPA), aiming at having greater awareness about the capabilities of the platform under control.

Moreover, OSM provides a wiki [58] with up-to-date documentation covering technical details concerning the project, along with an installation and a user guide. The initial expectation of the project is to provide new releases with a periodicity of six months.

Finally, we want to highlight that the 5GinFIRE consortium includes partners with direct involvement in the development of OSM, particularly Telefónica, as an OSM member, and UC3M, as an OSM participant.

## Summary

In the following, we enumerate the main features of OSM, as previously presented:

- ETSI-hosted open-source project.
- Practical open source implementation of a NFV MANO stack aligned with the ETSI NFV reference architectural framework.
- Supported by leading network operators, NFV cloud providers and research and academic centers (at the time of writing 26 members, 31 participants).
- Licensed under Apache License 2.0.
- Available up-to-date documentation.
- Supports a plug-in model framework to facilitate maintenance, future extensions and interoperability.
- Multi-VIM support.
- One-step installing process.
- Providing an intuitive easy-to-use GUI.

### 2.5.1.2 OpenMANO

#### Excellence

OpenMANO is an open source project led by Telefonica that aims at providing a practical implementation of a Management and Orchestration stack aligned with the NFV reference architectural framework under standardization at ETSI's NFV ISG [31]. The OpenMANO architecture (see Figure 10) is composed by three main software components:

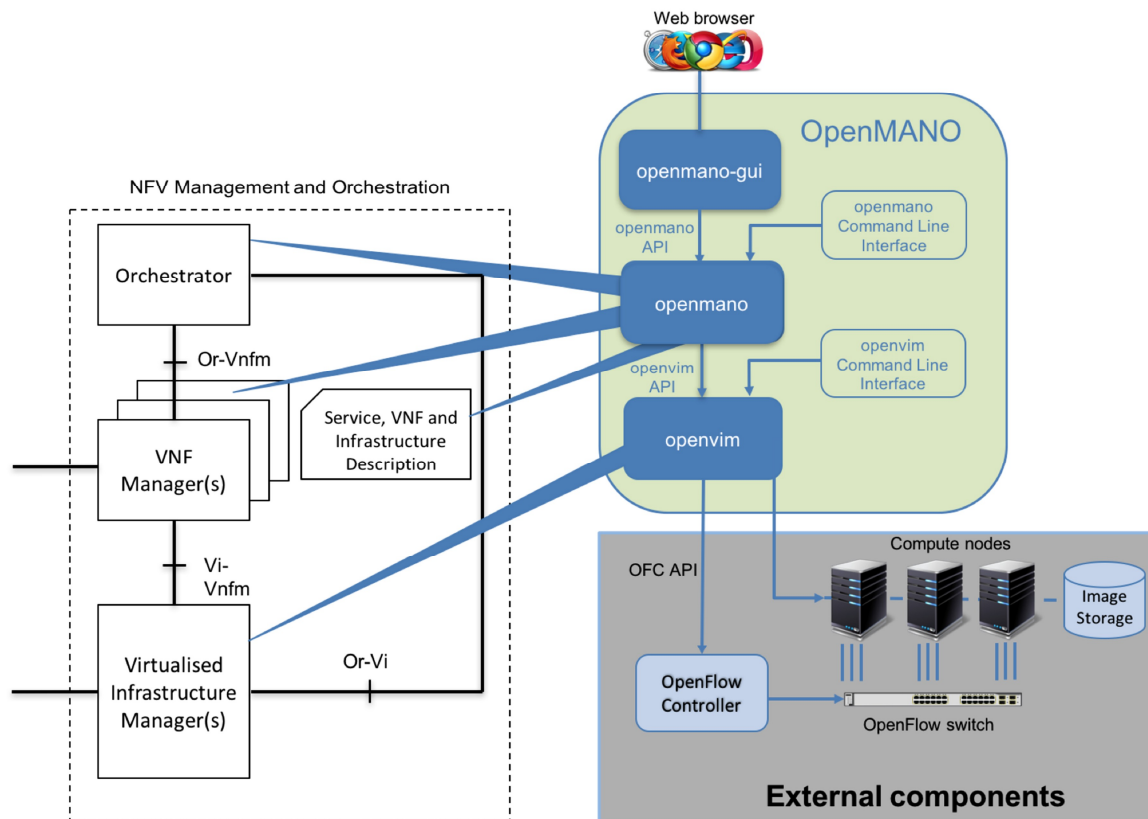


Figure 10: OpenMANO Architecture

**OpenMANO:** reference implementation of a Network Functions Virtualisation Orchestrator (NFVO). OpenMANO interfaces with an NFV Virtualized Infrastructure Manager and provides a northbound interface, where NFV services are offered including the creation and deletion of virtual network functions (VNF) templates, VNF instances, network service templates and network service instance. This component has been contributed to the open source community project Open Source MANO (OSM) hosted by ETSI, as it was commented previously in this document.

**OpenVIM:** reference implementation of an NFV VIM, which provides virtual computing and networking resources and supports the deployment of virtual machines, interfacing with an NFV Infrastructure (NFVI) and an OpenFlow controller. This implementation follows the recommendations of [32].

**OpenMANO-GUI:** web graphical user interface to interact with the OpenMANO API.

#### Evaluation

OpenMANO is a result of the innovation work of Telefónica in the field of NFV MANO. Being part of the Open Source MANO software stack, as it has previously been commented, its use in 5GinFIRE will be considered within the context of OSM.

## Summary

The main features of OpenMANO are summarized next:

- Open-source project.
- Practical implementation of a Management and Orchestration stack aligned with the NFV reference architectural framework standardized by ETSI
- Includes a GUI and the functionality of an NFVO and a VIM.

### **2.5.1.3**      *Rift.ware*

#### **Excellence**

RIFT.ware is an ETSI-compliant NFV MANO solution and commercial distribution of ETSI Open Source MANO (OSM) that radically simplifies the deployment of multi-vendor VNFs and orchestration of complex, multi-vendor network services in carrier and enterprise clouds. It supports everything needed for highly automated, end-to-end service delivery and lifecycle management. RIFT.ware features intelligent workload placement and integrated platform awareness (EPA), such as Intel Enhanced Platform Awareness, to optimally use available network and cloud infrastructure capabilities. [59]

#### **Evaluation**

Rift.ware is offered by Rift.io. RIFT.io is a founding member of the OSM project and contributed significant code, including network service orchestration, graphical user interface, and automation tools. RIFT.ware 4.3.3 is available as a commercially solution to accelerate market adoption of OSM and NFV through advanced, carrier-grade features that simplifies the onboarding and management of virtual network functions (VNFs) across one or more service provider data centers.

## Summary

The main features of Rift.ware are summarized next:

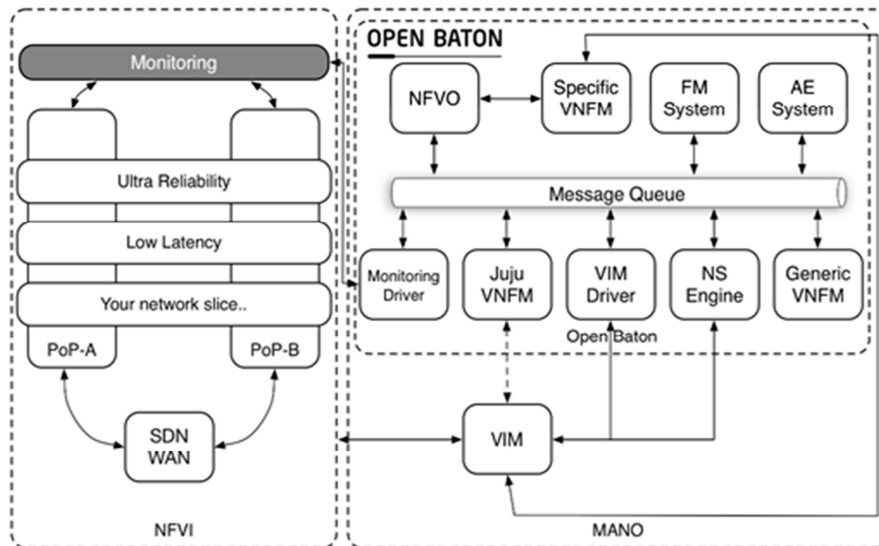
- Mature product.
- It has good support since it is implemented and supported by Rift.io.
- Fully compliant with OSM APIs and interfaces.
- It is offered as a commercial distribution.

### **2.5.1.4**      *Open Baton*

#### **Excellence**

Open Baton is an ETSI NFV compliant MANO framework. It enables virtual Network Services deployments on top of heterogeneous NFV Infrastructures.

It integrates with OpenStack, and provides a plugin mechanism for supporting additional VIM types. It supports Network Service management either using a generic VNFM or interoperating with VNF-specific VNFM. It uses different mechanisms (REST or PUB/SUB) for interoperating with the VNFMs. It integrates with additional components for the runtime management of a Network Service. For instance, it provides auto scaling and fault management based on monitoring information coming from the monitoring system available at the NFVI level.



**Figure 11: Open Baton Architecture**

Open Baton (see architecture in Figure 11) provides many different features and components:

- A Network Function Virtualisation Orchestrator (NFVO) completely designed and implemented following the ETSI MANO specification.
- A generic Virtual Network Function Manager (VNFM) able to manage the lifecycle of VNFs based on their descriptors.
- A Juju VNFM Adapter to deploy Juju Charms or Open Baton VNF Packages using the Juju VNFM.
- A driver mechanism for adding and removing different type of VIMs without having to re-write anything in your orchestration logic.
- A powerful event engine useful based on a pub/sub mechanism for the dispatching of lifecycle events execution.
- An auto scaling engine which can be used for automatic runtime management of the scaling operation operations of your VNFs.
- A fault management system which can be used for automatic runtime management of faults which may occur at any level.
- It integrates with the Zabbix monitoring system.
- A set of libraries (the openbaton-libs) which could be used for building your own VNFM.
- A Marketplace useful for downloading VNFs compatible with the Open Baton NFVO and VNFMs.

## Evaluation

Open Baton is a project developed by Fraunhofer FOKUS and TU Berlin. It is supported by different European publicly funded projects: NUBOMEDIA, Mobile Cloud Networking, CogNet, SoftFIRE. Open Baton is one of the main components of the 5G Berlin initiative.

It can support different OS such as:

- Linux OS: Direct installation of a standalone or complete Open Baton environment on top of a Linux OS (Ubuntu/Debian) using either the source-code or binary version.
- Mac OS: Install the Open Baton NFVO on MacOS using the provided brew formula.
- Docker: Launching a pre-configured Docker image containing a standalone Open Baton environment.

- Vagrant: Launching a vagrant box using the provided 'Vagrantfile' containing a standalone Open Baton environment.

Built from scratch following the ETSI MANO specification. The NFVO uses the ETSI NFV data model internally for the definition of the Network Service and Virtual Network Descriptors.

Allows interoperability being interoperable is one of the challenges brought by the fragmented ecosystem in the management and orchestration area. It requires a lot of work to make two different vendors solution working together need of a single vendor-independent platform.

Easily extensible based on a message bus architecture it provides several mechanisms for being extended with new functionalities and integrated in existing platforms.

## Summary

As we saw in previous section Open Baton has:

- Consistent roadmap evolution.
- Integration with different OS.
- The community is not so big and it is mainly leaded by German.
- Integrated with OpenStack.
- It is distributed under the Apache License 2.0

### 2.5.1.5 *ECOMP*

#### Excellence

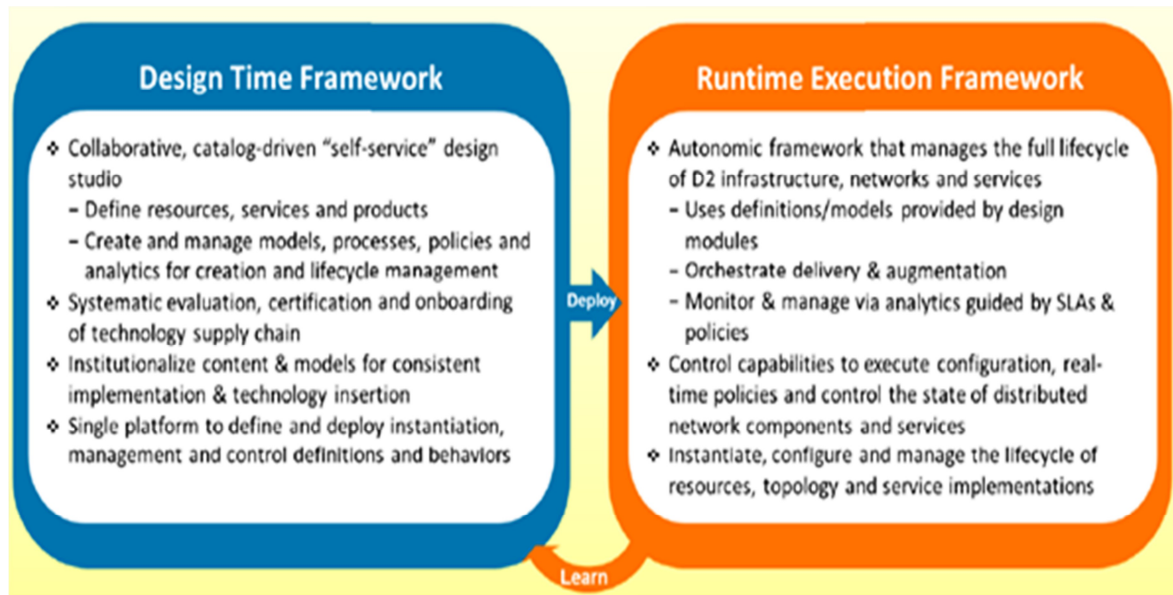
The ECOMP Software Platform delivers product/service independent capabilities for the design, creation and lifecycle management of the environment for carrier-scale, real-time workloads. It consists of eight software subsystems covering two major architectural frameworks: a design time environment to design, define and program the platform, and an execution time environment to execute the logic programmed in the design phase utilizing closed-loop, policy-driven automation.

The ECOMP architecture and Operational Management Framework (OMF) were defined to address the business/strategic objectives of AT&T Domain 2.0 (D2) as well as new technical challenges of managing a highly dynamic environment of virtual network functions and services, i.e., a software ecosystem where functions such as network and service provisioning, service instantiation, and network element deployment all occur dynamically in real-time.

The ECOMP Platform enables product/service independent capabilities for design, creation and lifecycle management. There are many requirements that must be met by ECOMP to support the D2/ECOMP vision. Of those many requirements, some are key in supporting the following foundational principles:

- The architecture will be metadata-driven and policy-driven to ensure flexible ways in which capabilities are used and delivered
- The architecture shall enable sourcing best-in-class components
- Common capabilities are 'developed' once and 'used' many times
- Core capabilities shall support many AT&T Services
- The architecture shall support

These capabilities are provided using two major architectural frameworks: (1) a design time framework to design, define and program the platform (uniform on boarding), and (2) a runtime execution framework to execute the logic programmed in the design time framework (uniform delivery and lifecycle management). Figure 12 shows the ECOMP Platform architecture.



**Figure 12: ECOMP Platform**

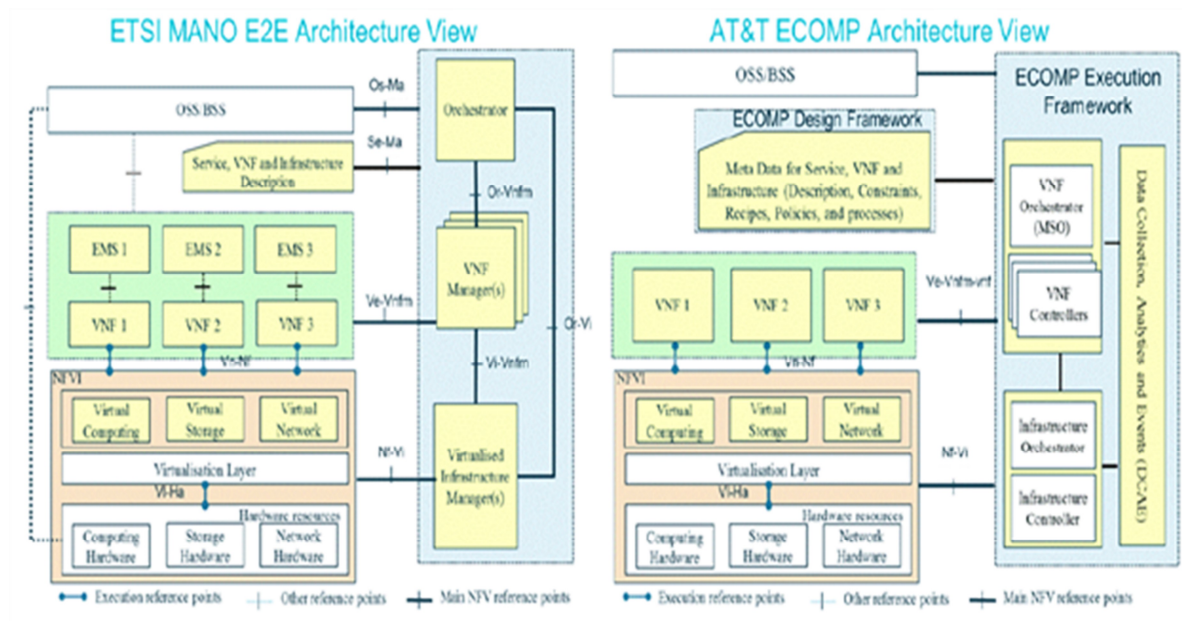
Ecomp has recently merged (in February ‘17) with open-o to create ONAP [127]. This new alliance is too young to get feedback and to be included in the scope of this document. This event does not alter the evaluation of ecomp and open-o done in this document.

## Evaluation

The main components of the ETSI-NFV architecture are: Orchestrator, VNF Manager, and VI (Virtualized Infrastructure) Manager. ECOMP expands the scope of ETSI MANO coverage by including Controller and Policy components. Policy plays an important role to control and manage behaviour of various VNFs and the management framework. ECOMP also significantly increases the scope of ETSI MANO’s resource description to include complete meta-data for lifecycle management of the virtual environment (Infrastructure as well as VNFs).

The ECOMP design framework is used to create resource / service / product definitions (consistent with MANO) as well as engineering rules, recipes for various actions, policies and processes. The Meta-data-driven Generic VNF manager (i.e., ECOMP) allows us to quickly on-board new VNF types, without going through long development and integration cycles and efficiently manage cross-dependencies between various VNFs. Once a VNF is on-boarded, the design time framework facilitates rapid incorporation into future services.

ECOMP can be considered as an enhanced Generic VNF manager as described by MANO NFV Management and Orchestration Architectural Options (see Figure 13).



**Figure 13: Comparison of ETSI MANO and AT&T ECOMP Architectures**

The ECOMP platform provides external applications (OSS/BSS, customer apps, and 3rd party integration) with a secured, RESTful API access control to ECOMP services, events, and data via AT&T gateways. Soon, ECOMP will be available in AT&T’s D2 Incubation & Certification Environment (ICE) making ECOMP APIs available to allow vendors, cloud providers, and other 3rd parties to develop solutions using ECOMP and AIC reference architecture (current and future-looking).

Orange and Bell Canada are testing ECOMP in their networks. Amdocs joined the AT&T efforts and has been contributing to ECOMP.

AT&T also informs that will provide the source code to Linux foundation to open source community implement and develop.

**Summary**

As we saw in previous section ECOMP has:

- Restrict control by AT&T.
- Linux Foundation will receive the code, but need time to absolve, adjust, etc.
- It is growing in Operators market but it is under AT&T usage.
- It doesn’t have open source community.

**2.5.1.6 Open-O**

**Excellence**

OPEN-O is a Linux Foundation Collaborative Project focused on building an open source, carrier grade orchestration platform. The primary goal is to establish an open framework to orchestrate end-to-end composite services across legacy networks, along with emerging SDN and NFV Infrastructure.

**Management Layers**

OPEN-O is positioned in the services layer to create and manage any service over any network. The Figure 14 describes how various open source projects fit in different layers to deliver the e2e management and control functions. OPEN-O on the south bound interworks with different SDN controllers, NMS /EMS systems, VNFM and VIM’s to create end-to-end services.



Figure 14: Management Layers

**OPEN-O Architecture Overview**

The OPEN-O architecture (see Figure 15) is compliant with the ETSI NFV Architectural Framework and Management and Orchestration (MANO), and has adopted standard service modelling languages YANG and TOSCA to ensure commonality and flexibility. OPEN-O has three main Orchestration functions namely Global Services Orchestrator (GS-O), SDN Orchestrator (SDN-O) and Network Function Virtualization Orchestrator (NFV-O) supported by Orchestrator Common functions. The Orchestrator common provides TOSCA related micro services for GS-O, SDN-O and NFV-O and includes TOSCA Parser (Project ARIA), TOSCA Catalog, TOSCA Graphical Designer, and an Inventory.

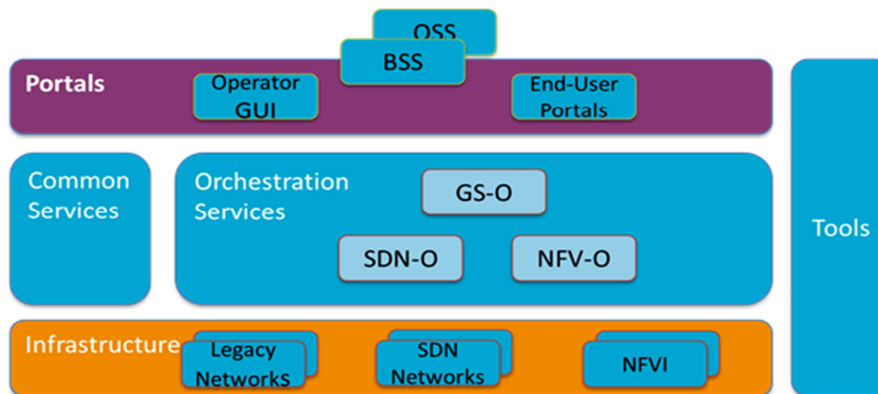


Figure 15: OPEN-O Architecture

**GS-O - Global Services Orchestrator**

GS-O is responsible for Global Service Orchestration to provide End-to-End orchestration of any service on any network. GS-O allows the user to describe the end-to-end (global) service in a single Global Service Description. GS-O decomposes the Global Service Description to SDN Service Descriptions and NFV Service Descriptions, and then forwards these requests to the correct instances of SDN-O and NFV-O.

The main components of GS- O include Service life cycle manager, service decomposer, service parser and NBI layer. The NBI layer interworks with Business operations layer through which GS-O receives the information about the services it needs to create to fulfil the product requests from business layer. GS- O Performs the creation & activation of global service instance and deactivate & deletion of global service instance.

**NFV-O - Network Functions Virtualization Orchestrator**

This NFV-O function is based on ETSI NFV MANO architecture & information model as a reference. It orchestrates Network Services composed of Virtualized Network Functions and is an integral part of



OPEN-O. The NFV-O function includes Network service life cycle manager, NFV resource manager, and NFV monitor. Using the south bound SBI, NFVO interworks with several NFV SDN controllers, Multi-vendor VNFM's and different VIM's.

The NS LCM (Network Service - Life Cycle Management) includes NS template design, NS package onboarding, NS instance creation, NS instance termination and NS package deletion. NS is composed of VNFs and the connections to be deployed. The LCM feature also includes a GUI portal that is used for NS template design (including workflow editing) and LCM operations.

The NFV-O also provides basic alarm monitoring and performance statistics of VIMs and Hosts.

NFV-O APIs/interfaces:

- NBI: Interfaces between GS-O and NFV-O based on REST API;
- SBI: Interfaces between NFV-O and drivers for (VNFM, SDNC, and VIM) on REST API;
- Information Model and Data Models:
  - Candidates under consideration: ETSI NFV 1.0/2.0, TOSCA NFV Profile v004/005, YANG for SFC.

### **SDN-O - Software Defined Network Orchestrator**

The SDN-O provides network service orchestration over SDN and Legacy networks. It will provide full service life-cycle functionality (design, provision, operate). Release 1 is focused on network integration and automated network provisioning. Future releases will address dynamic model driven service design, provisioning and assurance, as well as working with SDOs to align models and APIs.

SDN-O Features and Functionality:

- Provisioning and activation of e2e network connectivity services.
- Provisioning of both underlay (L3 VPN and L2 VPN) and overlay (VxLAN, IPsec, VPC, Service Chain) network connectivity
- Basic Resource Inventory
- Monitoring of utilization and optimization of the network path
- SDN-O APIs and TOSCA templates
- Simple SDN-O Portal
- Common SBI for VxLAN, IPsec, L3VPN, L2VPN, VPC, Service Chain

### **Common Services**

Provide the common services for all the other OPEN-O components, including Micro Service Bus, HA, Driver Manager, Log, Authentication, External System Register and Protocol Stack.

Common service Features:

- Service Registration - API, file, UI, Service Discovery, Service Call Routing, API Call Logging and Asynchronous Message Routing-Comet.
- Micro service Bus Cluster, Load balancing for stateless service and Service healthy monitoring.
- Provide User Management, login /logout and Session Management.
- RESTCONF Protocol Stack: A REST like protocol running over HTTP for accessing data defined in YANG using data stores defined in NETCONF.
- NETCONF Protocol Stack: The Network Configuration Protocol (NETCONF) provides mechanisms to install, manipulate, and delete the configuration of network devices.

### **Common TOSCA**

Common TOSCA provides shared components for designing, modelling, validating, executing and storage of VNFs, Network Services (NS) and Service Function Chaining (SFC) and related resources needed for GS-O, NFV-O and SDN-O.

Common TOSCA sub modules:

- TOSCA Parser and execution engine(ARIA) with OpenStack VIM Plugin and NETCONF Plugin
- Workflow Engine (ARIA)
- Graphical Modelling Designer
- Catalog
- Inventory
- Policy
- Analytics

Common TOSCA Features:

- Create service template and plan, Support layered/nested Service
- Provide Catalog Features: CSAR upload, download, delete, query, status management.
- Provide REST API interface to query information model.
- Provide artefacts download (software pkg., image, scripts, policy, plan, etc.)
- Provide Inventory functions, as resources centralized storage function, schema less data stores, CRUD (create, update, read, delete) REST interface for GS-O/SDN-O/NFV-O and Maintain the relationship of resource instance include the resources change notification.

### **Evaluation**

OPEN-O is an open source project backed by the Linux Foundation that enables telecommunications and cable operators to effectively deliver end-to-end services across Network Functions Virtualization (NFV) Infrastructure, as well as Software Defined Network (SDN) and legacy network services.

It has a consistent Governance structure and aligned with different developers such as: Huawei, Ericsson, ZTE, China Mobile, China Unicom, Gigaspaces, HKT, Intel, VMware, BOCO Inter-Telecom, Canonical, Solutions Cloudbase, Infoblox, Raisecom, Red Hat.

Different approved project under development (around 14 projects) with different developers from the market.

It has a consistent roadmap evolution and releases.

### **Summary**

As we saw about Open-O:

- Enable operators to capitalize on NFV and SDN architecture
- Multi-domain, multi-location, and end-to-end service composition
- Adopt industry-wide common information model and TOSCA/YANG (RFC 6020) data models to ensure extensibility, implementation efficiency, and interoperability
- Provide a common platform for VNF vendors, simplifying and accelerating VNF onboarding, development, and deployment
- Enhance the overall service lifecycle, through model-driven automation to enable reuse and incremental upgrades

- Provide abstraction to operate over diverse SDN, NFV, and legacy network.
- It is free Open source by Linux Foundation with strong market support and development.

### **2.5.1.7**      *Cloudify orchestrator*

#### **Excellence**

Cloudify enables to orchestrate an entire application lifecycle by modelling a topology once, and then deploying it to a target infrastructure of choice, all while managing, monitoring, scaling and healing everything in the same place. Cloudify is OpenStack native and works out-of-the-box with all the OpenStack APIs from Neutron through Murano.

Cloudify is open-sourced software, and its scope includes provisioning, configuring, and monitoring. It uses the Topology and Orchestration Specification for Cloud Applications (TOSCA), an emerging modelling language whose mission is to be able to model every aspect of an application. In terms of where Cloudify fits into the ETSI NFV MANO architecture, it provides a network functions virtualization orchestrator (NFVO) and a virtual network functions manager (VNFM). [60] [61]

#### **Evaluation**

Cloudify, based on TOSCA claims to be [62] the only real-world and most extensive technology agnostic implementation of ETSI's MANO specification. This is witnessed by Tier-1 telecom operators selecting Cloudify as their tool of choice for NFV orchestration. For example, there is the work done by Orange Labs using Cloudify for orchestration in their vIMS POC (and even fork the code). [63]

#### **Summary**

As described above Cloudify offers:

- Open source solution, which implements the ETSI's MANO specification.
- It offers standardized APIs and provides an NFVO component.
- TOSCA modelling languages is a first-class citizen for describing deployments.

### **2.5.1.8**      *Juju*

#### **Excellence**

Juju by Canonical [64] enables application developers and cloud administrators to publish various cloud-ready applications in a browsable categorized catalog. Charms are sets of scripts for deploying and managing services. With event handling built in, they can declare interfaces that fit charms for other services, so relationships can be formed. Bundles are collections of charms that link services together, so a user can deploy whole chunks of app infrastructure in one go.

#### **Evaluation**

Juju technology is already integrated in OSM as an implementation of the VNF Configuration & Abstraction component, as the Figure 16 shows. [65]

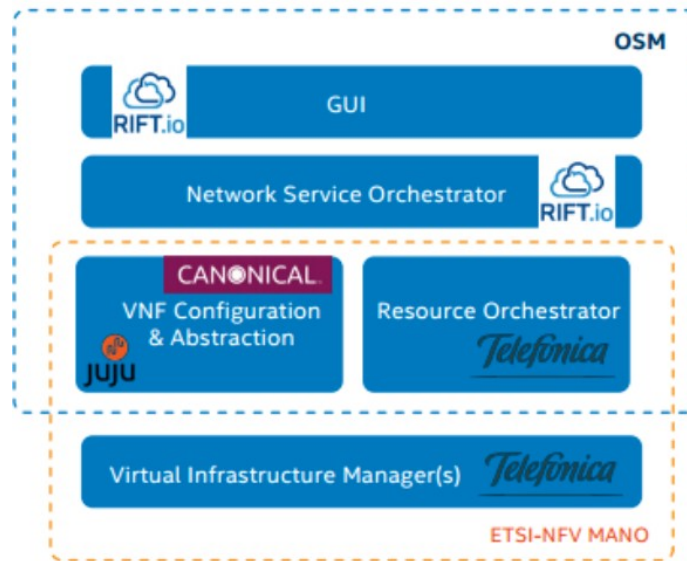


Figure 16: Juju in ETSI-NFV MANO Structure

### Summary

The main features of Juju are summarized next:

- Open source project.
- It provides open and compliant APIs.
- It is integrated with OSM for VNF configuration and management.

#### 2.5.1.9 *Murano*

### Excellence

OpenStack Murano [66] introduced an application catalog to OpenStack, which allows application developers and cloud administrators to publish their cloud-ready applications in a browsable, easily navigable and categorized catalog. With Murano, the new backend provides roles and permissions to enable the administration of the applications - so you can have an application developer role (who develops an application and uploads it to Murano) a cloud user role (who uses the uploaded applications) and an administrator role who manages the Murano service itself. This simplifies the management of apps on OpenStack significantly. Murano makes all this possible on OpenStack. However, when applications are deployed or shared between multiple clouds or not deployed on OpenStack at all, Murano cannot do much for you.

### Evaluation

Mirantis (which develops and maintains Murano) places Murano as an NFV orchestration project in OpenStack. The actual deployment itself will be done by the existing software orchestration tools (such as Openstack Heat). [67] Mirantis will contribute Murano application catalog models along with open source software which is currently used in production NFV deployments to help accelerate OSM's Resource and Service Orchestrators. Mirantis, one of the Openstack Foundation Board members, is working on creating a liaison between ETSI's OSM and Openstack's Murano Project. The initial project announced will integrate Telefonica's OpenMANO software with RIFT.io's Canonicals' Juju-generic VNF Manager, and the Rift.io orchestrator.

### Summary

The main features of Murano are summarized next:

- Murano seems an active project supported by Mirantis.
- It targets the NFVO component and OSM's Resource and Service Orchestrators.
- The liaison between ETSI's OSM and Openstack's Murano Project is still ongoing.

---

## **2.5.2 NFV Infrastructure**

This subsection presents the main projects and open source technologies that have been identified to be of potential interest to support the functionalities of a NFV infrastructure, as defined by the ETSI NFV architectural framework.

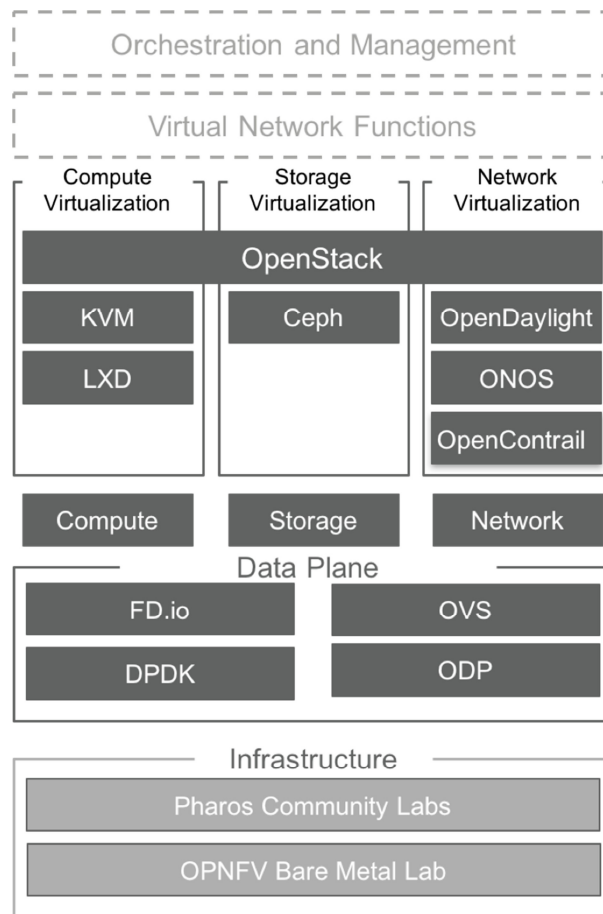
### **2.5.2.1 OPNFV**

#### **Excellence**

Open Platform for NFV [68] (OPNFV) is a collaborative project at Linux Foundation [69] supported by several important telco companies like Cisco, Intel, DELL, Nokia, Red Hat, SUSE and IBM, among others (the complete list of members can be found in [70]). OPNFV works closely with ETSI NFV ISG, IETF (Internet Engineering Task Force, [71]), MEF (Metro Ethernet Forum, [72]), among others Standards Developing Organizations (SDOs) and aims at providing a carrier-grade, integrated and open source platform based on open standards and software to establish an open environment for NFV solutions, while allowing continued integration, automated implementation and testing.

To accomplish this, OPNFV is committed to an 'Upstream First' methodology, integrating components from upstream projects such as OpenStack or OpenDaylight. With this philosophy, OPNFV adopts diverse upstream open source projects and devotes development resources towards integration and testing activities, instead of new developments, avoiding proprietary forks of the adopted products and components.

According to this philosophy, OPNFV provides a platform which can support a set of use-cases including: (1) lifecycle management of virtual network functions (VNFs), which include deployment, instantiation, start and shutdown, among others; (2) specifying and interconnecting VNFs, virtual network function components (VNFCs) and physical network functions (PNFs); (3) instantiating VNFs dynamically to meet the current performance, scale and networks bandwidth needs; (4) detecting faults and failure in the NFVI, VIM and other components of the infrastructure and recovering from those failures; (5) sourcing and sinking traffic from/to PNF to/from VNF; (6) and hosting different VNF instances from different vendors on the same infrastructure using the NFVI as a Service.



**Figure 17: OPNFV Platform Diagram (Release Colorado)**

Figure 17 shows the architecture of the current release of OPNFV, Colorado, at the time of writing. The figure represents components from upstream projects that are integrated in the platform, such as OpenDaylight, ONOS, OpenStack, Ceph, KVM, Open vSwitch and Linux to provide the NFV Infrastructure (NFVI) and Virtualized Infrastructure Management (VIM). As mentioned in the OPNFV documentation [73], the portfolio has been extended in the current OPNFV release to include Management and Network Orchestration MANO components primarily for application composition and management.

In addition, a key part within OPNFV platform is the Pharos Community Labs project [74] as well as OPNFV bare metal lab infrastructure hosted by the Linux Foundation. The former provides a federated NFV testing infrastructure of community labs around the world and aims at testing of the OPNFV platform, hosting continuous integration/continuous deployment (CI/CD) and helping to ensure OPNFV applicability across architectures, environments and vendors through a collection of labs and broad range of hardware.

**Evaluation**

OPNFV is an open source project where participation is not only open to organizations and companies, but also to end users and developers (even considering the connection of end user experimentation environments to the federated Pharos infrastructure).

As such, the goal of OPNFV is not to develop a specific NFV solution. Instead, OPNFV aims at building a reference NFV platform to accelerate time-to-market for NFV products and services, while facilitating the development and evolution of NFV components within relevant open source projects. The work on the OPNFV platform considers the following key aspects: openness, leveraging open

interfaces to enable the integration of diverse NFV components developed in upstream projects; integration, focusing on the configuration and deployment aspects that support the operation of the multiple hardware and software components that comprise a NFV infrastructure, the deployment and testing of OPNFV experimentation scenarios and the incorporation of new projects into the OPNFV platform; robustness, taking account diverse aspects related with scalability, throughput, fault tolerance, security, high availability, support of multi-site capabilities and upgrade-ability; and agility, by supporting appropriate DevOps continuous integration and deployment methodologies.

Source code is under Apache License 2.0 [57] and the current release of OPNFV (Colorado 3.0, 12/08/2016) is available through the OPNFV website [75]. Complementary to this information, OPNFV provides consumable releases every six months.

Given the commitment of OPNFV to an ‘Upstream First’ methodology, OPNFV integrates diverse components, products and solutions that may be of relevance to 5GinFIRE partners, making it an appealing solution to be considered within the project scope.

### **Summary**

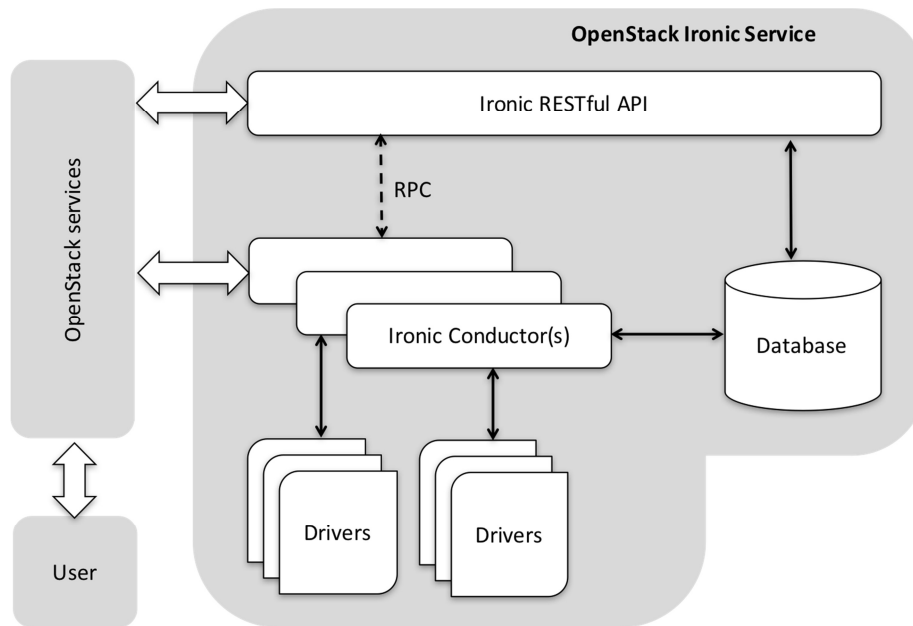
The main features of OPNFV are indicated below:

- Collaborative project at Linux Foundation supported by relevant telco companies.
- Carrier-grade, integrated and open source platform based on open standards and software to establish an open environment for NFV solutions.
- ‘Upstream First’ methodology.
- Source code licensed under Apache License 2.0.

#### **2.5.2.2 *Ironic (OpenStack)***

### **Excellence**

OpenStack bare metal provisioning (Ironic) [76] is an OpenStack project aiming at provisioning bare metal (physical) machines instead of virtual machines. Ironic may be used as a standalone software program, or it may be integrated with other OpenStack services (e.g., Nova). As mentioned in the OpenStack documentation [77], there are several use cases where the use of physical machines may be of relevance, such as supporting the rapid deployment of a high-performance cloud infrastructure or executing tasks that require specific-purpose hardware, among others.



**Figure 18: Logical architecture of Ironic**

Figure 18 shows the logical architecture of the OpenStack Ironic architecture, which comprises the following components:

- The Ironic RESTful API service. It may be used by a cloud administrator or other OpenStack services (e.g. Nova) to incorporate and interact with the physical machines managed by Ironic.
- The Ironic conductor service. This is the main entity in charge of handling the provision of physical machines per user requests. Multiple instances of this service running at separate hosts may be used, for the sake of redundancy and to support diverse sets of drivers that enable the operation of the available hardware.

To support operations related to bare metal provisioning, OpenStack Ironic makes use of diverse technologies and protocols, including: (1) Preboot Execution Environment (PXE) [33], an industry standard specified by Intel and SystemSoft to configure and boot a physical system over a communication network; (2) DHCP [34], to distribute the network information that is necessary to support bootstrapping procedures; (3) TFTP [35] to download the software that is required to boot a physical server (i.e., the Network Bootstrap Program); and (4) Intelligent Platform Management Interface (IPMI) [36], to support out-of-band management and monitoring of systems (e.g., remote power on).

## Evaluation

The source code of Ironic and complete documentation is available through the OpenStack website [76]. As an OpenStack project, the parameters relevant to the evaluation of this solution have been covered in Section 4.

## Summary

In the following, we enumerate the main characteristics of Ironic:

- OpenStack project aiming at provisioning physical machines.
- It may be used standalone or integrated with other OpenStack services.
- It makes use of standard technologies and protocols: PXE, DHCP, TFTP, and IPMI, among others.
- It is distributed under the Apache License 2.0



### **2.5.2.3      *Metal as a Service (MAAS)***

#### **Excellence**

Metal as A Service (MAAS) [78] is a hardware provisioning software from Canonical that aims at allocating physical servers to execute user applications, automating the deployment and dynamic provisioning of computing environments. As mentioned in MAAS documentation [79], MAAS turns bare metal into an elastic cloud-like resources and allows to manage a large number of physical machines by discovering, commissioning and deploying physical servers; matching workloads requirements by the dynamically re-allocation of the physical resources; and making physical resources available for future workloads by their retirement when they are no longer required.

In addition, MAAS is designed to be integrated with Juju (see section about Juju, previously commented in this document) to deploy complex services and scale them up or down as demand requires.

#### **Evaluation**

MAAS does not fully support the open-source paradigm, therefore it will not in principle be considered in the scope of 5GinFIRE.

#### **Summary**

The main features of MAAS are listed below:

- It aims at allocating (bare metal) physical servers.
- Up-to-date comprehensive documentation.
- As already commented, it does not fully support the open-source paradigm.

### **2.5.3      *Network Service Descriptors: Models and Catalogue solutions for Describing and Configuring service topologies***

---

The 5GinFIRE DoA defines the Application Composer Toolkit which will be used by end-users to create the composition of services needed to instantiate an application that will contain the Artefact Under Test (UT) (e.g. a service, VxFs, etc.). This request will be transformed to the service composition. This section explores technologies, specification and models that can be processed by orchestration engines as service templates to instantiate applications. In case of 5GinFIRE such technologies could be used to describe the service under test as well as the setup of the experimentation environment.

Network Service Descriptors can be described by various format and languages. For example, there is OpenStack's Heat Orchestration Template (HOT), TOSCA, CloudFormation(Amazon), Juju charms. Although there is no clear winner, TOSCA seems to win in popularity and acceptance due to involvement and support by many telcos.

#### **2.5.3.1      *TOSCA***

#### **Excellence**

OASIS has formed the Topology and Orchestration Specification for Cloud Applications (TOSCA) technical committee to from the TOSCA standard. Its focus is to create an interoperable specification based on a simple profile. Main activities are around:

- NFV and SDN which explores the description of Network Services composed of Virtual Network Functions. TOSCA on this topic work closely with ETSI NFV standards Working Group

- Definition of an instance model which provides a representation of a service deployment based on TOSCA
- Containers and Clustering which explores service descriptions for containers as well as consider the deployment
- Monitoring which proposes potential implementation details

A most notable deliverable from the TC today is the TOSCA Simple Profile in YAML Version 1.0 Committee Specification 01 [80]. The deliverable defines the authoring of TOSCA service template by using YAML.

### **Evaluation**

The TOSCA standardization effort is supported by many companies like IBM, Red Hat, VMWare, etc. Nevertheless, most notable and related to 5GinFIRE are the following efforts found so far:

- ARIATOSCA [81] is an open source implementation, for the command line, that can be used to translate TOSCA based orchestrations
- Openstack support. There is an official TOSCA Parser [82]
- OPEN BATON support: There is an official TOSCA support [83]
- Rift.io: as of version 4.2.2.0 in related Information model translation [84] RIFT.ware can read in MANO descriptors that are in TOSCA or YAML format. A developer can on board these descriptors in both NS AND VNF packages to the Launchpad.
- Juju: There is a juju-Tosca implementation [85] to import/export TOSCA specification, but seems rather old
- Openstack Murano seems not yet to support TOSCA, although it is in its roadmap [86] although it seems to be a solution on using Murano with Cloudify [61]

### **Summary**

The main features of TOSCA are summarized next:

- Promising for the use as an experiment descriptor.
- It is already being supported by many tools both commercial and open source.

### **2.5.3.2 The YANG data modelling language**

#### **Excellence**

The YANG data modelling language is used to model configuration and state data manipulated by NETCONF. The NETCONF protocol is a formal application programming interface (API) that allows configuration data information to be retrieved and manipulated. NETCONF provides full set of semantics for configuration management. YANG provides many features specific to configuration management.

#### **Evaluation**

While TOSCA is very good supporting the latter requirements, Netconf/YANG is very good at configuring network devices. Combining those two could support of the following:

- Provisioning IaaS components which mainly include compute network and storage. (This would be done with the TOSCA-based templating)

- Configuring network devices which include lots of configuration details per device type (This would be done via Netconf/YANG modelling)
- Topology-driven monitoring (TOSCA)
- Day to day operations that are performed on the topology graph like upgrades, self-healing scaling etc. (TOSCA)

Moreover, a TOSCA orchestration can communicate with Netconf/YANG-based components and drive network configurations and orchestration via Netconf/YANG-based products. In this way, we can leverage the best of the two worlds.

### **Summary**

The main features of YANG are summarized next:

- Mature data modelling language.
- It cannot be used to model a deployment, together with a descriptor.
- Like TOSCA could offer much more powerful scenarios.

### **2.5.4 5GinFIRE Candidate(s)**

---

The following table presents a summary of the technologies covered in this section, reflecting their contributions in the scope of the NFV reference architectural framework defined by ETSI:

**Table 2: NFV Comparative Analysis**

	NFV MANO			NFVI	Maturity & Applicability	Open Source license
	NFV Orchestrator	VNF Manager	Virtualized Infrastructure Manager			
<b>OSM</b>	Provides a Network Service Orchestrator (NSO), based on <b>RIFT.ware</b> , and a Resource Orchestrator (RO) based on <b>OpenMANO</b> ; allows the deployment of multi-site network services that span across multiple datacentres	Includes a VNF Configuration and Abstraction layer, based on <b>Juju Charms</b> from Canonical.	Includes <b>OpenVIM</b> as part of the OSM runtime environment; the current release also supports OpenStack and VMware vCloud Director; follows a plugin model to support the addition of other types of VIM.	-	High maturity. Open source. Two successful releases, including a commercial distribution by RIFT.io.  Directly applicable and almost direct feedback channel to the community  Apache License 2.0	Yes
<b>Open Baton</b>	Provides a NFVO implemented following the ETSI MANO specification	Includes a generic VNFM; may interoperate with VNF-specific VNFMs	Integrates with <b>OpenStack</b> ; provides a plugin mechanism for supporting additional VIMs	-	Medium. Academic project. Announcements of collaboration with OSM	Yes
<b>ECOMP</b>	-	Provides an enhanced generic VNF manager	-	-	Unknown. Code not yet available	No
<b>Open-O</b>	Provides a NFV Orchestrator based on ETSI NFV MANO; includes	Interworks with multi-vendor	Interworks with	-		Yes

	a network service life cycle manager, a NFV resource manager and a NFV monitor.	VNFMs	different VIMs		Low. Recent first release, as an integration of initial, seed components	
<b>Cloudify</b>	Provides a NFVO	Includes a VNFM	Integrates with <b>OpenStack</b>	-	High maturity. Cloudify Community version is open source. There is a premium commercial distribution for telcos. Unsure about feedback opportunities	Yes
<b>Murano</b>	Currently exploring the establishment of a liaison with ETSI <b>OSM</b> , with the target of helping to accelerate OSM’s NSO and RO.	-	Introduces an application catalogue to <b>OpenStack</b>	-	Medium. In the process of positioning in the MANO open source landscape	Yes
<b>OPNFV</b>	In the Colorado release, it includes MANO components primarily for application composition and management		Main focus on building NFVI and VIM; integrates components from upstream projects ( <b>ODL, OpenStack, Ceph Storage, KVM, Open vSwitch, and Linux</b> )		High. Fourth release about to be launched. Very active community and good prospects for feedback Apache License 2.0	Yes
<b>Ironic</b>	-	-	-	Aims at provisioning bare metal (physical) machines instead of virtual machines; may be used standalone, or may be integrated with other <b>OpenStack</b> services (e.g., Nova).	Medium, though it has a strong traction within the OpenStack community  Apache License 2.0.	Yes
<b>MAAS</b>	-	-	-	Hardware provisioning software from Canonical that aims at automating the deployment and dynamic provisioning of computing environments; designed to be integrated with <b>Juju</b> to deploy and scale complex services.	High maturity. Provided as commercial distribution. Feedback opportunities may be limited, though the strong implication of Canonical in OSM could help does not fully support the open-source paradigm.	Yes

The technical solution adopted in 5GinFIRE for the orchestration of network services, consists in the implementation of a single orchestration domain, where a NFV orchestrator manages and coordinates the creation of end-to-end network services, potentially involving multiple VNFs, across the diverse experimentation infrastructures provided by the project partners. With this purpose, the NFV orchestrator of 5GinFIRE will interact with the appropriate VIMs available at these experimental infrastructures, coordinating the allocation and setup of the computing, storage and network resources, which are necessary for the instantiation and interconnection of VNFs, Each partner running an experimental infrastructure will be in charge of the deployment and maintenance of a VIM, compliant with the necessary interfaces, as defined by the 5GinFIRE reference architecture.

According to our analysis of the main practical implementations of NFV MANO, available under the umbrella of open source projects, the technology chosen to provide the management and orchestration functionalities envisaged in 5GinFIRE is Open Source MANO (**OSM**). The main reasons that have motivated our selection are: (a) it is an ETSI-hosted project that provides a comprehensive and operational implementation of an NFV MANO software stack; (b) it is supported by a large community of leading network operators, NFV cloud providers and research and academic centres, which includes the direct involvement of two 5GinFIRE partners, i.e. Telefónica, as an OSM member, and UC3M, as an OSM participant; (c) it supports a plug-in model that allows integrating the diverse VIM solutions considered as 5GinFIRE candidates in the analysis presented in section 2.3.1 of this document (release ONE of OSM currently includes a reference VIM, i.e. **OpenVIM**, and supports **OpenStack** and **VMware** vCloud Director); (d) it offers the level of maturity and stability required for 5GinFIRE, as indicated by the availability of commercial distributions (e.g., *rift.ware*); (e) it supports the automated instantiation of end-to-end network services across multiple sites, each with a supported VIM; and (f) OSM source code is available to 5GinFIRE partners, granted under an Apache license, with simple installation processes and comprehensive documentation and support from the OSM community.

With respect to NFVI solutions, in addition to well-known open source platforms, 5GinFIRE will also consider the work being done under the umbrella of **OPNFV**, as a solution of particular interest that provides an NFVI platform integrating components from upstream projects that are relevant in the context of NFV. Furthermore, 5GinFIRE will consider the utilization of solutions for bare metal provisioning, particularly **Ironic**, given its capability to be used not only as standalone software program, but also integrated with other OpenStack services.

Using Open Source MANO (OSM), 5GinFIRE partners benefit from a production-level MANO stack supported by a wide open-source community. This MANO stack will be deployed on 5TONIC, and maintained and regularly updated with the latest functionalities and corrections. Some development shall be considered to address the specific 5GINFIRE requirements when they are needed. Among these, it might be considered to support a mapping from TOSCA-based NFV descriptions into OSM-supported YAML. This last development depends on the availability of specification provided by the TOSCA community for VNF descriptors based on the TOSCA model, which is currently being pursued by both OASIS and ETSI NFV. Once the 5GINFIRE MANO instance is deployed on the 5TONIC infrastructure, additional components like the portal are still to be integrated and some efforts to be considered on this aspect, in order to provide specific functionalities to 5GINFIRE experimenters to describe their use cases. Tools to develop this portal were not discussed in this state of the art but will describe later in the architecture section.

---

## **2.6 FIRE**

---

As presented in the DoW and in the workflow of Figure 1, 5GinFIRE will be as much as possible interoperable with existing FIRE standards and facilities. As the most notable effort in FIRE is the FED4FIRE initiative (FED4FIRE+ is its current funded EU project name) we present here some representative technologies that are considered to be used within 5GinFIRE, namely the Authentication scheme and RSPECs.

### **2.6.1 FED4FIRE and Authentication**

---

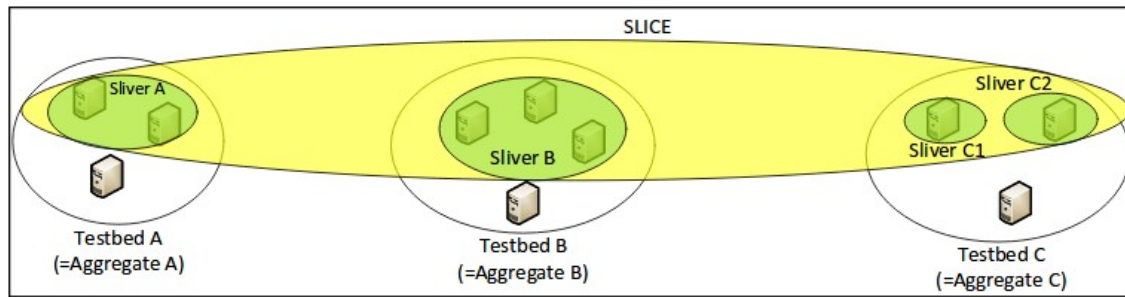
5GinFIRE will investigate if possible to federate with other FIRE testbeds via the Fed4FIRE AAI technology, thus accepting seamlessly FIRE users allowing the creation of experiments and facilitate integration of existing FIRE facilities. Fed4FIRE testbeds can grant access to actors outside of the federation if they trust them. Two are the most considerable components from FED4FIRE:

- Identity provider: An Identity Provider (IdP) is run by an organization to provide experimenters and services working on their behalf with the means to authenticate themselves within a security domain. Organizations register Experimenters and services with IdPs through trusted registration processes. Testbeds can deploy their own IdP (testbed A) or rely on 3rd party IdPs, for example those operated by the Federation Facilitator. IdPs issue security tokens to subjects that codify signed assertions about them, which can be presented to services at the point of use. The services then enforce policies related to level of trust in the IdP and the subject by associating rights (or declining rights) to each security token.
- Certificate directory: Each IdP provides a root of trust for subjects within a security domain. Each testbed must decide which IdP they trust to make assertions about subjects in authentication and authorization decisions. The current architecture relies on a public key infrastructure where the root of trust in each security domain is a Certificate Authority. The Certificate directory provides a mechanism to distribute root certificates for IdPs and avoid the need to manually exchange certificates between testbed providers and experimenters.

### **2.6.2 Concepts slice, sliver and Rspecs**

---

5GinFIRE would like in future to expose its available infrastructure and repository to other FIRE facilities. RSpec seems as a good candidate. FED4FIRE has already introduced the concepts of Slice, Sliver and RSpec. A slice is the tool that allows the connection of resources when they belong to the same experiment, often when the experiments use multiple testbeds. The sliver is the part of the slice containing information about a single testbed. Slivers use an RSpec (Resource Specification) on a single testbed to define the sliver on the testbed. The RSpec and thus the sliver can contain multiple resources. Note that there is an important difference between the concept sliver as defined in the Aggregate Manager API v2 and v3.



**Figure 19: Concepts of slices, slivers [49]**

Version 2 of the AM API supported only one sliver per slice per testbed [42]. Hence in this version of the AM API a sliver denotes all the slice's resources on a testbed. (e.g. the `deletesliver` call deletes all resources of a slice on a single testbed). In the figure above Sliver C1 and C2 cannot exist in AM API v2 (of course, a testbed can decide to work internally as such, but the slivers are not individually addressable through the API).

In version 3 of the AM API, one of the several important problems that was solved is that of adding resources of a testbed to a slice. In AM APIv2, as only a single sliver is supported per testbed and slice, it was not possible to add extra resources to this same slice for that testbed (and remove them afterwards). For this, API v3 now supports addressing of individual slivers, and the SFA calls such as `allocate`, `provision`, etc. can now work on multiple slivers on a single testbed. For instance, the call `allocate` returns a struct of slivers.

In practice, it is a little more complicated since testbeds have the total freedom to adopt one of multiple methods when allocating resources to slivers [44]. If you request an RSpec with multiple resources, a testbed can decide to create multiple slivers, only one, or any allocation strategy in between. This has then as a consequence that it depends on the testbed if an experimenter can remove single resources/slivers or only everything at once. E.g. on the virtual wall testbed, if you allocate 2 nodes with a link, you get back 3 slivers (2 for the nodes and 1 for the link).

## 2.7 Conclusions

The state of the art concerning Cloud, SDN, NFV and SFC described in the previous chapters has permitted to select the technologies and components we will use along 5GinFIRE project. This analysis was necessary and even required in order to define in this document a first version of the 5GinFIRE architecture and its use cases.

We considered different criteria to make those choices: Open Source, Maturity, API and standard compliance, features and the capacity to be mastered by Partners. We confronted them to different popular technologies or components of the moment and we made our selection. Those criteria were in one side generic and in another side specific four our project.

We selected:

- OpenStack, which is one of the biggest open source projects. It is the central player in the cloud technology.
- OpenDaylight, which is the most mature as well as the most SDN controller known by the partners. SDN is still an emerging technology so our choice is wise.
- Open Source MANO, not only because it's fitting our standards but because also we will profit from the knowledge of some of our partners who are leading this project.



Orchestration services are very new and we are still in the experimental phase. Along with OSM, comes a set of tools and components that will be used within 5GinFIRE because they are necessary for OSM to operate: Rift.ware, OpenMANO, Juju, Murano, Ironic and MAAS.

## **3 5GinFIRE Use Cases**

### **3.1 Introduction**

---

In the project we will automate the process of combining infrastructures (e.g. NFV, ITS and IoT) to make the experimentation environment readily available for experimenters around the world.

To define the architecture requirements, we consider a specific set of use cases that will represent the basis set of interactions between vehicles, IoT, and emergency services, which can be provided through virtual network functions in a shared infrastructure.

In order to account for a diverse set of requirements satisfied by 5GinFIRE environment, we will also investigate another type of vertical as an EVI candidate. This will be the Smart Cities use case which will encompass a set of sensors for both monitoring and actuating in a city, through municipality development with a large set of public policies, and investigating how the corresponding infrastructures could be integrated in the overall 5GinFIRE concept. These use cases will drive the 5GinFIRE architecture, some of them being deployed in the first phase of the 5GinFIRE (e.g. automotive use case) and some using specific inputs from the open calls (e.g. smart cities use case).

In this section we will provide detailed requirements of the Automotive EVI and of the Smart City use cases, and will identify scenarios and most promising VxF components of the automotive industry and the smart cities IoT environment to experiment our architecture.

### **3.2 Use Cases**

#### **3.2.1 Automotive Domain**

---

##### **3.2.1.1 Overview**

We will enhance and automate the process of combining infrastructures (e.g. NFV, ITS and IoT) for making the experimentation environment readily available for experimenters around the world.

To define the architecture requirements, we consider a specific set of use cases that will represent the basis set of interactions between vehicles, IoT, and emergency services, which can be provided through virtual network functions in a shared infrastructure.

One of the project's use cases makes use of both ITS, IoT, and that can be made available through SDN/NFV: assisted overtaking vehicle.

The objective of the assisted overtaking is to gather all needed information in real-time to a vehicle to make a decision of overtaking or not the front vehicle, while the visibility of the road from the vehicle is not sufficient to take a decision. Important information will be used to take the decision, such as: video images from the road ahead, vehicles ahead, obstacles, road conditions, crosswalk, turns, people and bicycles in the road, and even traffic signs such as stops and traffic lights.

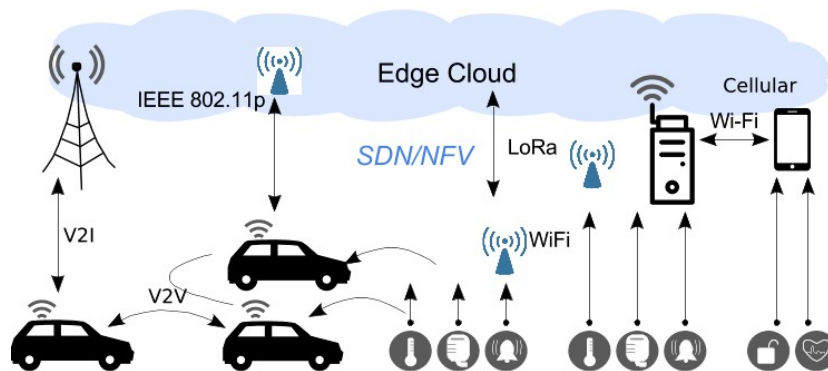
Each vehicle will have access to its own information such as velocity, GPS, camera and internal sensors. This information will be used by the embedded computer of the car to take local decisions, and it can also be advertised to the other vehicles. Each vehicle will also have access to information from the street and surroundings through video cameras on the other vehicles and sensors (people, bicycles, crossing roads and traffic lights, vehicles in the road, bad conditions in the road). This information can be sent and disseminated through vehicles (through DSRC or future 5G device-to-device (D2D) technology), or to a WiFi, DSRC or cellular station (or a future 5G station) in the street, and then be disseminated to the vehicles in the area.

With all this information, each vehicle can access its own information and the one gathered through other vehicles. Vehicles will use this information to provide assisted driving: vehicles take individual decisions according to its information and the one coming from other vehicles: cameras and sensors.

To design and implement such a scenario, several enabling capabilities and infrastructure will be provided. We consider the following interactions to gather and provide information:

- Sensors to the edge-cloud: send sensors information to be processed in the edge together with the vehicles and people information;
- Cameras and sensors to the vehicles: send information to the vehicle, such as real-time video images, strong brakes, malfunctions;
- Vehicles to and from the edge-cloud: send information about each vehicle and its sensors to the edge-cloud;
- Vehicles to vehicles: send information about each vehicle cameras and its sensors to other vehicles;
- Edge-cloud to vehicles: send information about the overall environment in the surrounding after fast processing, so that the vehicle can take decisions on the safe overtaking.

The next figure depicts some of these interactions and the elements involved in the road area.



**Figure 20: Interactions between sensors, vehicles and access points/road side units**

There are several service components required in the architecture to implement this use case. These are represented in the following figure.

The ITS and IoT infrastructures provide the verticals specific of each physical infrastructure. Information and processing are run in edge computing systems, which are composed by access points (APs), ENodeBs, servers, providing virtual functions through SDN and NFV. The core path is de-structured through SDN and NFV to enable flexibility, providing accommodation of new information and functions in an autonomous approach. The several running experiments in the vertical infrastructure will then be provided through experimental vertical instances, being deployed to both ITS and IoT, and open to build new instances.

This architecture will support the right functions at the right time, in the edge network and in embedded routers in the car:

- Interaction between car components and the information from the outside to take local decisions.
- Network and communications: V2V, V2I (through edge cloud).
- Video cameras and sensors integration: connected to access points or road side units, which can be stored in the mobile edge computing cloud in the street.
- Configuration, management, deployment tools to serve the actual experimental needs.

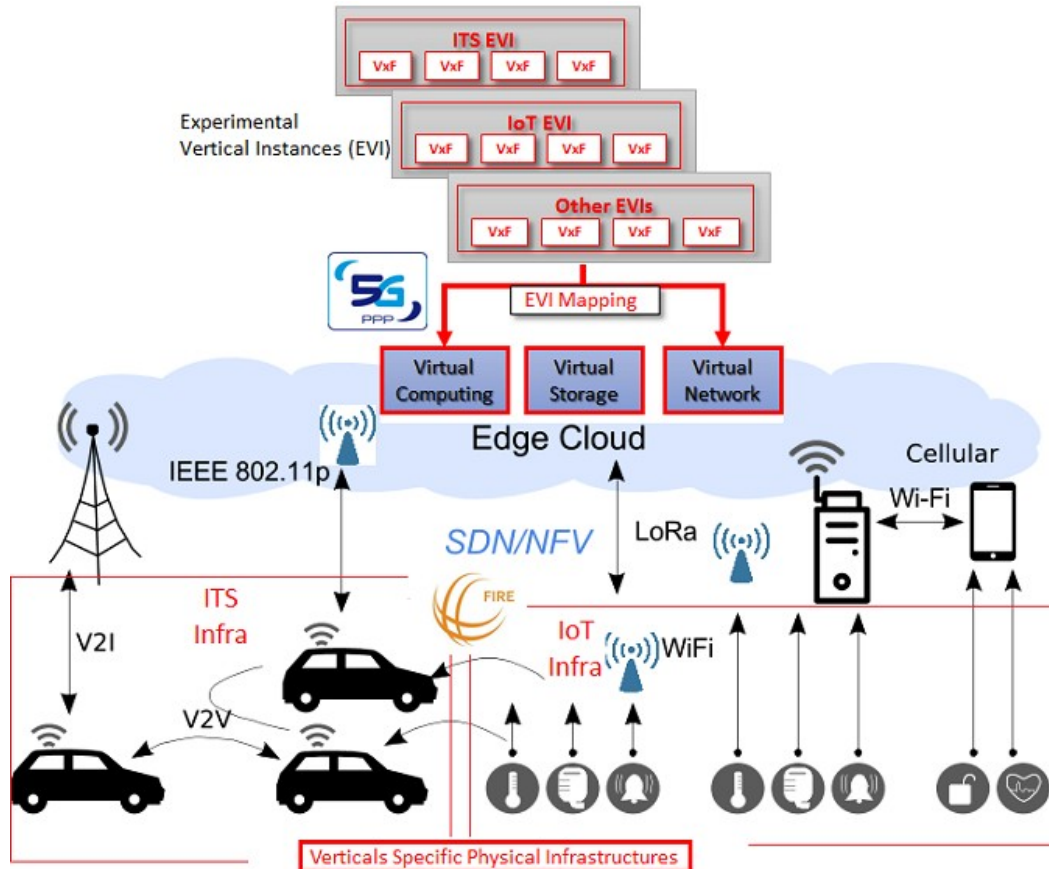


Figure 21: 5GINFIRE architecture in the ITS and IoT infrastructure

### 3.2.1.2 Use Cases

#### 3.2.1.2.1 Sensing-based Automotive Testbed for assisted driving

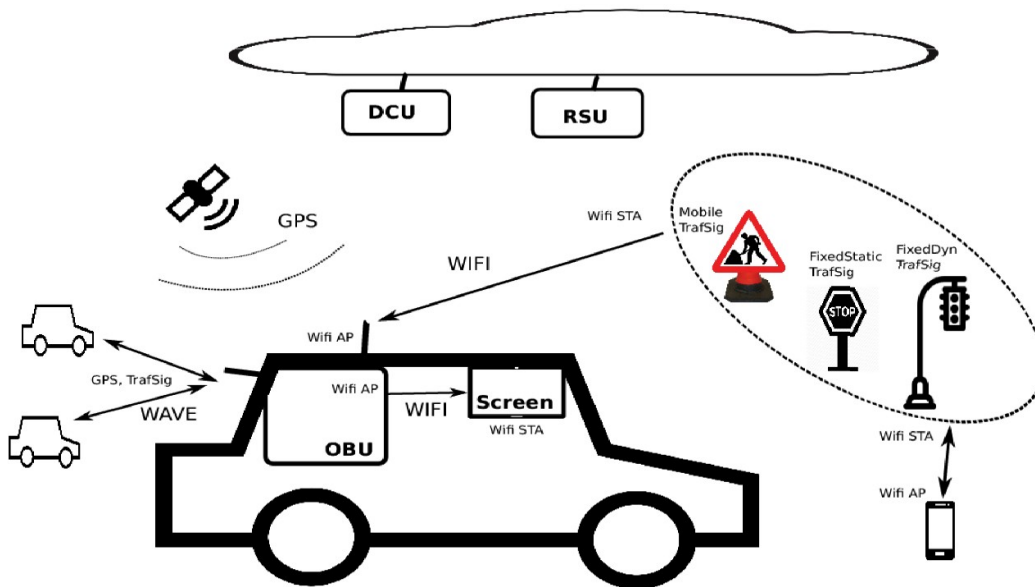
The concrete scenario for assisted driving is presented in Figure 22. Each vehicle contains an On Board Unit (OBU) that provides the communication between vehicles and between each vehicle and the infrastructure. The architecture of each OBU is described in section 3.3 (Testbed). The OBU also connects to an Android device, smartphone, through WiFi, that provides visual information for the driver.

The smartphone receives information from the OBU via WiFi in station mode. The OBU in the car receives a large set of information from the road-side units, related to vehicles traffic, traffic signs, road conditions, which is disseminated in a specific area.

The visual screen provides visual information about the main actors in car-assisted driving:

- Vehicle status which is presented in the car panel.
- Road/Highway status and traffic signs
- Environment status
- Traffic status

The visual screen will be implemented in a smartphone and presents traffic/road/environment information in a visual way. This information will be used by the OBU to take decisions on driving, and more specifically on overtaking, and the decision will be presented in the visual screen.



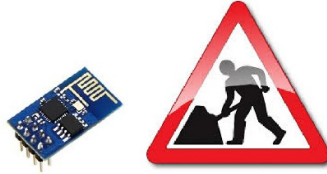
**Figure 22: Scenario for assisted driving**

Each OBU in the vehicle contains a GPS that provides its location. This information is sent between vehicles in beacons that are exchanged in the IEEE 802.11p interface. The vehicles can also exchange information related to the road sensors, traffic signals and environment sensors. This information is collected from the OBU in the vehicle and visually presented in the visual screen. The information is also used by the OBU to determine the obstacles, the road conditions, the positions of the other vehicles, and the restrictions of the roads.

The traffic signaling status integrates the road/highway status, such as restrictions on velocity, traffic signs such as stops, traffic lights or traffic signs that advises for road constructions. We consider 3 types of traffic signals (TrafSig):

- Fixed TrafSig (F-TrafSig): like a STOP signal.
- Fixed but Dynamic TrafSig (D-TrafSig): like a traffic light.
- Mobile TrafSig (M-TrafSig): like a “working” traffic signal.

These signals are initially emulated through ESP8266 devices, as depicted in Figure 23. In a later stage there is the possibility to evolve to a real solution with integration with traffic signs (e.g. Siemens systems in traffic lights integration).



**Figure 23: Emulation of traffic signal devices**

The information from these traffic signals will be sent to the vehicle through the road side units, or through communication with the OBUs. In the case of fixed and static traffic signals, and even in the case of mobile traffic signals with static information, they send their information to the road side units through WiFi or LoRa. The fixed static traffic signals can even be uploaded to the edge cloud and be integrated in the area maps that will be downloaded to the smartphone, and also sent to the OBU in the vehicle. The dynamic traffic signal will need to interact directly with the vehicle through WiFi or LoRa, so that the vehicle is able to know the current and real-time status of the traffic light.

The data collection unit (DCU) contains several real sensors of temperature, humidity, carbon monoxide, wind and pluviometry of the several streets, and they will provide relevant information related to the risk of driving in those atmospheric and environment conditions. This information will be sent to the road side units through WiFi or LoRa, and the RSU will then update the maps with these conditions, and the OBU in the vehicles. The OBU will also consider this information to assist the velocity and how careful the driver shall be to minimize the risks.

As an overview, the following interconnections will be provided:

- RSUs (with IEEE 802.11p, WiFi and LoRa) that gather the information from other cars, the environment sensors and the traffic signaling sensors;
- OBUs (with IEEE 802.11p, WiFi and 4G) in each car that gather information from other cars, from the RSUs, and from dynamic traffic signals;
- DCUs (with WiFi and LoRa) which contain the information of the environment sensors;
- WiFi hotspots that gather the information from sensors and traffic signs;
- Ethernet backbone and Internet access to interconnect the road side units, some DCUs and WiFi hotspots;

For the specific use case it is important to be able to define experiments, start the log with the important information to get the experiment running details, and get all the decisions information to analyze the experiment.

To provide this approach, through the OBUs several types of information will be available and be sent to a pre-defined server through different logging modules:

- Location logging: GPS location, heading, and speed;
- Traffic logging: traffic collected or sent through the OBU: vehicle users' traffic, RSUs/OBUs traffic load, including their mean packet size and mean packets rate;
- Networks logging: network to which a specific OBU is connected, through which technology and the number of hops, connections characteristics, such as the signal quality/signal strength, number of visible 802.11p neighbours;
- Sensors and traffic signs information logging;

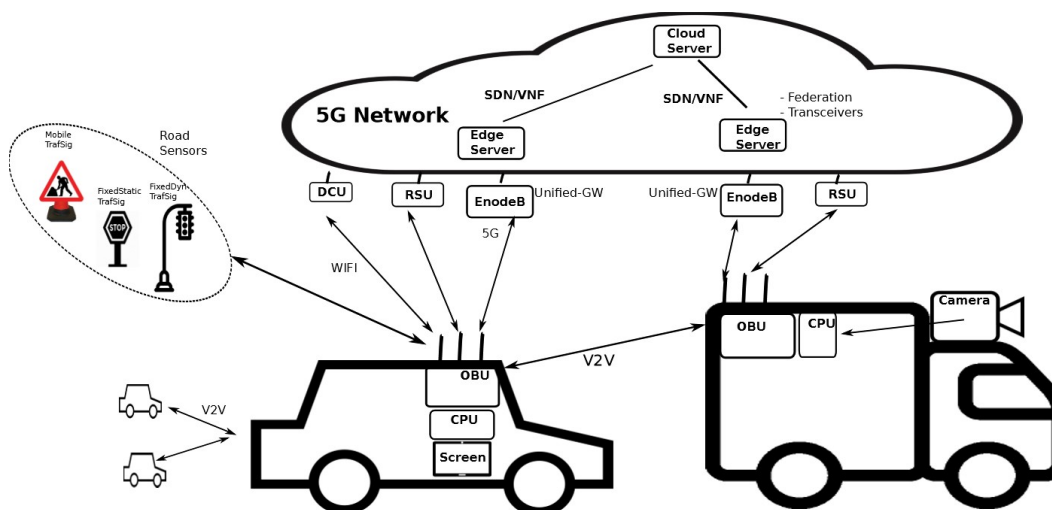
- Logging of information related to decisions on assisted-driving.
- In terms of configuration, several configuration modules will be made available:
- Define the sampling time of the extracted information (logging information);
- Define a specific server to receive the provided information (logging information);
- Define an experiment and configure the previous logging modules.

In this use case the 5G network will be required to be able to get and process all the received information and take the over-taking decision with a very small delay. The VxF will then be the sensing-based over-taking decision module, which can be implemented in the edge of the network connected to the road side units.

### 3.2.1.2.2 Video-camera-based Automotive Testbed for Assisted driving

The concrete scenario for video-camera-based assisted driving is presented in Figure 24. Each vehicle contains an On Board Unit (OBU) that provides the communication between vehicles and between each vehicle and the infrastructure. The OBU also connects to an Android device, smartphone, through WiFi, that provides visual information for the driver.

The vehicle contains a video camera of its front side. The video is streamed from the front vehicle to the rear vehicle through vehicle to vehicle communications, using IEEE 802.11p communication. The OBU in the car receives the video stream and sends it to the visual screen in the car, so that the driver can have real-time access to the visual information of the vehicle. This information will be used by the driver to take decisions on driving, and more specifically on overtaking.



**Figure 24: Scenario for video-camera-based assisted driving**

Both rear and front vehicles may belong to different brands and have no agreement between them, both in terms of communication (vehicle to vehicle) and in terms of joint services. This means that the transmission of the video streaming from the camera in the front vehicle to the visual screen in the rear vehicle may need a 3rd party agreement to agree on the service to be established between the vehicles. Moreover, the video quality to be chosen in the camera will depend on the capabilities of the video screen. Therefore, a transcoding system may be needed to adapt the video capabilities between both vehicles.

Both 3rd party service and video transcoding will need a 5G network approach: they will need a VxF to be available in the edge or even in the vehicle that will provide its capabilities with very low delay,

such that it will not interfere in the driving process. In this use case we consider that both approaches are possible.

When the vehicles are in range, the video streaming can be transmitted between them. However, before this happens, an application in the OBU sends a federation request to the 3rd party edge service, with the information on the video capabilities and the identification of the rear vehicle. This information can be sent from the OBU to the Unifier Gw, which will also be working as a VxF in the edge. The 3rd party module will contact, again through the Unifier Gw, the OBU on the rear vehicle to provide the agreements and understand the video capabilities. Then, it will perform the agreement and will instantiate the transcoding function between the vehicles. The video streaming will be activated and exchanged through vehicle to vehicle IEEE 802.11p communication. The driver will use the video to decide on the vehicle overtaking.

As an overview, the following interconnections will be provided:

- OBUs (with IEEE 802.11p) in each car that transmit the video streaming information;
- RSUs (Unifier Gw) (with cellular or 5G) that receive the request for federation and transcoding, and send the decision to the OBUs in the vehicles;
- RSUs (Unifier Gw) (through wired network) that contact the VxF for federation and transcoding;
- Ethernet/fiber backbone and Internet access to interconnect the road side units.

For the specific use case it is important to be able to define experiments, start the log with the important information to get the experiment running details, and get all the decisions information to analyse the experiment.

Through the OBUs several types of information will be available and be sent to a pre-defined server through different logging modules:

- Location logging: GPS location, heading, and speed;
- Logging of traffic from video streaming;
- Logging of traffic to and from the RSUs/Unifier Gw;
- Logging of information related to decisions on assisted-driving.
- In terms of configuration, several configuration modules will be made available:
- Define the sampling time of the extracted information (logging information);
- Define a specific server to receive the provided information (logging information);
- Define an experiment and configure the previous logging modules.

In this use case the 5G network will be required to be able to get and process very fast the federation and transcoding, so that the over-taking decision will be performed with a very small delay. The VxF will then be the 3rd party federation and the video transcoding, which can be implemented in the edge of the network connected to the road side units.

### **3.2.1.2.3 Functional safety and event recording services in virtualised multi-stakeholder network environments**

Functional safety is the part of the operational safety of a system that depends on the system and its components operating correctly in the event of human operator errors, hardware failures or environmental impacts, during normal system operation.



The objective of functional safety is the elimination of unacceptable risks of system failure and a resulting direct or indirect damage to humans, property or the environment

Functional safety is intrinsically end-to-end in scope and nowadays is accountable mainly on physical equipment such as electrical engines for aeroplanes, ships, electricity generator machinery and power plants. It is increasingly part of electronics equipment and extends in practice also to programmable systems.

However, complex ICT environments and future virtualised environments such as those being discussed in the context of ETSI NFV and 5G are still very far away from providing functional safety. In fact also current networks, such as 3G/4G or the Internet have largely not been designed with functional safety in mind.

Emerging, 5G influenced market segments expected to require functional safety in the near future are:

- the automotive industry with new technologies for auto-pilot requiring reliable navigation and traffic information
- the home automation segment of the CE industry being highly influenced by automation, well-being and security solutions requiring connectivity and operational reliability
- power grid network management and services provisioning, requiring optimal resources management for better customer prices and optimal utilisation of energy plants

Technical use case:

An end to end service includes a sensor (and actuator) responsible for monitoring (and reacting on) conditions in the energy grid. This device is networked via a mobile wide area access network, connected to a mobile edge cloud, connected via a core network, connected to a data-centre. How to ensure the functional safety across all segments in the chain?

Individual components in the chain may have own functional safety requirements to meet, but the overall system must be subject to functional safety as well. In particular systems that are purely implemented in software, such as network functions (VNFs - Virtual Network Functions) must also meet functional safety requirements.

Vertical industries:

Advanced Driving Assistance Systems (ADAS) today are being developed by the automotive industry with high functional safety requirements. The 5G automotive white paper refers to ADAS as a use case, which implies that functional safety must be provided by the 5G system at the same level as it is being designed at the component level for ADAS.

In summer 2016, the German Federal Ministry of Transport and Digital Infrastructure kicked off a public debate to introduce amendments to the road traffic law enabling the admission of autonomous cars on public roads. In January 2017 the road traffic law was amended for this purpose. However, opinions have been voiced [101] that perhaps a full new legal framework needs to be developed since the matter touches upon

- road traffic law,
- liability insurance law,
- product liability law and
- product safety law

Similar requirements exist in other verticals, such as energy, manufacturing, transportation, health, etc. These requirements exist for all systems that are part of a critical function in the vertical, beyond

any obvious services that can be served today like entertainment in cars, office communication in factories, etc.

#### Legal use case

Traditionally product liability (as judged in most court cases) is limited to “products” in the form of tangible personal property. However the correct functioning of an IoT device includes the functioning network and service.

- The product or service may become defective upon:
- Network or service failure (even temporal)
- Discovered security vulnerabilities

Smart (connected) devices will have a far reaching impact on manufacturers, service companies, insurers and consumers

How does the legal framework on liability need to evolve in order to cater for such liability chains? Beyond this, how would liability delegation work?

#### Virtual Event Recording Service - A case for a VxF

For clarifying the liability in a complex system one also needs an unforgeable event recording functionality, with similar properties like a black box in an aircraft. However because the 5G system is typically composed of functions operated in different administrative domains, a distributed event recording function is needed and which is composed of recording functions at the border of each domain.

For recording of events with the correct time-stamp a high accuracy time service in the 5G system is required. So far a high accuracy time service has not been a requirement in 5G.

Furthermore the current state of the art event recording devices are typically based on dedicated and embedded specialised hardware. In a highly virtualised environment though and because of the distributed nature an event recording service in 5G must be largely based on software, without excluding possibilities of dedicated hardware supported functions.

#### Objectives

- Collect and analyse requirements for functional safety in 5G from the verticals
- Solicit requirements for distributed event recording (black box) functionality
- Design a functional safety architecture
- Implement a proof-of-concept system, including a distributed event recording function

Through the OBUs several types of information will be available and be sent to the black box through different logging modules:

- Location logging: GPS location, heading, and speed;
- Logging of relevant events;
- Logging of internal and external sensors information.
- In terms of configuration, several configuration modules will be made available:
- Define the sampling time of the extracted information (logging information);
- Define a specific server that will work as black box to receive the provided information (logging information);
- Define an experiment and configure the previous logging modules.

In this use case the 5G network will be required to be able to get and process very fast the logging information, so that all the relevant events will be logged. The VxF will then be the black box logging module, which can be implemented in the edge of the network connected to the road side units.

### **3.2.2 Smart Cities Domain**

---

The 5G networks foresee the deployment of a network specific to different scenarios of use. These different scenarios are related to specific usage domain are called verticals. Examples of these vertical markets are energy, manufacturing, health care, agriculture, city and automotive. The focus is to design a network that is driven by the requirements of each vertical market in opposition to the previous network generations where the focus was to offer connectivity to support voice and data at increasing data rates.

From the network point of view, it must support different requirements that are posed, sometimes, by the same vertical. In the context of cities, also mentioned as Smart Cities when considering the junction of cyber physical systems based on the Internet of Things (IoT), an application with several full definition cameras that are monitoring the city will require a high throughput from the network. An application to control the flight of UAV that is monitoring the traffic in the city on a specific point will require an ultra-reliable and low latency from the network. A massive number of sensors that are monitoring different parameters of the city, such as air conditions, will require low power consumption and will send small chunks of data to the network.

This subsection will explore the Smart City vertical of the 5G by presenting the 5GINFIRE context where an experimenter will use real time data provided by the city and a network core, hosted in the 5GINFIRE based cloud. The experimenter will be able to work with different scenarios of a 5G oriented system. To present this, initially an overview of a sample use case scenario of city monitoring application will be described. In the sequence, common actors and stakeholders of the city environment that will interact with the 5GINFIRE testbed will be presented and a common set of network related Vertical Function of Verticals (VxF) of a Smart City Experimental Vertical Instance (EVI) will be described.

#### **3.2.2.1 Overview**

Smart City is a term with a broad definition and there are several initiatives around the world related to this area [94].

In the context of this document the focus is the use of Internet of Things (IoT) based systems that may help to enable the vision of a Smart City in the urban environment [96]. In this area, several uses cases, in different domains of applications can be described such as traffic, security, environmental monitoring, and parking among others.

Fixed sensors, such ones deployed at a street light, and mobile sensors, such as citizens' smartphones, can provide information that could enable the monitoring of the traffic in the city and real time information from this gathered data could be consumed by citizens, logistics companies and the public sector to help a better moving in the city. In the same way, the historical analysis of this data could help the public sector to plan and execute interventions in the city to evolve the traffic management.

Cameras deployed in the city can be used to improve security and safety of citizens if the data is consumed by different and specialized systems most often in a city Command and Control Center. Different sensors deployed in the city can provide information about noise, air pollution such as Carbon Dioxide (CO<sub>2</sub>) levels and water condition. This information can also enable different services and applications that might improve help to improve the daily life in the city and give conditions to city planners to act to improve the living in different places.

Parking sensors deployed along streets or inside parking areas can provide real time data of parking spots available enabling citizens to quickly find the best place to stop their car saving time and avoiding a search inside car that consumes time, fuel and produces more traffic and pollution.

The use cases above are small examples of how sensors deployed in a city can enable services and applications that may improve the life in the city.

In general, these systems share some common building blocks: sensors, gateways, compute resources, communication links between sensors and gateways and transport links between gateways and computing resources [96]. Different technologies are employed currently to support the communication links. Some are used in a small area such as Bluetooth Low Energy (BLE) and IEEE 802.11 (WIFI) and other in a wide area such as Low Power Wide Area (LPWA) and 3GPP based current mobile telecommunication networks (2G, 3G and 4G).

The next mobile telecommunication network, envisioned as 5G, will play an important role in the IoT area because it will focus on a broad range of scenarios with each their requirements [95].

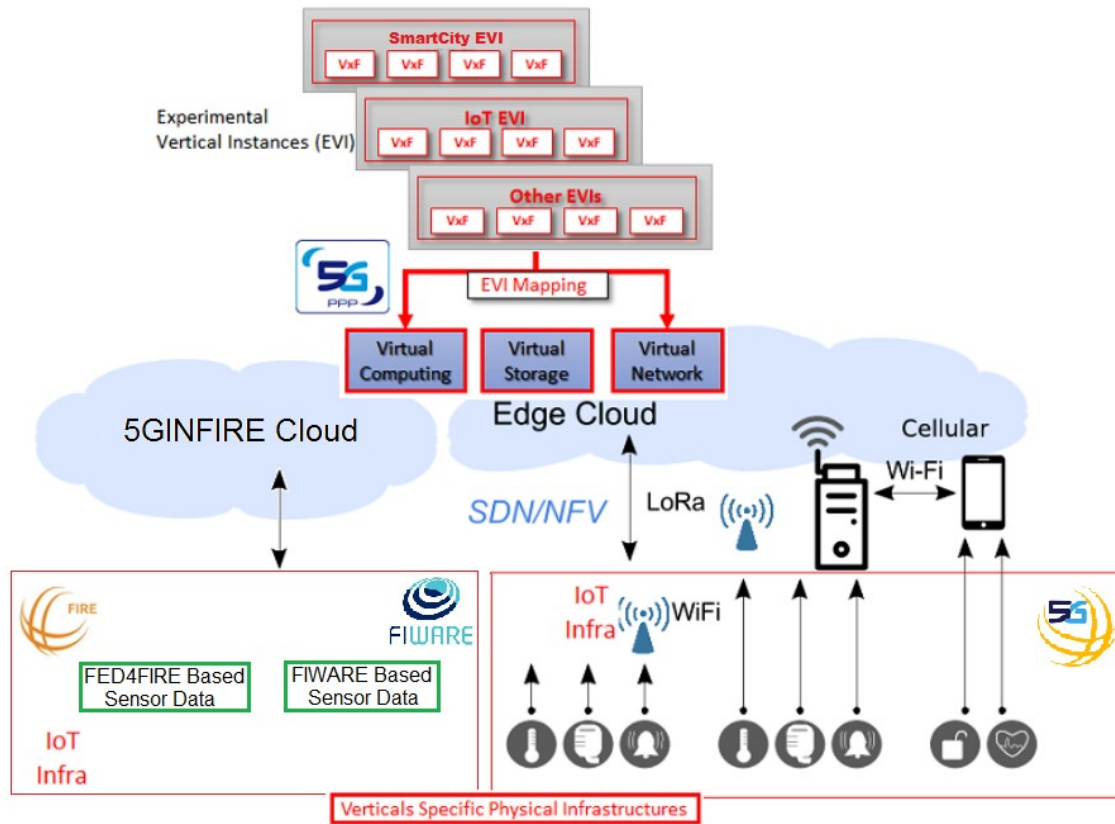
5GINFIRE aims to help this area by providing the conditions to experiment IoT based scenarios in the smart cities vertical supported by 5G based systems. This support will be done by putting together the actors, the resources and the technology in a cloud based environment.

Two different approaches will be used: one based on Virtual Sensors (VSensor) and the other using real sensors. The two approaches are depicted in Figure 25.

The first approach will be based on reusing current open data provided by sensors already deployed in several FIRE and FIWARE based testbeds. VSensors will be hosted by the 5GINFIRE infrastructure and will be connected to the data provided by the real sensors. This approach will allow an experimenter to create a Smart City EVI based on a set of data stored in the VSensors. This data was previously gathered from the real sensors.

To support the experimentation of 5G oriented systems in the Smart City vertical the 5GINFIRE core architecture will have to support two different types of integrations, as presented in Figure 25. One is based on a 5GINFIRE edge cloud that will be deployed in the cities where 5GINFIRE project partners are located and where they interact with the local stakeholders such as the municipality, citizens, fixed and mobile operators. The other integration will with other infrastructures data provide data the city. At this moment, this last integration will occur with FIRE and FIWARE based infrastructures.

Figure 26 presents the main components that will be required to be present in the 5GINFIRE architecture. Our requirements analysis indicates that there two main types of Smart City based infrastructures: some based on FED4FIRE and other are based on FIWARE compliant components that offers interfaces using CKAN based APIs or Next Generation Services Interface (NSGI). The 5GINFIRE middleware must have integration components to each one of these types of infrastructures. The information obtained using these integrations will be consumed by the VSensors Framework. This framework will provide the services that will enable the cycle of life of the experiment that will be executed by the experimenter.



**Figure 25: 5GINFIRE architecture in the Smart City and IoT infrastructure**

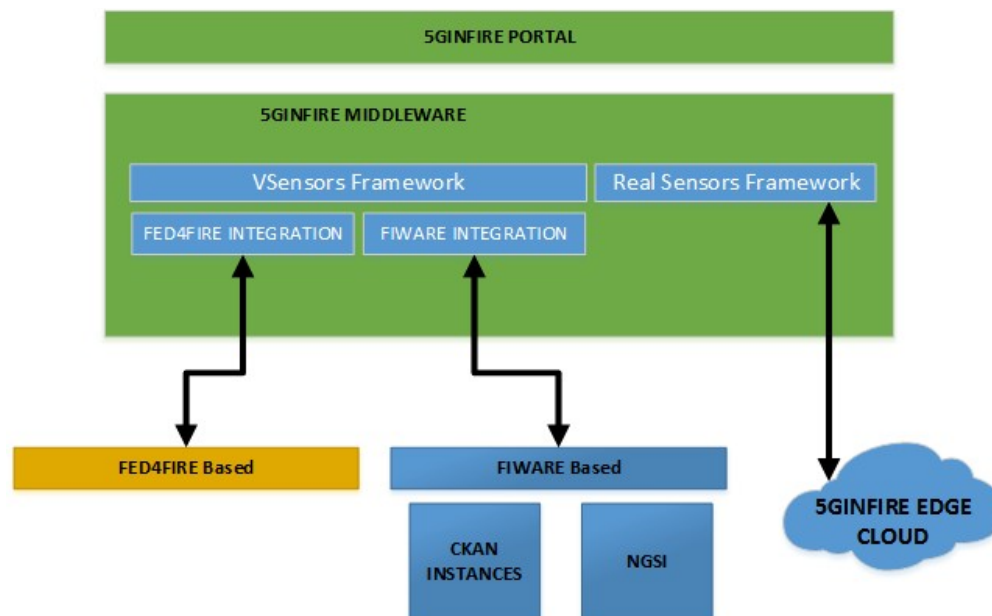
The Real Sensors framework will handle the data obtained in the 5GINFIRE Edge Cloud and will provide the services that will enable the experiments in the same way. Both frameworks will have to support the same services.

Smart City applications based on the IoT have a Sensor Layer where different devices connect to a gateway. The gateway interconnects these sensors with the infrastructure and has different roles such as protocol translations. The Gateway will be a VxF that will be available in the edge in both scenarios that considers the virtual and real sensors. Considering a FIWARE based backend, the gateway will translate protocols such as Message Queue Telemetry Transport (MQTT) and Constrained Application Protocol (CoAP) to Next Generation Service Interface (NGSI). This type of gateway if called by IoT Agent at the FIWARE platform but the VxF will have some additional roles such as being managed by the 5GINFIRE infrastructure.

Other functions are envisaged to be Vxfs that can be deployed in the Smart City such as a Load Balancer that can be able to balance traffic from multiple sensors considering the network conditions. A router responsible to IPv6 and IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) conversion [96] in both directions is also a candidate to be a VxF in scenarios using 6LoWPAN in accesses networks.

In some scenarios, according to the requirements of the solution a Firewall can be a VxF deployed in the Smart City.

By using the capabilities NFV and SDN different VxF can be foreseen on IoT based Smart City applications. The VxF can be responsible to other functions related to the infrastructure services [126] such as device management including fault and monitoring management among others.

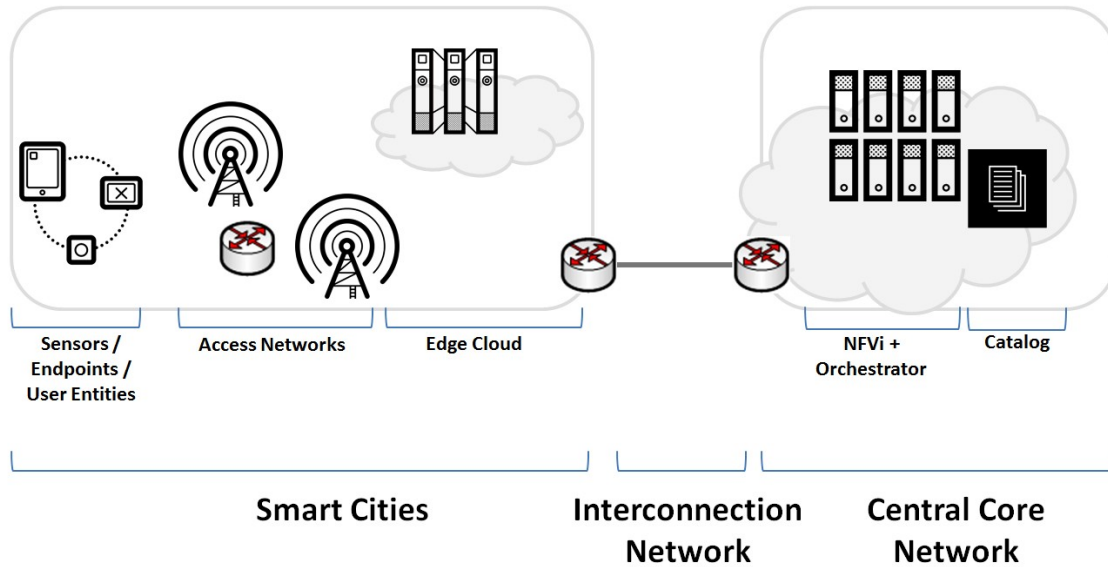


**Figure 26: 5GINFIRE IoT Integration Components**

The second approach will be based on using real data from sensors using a 5G based access network that will provide this data to 5GINFIRE to edge cloud deployed in the cities where project partners are located: Aveiro and Bristol in Europe and São Paulo and Uberlândia in Brazil. At these locations, some real sensors are already deployed. For example: in Aveiro, Portugal, there are sensors that provides information such as temperature, humidity, pressure, lighting, CO, NO<sub>2</sub>, H<sub>2</sub>, NH<sub>3</sub>, CH<sub>4</sub>; in Uberlândia, Brazil, there is information about urban buses positions and sewers sensors information at a district in the city called Granja Marileusa.

### 3.2.2.2 *Network and Service Functions/Modules*

In this part we will list the requirements needed by an environmental use case on smart cities and then on the core network. For each of them, we will describe any network and service functional requirements. This listing is going to be carry out in a broader approach for the purposes to not only target an environmental use case but as much of use cases that can be hosted on smart cities.



**Figure 27: Networks and Service requirement for smart cities**

**For the Smart city testbeds:**

As described by the Figure 27, a smart city can be simply be seen as a connected city that have various access network which allow sensors or endpoints to record data and allow applications to access them or be reached by user entities.

**Sensors / EndPoints / User Entities:**

Sensors: it is a connected object providing useful applicative data like temperature, air analysis, ...

EndPoints: it can be seen as a “modem” providing the connectivity to the access network, term used in the non-3GPP technologies like LoRa or Sigfox.

User Entities (UE): it is a smartphone, tablet or any device identified and to a 3GPP radio access network

In general, a smart city testbed shall provide a subset of each category but it could also be completed during open calls with Sensor providers (bringing and testing their sensors) and Apps providers (using the sensors already available in the smart cities, developing and testing their application).

But for the environmental use case, sensors collecting environmental data should be made available (by the smart city testbed or by sensors providers). Those sensors might imply physical endpoints depending on the type of sensors that will be used.

**Access Networks:**

Use cases require various types of access networks, according to the usage itself or the type of technology deployed by the city. Smart cities will need to equip the geographical area with a representative set of sensors with a large coverage in order to receive sufficient amount of data that can be then exploitable by applications. The access network shall include the front hauling network (like fibber between LTE ENodeB but for LoRa gateways it could be ADSL or even 3G).

**Access Network Types:**

Various wired or wireless access networks may be used, with following suggestion:

- LoRa or NB-IoT may be used for the environmental use case, but also for some telemetering use cases. In such a case, where sensors need to be placed in isolated areas, we prime wireless and powerless technology like Lora or NB-IoT.
- Wifi or LTE (or LTE advanced) can be used for the environmental use case, but also telemaintenance use cases. They are ideal for services that may require bidirectional streams with real time constraints (conversational service) and may rely on various technologies (VoLTE, VoWiFi, WebRTC, ...).
- LTE-M or fiber can be used for the environmental use case, but also for CCTV and security use cases. They can provide high resolution for video sensor (a camera), a high bandwidth and very low latency connection.

#### **Applications/Vxfs near to the data:**

Having computing near to the data sources can have several benefits compared to the traditional cloud based computing paradigm. By distributing computing tasks near to their data source, it offloads network traffic data on central datacenter as well as their computing tasks. But also it shortens application/VxF processing time and this can be particularly true when data provider (in here sensors) are working in a send/receive mode. For all of those reasons, it can be very interesting for a developer to put some of the applications/Vxfs computing tasks on the edge rather than letting them in a central cloud.

So in conclusion for some applications/Vxfs, the data will be better to be processed at the edge for shorter response time, more efficient processing and smaller network pressure. For some other, it won't be necessary, but still the capacity to put computation processing at the edge will still need to be considered.

#### **For the Core Network(s):**

##### **NFVi:**

For a testbed that is targeting a 5G NFV-Based architecture, a NFVi (NFV cloud infrastructure) is required to host the VNFs. The capacity of the NFVi shall be expressed in term of computing, storage and networks with:

- Servers or CPU cores,
- VM flavours (CPU, RAM, Disk) which are available for booking
- Number of VMs supported by the NFVi

##### **Catalog:**

A catalog of VNFs provided by partners and that can be hosted on the NFVi is required with the associated licences, packages, documentation, release notes and hosting requirements.

##### **Orchestrator:**

To complete one of our objectives which are to fit to ETSI's reference architecture it is required to deploy inside 5GinIFRE core network an orchestrator for VNFs including a SDN controller for interconnection.



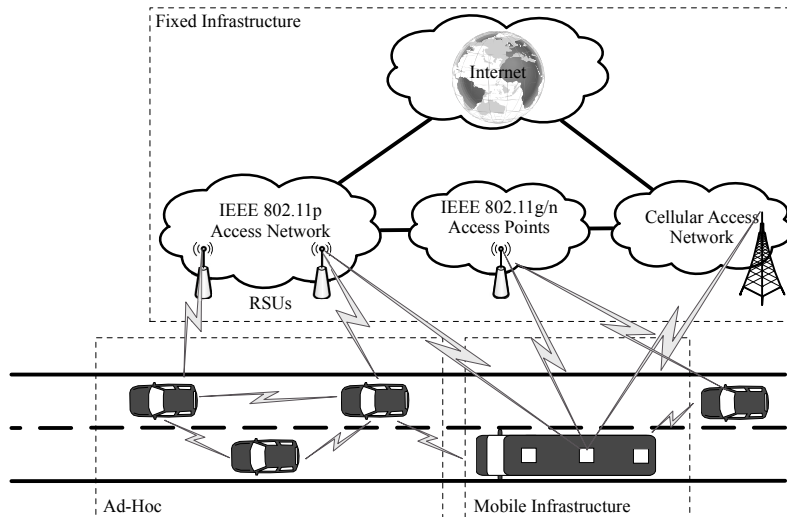
### **About the Interconnection:**

Interconnection between the smart cities testbeds and central cloud hosting Core Network features is required, through a secure link (site-to-site VPN for example) and with a determined Quality of Service in term of bandwidth, latency, jitter, and so on. “Determined” does not mean “guaranteed”, it is a way to qualify the type of application that could be experimented during open calls. Such an interconnection shall be based on GEANT and NREN services.

## **3.3 Testbeds in 5GinFIRE Use Cases**

### **3.3.1 Automotive**

The vehicular network that will be used in the automotive use case consists of On-Board Units (OBUs) in the vehicles and roadside units (RSUs) connected to the Internet through optical-fiber through an Ethernet interface. As shown in the Figure 28, the vehicles connect among each other via standard IEEE 802.11p/ WAVE links, and are connected to the RSUs and the Internet through IEEE 802.11p/ WAVE, IEEE 802.11g/ WiFi or cellular links.



**Figure 28: Vehicular Network Architecture**

IEEE 802.11p/ WAVE technology has been developed for vehicular dynamic environments: it is able to provide a communication range of up to 900m in line-of-sight, which is very important to reach vehicles in a road environment, and the connections can be established in 10-20msec, which are able to use the very small opportunities of communication. Moreover, vehicles can operate in a mesh and multi-hop topology, improving the communication range between the vehicles and the infrastructure. DSRC/WAVE works on an unlicensed band that is free of cost and reserved for intelligent transportation systems.

Cellular backhaul is available as a complement technology to the IEEE 802.11p when this is not available. WiFi is available in the OBUs and RSUs so that users inside the vehicles and users in the street near the RSUs, are able to connect to the Internet. Sensors installed in the vehicles and in the street can send data to the cloud through the vehicular network (through RSUs and OBUs WiFi hotspots).

Vehicles are also able to connect to the infrastructure through WiFi hotspots installed in the roads. However, since the range of WiFi is 10 times lower than the one of IEEE 802.11p, and the time of

connections establishment is 100 times larger (in the order of seconds), they only connect through WiFi hotspots when they are stopped or moving very slowly and near the WiFi hotspots.

Each vehicle is equipped with an OBU with multiple wireless interfaces, which enable the vehicles to communicate both with other vehicles circulating inside the city and with RSUs that are integrated in the city infrastructure. OBUs and RSUs have a similar hardware, except for the antennas, which have higher gains in the RSUs. An example of an OBU is depicted in Figure 29.



**Figure 29: OBU**

The OBU includes the following elements:

- Single-Board Computer (SBC)
- Dedicated Short Range Communication (DSRC) wireless interface (IEEE 802.11p)
- WiFi interface (IEEE 802.11a/b/g/n)
- 4G Interface
- GPS receiver
- Antennas for each technology (round antenna is for WiFi and rectangular antenna is for IEEE 802.11p).

The SBC contains the processing unit and is responsible for coordinating the various interfaces and access technologies. Moreover, it provides an in-vehicle WiFi hotspot for the users in the vehicles, and the sensors installed in the street and in the vehicles.

A mini-PCI 802.11p compliant wireless interface is connected via one of the mini-PCI slots. A standard 802.11g/n wireless interface is connected to one of the USB ports of the mainboard to provide communication between the OBU and other user devices. This interface can also be used to connect the vehicles to any available WiFi hotspot. A cellular interface is available to be used when required to control the OBU operations, and whenever no other connection type is available.

The GPS receiver is integrated with the IEEE 802.11p interface of the SBC to provide multi-channel synchronization. Synchronization to Universal Time Coordination (UTC) is mandatory for DSRC devices that switch between channels. The channel interval boundaries are derived from the GPS signals.

The OBUs are running a tailored Linux distribution based on Buildroot. The kernel was customized to include new features such as clock synchronization, as required by IEEE 802.11p. As Linux Wireless does not provide support for the IEEE 802.11p / WAVE norm, the ath5k driver was modified to accommodate that norm within the AR5414A-B2B Atheros chipset. The driver was further extended to meet the requirements of IEEE 802.11p/WAVE. The WAVE Short Messaging Protocol enables the exchange of short messages between nodes at the MAC layer, offering a fast way for vehicular nodes to safety-critical information.

The RSUs have the same hardware as the OBUs, except for the cellular interface (which they do not require) and the Ethernet interface (required to connect to a switch from the fiber infrastructure). In Porto, some RSUs are connected in municipality buildings and other RSUs are connected in traffic lights, which are then connected to a switch from the fiber infrastructure (see Figure 30). The RSUs in traffic lights are powered through PoE (Power over Ethernet).

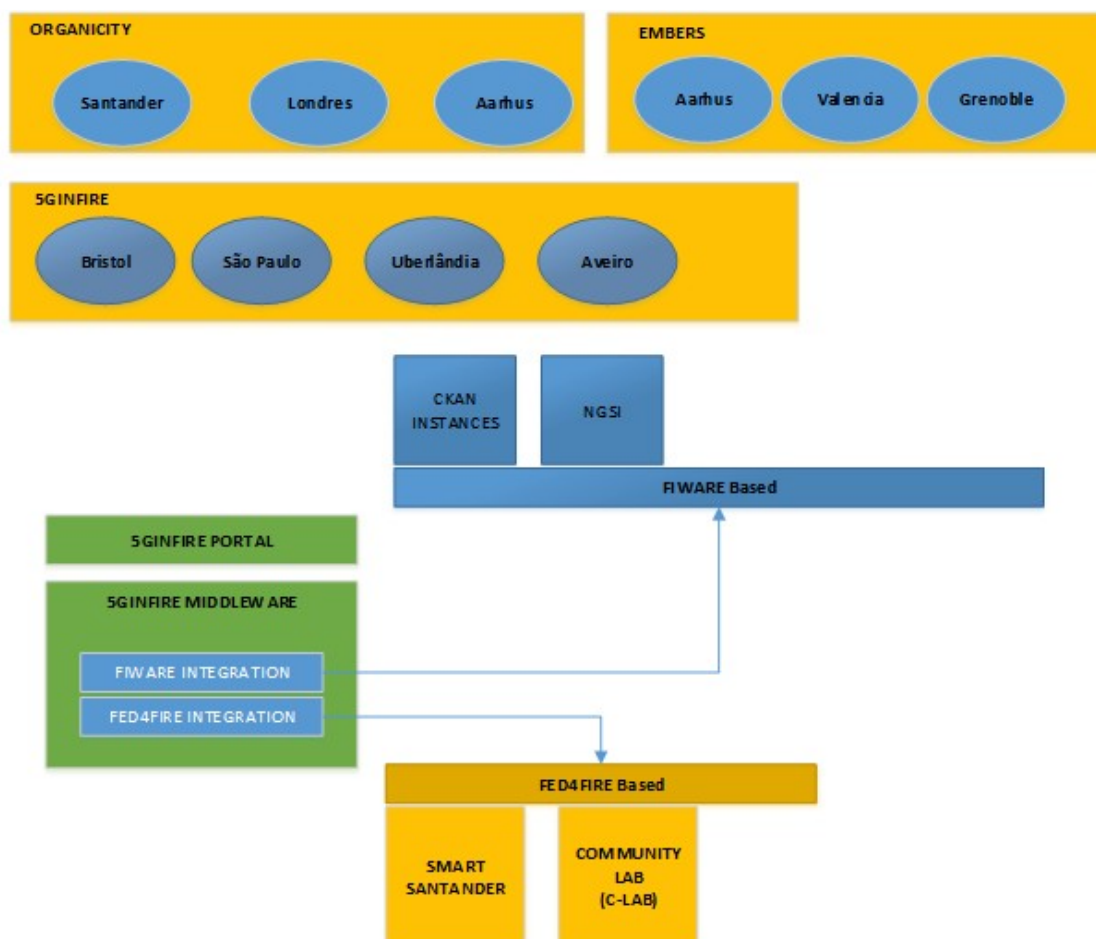


**Figure 30: RSUs mounted in the municipality building and in a traffic light**

### **3.3.2 Smart Cities**

Currently there several testbeds that focus Smart Cities based scenarios. The following FIRE related project can be integrated with the 5GINFIRE infrastructure: ORGANICITY Project [104], EMBERS Project [105], FESTIVAL [106], FIESTA-IoT [107], Smart Santander [108] and Community Lab (C-LAB) [109]. Figure 31 gives an overall view of testbeds the type of integration that will be required. Using the open data available from these projects several Smart City based use case can be experimented inside the 5GINFIRE infrastructure using the VSensor approach.

ORGANICITY has open data available from CKAN instances [100] from the cities of Aarhus, London and Santander. These data sets are FIWARE based CKAN instances and they can have data from a broad range of different domains in these cities such as mobility, traffic flow, air quality, environmental monitoring among others. Using the same FIWARE enabled CKAN instances, the EMBERS project also has open data from cities such as Aarhus (real time and historic traffic and pollution data), Grenoble (real time and historic simulated parking data) and Valencia (real time traffic information).



**Figure 31: 5GINFIRE Testbeds for Smart Cities**

The cities of Aveiro, Bristol, São Paulo and Uberlândia where 5GINFIRE project partners are located can support 5G experiments, as indicated in the Figure 31. In this case, the experimentation will be based on an Edge Cloud available in these locations, thus enabling the use of real sensors directly integrated with the 5GINFIRE cloud.

At Uberlândia, the UFU Future Internet Testbed [110] was built during the participation in the OFELIA [111] and FIWARE [112] projects and within the association with the FIBRE project [113]. The infrastructure is located at Santa Monica Campus in Uberlândia, Minas Gerais, and Brazil.

The FIWARE Lab consists of a cloud infrastructure with 92 cores, 528 GBytes of RAM and 8 TBytes of disk storage. The FIWARE Laboratory has open data provided by the Municipality of Uberlândia, such as public buses position. The testbed is connected with Granja Marileusa [114], a smart district located in Uberlândia where different sensors are being deployed such as sewers information. All this open data is available through FIWARE based CKAN instances.

The FIBRE island is an infrastructure with two 24-port DATACOM switches with OpenFlow support, two experimental switches based on NetFPGA and four EDOBRA switches. The EDOBRA switches were built on the OFELIA project and are based on a four-port wireless Gigabit router (TP-LINK) that supports the OpenFlow protocol and contains ODTONE [115], An open source implementation of the IEEE 802.21 protocol. The processing capacity is based on 36 cores, 96 GB of RAM and 7 TBytes of storage. This infrastructure, built on the OFELIA project, is integrated into the FIBRE project and allows the realization of different types of experiments using the network equipment described

above in different network topologies. The experiments can also use resources and networks for the other islands of Brazil using the FIBRENET [116] maintained by the National Research Network [117]. FIBRENET also allows connection to different infrastructures both in Europe and the United States.

The platform in Porto and Aveiro leverages static sensors for gathering environmental data. This platform is composed of static data collection units (DCUs) that monitor a number of environmental parameters (e.g. temperature, wind, luminosity, noise, pollutant emissions) and store the measurements in a local database. A DCU is composed by a power supply, a processing unit (Raspberry Pi), the sensors, and a control board to interface the processing unit and the sensors. The sensors are of three classes: (i) meteorological: thermometer, hygrometer, wind vane, anemometer, rain gauge; (ii) life quality: sonometer, lux meter, solar radiation; (iii) air quality: particulate matter sampler, CO, NO<sub>2</sub> and O<sub>3</sub> gaseous meters. The wind vane, anemometer and rain gauge are mounted in a dedicated structure for freedom of movement. The sonometer is mounted below the main casing, to maximize exposure to road noise and ensure protection from rain. The remaining sensors are enclosed in a separate vented shelter (to allow air flow) mounted on top of the main casing.

---

### **3.3.3 Additional Testbeds Required in 5GinFire**

---

#### **3.3.3.1 The 5TONIC laboratory**

The global 5G Telefonica Open Innovation Laboratory (5TONIC) has been established in Madrid (Spain) as a leading European hub for knowledge sharing and industry collaboration in the area of 5G technologies. The laboratory provides an open research and innovation ecosystem for industry and academia to promote joint project development, joint entrepreneurial ventures, discussion fora, and provide a site for events and conferences, all in an international environment of the highest impact. The laboratory also serves to evaluate and demonstrate the capabilities and interoperation of pre-commercial 5G equipment, services and applications. Currently, the 5TONIC laboratory has eight members: Telefonica, Institute IMDEA Networks, Ericsson, Intel, Commscope, University Carlos III of Madrid, Cohere Technologies, and Artesyn Embedded Technologies.

The 5TONIC laboratory includes a solid baseline of facilities and infrastructure and equipment to support advanced experimentation in the 5G virtual network function area. The description of the main infrastructure connected to 5TONIC that will be offered for experimentation in 5GinFIRE is detailed in section 4.

#### **3.3.3.2 Bristol is Open Testbed**

##### **Description**

Bristol Is Open is an open programmable city region which is welcoming a range of partners to the project, including large telecom and software companies, small hi-tech start-ups, public service delivery organisations, academics and others. The use cases will have a slice of the network to work with. The active, wireless and mesh network will be technology agnostic, built on open network principles, using software defined network technologies, that enable network function virtualisation.

Small sensors, including the smart phones and GPS devices of willing participants, will supply the three new fast networks in the centre of Bristol, with information about many aspects of city life, including energy, air quality and traffic flows. A city operating system will dynamically host this machine-to-machine communication, allowing the development of a wide range of applications. All the data generated will be anonymised and made public through an 'open data' portal.

The facility aims to create a unique, fully flexible, programmable and open experimental platform for all networks and IT technologies. It enables user-defined experimentation with physical and

emulated technologies under realistic, controllable, predictable, secure and repeatable conditions. Key capabilities of the facility are:

- Demonstrate radically new network architectures;
- Measure and demonstrate large-scale, multi-technology network systems at laboratory, city scale and UK national level;
- Support research on IT Infrastructures at scale including new-generation computing systems (e.g Data Centres, High Performance Computing (HPC) and extensions to exascale & quantum computing);
- Understanding behaviour of hybrid complex systems (networks+IT);
- Experiment on Seamless convergence of heterogeneous technology domains (i.e. wireless, optical, Data centre);
- Breaking traditional barriers between infrastructure, control and service layers.

The facility is already in place and is funded by a number of UK government initiatives, Industry and EU projects. It is a joint venture between the Council and the University of Bristol. The joint venture board includes: representatives from University of Bristol, Bristol City Council, local digital organisations and local enterprise partners. This board is supported by an advisory panel comprising of long-term partners and invited experts.

## Facilities

### 1. Bristol Is Open (BiO) Bristol City Experimental Network

Bristol is Open (BiO) is a joint venture between the Bristol City Council and the University of Bristol. It aims to create a living lab R&D Testbed targeting City-driven digital innovation as a City Experimentation as a Service (CEaaS). It provides a managed multi-tenancy platform for the development and testing of new solutions for information and communication infrastructure, and thus forms the core ICT enabling platform for Future Cities agenda. At the infrastructure level BiO comprises convergence of five distinctive SDN enabled infrastructures as shown in Figure 32:

- Heterogeneous optical and layer 2 infrastructure comprising fibre, flexi WDM, 10/40G Ethernet technologies
- Heterogeneous wireless infrastructure comprising WIFI, LTE and 60Ghz technologies
- IoT sensor mesh infrastructure
- Network emulator comprising server farm and FPGA / SoC / NP farm
- Blue Crystal HPC facility

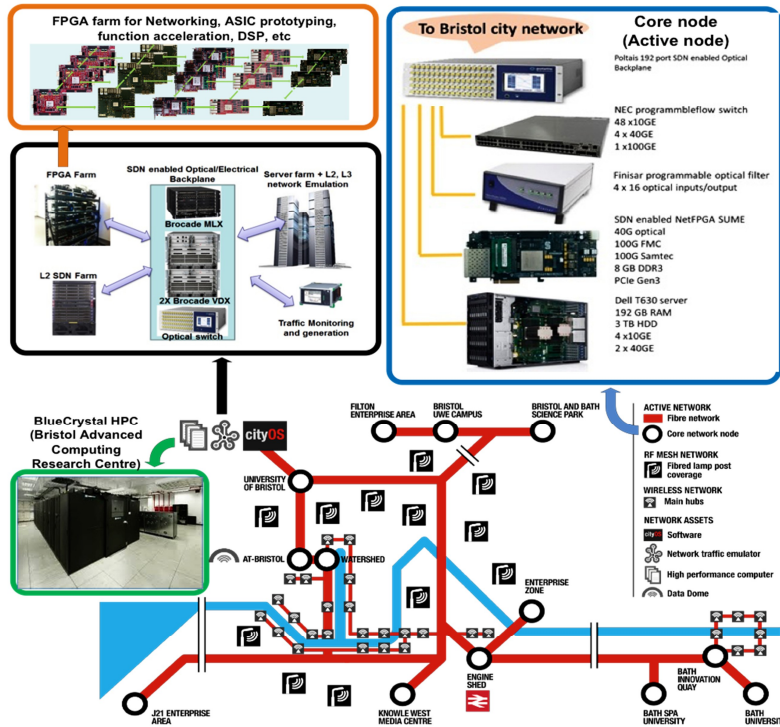


Figure 32: BiO Infrastructure

2. Heterogeneous optical and layer 2 infrastructure:

The Bristol city network comprises active nodes (i.e. Core Network Nodes) across the city in which are connected through optical links (Figure 32). Each of the active nodes in the topology use a number of advanced network and IT modules as shown in Figure 32.

- The SDN enabled optical switch back plane with high switching speed (25ms) and low loss (~0.5 dB) enables high throughput and low latency communication between the network nodes, whilst providing an optical back plane for the devices in the rack.
- The SDN enabled NEC Programmable Flow switch with 48 10GE, 4 40GE interfaces
- The T630 is one of the most recent models of Dell HPC grade servers, which uses 2 Intel Xeon E5 processor chips with 16 cores, 192 GB RAM, 3 TB hard disk, 4 10GE and 2 40GE NICs. The server motherboard supports more than 8 PCIe slots, and 4 of them are 16 lane PCIe3. The server support virtualisation software such as Xen and KVM, and is remotely controllable through the integrated Dell Remote Access Controller (iDRAC). At NIC interface level the server supports SDN

3. Heterogeneous wireless infrastructure

A cluster of Wi-Fi Access Points provides combined 2 & 5 GHz operation (supporting the IEEE 802.11ac Wave 1 profile). The facility also supports experimentation platforms for new 5G and beyond access technologies; for example millimetre wave based access solutions with beam tracking, as well as new technology enablers such as Massive MIMO for ultra-high density networks in the 2 GHz band.

The entire platform will use Software Defined Network (SDN) control principles.

The wireless network will use a combination of the following advanced technologies to provide access and backhaul RF infrastructure:

- SDN enabled 802.11ac (2GHz & 5GHz) : AP832e by Meru networks implements dual-radio, dual-band IEEE standards 802.11a/b/g/n/ac-Draft access point with six RP-SMA connectors and six external omnidirectional antennas. AP832e is capable of supporting two concurrent 5 GHz 3x3 MIMO (3ss) radios with an aggregate 2.6 Gbps data rate. For the most demanding business applications like video and voice with Wi-Fi Multimedia (WMM) quality of service configurable per user is also available.
- Microwave 59-63GHz: iPASOLINK SX Supplied by NEC are used in the BIO testbed to provide wireless backhauling capabilities to the nodes around the quay side. The supplied modules utilise high speed modulation format (QSPK to 256 QAM) which enable up to 400Mb/s point to point communications with highly narrow beam-width antenna. These modules support 8 internal QoS class mappings and packet classification function based on header information following the 802.1p standard. They connect to the WAN network through GbE interfaces and support synchronous Ethernet for transferring network clock.

#### 4. FIWARE-IoT Platform

For gathering and storage different data type, we will use a FIWARE instance deployed inside the Bristol is Open cloud using different platform assets like Short Term historic (aka Comet), Context Broker and IoTagents.

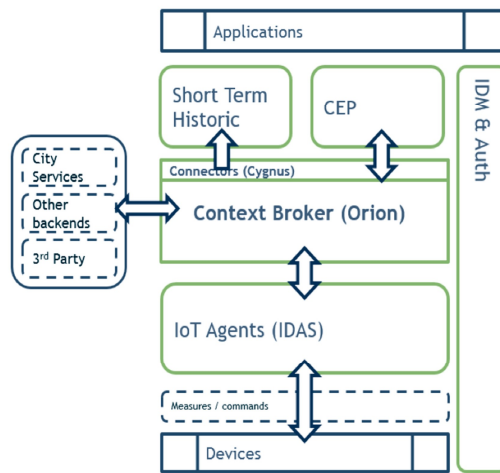
- Context Broker: The Orion Context Broker is an implementation of the Publish/Subscribe Context Broker GE, providing the NGSi9 and NGSi10 interfaces. Using these interfaces, clients can do several operations:
  - Register context producer applications, e.g. a temperature sensor within a room
  - Update context information, e.g. send updates of temperature
  - Being notified when changes on context information take place (e.g. the temperature has changed) or with a given frequency (e.g. get the temperature each minute)
  - Query context information. The Orion Context Broker stores context information updated from applications, so queries are resolved based on that information.
- IoT Agents: The platform supports several IoT protocols with a modular architecture where the modules are called “IoT Agents”. Therefore, integrators need to determine first which protocol they will be using to connect devices and select the right IoT Agent to use. At present, the following IoT Agents and supported IoT protocols in the IoT Platform are:
  - Ultralight 2.0 [118]
  - JSON [119]
- Comet: component of the FIWARE ecosystem in charge of managing (storing and retrieving) historical raw and aggregated time series information about the evolution in time of context data (i.e., entity attribute values) registered in an Orion Context Broker instance.



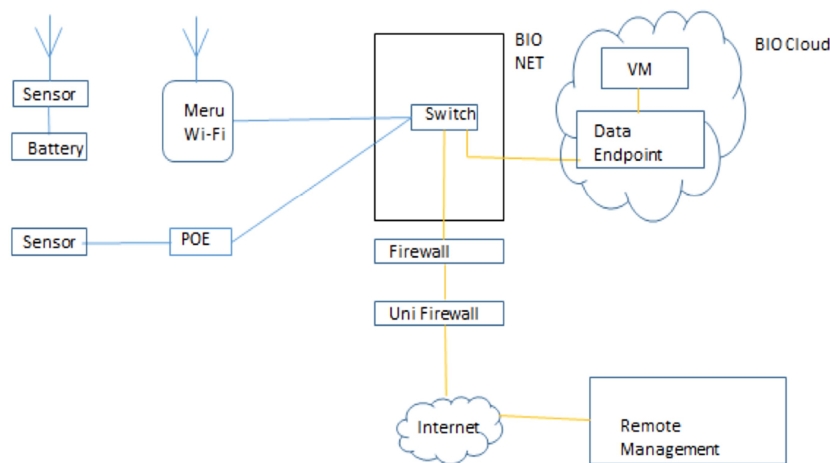
**BiO and 5GinFIRE Smart City Use Case**

By deploying sensors to the local BIO network the data generated will flow to the FIWARE IoT stack and arrive to the application layer. At this point the application will expose the data through a REST api from the Context Broker (Orion) [102] and manage it accordingly. Finally the network provisioning and network management is done by a SDN controller this enhances the infrastructure with more granular security, lower operating costs, cloud abstraction and guaranteed content delivery. Figure 33 highlights the architecture of the IoT platform. Figure 34 is a general description of the infrastructure.

Everything is deployed and ready for use once the sensors are connected and set to send data to the IoT Agent (IDA). At the application layer the data coming from the sensors can either be used by an open source application for example freeboard [120] or the developer can use the data and visualize it through the individual application.



**Figure 33: FIWARE IoT Platform**



**Figure 34: FIWARE IoT General Infrastructure**

Sensor data will be transferred to a VM located on the BIO infrastructure via the internet using the VMs public IP and a specific port for the endpoint. The user will have access to the virtual machine “agent” via two methods depending on the use case: An IPSEC site to site VPN can be configured to the customers OpenStack project where the VM is hosted. BIO can supply instructions for the VPN setup. The VPN is also provisioned by the SDN controller providing dynamic capabilities to the network. An alternate option is BIO could provide the VM with a public IP so it is remotely accessible over the internet. BIO will provide credentials to access the VM host.

## 4 Experimental Infrastructure Architecture, Model Entities Specification and Design

### 4.1 Introduction

---

The section presents 5GinFIRE architectural approach, processes and envisaged roles and usage scenarios of 5GinFIRE platform. It is comprised by technical and non-technical perspectives.

### 4.2 5GinFIRE actors and terminology

#### 4.2.1 Actors

---

5GinFIRE offered services and tools target to accommodate the following envisaged user roles. All users are assumed to be of an Authenticated role:

- **Experimenter:** This role represents the user that will utilize our services and tools to deploy an experiment. That is the experiment description in terms of e.g.: NSD (Network Service Descriptor) or TOSCA Specification
- **VxF developer:** This role is responsible to upload VNF and NSD Descriptors in the 5GinFIRE services
- **Testbed provider:** This role represents users that are responsible for testbed administration, configuration, integration, adaptation, support, etc
- **Services administrator:** This role represents the user that are responsible for maintenance of the 5GinFIRE services

Finally an anonymous user role exists who has some really simple usage scenarios (e.g. signup through the portal)

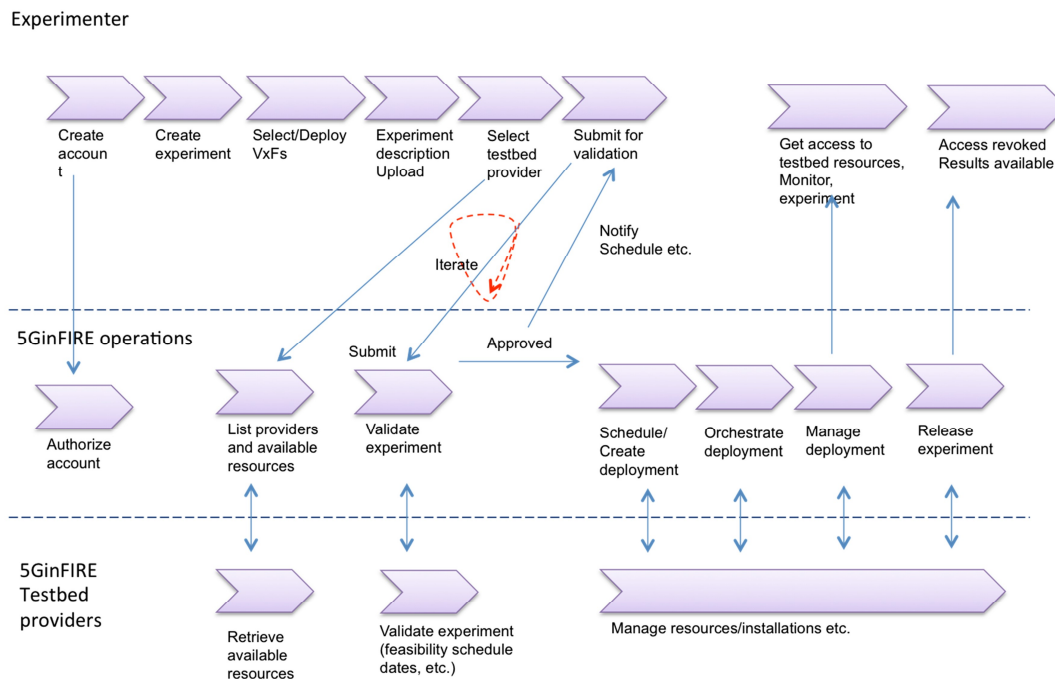
#### 4.2.2 Terminology

---

**Experiment:** In 5GinFIRE it is defined as a set of experimentation activities that will be conducted during an allocated time-slot (probably spanning for several days); the experiment may probably involve multiple 5GinFIRE test-beds; it might require the utilization of several network services, which will be indicated during the experiment definition and will be validated by the 5GinFIRE operations; these network services may be used by the experimenter during the allocated timeslot. it is a set of experimentation activities that will be conducted during an allocated time-slot (probably spanning for several days); the experiment may probably involve multiple 5GinFIRE test-beds; it might require the utilization of several network services, which will be indicated during the experiment definition and will be validated by the 5GinFIRE operations; these network services may be used by the experimenter during the allocated timeslot.

**VxF:** complex constellations of virtual functions, all running on a mix of real and virtual network or computing elements. We refer to virtual functions as VxFs when we do not want to distinguish between network-centric functions and vertical-centric functions.

### 4.3 5GinFIRE Experimentation Workflow



**Figure 35 : 5GinFIRE experimentation workflow overview**

Figure 35 displays an envisaged overview of the workflow process. There are three main horizontal lines: the experimenter, the 5GinFIRE operations and the 5GinFIRE testbed providers that interact during an experimentation life-cycle. At the simplest case, users signed-up to the platform will be approved by 5GinFIRE Operations. To perform an experiment on top of the 5GinFIRE infrastructure at its simplest form the user needs to create an experiment, e.g. some experiment metadata, scheduling, purpose, etc and select available VNFs or deploy new ones. Then he needs to compose the experimentation solution. This can be done either as an OSM Network Service Descriptor or in terms of a TOSCA specification. In a first version of the architecture, the user will provide an OSM-supported YAML description of the network service, potentially aided by a graphical composer. Subsequent refinements of the architecture may consider the utilization of TOSCA-based NFV descriptions, which would be mapped to into OSM-supported YAML by specific 5GinFIRE middleware (this development will depend on the availability of specification for VNF descriptors based on the TOSCA model, which is currently being addressed by both OASIS and ETSI NFV). As soon as everything is in place for an experiment description, the experimenter selects the testbed facility based on resource availability after the experiment is submitted for validation.

5GinFIRE prepares a process for validating an experiment in terms of various rules such as schedule, resource availability, etc. The validation process is closely performed together with the target testbed providers. We expect this to be iterative in various cycles involving the experimenter by either asking questions or modifying any experiment details and parameters.

As soon as an experiment is approved, it is scheduled by 5GinFIRE operations for deployment. Through the portal or OSM the 5GinFIRE operations will create a deployment (i.e. uploading descriptors etc) and OSM will later on orchestrate it (trigger the services instantiation). We expect that there will be a close collaboration during the management of the orchestration/deployment with the testbed providers. After deployment the resources are available and accessible to the experimenter.

In the end of the experiment schedule, the resources of the experiment are released and access is revoked. We expect though that any available results of the experiment will be available to the experimenter.

#### **4.4            *5GinFIRE experimentation platform usage scenarios and requirements***

---

This section presents use cases that the described actors will perform through 5GinFIRE provided tools and services. It will help at a later stage to verify our proposed solution. Figure 36 displays a UML use case diagram with the identified actors. It is expected that supported features, capabilities and provided services of the proposed 5GinFIRE architecture arrive from the Task 2.2 use-case experiments driven approach we have described. More requirements for the architectural components and the processes are defined from the usage scenarios that we describe here. In the tables apart from the responsible actor and a description we have identified a target responsible service.

Please also notice that the roles of Experimenter, VxF developer, Testbed provider and Services Administrator are all subclasses actors of the Authenticated role. That means that they get their role as soon as they are authenticated in the platform.

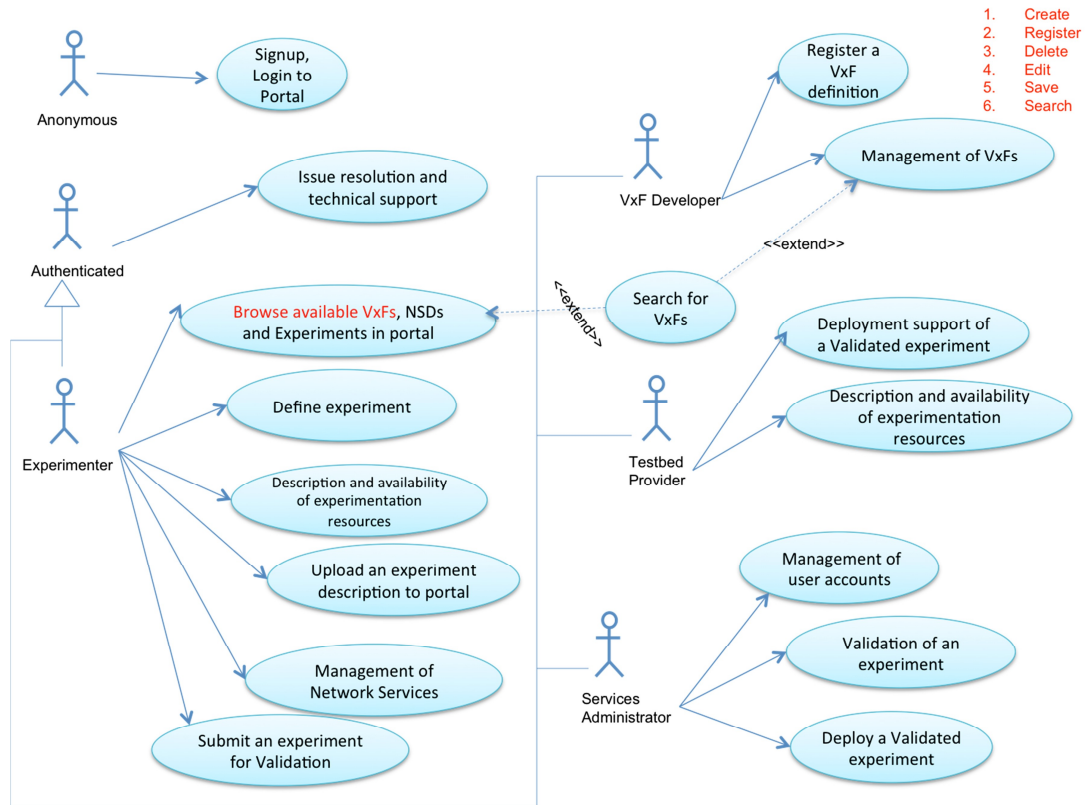


Figure 36: Use cases diagram

Generic usage scenarios

<b>No</b>	#1001
<b>Title</b>	Signup, Login to Portal
<b>Actor</b>	Anonymous User
<b>Description</b>	<p>The anonymous user can create an account and login as one of the roles: Experimenter, VxF Developer, Testbed provider, Services administrator. During sign up the user can express the interest of such a role or roles. The Portal Services administrator needs to approve a user and his role(s) (see #5010).</p> <p>More specifically, the user can create an account:</p> <ul style="list-style-type: none"> <li>- Specifying a username and a password, along with any personal data and indented role(s)</li> <li>- From federated FED4FIRE credentials</li> </ul>
<b>Target service</b>	Portal

<b>No</b>	#1002
<b>Title</b>	Issue resolution and technical support
<b>Actor</b>	All Authenticated
<b>Description</b>	An experimenter or a VxF developer should be able to identify operational issues and obtain technical support, using the diverse communication channels established for these purposes (i.e. through Wiki, email or the issue tracking tool of the project). Issue resolution and technical support may involve testbed providers and/or the services administrator.
<b>Target service</b>	Issue tracking tool and the 5GinFIRE wiki

#### Experimenter usage scenarios

<b>No</b>	#2010
<b>Title</b>	Browse available VxFS, NSDs and Experiments in portal
<b>Actor</b>	Experimenter
<b>Description</b>	An experimenter can browse all available VxF descriptors and NSDs registered at the portal marketplace. It can browse its uploaded experiments or any publicly shared experiment descriptions.
<b>Target service</b>	Portal

<b>No</b>	#2020
<b>Title</b>	Define experiment
<b>Actor</b>	Experimenter
<b>Description</b>	An experimenter should be able to define an experiment. An experiment definition may include: a) time requirements (e.g. list of proposed dates and duration of the experiment); b) indication of the Network Services that will be instantiated during the experiment; c) description of the testbeds, facilities and resources required for the experiment; etc.  The service chaining can be defined by submitting an OSM-supported YAML description or possibly enhanced by using a graphical interface
<b>Target service</b>	Portal

<b>No</b>	#2030
<b>Title</b>	Description and availability of experimentation resources
<b>Actor</b>	Experimenter / Testbed provider
<b>Description</b>	Using the portal, an experimenter should be able to get a description and a

	tentative availability of experimentation resources in the 5GinFIRE platform (i.e. the availability of the different 5GinFIRE testbeds and facilities). This information may serve as a reference to the experimenter, for the purposes of experimentation planning, but might not be the final scheduling as this will be decided by the experiment validation phase. Information about the availability of resources may be updated by testbed providers according to their scheduled experimentation activities (see #4020).
<b>Target service</b>	Portal

<b>No</b>	#2040
<b>Title</b>	Upload an experiment description to portal
<b>Actor</b>	Experimenter
<b>Description</b>	It will be possible to upload just a YAML descriptor created on some other tool, through the portal. The experimenter will upload an experiment description based on YAML. More Experiment metadata: name, creation date, etc must also be available during upload
<b>Target service</b>	portal

<b>No</b>	#2050
<b>Title</b>	Management of Network Services
<b>Actor</b>	Experimenter
<b>Description</b>	An authorized experimenter should be enabled to upload to the 5GinFIRE repository a deployment template for a Network Service (NS), i.e. an NS descriptor, including the description of the VxFS that comprise the service and their interconnection. In a first stage of the architecture, the user will provide an OSM-supported YAML description of the network service. This functionality may be aided by a graphical composer. The user should be able to refer to the NS in the 5GinFIRE repository for subsequent use. By default, a network service is only visible to its creator. Complimentary, the experimenter should be able to edit and delete a NS.
<b>Target service</b>	Portal, OSM NS graphical descriptor

<b>No</b>	#2060
<b>Title</b>	Submit an experiment for Validation
<b>Actor</b>	Experimenter / Testbed provider
<b>Description</b>	The experimenter should be able to request the validation of the experiment, which will be evaluated offline by the responsible entity of 5GinFIRE operations,



	probably with the support of the testbed providers (e.g. to check the suitability of the experiment to be executed over the 5GinFIRE facilities). Independently of the established validation process, the experimenter should be able to check the status of the validation request from the portal. In case of successful validation, the information provided to the user may also include indication of time slots allocated to the experiment.
<b>Target service</b>	Portal, and Issue tracking tool

<b>No</b>	#2070 (same as #3030)
<b>Title</b>	Search for VxFs
<b>Actor</b>	Experimenter
<b>Description</b>	An authorized Experimenter should be able to search for a VxF definition in the repository.
<b>Target service</b>	Portal

#### VxF developer usage scenarios

<b>No</b>	#3010
<b>Title</b>	Register a VxF definition
<b>Actor</b>	VxF developer
<b>Description</b>	An authorized VxF developer should be able to create a VxF definition in the repository. The VxF developer registers any metadata of the VxF together with a packaging archive to the repository.
<b>Target service</b>	Portal and 5GinFIRE wiki

<b>No</b>	#3020
<b>Title</b>	Management of VxFs
<b>Actor</b>	VxF developer
<b>Description</b>	The developer should also be enabled to update a VxF descriptor, along with the archive of the VxF, to the 5GinFIRE repository. The VxF developer may or may not set the VxF descriptor as public/visible to other users of the platform (it may want to test the implementation of a VxF before making it available to experimenters and/or other VxF developers). Analogously, the VxF developer should be able to edit or delete a VxF descriptor (along with its corresponding archive images).
<b>Target service</b>	Portal

<b>No</b>	#3030
<b>Title</b>	Search for VxFs
<b>Actor</b>	Experimenter
<b>Description</b>	An authorized VxF developer should be able to search for a VxF definition in the repository.
<b>Target service</b>	Portal

#### Testbed provider usage scenarios

<b>No</b>	#4010
<b>Title</b>	Deployment support of a Validated experiment
<b>Actor</b>	System Administrators
<b>Description</b>	In case of successful validation of an experiment by System Administrators and Testbed Providers, the experiment is marked as Validated. The deployment process may require support from the testbed providers, prior, during and/or after the experiments.
<b>Target service</b>	Portal and Issue management system

<b>No</b>	#4020
<b>Title</b>	Description and availability of experimentation resources
<b>Actor</b>	Experimenter / Testbed provider
<b>Description</b>	The description and availability of resources may be updated by testbed providers according to their scheduled experimentation activities.
<b>Target service</b>	Portal

#### Service Administrator

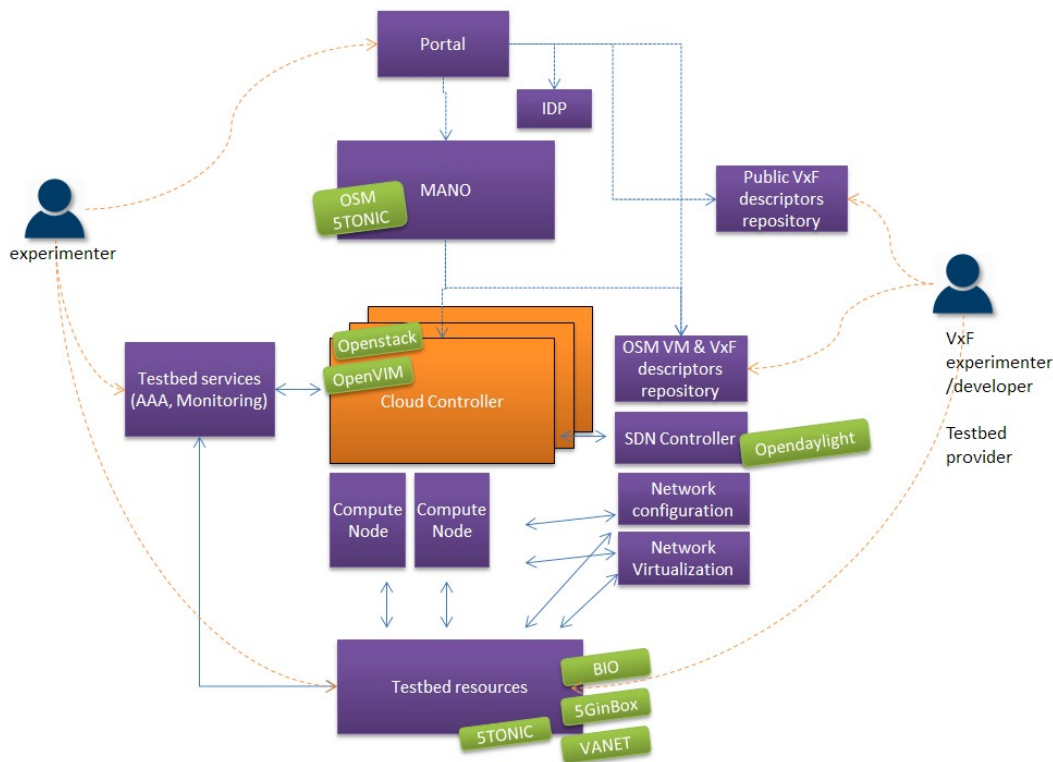
<b>No</b>	#5010
<b>Title</b>	Management of user accounts
<b>Actor</b>	Service Administrator
<b>Description</b>	Authorizes and manages users and their roles

<b>Target service</b>	Portal
-----------------------	--------

<b>No</b>	#5020
<b>Title</b>	Validation of an experiment
<b>Actor</b>	Service Administrator
<b>Description</b>	The experiment will be evaluated offline by the responsible entity of 5GinFIRE, probably with the support of the testbed providers (e.g. to check the suitability of the experiment to be executed over the 5GinFIRE facilities). Portal and Issue management services will be used to communicate with the experimenter and validate it
<b>Target service</b>	Portal and Issue Management Service

<b>No</b>	#5030
<b>Title</b>	Deploy a Validated experiment
<b>Actor</b>	Service Administrator
<b>Description</b>	In case of successful validation of an experiment by 5GinFIRE responsible System Administrators and Testbed Providers, the experiment is marked as Validated.  5GinFIRE responsible System Administrators should be able to instantiate the network services indicated by the experiment definition, during the time slots assigned to the experimentation activities.
<b>Target service</b>	Portal

## 4.5 5GinFIRE architecture



**Figure 37: Architectural approach of 5GinFIRE**

Figure 37 displays the core architectural components envisaged so far in 5GinFIRE together with their connections. Next section describes each component and its interfaces

### 4.5.1 Internal and external components and their interfaces

#### 4.5.1.1 Portal

A portal in terms of web application that end users (Experimenters, VxF developers) can subscribe, manage experiments, browse our repository, monitor experiment results, etc. It will also offer services that will allow admins to manage the offered 5GinFIRE platform as well as to manage the repository. It will provide Access to the 5GinFIRE repository of Vxfs metadata and templates, categorized in EVIs through well specified APIs. Finally it will contain a

AAA mechanism with other FIRE testbeds via the Fed4FIRE AAI technology, thus accepting seamlessly FIRE users allowing the creation of federated experiments and facilitate integration of existing FIRE facilities

#### Interfaces of Portal

**OSM:** uses this interface to communicate with OSM via the OSM API in order to push experiment descriptions, VxF descriptors, etc

**IDP**; An identity provider mechanism for authenticating 5GinFIRE users

**Public VxF descriptors repository**: contains all VxFs registered by 5GinFIRE users

**OSM VxF descriptors repository**: It is the OSM repository where VxF that will be instantiated need to be committed

#### **4.5.1.2 IDP**

An Identity provider (IdP), used to provide identifiers for users that will interact with 5GinFIRE services and provide other information about the user

##### **Interfaces of IDP**

**Portal**: authenticating portal users

**Support tools**: Ticketing system authentication

#### **4.5.1.3 Public VxF descriptors repository**

It will hold a catalog of all registered VxF descriptors. These descriptors will be available to be used by experimenters. It can have a YAML format and be compliant with other efforts like the one from the SONATA project.

##### **Interfaces of Public VxF descriptors repository**

**Portal**: Will accept request of managing VxF descriptors and their archives

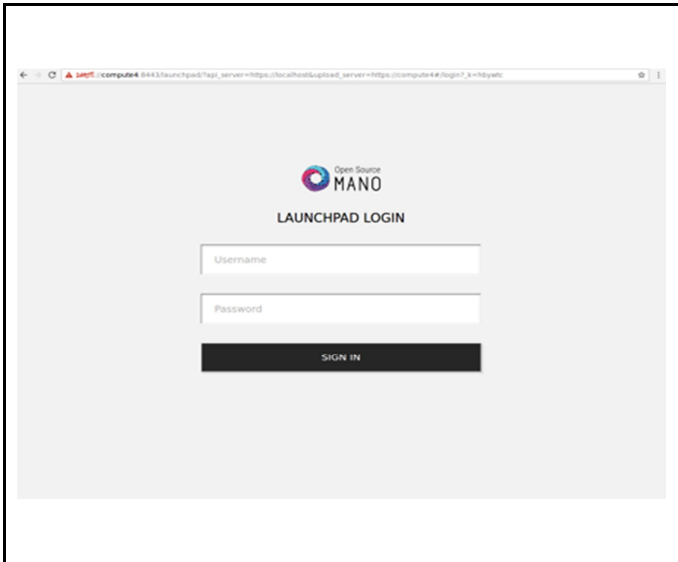
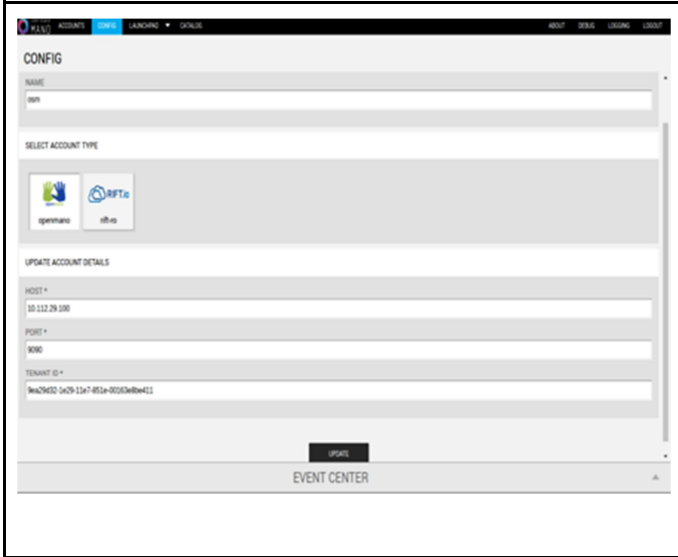
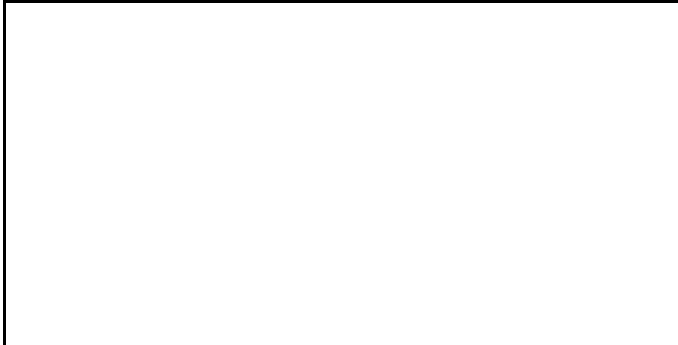
#### **4.5.1.4 5GinFIRE MANO platform**

MANO (*Management and Orchestration*) is one of the core concepts of the ETSI NFV reference architectural framework, in charge of allocating and configuring the infrastructure resources used by a virtualized network service, deploying and interconnecting the associated virtual network functions (VNFs) and their components, and managing the lifecycle of these functions and services on the NFV infrastructure. Considering our detailed state-of-the-art analysis on the open-source MANO frameworks currently under development, we have chosen Open Source MANO (OSM) as the base software platform to build the MANO component of the 5GinFIRE architecture (the reasons for this selection are motivated in Section 2.5.4).

The MANO platform of 5GinFIRE will receive orchestration actions from the 5GinFIRE portal (e.g., to create/delete a VNFD in/from the OSM catalog, to create/delete an NSD in/from the OSM catalog, to instantiate a NS, etc.). Through this interaction with the portal, the Network Service Orchestrator (NSO) of the OSM stack will receive appropriate information to instantiate the different network services comprising a given experimentation scenario, and the corresponding events related to the lifecycle of the services. The NSO will take care of the delivery of the services, interacting with the Resource Orchestrator (RO) and the VNF Configuration & Abstraction components of the OSM architecture. The RO will coordinate the allocation and setup of the computing, storage and network resources, which are necessary for the instantiation and interconnection of VxFs, interacting with the appropriate Virtualized Infrastructure Managers (VIM) available at the experimental infrastructures connected to it (as it is detailed in Section 4.5.1.5, each partner providing an experimental facility in 5GinFIRE will deploy and maintain a VIM).

**OSM operations and relation to the 5GinFIRE experimentation workflow.**

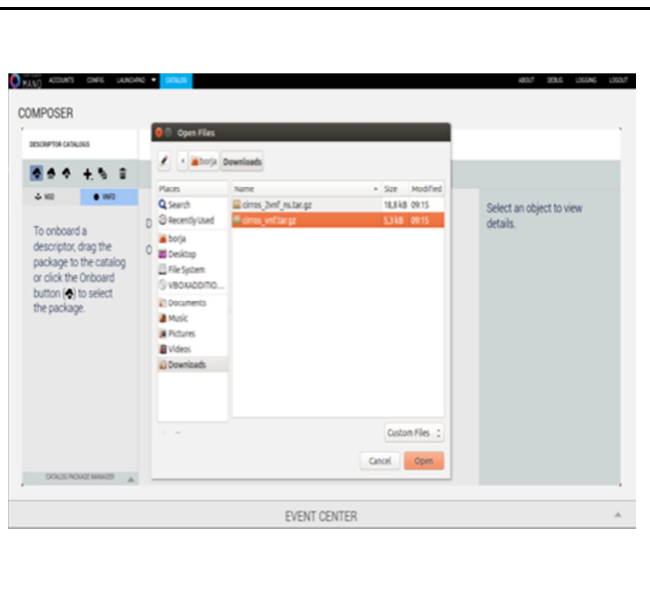
The current release of OSM (OSM release ONE, at the time of writing) supports the diverse actions identified in the 5GinFIRE experimentation workflow (see Section 4.3), related with the orchestration and lifecycle management of network services. In the following, we provide a set of snapshots, obtained from the graphical user interface of OSM, to serve as illustrative examples of the main operations that can be executed using the OSM software stack.

	<p>Environment Login. Operations over the OSM deployment are done under an administrator profile.</p>
	<p>Configuring a Resource Orchestrator (RO) account (e.g. OpenMANO).</p>
	<p>Configuring a VIM tenant for the resource orchestrator (the configured VIM will manage a datacenter where NSs will be</p>

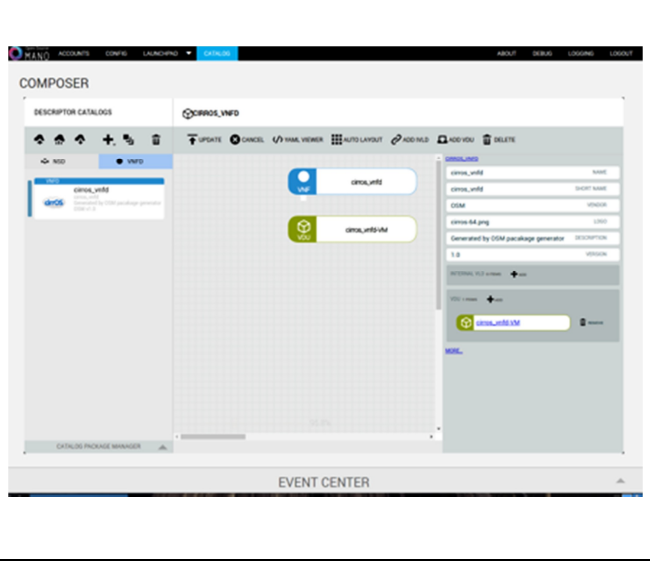
```

root@0:~# openmano config
OPENMANO_HOST: localhost
OPENMANO_PORT: 8080
OPENMANO_TENANT: osm
OPENMANO_DATACENTER: None
root@0:~#
root@0:~#
root@0:~# openmano datacenter-create openvln-site http://280.288.1.122:9080/openvln --type openvln --description "V2X type openvln"
6a204322-2ec9-11e7-8099-00163e0be411 openvln-site
root@0:~#
root@0:~# openmano datacenter-attach openvln-site --vln-tenant-id 628c29aa-1e96-11e7-836e-00002799c9be
6a204322-2ec9-11e7-8099-00163e0be411 openvln-site
root@0:~#
root@0:~#
root@0:~# openmano datacenter-llist
6a204322-2ec9-11e7-8099-00163e0be411 openvln-site
root@0:~#
    
```

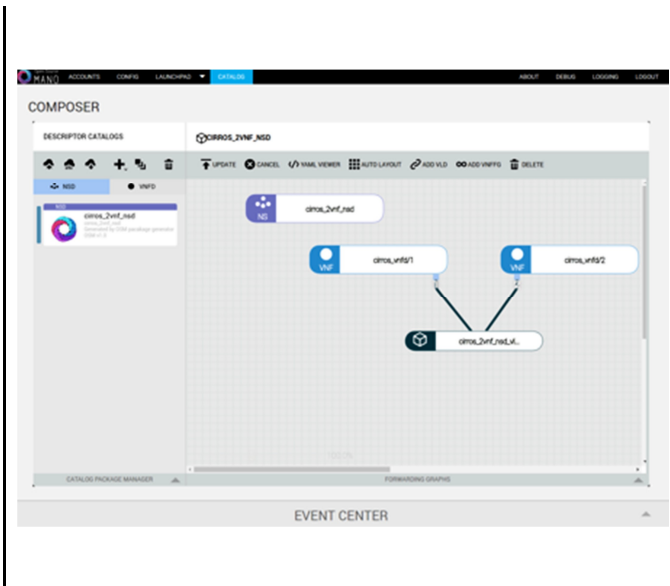
deployed). In the case of using OpenMANO, this configuration is done through the command line interface, as shown in the snapshot.



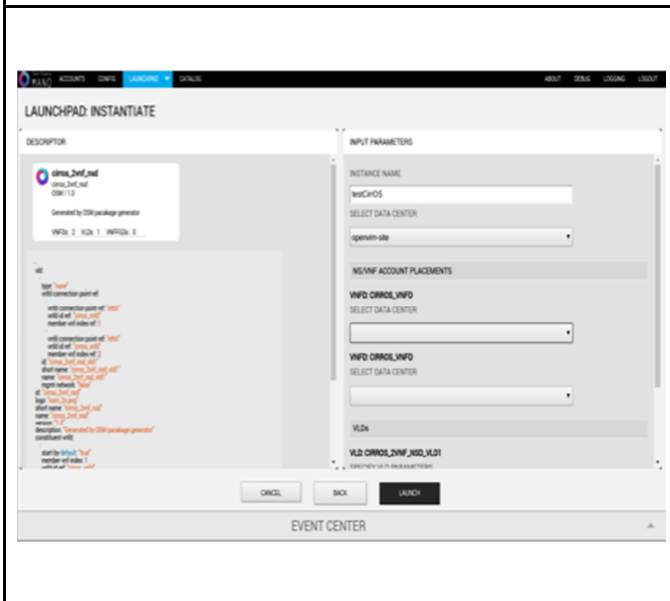
ON-boarding VNFs and NS descriptors to the OSM catalogue through packages distributed in tar.gz format.



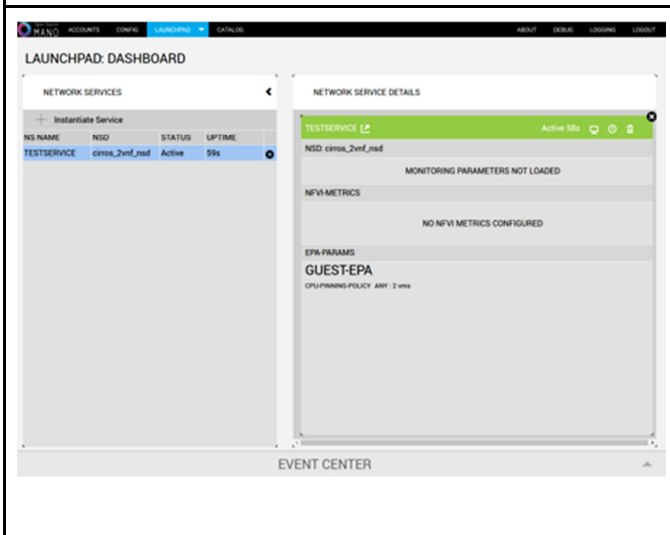
Browsing and managing available VNFs



Composing network services in terms of VNF Descriptors



Setting up the NS instantiation with the input parameters, supporting multi-site deployments if more than one datacenter has been configured.



Monitoring and lifecycle management of network services



The above mentioned examples show that OSM supports the main operations, related to the deployment and management of network services and virtualized network functions that are required to implement the experimentation workflow of 5GinFIRE. Besides the graphical user interface, these operations can also be executed using a command line interface (CLI). In addition, OSM Release ONE provides a REST Application Programming Interface [122] (API). This API will serve to implement the interactions between the 5GinFIRE portal and the MANO platform, enabling the execution of the following actions from the portal: NS and VNF package management, NS Descriptor management, NS lifecycle management, VNF Descriptor management and VNF lifecycle Management. Adaptations to the OSM software base to harmonize the OSM deployment with the specific particularities of the experimental framework of 5GinFIRE will be targeted by Work package 4 (for instance, to enable the identification and tracking of experimenters when deploying experimentation scenarios).

#### **Evolution of OSM and relation to 5GinFIRE.**

The short and medium term evolution of OSM aims at making new releases of the MANO software stack to be “deployment ready”, and is governed by the requirements that for each successive release are agreed by the OSM EUAG (End User Advisory Group). 5GinFIRE will put in place a continuous integration mechanism for the MANO platform, able to guarantee the availability of a state-of-the-art MANO platform as the OSM software base evolves, making it compatible with the specific requirements of an experimental facility like 5GinFIRE. The main directions identified by the OSM community for Release TWO and beyond are [123]

- Support of dynamic VNF and network service definition.
- Service assurance features.
- Enhanced security with role-based access control and authentication.
- Support of new components via the plugin model, already present in Release ONE for incorporating new VIM components.
- Support of different approaches to service chaining.
- Support of recursive Network Services, allowing the creation of complex services from simple templates.
- Enhancing user experience.
- Modeling of VNFs and Network Services. In this particular aspect, the OSM community is committed to follow ETIS NFV results, and therefore the foreseen convergence among different data models for VNF and service descriptors.

#### **Interfaces of the 5GinFIRE MANO platform**

**Portal:** this interface is based on the REST API of OSM, and enables the MANO platform to receive requests from the portal to execute orchestration actions.

**VIMs:** interfaces towards the VIM endpoints, to request the allocation and release of computing, storage and network resources at the partners' NFV Infrastructures.

##### **4.5.1.5 VIMs (Cloud Controllers)**

Following the architectural design of Figure 37, each partner providing an experimental infrastructure to 5GinFIRE will be in charge of the deployment and maintenance of a Virtualized Infrastructure Manager (VIM) supported by the 5GinFIRE MANO platform. Being based on the current release of OSM, release ONE, our MANO platform will natively support OpenVIM (the reference VIM of OSM), OpenStack and VMware's vCloud Director.

This way, each VIM deployed at a partner infrastructure domain, will provide a compliant northbound API that may be used by the 5GinFIRE MANO deployment to control and manage the allocation of computing, storage and network resources at the partner NFV Infrastructure (NFVI).

#### **Interfaces of VIMs**

**5GinFIRE MANO:** Support the interactions with the 5GinFIRE MANO platform.

**Testbed resources:** to control and manage the computing, storage and network resources of a partner NFVI.

#### **4.5.1.6 Testbed services**

These are some testbed specific services that could be handover to the experimenter in order to ease the operations during experimentation

#### **4.5.1.7 Testbed resources**

The available resources for experimentation located in each target testbed

## **4.6 Exposed VxFs and APIs to the experimenters**

### **4.6.1 5G-In-A-Box**

---

Different UEs can connect from Wi-Fi and/or 4G radio to b<>com \* Unifier GW \* . b<>com \* Unifier GW \* manages authentication, sessions establishment and provides IP connectivity (IPv4 only) to:

- other servers hosting application(s) and/or
- Internet access.

Thanks to this IP Connectivity provided to Applications, the Application providers (e.g. OpenCall) can provide features to the connected UE using different protocols on top of this IP Connectivity.

Examples applications are:

- web server (http, ftp, ... protocols),
- sip phone server (tcp/sip, udp/rtp protocols for instance),
- Video streaming or others.

It can be used also by other specific Applications using information from the connected UE (localization information, information from UE camera ...) to aggregate them and/or provide contextual service to the users.

In conclusion, the b<>com \* Unifier GW \* does not have any APIs to expose to the experimenters. However the b<>com \* Unifier GW \* simplify connection to APIs available either on its access networks either on PDNs that it provides.

## 4.7 Testbed components and extensions to support 5G experimentation

### 4.7.1 7.1 5G-In-A-Box

#### 4.7.1.1 *b<>com \* Unifier GW \**

The *b<>com \* Unifier GW \** is a virtualized network element in charge to aggregate different network accesses (LTE, Wi-Fi, fixed networks). Thanks to virtualization, it can be deployed at different locations in Operator's datacenter, in distributed Point of Presences (PoP) or even closer to the users, at enterprise premises for instance. The *b<>com \* Unifier GW \** provides access authorization, user authentication and IP address allocation. This enables the Unifier Gateway to route user's traffic with different policies whatever the used access network is cellular or fixed/Wi-Fi.

The Unifier Gateway is expected to:

- Manage a Wi-Fi/LTE infrastructure built from standard equipment
- Unify user subscriber management for the different accesses
- Separate completely control plane from user plane traffic.
- Be deployable as a VNF in a datacenter or in an OTS server
- Secure access management
- Be able to dynamically adjust behavior depending on monitoring (e.g. through probes measurements)
- Have a platform that is available for external high-value services, and that auto-configures services deployment

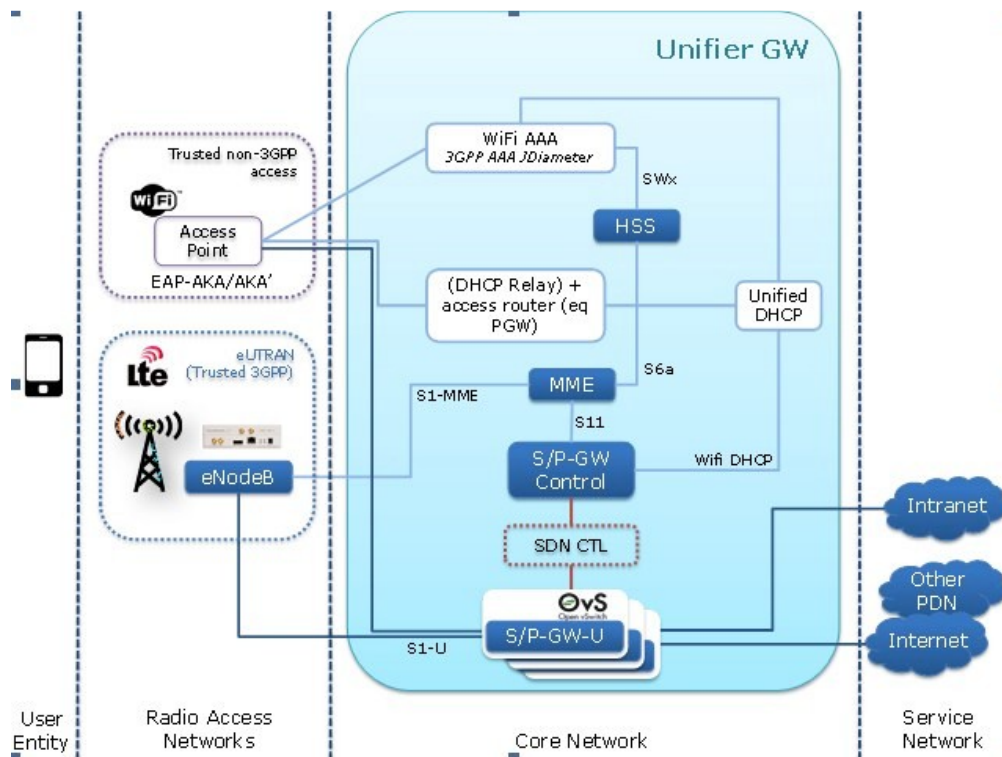


Figure 38: *b<>com \* Unifier GW \** functional Architecture Overview

#### 4.7.1.2 **5G-In-A-Box overall presentation**

5G-In-a-Box is built from b<>com \* Unifier GW \*. It is a set of VNFs providing various radio access technologies and core network features by embedding EPC with MME , HSS , S/P GW functions (virtualized : vEPC and “SDNized” : S/P GW-C and S/PGW-U) as well as AAA , DHCP and NAT. These VNFs have the ability to handle standalone private LTE and Wi-Fi networks but, within the scope of 5GinFire, the target is to insert it in a more global framework.

The scope of the deployment is:

- On one hand (see section 4.7.1.3), interface with established testbeds like BiO and/or City of Porto Automotive Testbed,
- not deploying new radio access infrastructure but interconnecting with deployed one's
- Providing connectivity in order to integrate Application Services at the edge of the network, close to the user in order to benefit from a low latency (“OverTheTop applications at the EdgeCloud”)
- On the other hand, interface with 5TONIC core platform (see section 4.7.1.4) to deploy and orchestrate VNFs installed in the established “local” testbeds (EdgeClouds managed from a central core platform)

In a MEC architecture, in order to get a low latency, the VxF may be hosted near to the edge, so close to the user. The Unifier Gateway which can have the role of an edge gateway with various radio access networks could handle the unified mobile access and provide the unified network interface for Over the Top Application Services. b<>com \* Unifier GW \* does not provide hosting capacity to host the different Applications Services. It will only provide the IP connectivity for external applications which will implement different use cases. However, b<>com \* Unifier GW \* deployment may be customizable (data-path management must be to the edge, for control management, it may depends of context).

An example of use case in the context of Smart City could be to illustrate benefits of dual radio coverage (WiFi/LTE) with unique secured authentication to provide some advanced services. An example could be (depends on what will be described in Use Cases and OpenCall candidates) to provide a VoIP application service taking benefit of b<>com \* Unifier GW \* for instance for City employees for “on site” actions with real time/live shared information (video ...) using UEs (eventually “advanced UEs” as google glass).

So, the Unifier Gateway could provide:

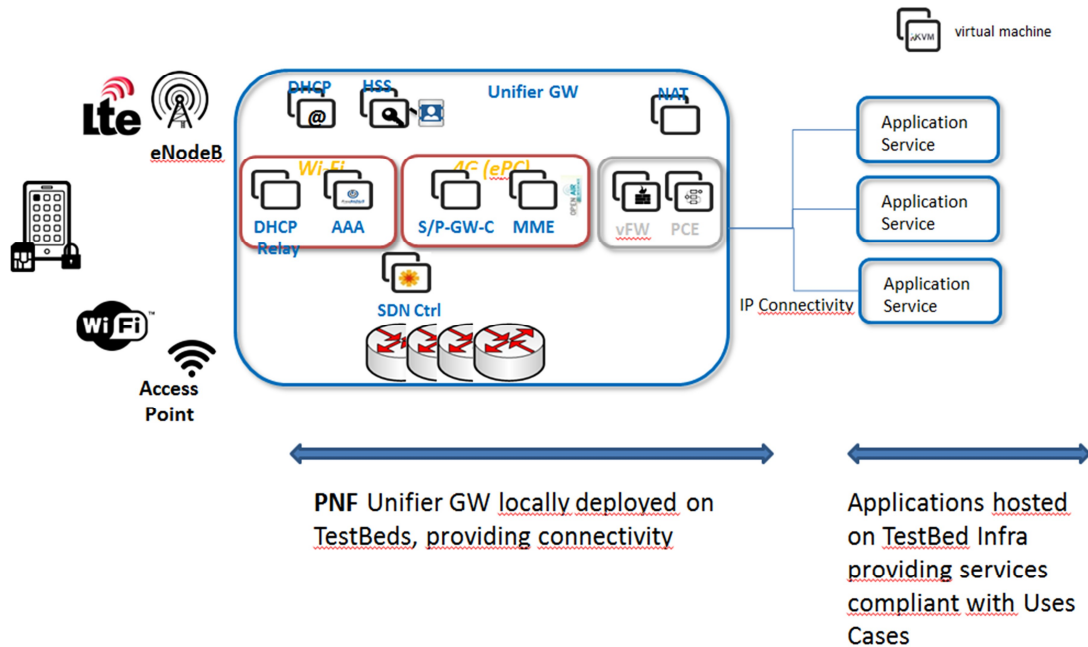
- A common unified mobile access to a Cloud Edge where Over the Top Network Applications Services (AS) may operate; these AS would be supervised and managed by 5TONIC Orchestration.
- A network interface available for VxF in order to provide Network Application services locally with better network latency. These VxF are so Cloud Edge resources hosted on Edge compute node(s), deployed on local IASS and managed by the VIM manager run by 5TONIC

In addition, these VxF are hosted on these Cloud resources and shall be under Service Orchestrator run by 5TONIC

**4.7.1.3 b<>com \* Unifier GW \* as a PNF (e.g. 5G-in-a-box from DoA)**

The b<>com \* Unifier GW \* may be installed on the local test bed as an autonomous embedded VNF running on a dedicated hardware resource (VNF provided by B-COM as a package with hardware platform). The Unifier GateWay shall be managed as a whole (hardware/software) by the Network Service Orchestrator. The Unifier Gateway will manage locally the users (embedded HSS). In such a case, there will be no capability of roaming, consistent unified users database between the different tests beds: no central common HSS.

The Unifier Gateway shall also establish the data path from local radio access networks to PDN (i.e. Internet local access in order to avoid any “tromboning” effect with a central PDN gateway); this data-path established possibly through VxF (depending on use cases).



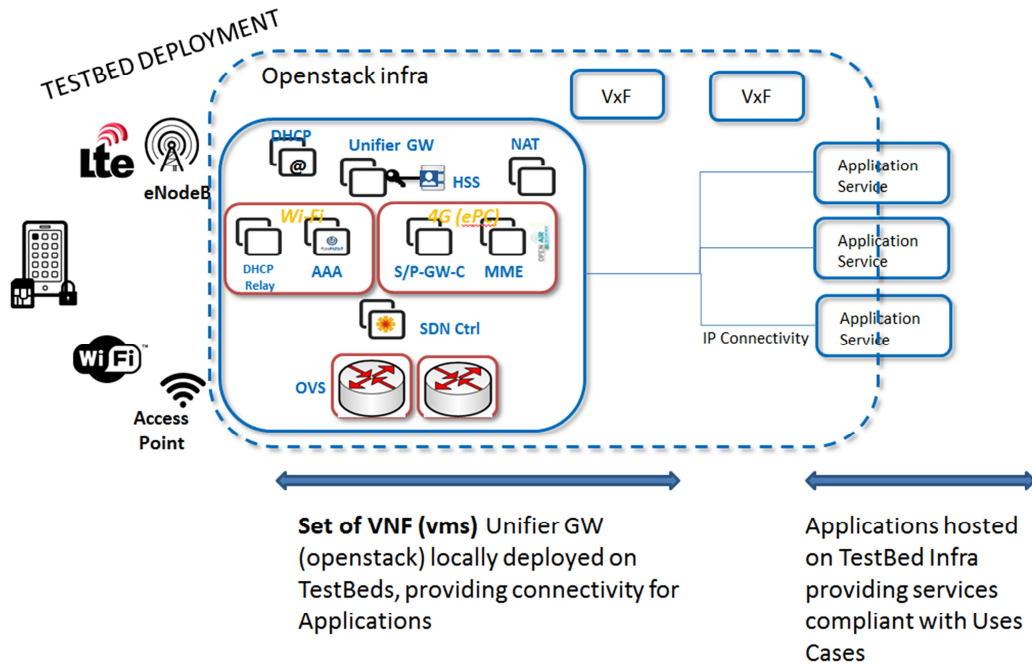
**Figure 39: 5G-In-A-Box deployed as a PNF locally to testbed**

**4.7.1.4 b<>com \* Unifier GW \* as a set of VNFs**

b<>com \* Unifier GW \* VNFs will be adapted to be deployed on an Openstack infrastructure. Different types of deployment may be experimented depending on the available infrastructures and capabilities from TestBeds:

- If Openstack infra is available at TestBed level, b<>com \* Unifier GW \* VNFs are deployed on it
- If no Openstack infra is available, b<>com \* Unifier GW \* may be provided as a “5G-in-a-box” with its own Openstack infra.

The whole b<>com \* Unifier GW \* will be instantiated at a TestBed level (a b<>com \* Unifier GW \* instance (e.g. set of VNFs) per Testbed).



**Figure 40: b<>com \* Unifier GW \* deployed as a set of VNF locally to testbed**

The b<>com \* Unifier GW \* deployment will be handled by OSM (at 5TONIC level) instantiating the different b<>com \* Unifier GW \* VNFs at testbed side.

**4.7.2 Testbed components in the 5TONIC laboratory**

The 5TONIC laboratory includes a solid baseline of facilities and infrastructure that supports advanced experimentation in the area of 5G network technologies. In this respect, the laboratory offers a data center with space for 10 racks, including one rack for communications with external networks. These racks, which are allocated to individual 5TONIC members, may be flexibly interconnected according to any experimentation requirements. Additionally, 5TONIC also provides its members with access to a common infrastructure with specific-purpose hardware, to assist in experiments, trials and demonstrations with 5G products and services. Due to confidentiality reasons, we cannot disclose all the software and hardware available in the laboratory, which also

includes experimental prototypes from the industrial and academic members. Consequently, in the following we keep our description focused, highlighting the main infrastructure and equipment at 5TONIC that will be offered for experimentation within the experimental infrastructure architecture of 5GinFIRE.

With respect to the **MANO software stack**, the UC3M rack will include a server with 32 cores, 128 GB RAM, 2 TB NLSAS hard drive and a network card supporting Intel DPDK with 4 GbE ports. This server will support the deployment of the OSM software, as well as the VIM solutions that will be tested and evaluated in Work Package 4.

The NFV infrastructure (**NFVI**) equipment will include 2 high-profile servers to test real deployments, equipped with 8 cores in a NUMA architecture, 128 GB RDIMM RAM, 4 TB SAS and 8 10Gbps Ethernet optical transceivers with SR-IOV capabilities. These servers are currently interconnected to support the deployment of the data planes, by using a switch with 24 10Gbps Ethernet optical ports. The NFVI is completed with 4 24-port 1GbE OpenFlow switches, which may be used to support SDN experimentation. The above mentioned NFVI equipment forms part of the laboratory infrastructure that is common to 5TONIC members.

We want to highlight that the UC3M rack also includes equipment to support the UC3M experimentation activities in other 5G research projects, particularly 5G-Crosshaul[124] and 5GEx[125]. Eventually, when this equipment is available, it could be considered to be incorporated to support specific experiments in 5GinFIRE. These equipment encompasses specific-purpose hardware, including: a rack server with 16 cores, 128 GB RAM, 2 TB NLSAS hard drive and 4 GbE ports supporting Intel DPDK; a set of lower-capacity servers with 8 cores, 16GB RAM and 4 TB NLSAS, 4 GbE ports with Intel DPDK; as well as commodity hardware, including 15 mini-ITX computers equipped with 8GB RAM, 128GB SSD and a module of 4 1GbE ports with Intel DPDK, which may allow a cost-effective approach to configure different network topologies of variable size and capacity.

In addition, 5TONIC provides multi-site capability by incorporating infrastructure and equipment located at TID dependencies in downtown Madrid, that can work independently or in connection with the 5TONIC main site. As in the case of the UC3M rack above, when this equipment is available and not committed to other experiments, it can be incorporated to support specific experiments in 5GINFIRE. This test-bed, known as the TID Future Network Lab, includes four high-end standard servers for NFV and SDN applications, five dedicated OpenFlow switches, and a scalable platform to support OpenVSwitch nodes on Intel-based micro-computers. In addition, a metro-core network setup composed by IP/MPLS and optical devices is available, together with several GMPLS nodes running software developed internally, built on emulated nodes running Ubuntu Linux.

---

## **4.8**      ***Resources reservation***

---

For the initial versions of the 5GinFIRE platform there will be not any specific automated resource reservation mechanism. We do not expect this to be a barrier since most experimentation deployments will be elastically deployed to cloud infrastructures. Nevertheless, the experiments will be validated and scheduled if there is a limited resource capability and as the work continues we will revisit the issue.

## 4.9 *Integration and deployment strategy*

### **Integration**

---

All components developed within the project will be integrated at the API level. For the portal and public repository to OSM the OSM API will be used.

From OSM to testbed VIMs (Visualized Infrastructure Managers) the standardized ETSI VIM interface will be used.

The following VIMs will be currently deployed to our testbeds:

**Table 3: Testbed VIM version**

<b>Testbed</b>	<b>VIM version</b>
BiO	The VIM version will be provided through OpenStack Liberty that currently includes TACKER 0.7 which has interoperability with ETSI MANO architecture framework.
ITAv	The VIM version provided will be the latest OpenStack release (currently Ocata) for the duration of the project. This deployment will support development tasks, and provide an environment for testing before code contributions to external projects.
5TONIC	Different VIM solutions will be deployed and tested, according to the VIMs utilized by partners providing experimental infrastructures at the 5TONIC testbed. This will serve to verify their appropriate interoperation with the MANO architectural component of 5GinFIRE.

### **Deployment**

---

For the deployment of the platform it is decided that the 5GinFIRE portal will be hosted by UoP which has also the development effort.

OSM will be deployed at 5TONIC. ETSI Open Source MANO group (ETSI OSM) has announced the availability of OSM Release TWO by the end of April 2017. Therefore we will use that version of OSM for the deployment of the 5GinFIRE MANO platform, although given the deadlines established for this deliverable, this document makes reference to OSM Release One (the essential features of OSM are maintained in the new version, which does not change the analysis and conclusions presented in this document).



## 4.10 Experimentation Support components and services

For 5GinFIRE operations to support end-users, we need to install some support tools, like wikis and ticket systems. Given partner experience we will deploy the following tools:

- Helpdesk / Ticketing: GLPI
- Wiki: MediaWiki

### 4.10.1 Helpdesk

#### Tool choice

Projects that will result from 5GinFIRE OpenCalls should be able to request assistance in case something is not working as it should be or to perform new requests. As the goal of 5GinFIRE platform is to host operational 5G use cases, it is highly important to track a request/incident all along until its resolution. In order to do so, solutions like mailing lists are easy to setup but they are not efficient in terms of tracking.

The best approach is to use a ticket tracking tool. For the 5GinFIRE platform operation purposes, the ticket tracking tool used will be GLPI, which provides an IT approach of issues management rather than a software development one. This should fit pretty well with the kind of issues handled within the testbed.

Users will be able to submit requests / incidents and they will be escalated according to their nature to the right team. Check section to get the details of the process flow.

#### 5GinFIRE platform support

An operational platform requires a support service in order to operate properly. Usually the support is composed of several layers or stratum that range from general issues/request troubleshooting, to specialized troubleshooting. Based on the existing roles, the following layers could be adopted:

- Layer 1: Handled by the 5GinFIRE Support group which is in charge of receiving the initial request and either resolve the issue or address it to appropriate group on the upper layers.
- Layer 2: Requests addressed to this layer will be managed by the Testbed Operator if it deals with an issue / request about the testbed execution environment or by the 5Tonic Operator if it is an issue for instance about VNF deployment.

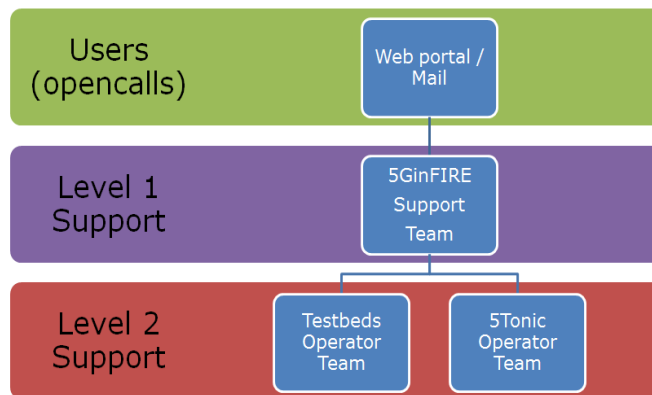
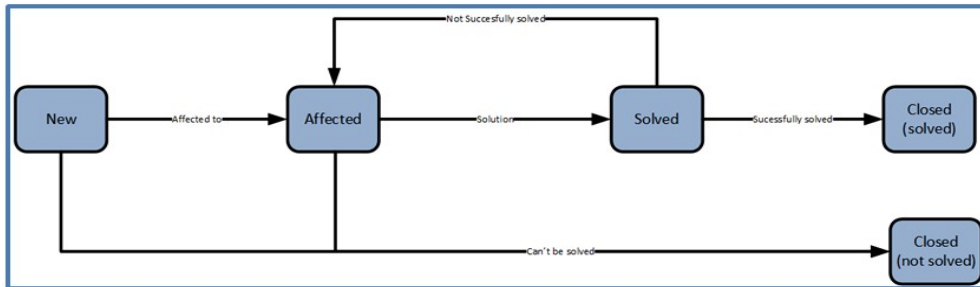


Figure 41: 5GinFIRE environment support organization

As defined in Tool choice a helpdesk tool will be available to help with the issue management and tracking. A basic support process flow is required in order to grant the proper management on the request.



**Figure 42: Support basic process flow**

The previous figure provides the support basic process flow which can be described as follows:

- A user opens a ticket for a new request / incident. The ticket is created and its status is new.
- The ticket will be evaluated by the 5GinFIRE Support team (layer 1 support) and affected to the appropriate team. The ticket status will become affected.
- Once a solution is identified and applied, the ticket will be solved.
- If the solution satisfies the user's expectations, the ticket becomes closed. Other ways it will come back on status affected.

While the ticket is open, it is possible to:

- Request complementary information to the user to better qualify the ticket.
- To close the ticket because there is not a feasible solution or because the demand is simply out of scope.

As many other tool proposed for the 5GinFire project, the ticketing system requires personal credentials to access the helpdesk tool.

When it is about deploying a use case or maintaining one, all requests coming from OpenCall's users should be performed via the helpdesk. Some of requests like account creation or new site interconnection are supposed to be addressed by mail.

#### 4.10.2 Wiki

##### ***Tool choice***

Users of 5GinFIRE and projects that will result from 5GinFIRE OpenCalls should be able to read public documentation of tools usage. This is done usually through Wikis.

For 5GinFIRE we proposed to adopt MediaWiki [93] a free software open source wiki package, originally for use on Wikipedia.

As many other tool proposed for the 5GinFire project, the Wiki system requires personal credentials to access the tool and contribute documentation. However for reading documentation it is not necessary to sign in into the system

## 5 Conclusion

---

This document has provided a first release of the 5GinFIRE Experimental Infrastructure architecture that will be deployed and to which reviewed 5GinFIRE requirements are addressed. The main ones considered were to build an Open 5G NFV based ecosystem, using Open Sources Software and where instantiating softwarised architecture from vertical industry and conduct experiments are possible. But it has been also considered that the proposed architecture fulfil requirements imposed by the testbeds and use cases.

On top of that, in this document, different software or components have been analysed for implementing the 5GinFIRE Infrastructure and it was explained how they were chosen considering the minimum development or integration time they needed as well as their stabilities and maturities.

Moreover, 2 use cases have been described and in particular the one concerning the automotive EVI where some promising VxFs were identified.

This document enables activities in the other work packages since they are directly concerned on choices that were made here:

- In the WP3, because it is where all the tooling will be developed and adapted when needed then deployed.
- In the WP4, it is where the Core MANO Service Management and Orchestration will be installed
- In the WP5, because it is where we will ready the 5G infrastructure to host experimentation for the Open Calls.

## References

- [1] OpenDaylight, «Platform Overview, » Linux foundation. Available: <https://www.OpenDaylight.org/platform-overview>.
- [2] L. Foundation, «Release Archives, ». Available: <https://www.OpenDaylight.org/software/release-archives>.
- [3] SDxCentral, «Market Report | The Future of Network Virtualization and SDN Controllers, » 2016.
- [4] L. Foundation, «OpenDaylight Features List, ». Available: <https://www.OpenDaylight.org/OpenDaylight-features-list>.
- [5] O. Foundation, «OVSDB Integration: Design, ». Available: [https://wiki.OpenDaylight.org/view/OVSDB\\_Integration:Design](https://wiki.OpenDaylight.org/view/OVSDB_Integration:Design).
- [6] Huawei, «Intent NBI for Software Defined Networking, ». Available: <http://www-file.huawei.com/~media/CNMG/Downloads/Technical%20Topics/Fixed%20Network/Intent%20ONBI%20for%20Software%20Defined%20Networking-whitepaper>
- [7] ETSI, «Network Functions Virtualisation (NFV); Architectural framework, ». Available: [http://www.etsi.org/deliver/etsi\\_gs/NFV/001\\_099/002/01.02.01\\_60/gs\\_NFV002v010201p.pdf](http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.02.01_60/gs_NFV002v010201p.pdf).
- [8] M.-P. Odini, «Open Source MANO, » IEEE, 07 2016. Available: <http://sdn.ieee.org/newsletter/july-2016/opensource-mano>.
- [9] «OpenVIM installation (Release One), » Open Source MANO. Available: [https://osm.etsi.org/wikipub/index.php/OpenVIM\\_installation\\_\(Release\\_One\)](https://osm.etsi.org/wikipub/index.php/OpenVIM_installation_(Release_One)).
- [10] ONOS, «onos projects - downloads, » ONOS. Available: <https://wiki.onosproject.org/display/ONOS/Downloads>.
- [11] ONOS, «Introducing ONOS - a SDN network operating system for service providers, » ON.LAB, 11 2014. Available: <http://onosproject.org/wp-content/uploads/2014/11/Whitepaper-ONOS-final.pdf>.
- [12] ONOS, «ONOS Project - System components, » ONOS. Available: <https://wiki.onosproject.org/display/ONOS/System+Components>.
- [13] ONOS, «ONOS projects - Downloads,» ONOS. Available: <https://wiki.onosproject.org/display/ONOS/Downloads>.
- [14] Open Networking Foundation, «2016 SDN Controller Landscape - Is there a winner,» 2016.
- [15] Bondkovskii, A. et al., 2016. Qualitative comparison of open-source SDN controllers. In NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium. NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium. pp. 889-894.
- [16] JUNIPER, 2013. Juniper Contrail Configuration API Model. Available at: <http://configuration-schema-documentation.s3-website-us-west-1.amazonaws.com/R3.2/> [Accessed March 20, 2017].
- [17] JUNIPER, 2015. Juniper Networks Contrail - Technical Documentation. Available at: [http://www.juniper.net/techpubs/en\\_US/release-independent/contrail/information-products/pathway-pages/index.html](http://www.juniper.net/techpubs/en_US/release-independent/contrail/information-products/pathway-pages/index.html) [Accessed March 20, 2017].
- [18] Khondoker, R. et al., 2014. Feature-based comparison and selection of Software Defined Networking (SDN) controllers. In 2014 World Congress on Computer Applications and

- Information Systems (WCCAIS). 2014 World Congress on Computer Applications and Information Systems (WCCAIS). pp. 1-7.
- [19] OpenContrail, 2013a. Contrail Controller License. GitHub. Available at: <https://github.com/Juniper/contrail-controller/blob/master/LICENSE> [Accessed January 27, 2017].
- [20] OpenContrail, 2013b. Contrail vRouter License. GitHub. Available at: <https://github.com/Juniper/contrail-vrouter/blob/master/LICENSE> [Accessed January 27, 2017].
- [21] OpenContrail, 2015. OpenContrail. Available at: <http://www.opencontrail.org/> [Accessed January 27, 2017].
- [22] Singla, A. & Rijsman, B., 2015. OpenContrail Architecture Document. Available at: <http://www.opencontrail.org/opencontrail-architecture-documentation/> [Accessed January 27, 2017].
- [23] Xie, J. et al., 2015. Control plane of software defined networks: A survey. Computer Communications, 67, pp.1-10.
- [24] Ryu SDN controller Official Website <https://osrg.github.io/ryu/>
- [25] Ryu book Using OpenFlow 1.3RYU project team <https://osrg.github.io/ryu-book/en/Ryubook.pdf>
- [26] RYU OpenFlow Controller, Dean Pemberton NSRC, Andy Linton NSRC, Sam Russell -REANNZ University of Oregon
- [27] Comparison of Ryu and OpenDaylight Northbound APIs <https://blog.zhaw.ch/icclab/comparison-of-ryu-and-OpenDaylight-northbound-apis/>
- [28] OSGi Alliance <https://www.osgi.org/developer/specifications/>
- [29] ETSI GS NFV 002 V1.2.1, "Network Functions Virtualisation (NFV); Architectural Framework", version 1.2.1, Dec. 2014.
- [30] Adrian Hoban, Alfonso Tierno Sepulveda, Gerardo García de Blas, Kiran Kashalkar, Mark Shuttleworth, Matt Harper, Rajesh Velandy, "OSM Release One, A Technical Overview", Oct. 2016.
- [31] D. R. Lopez, "OpenMANO: The Dataplane Ready Open Source NFV MANO Stack," in IETF Meeting Proceedings, Dallas, Texas, USA, March 2015.
- [32] ETSI GS NFV-PER 001 V1.1.2, "Network Functions Virtualisation (NFV); NFV Performance & Portability Best Practices", version 1.1.2, Dec. 2014.
- [33] Intel Corporation, "Preboot Execution Environment (PXE) Specification", versión 2.1, sep 2009.
- [34] R. Droms, "Dynamic Host Configuration Protocol", Internet Engineering Task Force, Request for Comments 2131, March 1997.
- [35] K. Sollins, The TFTP Protocol, Internet Engineering Task Force, Request for Comments 1350, July 1992.
- [36] Intel, Hewlett-Packard, NEC, Dell, "Intelligent Platform Management Interface Specification Second Generation", version 2.0, April 2015, <http://www.intel.com/content/www/us/en/servers/ipmi/ipmi-technical-resources.html>
- [37] <http://www.ict-fire.eu>
- [38] J. M. Marquez-Barja, N. Kaminski, F. Paisana, C. Tranoris, and L. A. DaSilva, "Virtualizing testbed resources to enable remote experimentation in online telecommunications education," in IEEE

- Global Engineering Education Conference (EDUCON15), Mar. 2015, pp. 836-843. [Online]. Available: <http://dx.doi.org/10.1109/educon.2015.7096069>
- [39] <http://www.ict-fire.eu/projects>
- [40] <http://www.fp7-ofelia.eu/>
- [41] <https://openflow.stanford.edu/display/FOAM/Home>
- [42] T. Rakotoarivelo, M. Ott, G. Jourjon, I. Seskar, "OMF: a control and management framework for networking testbeds", in ACM SIGOPS Operating Systems Review 43 (4), 54-59, Jan. 2010. [Online]. Available: <http://dx.doi.org/10.1145/1713254.1713267>
- [43] R. McGeer, M. Berman, C. Elliott, and R. Ricci, "The GENI Book", Springer, 2016. ISBN 978-3-319-33767-8. Available: <http://doi.10.1007/978-3-319-33769-2>.
- [44] H. Mussman, "Extending GENI to Support Large-Scale Research and Service Experiments", in TRIDENTCOM, 2012. Available: [http://groups.geni.net/geni/raw-attachment/wiki/GEC13Agenda/WiMAXPlanning/020112b\\_ExtendingGENI\\_TridentCom2012.pdf](http://groups.geni.net/geni/raw-attachment/wiki/GEC13Agenda/WiMAXPlanning/020112b_ExtendingGENI_TridentCom2012.pdf)
- [45] M. A. Marotta, F. J. Carbone, J. J. C. de Santanna and L. M. R. Tarouco, "Through the Internet of Things - A Management by Delegation Smart Object Aware System (MbDSAS)". In 2013 IEEE 37th Annual Computer Software and Applications Conference (pp. 732-741). Kyoto, Japan ,2013.
- [46] "GENI Design Activities", <http://groups.geni.net/geni/wiki/GeniDesign> , last accessed February 3rd 2014
- [47] W. Van de Meerssche, et al., "Federation AM APIs", <https://fed4fire-testbeds.ilabt.iminds.be/asciidoc/federation-am-api.html>
- [48] "GENI Aggregate Manager API Version 3", [http://groups.geni.net/geni/wiki/GAPI\\_AM\\_API\\_V3](http://groups.geni.net/geni/wiki/GAPI_AM_API_V3) , last accessed February 3rd 2014
- [49] D2.9 - Final federation architecture <https://www.fed4fire.eu/wp-content/uploads/2016/10/d2-9-final-federation-architecture.pdf>
- [50] Architecture for the Heterogeneous Federation of Future Internet Experimentation Facilities
- [51] [https://www.fed4fire.eu/fileadmin/dissemination\\_items/2013-07-FutureNetworkSummit2013\\_FinalPaper\\_ref\\_168\\_published\\_IEEE\\_Xplore.pdf](https://www.fed4fire.eu/fileadmin/dissemination_items/2013-07-FutureNetworkSummit2013_FinalPaper_ref_168_published_IEEE_Xplore.pdf)
- [52] <https://osm.etsi.org>
- [53] <https://portal.etsi.org/TBSiteMap/OSM/ListofOSMMembers.aspx>
- [54] <https://riftio.com>
- [55] <https://github.com/nfvlab/openmano>
- [56] <https://jujucharms.com>
- [57] <http://www.apache.org/licenses/LICENSE-2.0>
- [58] [https://osm.etsi.org/wikipub/index.php/Main\\_Page](https://osm.etsi.org/wikipub/index.php/Main_Page)
- [59] <https://www.riftio.com/rift-ware-orchestration-automation/>
- [60] <https://www.sdxcentral.com/articles/news/surprising-whos-using-mano-code-cloudify/2016/05/>
- [61] <http://getcloudify.org/network-function-virtualization-vnf-nfv-orchestration-sdn-platform.html>

- [62] [http://getcloudify.org/cloud\\_orchestration\\_cloud\\_automation.html](http://getcloudify.org/cloud_orchestration_cloud_automation.html)
- [63] <https://github.com/Orange-OpenSource/opnfv-cloudify-clearwater>
- [64] <http://www.ubuntu.com/cloud/juju>
- [65] [https://osm.etsi.org/wikipub/images/d/d4/OSM%2816%29000018r1\\_MWC16\\_architecture\\_overview.pdf](https://osm.etsi.org/wikipub/images/d/d4/OSM%2816%29000018r1_MWC16_architecture_overview.pdf)
- [66] <https://wiki.openstack.org/wiki/Murano>
- [67] <https://www.mirantis.com/blog/open-source-mano-osm-to-work-on-nfv-orchestration/>
- [68] <https://www.opnfv.org/>
- [69] <https://www.linuxfoundation.org/>
- [70] <https://www.opnfv.org/about/members>
- [71] <https://www.ietf.org/>
- [72] <https://www.mef.net/>
- [73] <http://artifacts.opnfv.org/opnfvdocs/colorado/2.0/docs/overview/index.html#document-overview>
- [74] <https://www.opnfv.org/community/projects/pharos>
- [75] <https://www.opnfv.org/software>
- [76] <https://wiki.openstack.org/wiki/Ironic>
- [77] <http://docs.openstack.org/developer/ironic/deploy/user-guide.html>
- [78] <https://maas.io/>
- [79] <https://docs.ubuntu.com/maas/2.1/en/>
- [80] [https://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=tosca](https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca)
- [81] <http://ariatosca.org/>
- [82] <https://wiki.openstack.org/wiki/TOSCA-Parser>
- [83] <http://openbaton.github.io/documentation/tosca-CSAR-onboarding/>
- [84] <http://repo.riftio.com/releases/open.riftio.com/4.2.2/RIFTware4.2.2.0-ReleaseNotes.txt>
- [85] <https://github.com/juju/juju-tosca>
- [86] <http://murano-specs.readthedocs.io/en/latest/specs/mitaka/support-tosca-format.html>
- [87] <https://fed4fire-testbeds.ilabt.iminds.be/asciidoc/federation-am-api.html>
- [88] <https://github.com/open-multinet/federation-am-api>
- [89] [https://fed4fire-testbeds.ilabt.iminds.be/asciidoc/federation-am-api.html#\\_basic\\_am\\_call\\_concepts](https://fed4fire-testbeds.ilabt.iminds.be/asciidoc/federation-am-api.html#_basic_am_call_concepts)
- [90] <https://fed4fire-testbeds.ilabt.iminds.be/asciidoc/rspec.html>
- [91] <http://www.protogeni.net/wiki/Flack>
- [92] <http://trac.gpolab.bbn.com/gcf/wiki/Omni>
- [93] <https://www.mediawiki.org/wiki/MediaWiki>

- [94] Albino, Vito, Umberto Berardi, and Rosa Maria Dangelico. 2015. "Smart Cities: Definitions, Dimensions, Performance, and Initiatives." *Journal of Urban Technology* 22 (1): 3–21. doi:10.1080/10630732.2014.942092.
- [95] Palattella, M. R., M. Dohler, A. Grieco, G. Rizzo, J. Torsner, T. Engel, and L. Ladid. 2016. "Internet of Things in the 5G Era: Enablers, Architecture, and Business Models." *IEEE Journal on Selected Areas in Communications* 34 (3): 510–27. doi:10.1109/JSAC.2016.2525418.
- [96] Zanella, A., N. Bui, A. Castellani, L. Vangelista, and M. Zorzi. 2014. "Internet of Things for Smart Cities." *IEEE Internet of Things Journal* 1 (1): 22–32. doi:10.1109/JIOT.2014.2306328.
- [97] Deloitte, « Smart Cities: How rapid advances in technology are reshaping our economy and society» [En ligne]. Available: <https://www2.deloitte.com/content/dam/Deloitte/tr/Documents/public-sector/deloitte-nl-ps-smart-cities-report.pdf>
- [98] Nokia, « Ultra-broadband networks empower smart cities» [En ligne]. Available: <https://insight.nokia.com/ultra-broadband-networks-empower-smart-cities>
- [99] Huawei, « Enabling Smart Cities with Mobile Broadband,» [En ligne]. Available: [http://enterprise.huawei.com/ilink/enenterprise/download/HW\\_373080](http://enterprise.huawei.com/ilink/enenterprise/download/HW_373080)
- [100] [http://organicity.eu/wp-content/uploads/D2.3\\_OrganiCity-EaaS-facility-Basic.pdf](http://organicity.eu/wp-content/uploads/D2.3_OrganiCity-EaaS-facility-Basic.pdf)
- [101] Selbstfahrende Autos, Hersteller sollen unbegrenzt haften, *Handelsblatt*, 21st July 2016, p.13
- [102] [https://www.ncsc.gov/nittf/docs/CNSSI-4009\\_National\\_Information\\_Assurance.pdf](https://www.ncsc.gov/nittf/docs/CNSSI-4009_National_Information_Assurance.pdf)
- [103] Functional Safety brochure by the International Electrotechnical Commission (IEC) - [http://www.iec.ch/about/brochures/pdf/technology/functional\\_safety.pdf](http://www.iec.ch/about/brochures/pdf/technology/functional_safety.pdf)
- [104] ORGANICITY Project (<http://organicity.eu>)
- [105] EMBERS Project (<http://embers.city/>)
- [106] FESTIVAL (<http://www.festival-project.eu>)
- [107] FIESTA-IoT (<http://fiesta-iot.eu>)
- [108] Smart Santander (<https://www.fed4fire.eu/smart-santander/>)
- [109] Community Lab (C-LAB) (<https://www.fed4fire.eu/community-lab/>)
- [110] UFU Future Internet Testbed (<http://www.xipi.eu/Infrastructures/UFU-Future-Internet-Testbed>)
- [111] OFELIA (<http://www.fp7-ofelia.eu/>)
- [112] FIWARE ([www.firmware.org](http://www.firmware.org))
- [113] FIBRE project (<http://fibre-ict.eu>)
- [114] Granja Marileusa ([www.granjamarileusa.com.br](http://www.granjamarileusa.com.br))
- [115] ODTONE (<http://hng.av.it.pt/projects/odtone>)
- [116] FIBRENET (<http://fibre.org.br/infrastructure/resources/>)
- [117] National Research Network (RNP – <https://www.rnp.br/en>)
- [118] Ultralight 2.0 (<https://github.com/telefonicaid/iotagent-ul>)
- [119] JSON (<https://github.com/telefonicaid/iotagent-json>)
- [120] freeboard (<https://freeboard.io/>)



- [121] Diogo Lopes, Susana Sargento, “Network Mobility for Vehicular Networks”, IEEE International Symposium on Computer and Communications (ISCC), Madeira, Portugal, June 2014.
- [122] SO REST API (March 1, 2017): <https://osm.etsi.org/wikipub/images/2/24/Osm-r1-so-rest-api-guide.pdf>
- [123] Francisco-Javier Ramón Salguero, Introducing Open Source Mano, Presentation at the OSM Workshop, SDN World Congress 2016, The Hague (October 10th, 2016)
- [124] <http://5g-crosshaul.eu>
- [125] <http://www.5gex.eu>
- [126] Omnes, N., M. Bouillon, G. Fromentoux, and O. L. Grand. 2015. “A Programmable and Virtualized Network IT Infrastructure for the Internet of Things: How Can NFV SDN Help for Facing the Upcoming Challenges.” In *2015 18th International Conference on Intelligence in Next Generation Networks*, 64–69. doi:10.1109/ICIN.2015.7073808
- [127] <https://www.onap.org/>