**5G-PPP Software Network Working Group**

# Cloud-Native and Verticals' services

## 5G-PPP projects analysis

**August 2019**

Date: 2019-08-31 Version: Draft/1.0

## Abstract

As part of the 5G-PPP Initiative, the Software Network Working Group prepared this white paper as a follow-up of 2018 Cloud-Native transformation white paper to analyze how 5G-PPP projects[1] interpret Cloud-Native design patterns and identify adoption barriers. The Software Network Working Group conducted a survey to collect technical inputs from 5G-PPP Phase 2[2] and Phase 3[3] projects on their:

- supported vertical use-cases,
- adopted virtualization technologies,
- followed architecture patterns.

Results are two-fold:

1) Projects are clustered according to their architecture patterns:
    a. Most project prototypes evolved from ETSI MANO relying on an Openstack VIM exclusively to include Kubernetes - on bare metal and public cloud - as a new VIM in parallel to Openstack.
    b. Meanwhile, they kept orchestration intelligence centralized in a VNFM-like box.
    c. Therefore, only a few of them fully exploited Kubernetes as a complete autonomous platform with its own orchestration intelligence able to host both Containerized Network Functions and classical VM-based VNFs.
2) Analysis:
    a. We acknowledge a *reluctance* for using fully Cloud-Native design provided by e.g. Kubernetes. This reluctance has been analyzed to extract the underlying reasons motivating projects to select this intermediate step where Kubernetes is considered only as a VIM. These reasons are presented as the barriers to adopt the Cloud-Native patterns.
    b. These barriers are essentially *the lack of standard* and *technological maturity*, implying *human adaptation resistance*.

[1] NGPaaS [1], 5G-MEDIA [2], 5G-Transformer [3], 5G-City [4], 5G-Picture [5], 5GTANGO [6], SAT5G [7], MATILDA [8], Slicenet [9], METROHAUL [10] , NRG5 [80], and Phase 3 project 5GENESIS [11].

[2] 5GPPP Phase 2 Projects - https://5g-ppp.eu/5g-ppp-phase-2-projects/

[3] 5GPPP Phase 3 Projects - https://5g-ppp.eu/5g-ppp-phase-3-projects

# Table of Contents

# List of Acronyms and Abbreviations

| | |
|---|---|
| 3GPP | 3rd Generation Partnership Project |
| 5G-PPP | 5G Infrastructure Public Private Partnership |
| 5GT-MTP | 5G-TRANSFORMER Mobile Transport and Computing Platform |
| 5GT-SO | 5G-TRANSFORMER Service Orchestrator |
| 5GT-VS | 5G-TRANSFORMER Vertical Slicer |
| API | Application Programming Interface |
| CAM | Cooperative Awareness Message |
| CI/CD | Continuous Integration/Continuous Delivery |
| CIM | Cooperative Information Manager |
| CNF | Container Network Function |
| COE | Container Orchestration Engine |
| CSP | Cloud Service Provider |
| DENM | Decentralized Environmental Notification Message |
| DPDK | Data Plane Development Kit |
| EC | European Commission |
| EU | European Union |
| eNB | Evolved Node B |
| ETSI | European Telecommunications Standards Institute |
| EVS | Extended Virtual Sensing |
| IoT | Internet of Thing |
| K8s | Kubernetes |
| KPI | Key Performance Indicator |
| LCM | Life Cycle Management |
| LTE | Long Term Evolution |
| MANO | Management and Orchestration |
| MCPTT | Mission-Critical Push-to-Talk |
| MEC | Multi-Access Edge Computing |
| NFV | Network Function Virtualization |
| NUMA | Non-Uniform Memory Access |
| PaaS | Platform as a Service |

| | |
|---|---|
| OSS | Operations Support System |
| KPI | Key Performance Indicator |
| RAN | Radio Access Network |
| SBA | Service Based Architecture |
| SFC | Service Function Chaining |
| SLA | Service Level Agreement |
| SLPOC | Single Logical Point of Contact |
| SM | Service Manager |
| SOA | Service Oriented Architecture |
| VNF | Virtual Network Function |
| VNFFG | Virtual Network Function Forwarding Graph |
| VNFC | Virtual Network Function Component |
| VM | Virtual Machine |
| UE | User Equipment |
| WAN | Wide Area Network |
| WG | Work Group |
| WP | White Paper |
| ZOOM | Zero touch Orchestration Operation and Management |

# 1 Introduction

**Importance of Verticals in 5G**: Verticals are an essential driving force behind 5G systems. Whether it is augmented reality, IoT, V2x, or any other application, it is clear by now that the human-to-human communication is only a fraction of what 5G promises to deliver. Most of the traffic or data flowing through 5G networks will be sourced from those different verticals business and applications.

**Benefits of Cloud to Verticals**: More than ever before, verticals are competing at an accelerated pace, each trying to get the biggest market share in its domain. There are currently vertical industries that rely on 4G systems and that will continue to exist in 5G. Those verticals that built their software-based systems on "legacy" technologies will need to adapt or migrate towards newer technologies as 5G systems matures. Newcomers to appear after 5G becomes available, however, must envision building their industry in the most flexible and adaptive way leveraging 5G capabilities. These features as well as short time-to-market and other benefits that Cloud-Native principles offer will keep verticals in the competition and allow them to deliver valuable quality services in a short time.

**Are Verticals ready?** So, verticals are the main beneficiary of 5G which, from a software perspective, is intended to adhere to an architecture based on Cloud-Native principles to leverage the ubiquity of Cloud Computing. It is then important to ask where are the verticals with respect to Cloud-Native? How far have they gotten in using this technology? Where is the potential? What are the challenges or impediments? And what is missing to bring them closer to embracing Cloud technologies and Cloud-Native services?

The first whitepaper produced by this working group summarized Cloud-Native technologies. To follow-up, this whitepaper aims at answering the previous questions and exploring to what extent Cloud-Native technologies are being used based on the experience and feedback of 5G-PPP projects, many of which have verticals contributing to their outcomes.

**What technologies do verticals use today?** The Software Network Group has conducted a survey to collect technical input from the different participating projects about their use cases, the virtualization technologies they use, and the architecture patterns they follow. The method consists of finding commonalities between the verticals/use cases and then cluster the verticals accordingly. This helps to position the verticals with respect to Cloud-Native adoption. Each participating project in the WG described a vertical use case experimented in the project. The use cases provide information about their deployment scenario, technologies used, and any useful information. The use-cases are grouped per industry to find similarities or differences per industry. Once this was completed, a technological analysis was done to cluster verticals according to the technologies they use in the implementation of their use cases. Then vertical partners would be involved, sometimes in a follow-up survey, to validate technologies and patterns choices. This led us to identify challenges or impediments, if any, that need to be overcome to bring them closer to become Cloud-Native. Finally, we propose guidance on the Cloud-Native journey.

The remaining part of this paper is organized as follows:

1. Section 2 provides a reminder on the definition and technologies used in the Cloud-Native ecosystem.
2. In Section 3, the use cases are described in details and the key technologies of each use case are identified.
3. In section 4, the analysis of the input from section 3 is conducted and vertical clusters are identified.
4. Section 5 discusses some of the challenges and barriers faced by verticals that prevent them from being Cloud-Native compliant.
5. Conclusions are summarized in Section 6.

# 2 Cloud-Native in a nutshell

## 2.1 What it means

"Cloud-Native" is the name of an approach to designing, building and running applications/virtual functions fully exploiting the cloud delivery model. Cloud-Native approach is the way applications are created and deployed, not where they are executed. Cloud-Native apps are developed with tools that allow them to take full advantage of cloud benefits, meaning they can be built and changed more quickly, are more agile and scalable, are more easily connected with other apps. New operational tools and services like continuous integration, container engines and orchestrators are pillars of this transformation. The overall objective is to improve speed, scalability and finally margin as exposed in the Blog of Anne Curie (https://container-solutions.com/what-is-cloud-native/) and presented below[4]:

- **Speed**: companies of all sizes now see strategic advantage in being able to move quickly and get ideas to market fast. By this we mean moving from months to get an idea into production to days or even hours. Part of achieving this is a cultural shift within a business, transitioning from big bang projects to more incremental improvements. Part of it is about managing risk. At its best a Cloud-Native approach is about de-risking as well as accelerating change, thus allowing companies to delegate more aggressively and thus become more responsive.
- **Scale**: as businesses grow, it becomes strategically necessary to support more users, in more locations, with a broader range of devices, while maintaining responsiveness, managing costs, and not falling over.
- **Margin**: in the new world of infrastructure-as-a-service, a strategic goal may be to pay for additional resources only as they're needed – as new customers come online. Spending moves from up-front CAPEX (buying new machines in anticipation of success) to OPEX (paying for additional servers on-demand). But this is not all. Just because machines can be bought just in time does not that they're being used efficiently. Another stage in Cloud-Native is usually to spend less on hosting.

At its heart, a Cloud-Native strategy is about reducing technical risk. In the past, our standard approach to avoiding danger was to move slowly and carefully. The Cloud-Native approach is about moving quickly but taking small, reversible and low-risk steps. This can be extremely powerful, but it isn't free, and it isn't easy. It's a huge philosophical and cultural shift as well as a technical challenge.

## 2.2 Technology for Cloud-Native

Cloud-Native Computing Foundation's Cloud-Native Landscape project suggests a roadmap for the Cloud-Native journey [40]. This roadmap encompasses technologies, tools, and patterns that help implementing Cloud-Native development and their deployment environment, and categorizes them:

1. **Containerization**: Containers are a lightweight virtualization alternative to VMs [41]. They leverage two Linux kernel features: namespace for enabling isolation of an application by limiting its view of the Operating System environment like process trees, networking resources (e.g. interfaces, IP addresses, routing table, etc.), and cgroups for limiting and prioritizing system resources like CPU, memory, I/O, etc. While VMs package an entire

[4] The text described is extracted from the Blog of Anne Curie (https://container-solutions.com/what-is-cloud-native/)

operating system along with the function/application, containers only package the application with its application-specific OS dependencies because running containers rely on the host kernel. Containers provides many advantages over VMs [62]. Docker is one of the most widely used container engines. A light weight alternative to Docker is offered by CNCF CRI-O [45]. CRI-O is an implementation of the Kubernetes CRI (Container Runtime Interface) to enable using OCI (Open Container Initiative) compatible runtimes. It is a lightweight alternative to using Docker as the runtime for Kubernetes. It allows Kubernetes to use any OCI-compliant runtime as the container runtime for running pods. Today it supports runc and Kata Containers as the container run times but any OCI-conformant runtime can be plugged in principle.

2. **CI/CD**: Continuous Integration/Continuous Delivery is a pipeline process that automates the steps involved between compiling a source code and having it deployed in production environment [42]. CI is a practice that encourages developers to integrate their code into a main branch of a shared repository early and often. CD is an extension of continuous integration. It focuses on automating the software delivery process so that teams can easily and confidently deploy their code to production at any time. Therefore, CI/CD can take the code pushed into a repository, compile it, pass it through a test suite, and if tests were successful build a container image and deliver it/deploy it appropriately. Jenkins is one of famous CI/CD tools [43].
3. **Orchestration**: Orchestration is involved when things need to take place in a certain order. (micro)service orchestration refers to the coordination of multiple services through a centralized mediator such as a service consumer or an integration hub. Kubernetes [44] is widely deployed at scale and used both in testing and production environment.
4. **Observability & Analysis**: Monitoring, logging and tracing are three crucial requirements of any service deployment in general, and particularly for VNFs. They enable observability and allow the analysis of a service state. A whitepaper [46] identified these three features among others as required for telco-grade PaaS support.
5. **Service Mesh & Discovery**: Complex services are made up of a high number of microservices. At a large scale, manually configuring and providing infrastructure services (e.g. routing, security policy, load balancing, etc.) to these microservices is practically not possible. In [63] it is explained how Service Mesh provides a configurable infrastructure layer integrated from within the compute cluster itself through APIs that doesn't require any additional appliances [47]. Linkerd [48], Envoy [49], and Istio [50] are commonly used for mesh architecture. Another feature that is needed when a high number of microservices need to communicate is service discover. This is similar to the DNS system in concept but is designed to cope with the dynamic nature of microservices. Services are registered in a service registry, that is a database of available services, and a service's client queries the registry to determine the service's location. Examples of a service discovery engine are CoreDNS [51], Consul [52], or etcd [53].
6. **Networking**: Traditional IP routing-like networking cannot cope with the requirements of Cloud-Native systems where instances of (micro)services are created, moved, and stopped very quickly. There are currently two networking models which provides specifications for container networking solutions: Container Network Model (CNM) and Container Network Interface (CNI). CNM is developed and being only used by Docker, while CNI is becoming the de facto standard of networking and is extensively used by other container management systems, particularly Kubernetes. Some implementations of CNI are Calico [55], Flannel, and WeaveNet.
7. **Distributed Database**: The cloud is distributed by nature. Data Centers and clusters span multiple geographical locations. A distributed database provides scalability, resiliency, and performance required for this kind of environment, and which cannot be found in centralized database. Cassandra is a well-known distributed NoSQL database. CNCF suggests *Vitess* for MySQL database.

8. **Messaging:** By design, microservices are not deployed in isolation. They communicate and exchange message with each other very often. For Web application, REST is usually used and provides the required performance. However, this is not always the case, especially for telco services and VNFs. gRPC is a high-performance Remote Procedure Call system that was initially developed by Google and is now open-source. Kafka framework gains now more and more momentum.
9. **Container Runtime:** This is the ecosystem that enables building images and running them. There are many container runtime engines with different level of maturity and features. Linux Container LXC comes built-in in the Linux kernel, but it is not supported by orchestration engines. Other feature-rich container runtimes include docker, rkt and containerd.
10. **Service based architecture:** As a novel design pattern the SBA brings a service-oriented architectural concept in the mobile systems. Previous cellular architectures followed the classical function-reference point and node-interface model. Such models are proven inflexible: an addition of a new service requires specifications of many new interfaces to other nodes. Instead, in SBA services are accessible over a common message exchange interface. A specification of a new service only requires the specification of its own interface. Similarly, an addition of an NF to a system only requires the NF to be available over the common message interface. The SBA supports more flexible designs and more flexible service instantiations.
11. **Software Distribution:** When installing any piece of software on any system, its origin must be authenticated, and its integrity must be verified to avoid any compromise of the system by malware or other means. Therefore, the software provider must use and provide tools that enable the users to verify its identity and be sure that the software has not been modified by someone else. Notary is client-server-based system that relies on TLS and digital signature for secure communication and verification of the software's integrity.

# 3 Vertical use-cases: case of study

## 3.1 Introduction

5G is providing new experiences, such as AR and VR offerings, to our traditional consumer customers. But our mobile cellular industry is also pushing into multiple new verticals with distinctive service categories, like future factories, eHealth, automotive, Mission Critical, Immersive Media etc, with the expectation of addressing these markets in the next three years or so.

So how will the telco industry that has essentially created and matured three mobile services over the past thirty years (voice, short messaging and internet connectivity), deal with this huge challenge of developing multiple markets in a few short years?

To meet the scalability, the flexibility and the performance to cost-effectively deliver 5G services, a Cloud-Native framework able to integrate all VNFs is required. This would enable operators to create, contract for, or require as a condition of use, a standardized "adapter" that would expose all control, parametric, and management APIs and data in a common way

With 4G, a Cloud-Native core architecture is becoming the preferred option, but with 5G a Cloud-Native design is a requirement. Only by redesigning the software architecture and core functions using Cloud-Native design principles and IT web-based development and methodologies, can CSPs gain the necessary agility to rapidly deliver new services and reduce their time-to-market.

In this section, we describe some selected vertical use-cases implemented in Phase 2 projects, namely: NGPaaS [1], 5G-MEDIA [2], 5G-Transformer [3], 5G-City [4], 5G-Picture [5],

5GTANGO [6], SAT5G [7], MATILDA [8], Slicenet [9], METRO-HAUL [10], NRG5 [80] and Phase 3 project 5GENESIS [11].

The objective here is to give a brief description of the selected use-case which regrouped per vertical area. For each use-case, we present a short description and the key technologies deployed to implement it.

For a detailed presentation and explanation, we invite the reader to visit the website of the different projects respectively where they have access directly to the public documentation and to the contact names.

## 3.2 Public Safety

### 3.2.1 Mission-Critical Push-to-Talk

#### 3.2.1.1 Use-case description

This use-case has been developed in NGPaaS project [1]. MCPTT is a Land Mobile Radio solution that is deployed to provide communication infrastructure for emergency first responders. This use case consists of deploying a communication system on demand to enable intensive voice communication inside a group and between different groups of firemen arriving at a very large factory that has been set on fire.

Different scenario can be illustrated depending on where the RAN is hosted. The prototype shows the case where the Fire Truck has its own data centre with the capability to host VNFs. In this case, MCPTT Control plane VNFs are hosted in a central data centre and the other VNFs: U-plane as well as vRAN and MCPTT Apps VNFs, are hosted in the Fire Truck data centre.

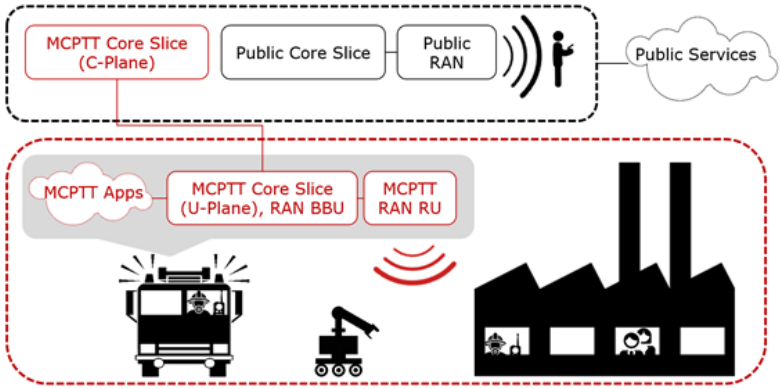


**Figure 1: Split MCPTT core and dedicated LMR RAN deployment example**

Figure 1 represents the use case, where the MCPTT Core components of the User Plane (UP) as well as the MCPTT application components and the MCPTT RAN components are positioned on a Fire Truck data center. Regarding the MCPTT Core control plane components (HSS, AMF, SMF), they are deployed on a distinct data center with a more central position as it should connect to multiple Firemen Truck data centers (this case is not represented in the Figure 1).

The elements/ Network Functions composing the service are:

- The elements from MCPTT Core Slice (Control plane):
  - HSS (Home Subscriber Server) playing as the legacy component of the AUSF (Authentication Server Function) and UDM (User Data Management) functions.
  - AMF (Access & Mobility Management Function).
  - SMF (Session Management Function).

- The elements from MCPTT Core slice (User plane)
  - UPF (User Plane Function)
- The elements from MCPTT RAN
  - eNB playing as the legacy component of the RAN 4G
- The elements from MCPTT Apps
  - Push to talk application customized for MCPTT use case

#### 3.2.1.2 Key technologies deployed

Considering the use case description, Figure 2 depicts the architecture of the service deployment. In the figure the PaaS refers to Kubernetes. All the functions are deployed as containers.

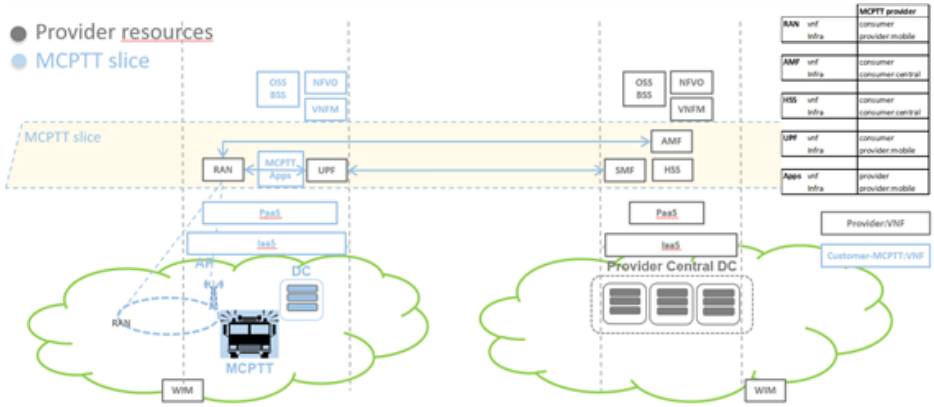


**Figure 2: MCPTT use case running on two Kubernetes platforms**

In this case, we deploy two Kubernetes clusters: one at a central/core cloud (could be OpenStack or public cloud) or and one at an edge cloud (Bare Metal). The MCPTT CP components are then deployed as containers on the core Kubernetes cluster, the MCPTT RAN is deployed as containers on the edge Kubernetes cluster. The latter one implements telco-grades features like CPU pinning, Network Service Mesh and NUMA to support RAN components latency requirements.

Before or during the rescue operation, the service provider will deploy on-demand different functions composing MCPTT service. Once deployed, the firemen can use their smartphones/hardened devices to connect and use the service to communicate and facilitate their rescue mission. The deployment of the different service components lasts less than 3 min.

### 3.2.2 Mission Critical Video

#### 3.2.2.1 Use-case description

This use case is implemented in one of the platforms created by 5GENESIS project located in the city of Málaga [75]. Its main objective is to create an infrastructure compliant with the 3GPP Rel.13 and Rel.14 including support of MCPTT and Mission Critical Services (MCS) such as MCVideo. These evolved mission critical services highly require eMBB and Low Latency communications features within the 5G ecosystem.

For one side, there will be a deployment of Mission Critical Services (MCS) that will allow smartphone public safety users to access professional mission critical communication responder's operation as it is depicted in Figure 4. The requirements need to be supported:

- Being able to allow the coordination of public safety teams demanding real-time communication in the field:
  - Smartphone users access secure, real-time private or group communications groups or/and exchange video with their teams and with a central command.

  - Central command assists tactical leaders in having a more complete view of the situation and thus making informed decisions.
  - Decision making is improved with additional information.
- Improving the mission-critical organizations' efficiency and effectiveness thanks to the MCS messaging features, for example by:
  - sharing a picture/video of a person or a suspect.
  - sharing a video of an accident or event.
  - receiving a map location from a field user
- Maintaining unbroken visual and spoken communication between team members during public safety operations.

For the other side, there will be a scenario with the same MCS communications being deployed behind a MEC server. That means these services will be provisioned and instantiated in circumstances where architecture with MEC capabilities is involved (e.g. trigger decisions based on overloaded network, latency requirements, sudden critical circumstances in a precise area, etc.). To this end, the use case will involve two different instances of VNFs: one heading non-MEC facilities (in this case located in the University of Málaga) and the other heading the MEC server (in this case located at Málaga City Centre).

### 3.2.2.2 Key technologies description

The use case will follow the architectural approach and component standardized and described in 3GPP Rel.13 and Rel. 14. This architecture comprises MCPTT/MCS application servers (ASs) – both controlling and participating - and MCPTT/MCS UEs deployed over an MCPTT/MCS-compliant SIP Core (e.g. IMS core) and a professional LTE access network that supports QoS-enabled unicast connectivity with PCC and native multicast support with eMBMS. The use case can be divided into two main logical divisions; the server-side and the client-side.

Within the server side, an MCS server will provide the application-level components required to deploy the main features in the 3GPP Rel.14 MCVideo. Other basic function and interfaces, such as MCS configuration and management servers, are also included in the solution to give support to the standardized solution, namely the Identity, Group, and Configuration management servers. Also, the solution includes the MCS specifications for user registration, authorization, affiliation, location configuration provisioning, and location reporting among other features.

The MCS system is deployed as a series of server components, each of them fulfilling a different functional role. All the server components may be deployed as standalone services in the host system, as an all-in-one Virtual Machine or a VNF-single or group of VNFs. Figure 3 contains the components and the target interfaces to be integrated with the LTE/5G side to be considered standardized mission-critical services.

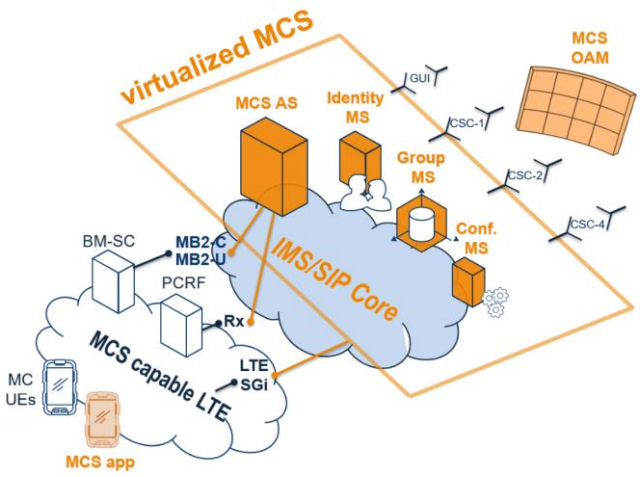


**Figure 3: MCS use case high-level architecture.**

Within the client side, the MCS is divided by two main components: the selected UEs (HW) and the MCS client App running on it (SW).

To perform this use case, the platform has created an infrastructure with 4 main sites: indoor and outdoor deployment within the University of Málaga, the city center of Málaga and Telefonica premises hosting mCORD-like installations to hold the use case in the edge environment. Within the use -case, the different VNF needed will be defined in the Network Service (NS) and deployed as a VMs by the OSM orchestrator. For more details, we invite the reader to go through this reference [75].

The final use case will provide two PoP, one close to the University of Málaga and the other one at Telefónica Edge premises. The normal workflow of first-responders will target the 5G Core at university premises and will only use the edge capabilities when the monitored crowded event or emergency requires to deploy so. Therefore, the workflow will dynamically switch a chunk of its load of entirely, targeting the newly instantiated edge MCS VNF at the edge to support lower latency and greater capacity in the field. Besides, considering the technical requirement that the MCS communications request in terms of QoS and service-oriented QCI selection in bearer setup, the infrastructure will automatically create mission-critical slices as well as to ensure the abovementioned technical requests. Moreover, the MCS paradigm does not only support single-level priority. Thus, the creation of slices will also cover the configuration of communication priority and pre-emption with the support of the end-to-end communication chain.

Finally, the use case also considers such situation where even the edge deployment is not enough to handle the load of a very crowded event with the whole police department working on the field and requesting MCVideo calls with demanding capacities. To this end, the orchestrator will monitor the service load and scale-up by creating/launching another MCS instance or by expanding the already deployed one.

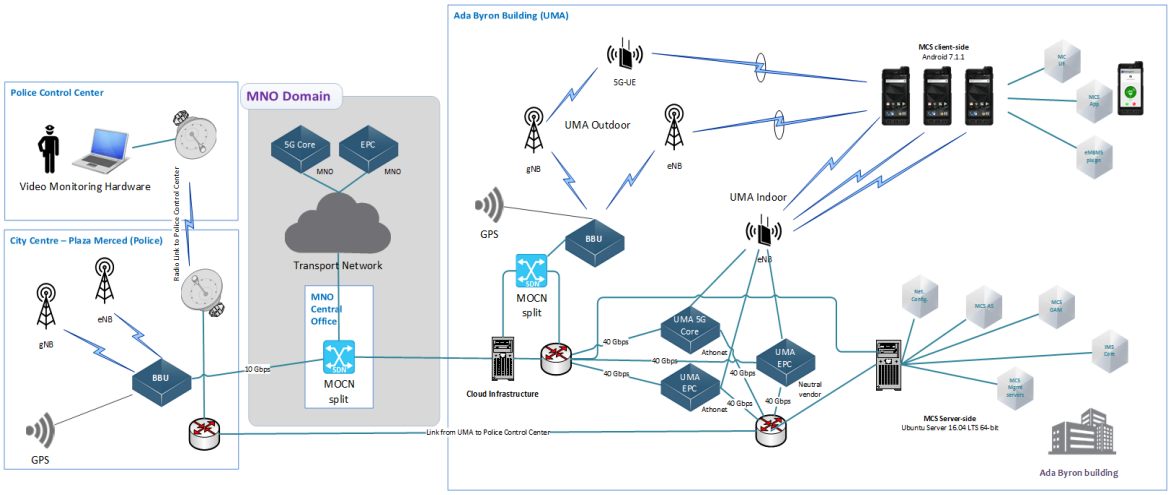


**Figure 4: Deployment Architecture of the MCS Use Case with its different domains and components.**

## 3.2.3 Video Security for Smart Cities: Real-time low-latency object tracking

### 3.2.3.1 Use-case description

The Metro-Haul technology will be showcased in a future smart city scenario, where security is provided by real-time object recognition and tracking. Simultaneous real-time access to the data of several fixed and mobile cameras allows tracking of objects and persons as well as the automatic recognition of critical events encompassing areas larger than a camera's field of view. Individual cameras provide tracking of anonymous objects and people; their combination with other data sources enables simultaneous recognition and identification. Thus, the system may also

be used for civil or commercial applications such as public transport optimization, traffic management and customer flow analysis. For real-time intervention relevant data could also be forwarded to mobile output devices such as smart phones or VR glasses.

The Metro-Haul approach leverages edge and cloud computing as it is shown in Figure 5. The use-case supports i) Real Time (RT) video surveillance with RT object tracking, automated face recognition and handover between multiple flexible camera devices, ii) Low-latency connectivity between fixed and mobile cameras assigned to different Access/Metro Edge Nodes (AMEN) and Metro/Core Edge Nodes (MCEN) with their associated DC storage and computing resources and ii) Enable flexible distributed cloud computing capabilities for real-time dynamic object tracking applications.

User community: Police and public security forces, crisis management and public infrastructure recovery after disaster events, public event management, public and private transport management, flow optimization of all involved traffic participants, smart city and traffic surveillance.

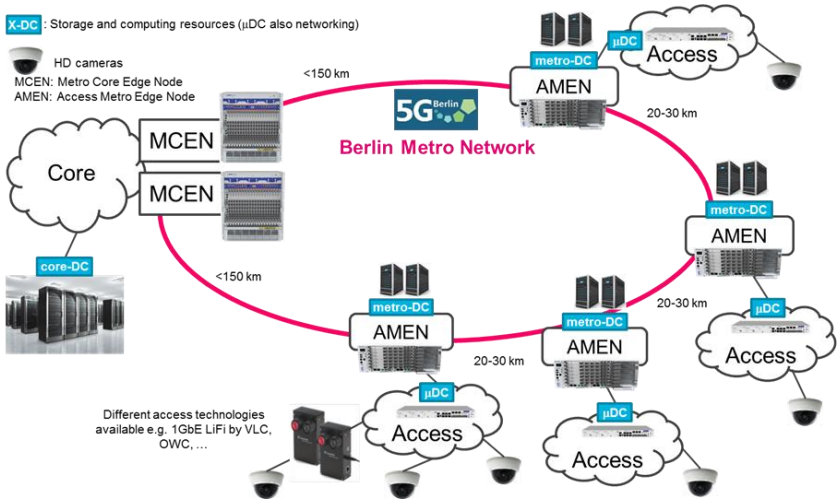


**Figure 5: Metro-Haul Real-time low-latency object tracking Use Case**

### 3.2.3.2 Key technologies deployed

The real-time low-latency object tracking use case within the Metro-Haul highlights a flexible deployment of an optical network slice instance, supporting a high-bandwidth low latency service profile. This is a public safety application running over a next-generation metro network. The service provides security and surveillance, including multi-camera target search, teleoperation, mobile device integration and deep video analytics for pattern recognition.

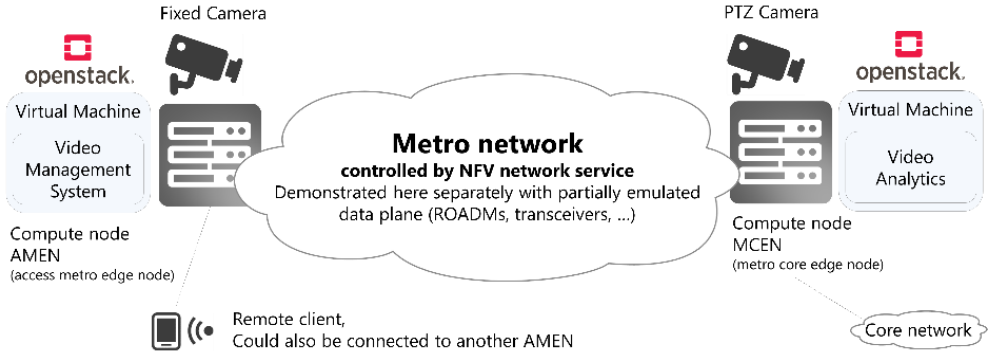


**Figure 6: Leveraging multi-layer network slicing for improving public safety**

The end-to-end security service is underpinned by the dynamic optical transport network composed of disaggregated elements, seeFigure 6. The optical transport elements include multi-

degree micro ROADM technology, managed using the open source OpenROADM YANG models and configurable via the IETF NETCONF protocol. The bit-rate speeds on the access ring range between 100G and 200G.

The MCEN Nodes include compute and storage capabilities orchestrated via OpenStack. The AMEN nodes are running the fixed and PTZ (Point Tilt Zoom) cameras, along with the camera device controller and video analytics intrustion and facial recognition.

The video signal is a Carrier Ethernet-Virtual-Circuit (EVC) orchestrated across the access and transport network, via the fully automated by the control-plane, extended with a Monitoring and Data Analytics (MDA) platform able to collect performance data from the devices that can be analysed using bespoke Machine Learning (ML) algorithms to proactively detect anomalies in the network.

The network performance measurements (telemetry data) are analysed by the Metro-Haul ML algorithms in the MDA agents; specifically, delay measurements are used as a metric that needs to be closely followed to detect degradations that might impact the security service. When degradation is detected a notification is sent towards the MDA controller and could be used, e.g., for triggering an in-depth analysis to localize the root cause of the failure, and also switching to a new path or requesting an entirely new connectivity service, thus ensuring service quality and resilience.

## 3.3 Media

### 3.3.1 Immersive Media

#### 3.3.1.1 Use-case description

Tele-Immersive (TI) applications are immersive media network-based applications that enable multi-party real-time interaction of users located in different parts of the globe, by placing them inside a shared virtual world. For each user participating in a TI application, her appearance is captured via multiple RGB-D sensors and a “3D replica” is created, by utilizing full-body 3D-reconstruction algorithms that produce the person’s digitized 3D Time Varying Mesh (TVM). TI applications produce a large volume of heterogeneous data, thus, creating a challenging networking scenario. Although TVM data can be compressed per-frame via state-of-the-art static mesh compression algorithms, these techniques lack optimal performance since they do not exploit correlations of the data over time. As of today, TVM compression algorithms that exploit the time dimension of TVMs are very few and these TVM compression schemes are not quite ready yet to support real-time applications. To cope with the large volume of data produced by a TI application, an upgrade to a more efficient network infrastructure along with the invention of new TVM compression schemes are required 5G networking technology appears now more than ever to be a necessity for real time TI applications. These have requirements for (i) very high bandwidth, (ii) ultra-low latency, (iii) ultra-high reliability and (iv) broadband access in highly populated areas. For the second part, novel, efficient, real-time, and potentially adaptive, TVM compression schemes need to be developed that in conjunction with the virtualisation technology offered by 5G networks will make TI applications a reality.

In this scenario, users are playing a Tele-Immersive game where their realistic appearance is reconstructed and embedded inside a common virtual game environment, see Figure 7. Inside the virtual world, the 3D replicas of the players are placed on top of sci-fi hover boards. The players can navigate and interact with the game environment by using body postures, i.e. bending on their knees to move forward and leaning left and right to turn accordingly. Each user-player is physically located in a TI capturing station that captures, processes, encodes and transmits the 3D appearance data to the other players. The body posture data used for navigation are also transmitted to an authoritative game server that is responsible for synchronizing the game state across players. Each player is participating in the common virtual game world via a game client.

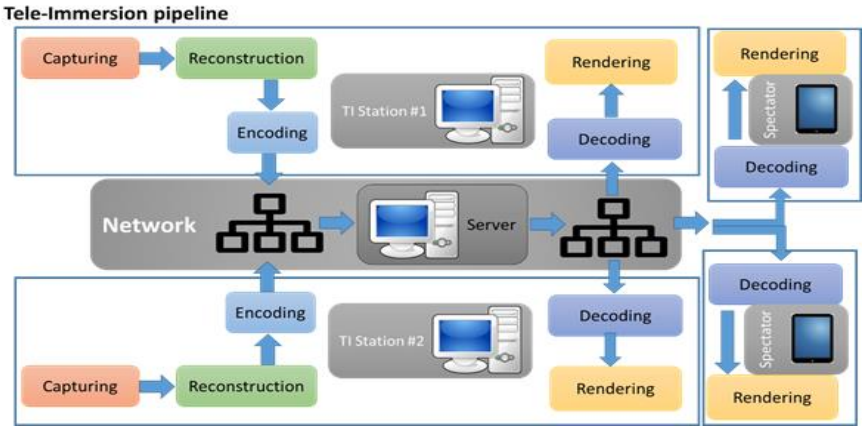


**Figure 7: Overview of the Immersive Media use case in 5G-MEDIA**

AR/VR immersive media use cases are also a part of 5GTANGO where it is demonstrating how 5G networks will enhance the experience of end users by improving their impressiveness into multiple 360º and non-360º video streams and the integration of their social media channels.

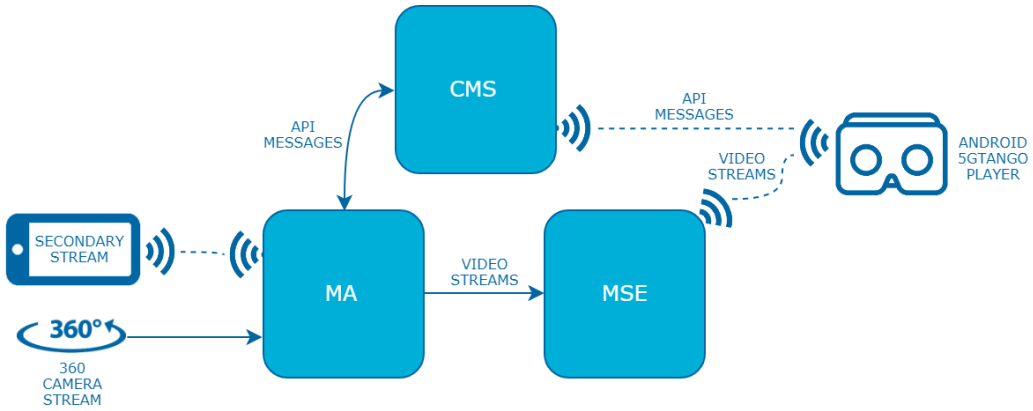


**Figure 8: 5GTANGO Immersive media pilot network service overview**

The 5GTANGO immersive media pilot service is composed of three main VNFs, as shown in Figure 8.

- The Content Management System (CMS) is the entry point of the service. The VNF gives the possibility to register new cameras, connect new Media Streaming Engines and provides the list of the available videos to the final users through a RESTful API. The CMS has a web interface to manage the users, cameras and statistics about the service.
- The Media Aggregator (MA) receives the RTMP video streams from the cameras, and also redirects those videos to the different Media Streaming Engines or to another Media Aggregator in another network node when needed. The MA has a RESTful API to interact with the CMS and the monitoring system.
- Media Streaming Engine (MSE) receives the RTMP video flows from the different Media Aggregators and implements the adaptive streaming algorithm. On a first step, the input video is transcoded to the different qualities and then an HLS playlist is generated with the different video segments. The MSE contains also a simple RESTful API to generate the VNF statistics and to send them to the monitoring system. This VNF is deployed on the edge of the network to bring the heavy transcoding task near to the end-user.

### 3.3.1.2 Key technologies description

Media specific VNFs involved in 5G-MEDIA Immersive Media use case are vTranscoder, vBuffer (buffering a few seconds of media for replay as required), and vReplay (a replay clip generating VNF that uses media buffered by vBuffer). This use case demonstrates a network

aware media application development through the deployment of media specific VNFs and showcases the potential of the FaaS development model where VNF instances comprising a session will be instantiated based on the trigger/rule logic that increases modularity and overall offers more cost-efficient operation. vTranscoder and vBuffer VNFs will be instantiated at the beginning of a gaming session. A vReplay VNF will be instantiated on-demand in response to the in-app events of interest (e.g., a user scores a hit against the opponent) and produce a clip that will be stored in a database accessible to the spectators for viewing at their discretion. Likewise, spectators' transcoders can be instantiated on demand for the duration of a short game bout, as well as the transcoder and buffer VNFs of the players. This flexibility allows to slash operational costs for the game providers hosting it on top of the 5G-MEDIA platform.

The vReplay VNF is well suited to a Cloud-Native FaaS approach. The game events stream from which replay clips are being created are highly dynamic. Thus, it does not make sense economically to allocate permanent resources of the platform to execute vReplay continuously. Rather, vReplay should be instantaneously elastic, highly reactive, and event driven. This will allow the game provider to save on operational costs considerably and therefore compete more efficiently while preserving the required QoS to the gamers.

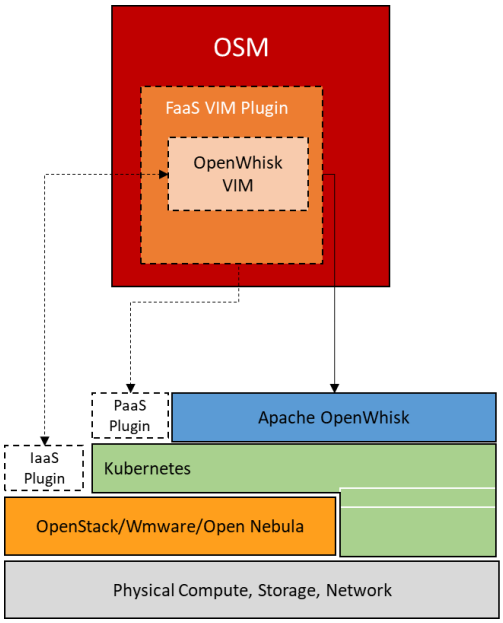


**Figure 9: FaaS VIM**

The game sessions are expected to be short-lived bouts. However, depending on the gamers' habits and preferences, some gamers can play just a few occasional bouts, while others can be expected to binge on the game, playing multiple bouts one after another for a relatively long duration. The 5G-MEDIA platform will help the game providers to learn the gamers' behavior and use optimization algorithms to decide on whether a new game session should be started using FaaS VNFs version (including vTranscoder and vBuffer) or a regular VNF should be used. For each resource and for each billing plan, a break-even point can be identified and based on a predicted time utilization of the resources, a dynamic decision can be made on whether FaaS or regular VNFs should be used for which session with the overall goal to save the cost to the game provider a well as to the platform provider.

In Figure 9, the reference implementation (i.e., the software architecture) of the FaaS VIM in 5G-MEDIA is depicted. OSM is used as VNF MANO Resource Orchestrator (Release 5 was used for actual implementation as we go to press). A FaaS VIM plugin is implemented with Apache OpenWhisk [65] being used as the FaaS framework. It was required to extend OpenWhisk with

some capabilities that were not available out of the box. Support for GPU, networking, and Day 1 configuration have been added to support the use cases at hand.

Two different architectures, based on the SONATA Service Platform, are proposed in 5GTANGO for VR/AR:

- *Kubernetes based*: Figure 10 shows the Kubernetes deployment of the network service using SONATA Service Platform and Kubernetes as a single VIM. In this deployment, the three VNFs are deployed as containers and interconnected using a weave network.

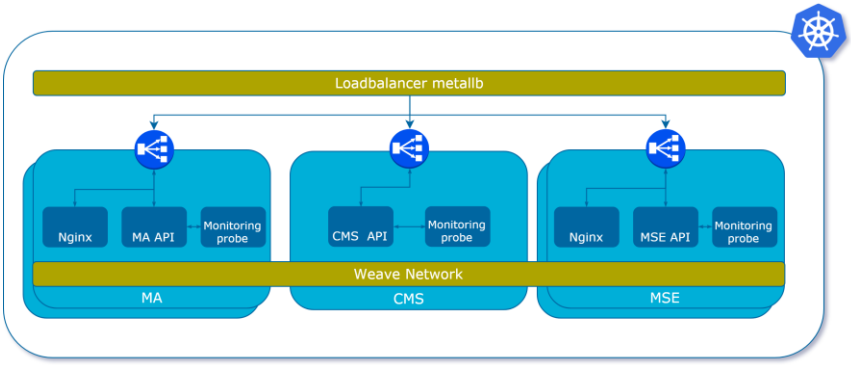


**Figure 10: 5GTANGO Immersive media deployment on Kubernetes**

- *Hybrid deployment*: Figure 11 shows this hybrid deployment, which uses SONATA Service Platform to deploy the network service on different VIMs. CMS is deployed as a Virtual Machine in OpenStack, while MA and MSE are deployed as Kubernetes containers.

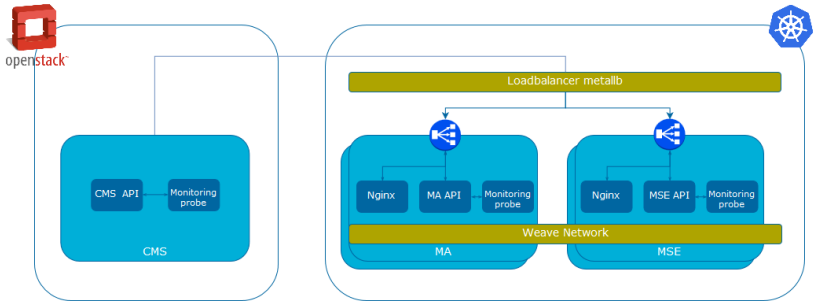


**Figure 11: 5GTANGO Immersive media hybrid deployment**

## 3.3.2 Remote and Smart Production in Broadcasting

### 3.3.2.1 Use-case description

Due to the steadily rising cost pressure, broadcasters are looking for new, low-cost and time-saving production methods, which include participatory and user-generated media archives in the production. These production methods are known under the term smart production. Remote production is a subgroup of smart production where the on-site production is handled remotely from the broadcaster's facilities.

Figure 12 shows an overview of the scenario covered by this use-case. Broadcast productions usually need large teams and long preparation times for the placement and adjustment of audio and video equipment for outside broadcasts. Another time-consuming part is the set-up and facilitation of a control room for the audio- and video-engineers as well as the directing team. To cut down complexity and costs, more and more productions take place remotely. In a remote production, the control room is on a fixed location, usually in the facility of the broadcaster. The control of the equipment on the venue itself happens remotely from this room. But establishing

these remote links with the needed performance often requires dedicated connections and, today, this is only feasible if the productions recur regularly. VNFs implementing Compression Engines have the potential to replace dedicated encoder hardware and optimisation algorithms based on QoS monitoring can help to overcome the current internet best-effort principle and ensure the required performance needs are met.

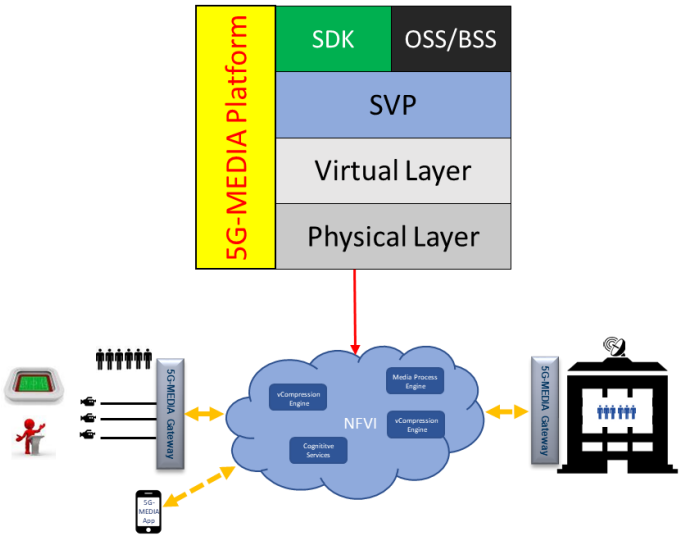


**Figure 12: Overview of the Remote and Smart Production use case in 5G-MEDIA**

Another smart production area is dealing with the needs of reporters and journalists in the field, who need to get high quality content back into the studio according to the high broadcasting requirements. Today interviews on the street are often recorded and then, only later, processed and edited in the studio. Often highly relevant and live content, e.g. for breaking news, has poor or unreliable quality yet it is often included in broadcasts as it is the only available footage of an important or newsworthy event. 5G-MEDIA has set out to improve this by ensuring that the reporter's audio-visual material gets delivered in high quality while also providing this as a reliable tool where now still dedicated lines and equipment are used.

5G-MEDIA addresses these limitations by leveraging new options for more flexible, ad-hoc and cost-effective production workflows by replacing dedicated lines and hardware equipment with software functions (VNFs) facilitating (semi-) automated smart production in remote locations. Virtualized and flexible media services reduce complexity for the user and ensure operational reliability (QoS, QoE). The network used for the use case is part of the network of an operator, where the virtual functions will be instantiated to provide the functionalities presented, closer to the required content production source.

This use case is about the contribution of content in broadcast production quality from remote locations to a broadcasting studio. It therefore requires high bandwidth streams between the venue and the broadcaster for both fixed camera/studio locations and for mobile-generated content. Low-latency paths will be required as well for signaling traffic between the broadcaster and venue locations for control purposes. vCompression Engines use smart machine learning algorithms to optimize the compression and encoding of audio-visual content to comply with dynamically-changing available bandwidth with the goal of meeting broadcast quality standards.

### 3.3.2.2 Key technologies description

The 5G-MEDIA platform offers open source FaaS technology, such as OpenWhisk as an integral part of the cloud services. FaaS will be deployed for scenarios that encompass use cases that happen spontaneously and require immediate setup of elastic virtual infrastructure. In these scenarios, it is expected that the video stream will have short-term and varying quality requirements. Potentially, a large quantity of short clips produced one after another by the mobile device of a journalist in the field will need to be handled efficiently. FaaS is applied at the edge

to automatically select acceptable quality of the clips and, e.g., extract faces from the video stream on a frame by frame basis from the video segments. The extracted faces are being transferred to the central cloud for further face recognition against a large database of celebrities maintained in the central location, tagging of the celebrity images and automatically preparing additional background information on the celebrity for use by the director. This approach saves bandwidth to the central cloud and eliminates resources waste by only using the virtual infrastructure that is required when it is required. Being intrinsically more lightweight than VM based approach, a container based FaaS technology, such as OW will be much more reactive, which is a critical feature in this scenario.

FaaS allows actions to be defined and then organised into sequences (or more complex topologies, called combiners) and create rules that define "if-this-then-that" type of policies. These features will be used to orchestrate different virtual functions and other parts of 5G-MEDIA platform in a genuine event-driven manner naturally suitable to the scenario. The cognitive services VNFs, Speech to Text Engine and Image/Face recognition Engine will be implemented as Docker actions in OpenWhisk.

## 3.3.3 Stadium/Mega Event Network Aware Applications

### 3.3.3.1 Use-case description

A mega event such as a football match can lead to a spike in users in a local area in and around the event space (e.g. stadium) as it is depicted in Figure 13. Due to cost factors and rarity of such events it is not possible for required resources to be provisioned permanently. This is especially true for the access network in the event area which can get congested very quickly (e.g. inside the stadium bowl) given media rich applications and availability of smartphones.

A problem like this can be solved by either deploying, dynamically, additional resources, especially physical resources at the edge (e.g. wireless access points, pico cells etc.) and provide additional connectivity to the Internet to allow networks applications path to their servers even during times of peak use. Or the network can be made smarter at recognizing traffic that should be prioritized and traffic that should be allowed to self-limit.

As an example, during events people like to make 'live' videos and share them on various social-media platforms. Event organisers like to provide a 'spectators' experience for publicity using video crowdsourced from the spectators. This is often done in real time during the event.

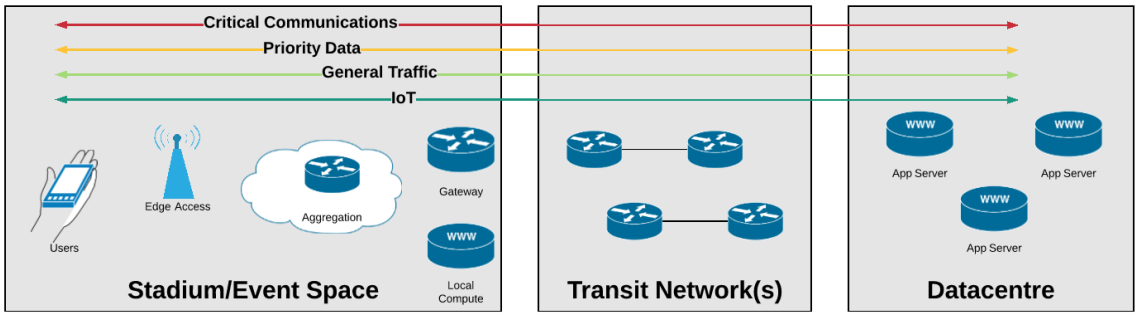


**Figure 13: Different components required for an event network**

For example. Watchity is an application that supports this use-case. But given the lack of resources to deal with peak traffic, the application users are not guaranteed performance. Furthermore, the videos may not have high quality content, may be similar (e.g. from the same location) or have objectionable content.

The above scenarios provide two use-cases for such an application:

- **Services mapped to slices** – running over a heterogenous network with different levels of priority (network aware of application) and involving different VNFs for controlling the application traffic including dropping traffic on request. The Crowdsourced video

application's traffic will be detected and prioritised over the wired and wireless domains that could include one or more 'transit' networks that should offer QoS-based connectivity. This will ensure application users get good user experience even as other services (e.g. general internet access) are de-prioritised.

- **Network aware applications** – Application is aware of APIs offered by network. In this case the network will allow the application admins to drop different video streams at the edge which will reduce the load on the wireless access network. This will be done by providing a programmable API from the controller which the Watchity Editor/Curator application will use to block 'low quality' (in terms of content and not video quality) videos. This highlights the feedback and programmable network concepts. This can be used to decongest the application slice and enforce fair-use policies.

### 3.3.3.2 Key technologies description

This section describes the technical aspects of the demonstration of Stadium/Mega Event Network aware applications for the 5G-PICTURE project. The demos will be carried out on-site at the Ashton Gate Stadium in Bristol, UK in March 2020.

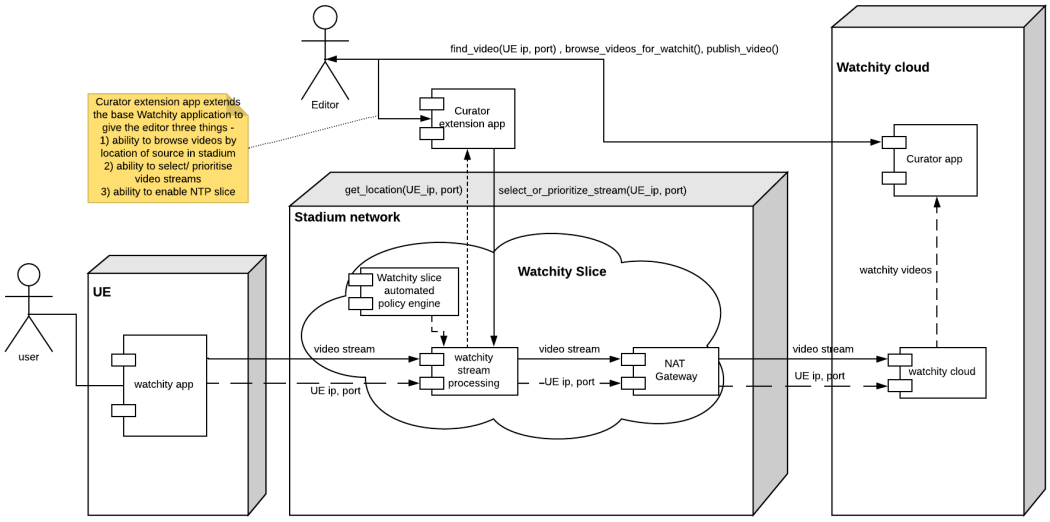


**Figure 14: Outline of the different components in the 5G-PICTURE Stadium Demonstration**

In the demo we will attempt to deploy all the components of a mega event network including, see Figure 14: Edge access, Edge compute, Transit network and Cloud compute.

Edge access includes WiFi access points and backhaul devices from i2CAT. Edge compute will be used to run the required software components to detect and curate traffic (VNFs). Zeetta Networks will provide the transit network which will provide QoS-based connectivity via COP interface which will be mapped to slice requests. Watchity application will provide cloud-based servers that will be the backend of the smartphone application.

Software components involved are:

1) Zeetta's NetOS Controller and Dynamic Slicing Engine (running in Docker, orchestrated by Kubernetes)
2) i2Cat Wifi Access Controller and Backhaul Orchestrator
3) Phishahang over Kubernetes for VNFs
4) Watchity Crowdsourced Video Application

## 3.3.4 Crowdsourced video broadcasts

### 3.3.4.1 Use-case description

Consider a sporting event in a crowded stadium. Often, spectators will film on their smartphones, later posting these videos to YouTube, Facebook, etc. These videos capture unique views potentially missed by professional videographers. Given a multitude of sources, such video content is difficult to browse and search. Crowdsourced live streaming (CLS) platforms allow users to broadcast their content to massive viewers, greatly expanding the content and user bases. CLS services enrich social interactions across communities by allowing users to share common interests and stories. If organizations hosting such events could easily capture all this video content in an interactive and highly shareable visual story, this would boost its social reach immensely. Services like Watchity and Periscope provide cloud based CLS systems to extract content-specific metadata from crowd sourced videos by forming "clusters" of synchronized streams with related content of the event. These are made available online so that event organizers can combine them to portray their brand or message. Synchronizing such streams and capturing them as a cloud service requires a high capacity network which adapts and scales dynamically to accommodate multiple synchronized streams.

For crowdsourced video broadcasts, a number of key capabilities are required, including: i) on-demand computing capability to transcode, aggregate and synchronize video feeds, ii) low latency network paths, from video source to aggregation point on the metro edges, to synchronize multiple video feeds as a cluster, iii) high capacity network paths to manage the scale of upload video feeds and finally, iv) capability to gather metadata from the video service to feed as inputs to End-to-End path setup. Once the crowdsourced video is ready for transmission, the quality and format of source stream transmission will require additional computational resource to transcode it into multiple industrial standard quality versions to serve viewers with distinct configurations.

### 3.3.4.2 Key technologies description

The Metro-Haul crowdsourced video broadcast use case demonstrates the need for a top-down orchestration layer, which then coordinates the end-to-end architecture comprising of edge computing resources, and virtualized network and service functions, a high-capacity reconfigurable and flexible optical metro network, and logical low-latency connections supporting a variety of bit-rates.

The initial service request for crowdsourced video broadcast is received via the Metro-Hual "Intent" northbound API, allowing a higher-layer application to request a service using basic descriptors ("what the application wants"). This service request is then considered by the orchestrator and translated ("how to deliver the application request") via the policy manager. The end-to-end resources are then provisioned to provide Cloud-based VNF-based service functions (instantiated via Open Source Mano – OSM), computing resources for transcoding, and optical connectivity based-on service bit-rate requirements via the Transport SDN Controller.

Control and provisioning of the optical nodes and transceivers solutions is achieved via an open south-bound API, and open optical transport YANG-based resource models, coordinated via the Transport SDN Controller. This allows the Metro-Haul orchestration-layer to manage a variety of open hardware and vendor specific optical transport equipment.

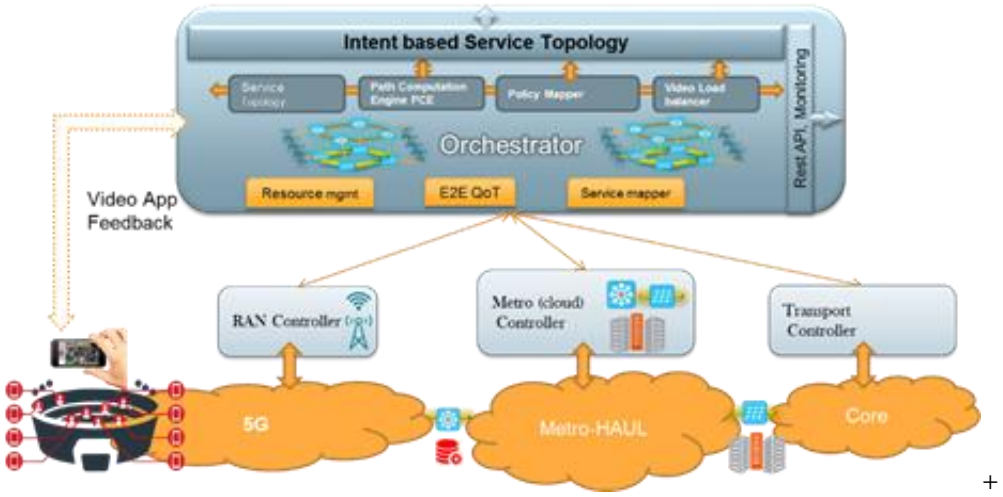


**Figure 15: Metro-Haul Crowd-sourced video Use Case**

Figure 15 highlights the top-down orchestration layer, functional components, and overall end-to-end architecture for the crowdsourced video broadcast use case.

## 3.3.5 Entertainment use-case

### 3.3.5.1 Use-case description

This use case [28] provides fans attending a sports event with ultra-high-fidelity media content to allow them an immersive experience. The service includes functions for video recording, enhancing, storing, and distributing with multiple caches, see Figure 16. These functions may be considered as user-plane functions. In addition, there are functions for controlling video storage and caching and for providing access control to the local video distribution.

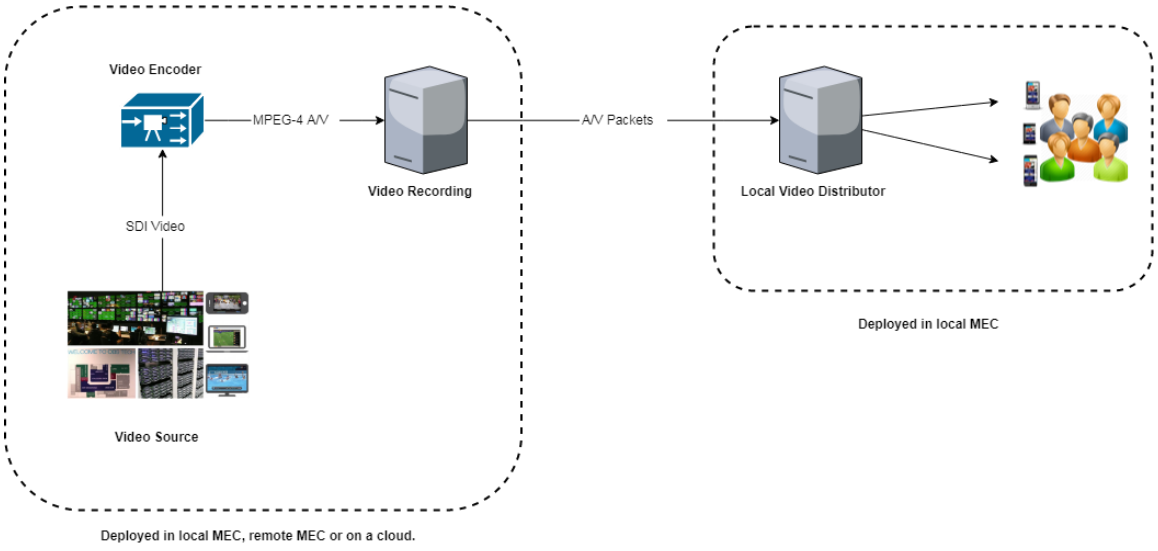


**Figure 16: 5G-TRANSFORMER Entertainment Use Case**

A single service instance may be deployed to multiple PoPs, especially a PoP local to the sports event and a PoP of a cloud provider for extended capacity.

### 3.3.5.2 Key technologies description

In the entertainment use case, the 5G-TRANSFORMER system is deployed using Kubernetes to control compute resources. To benefit of the more light-weight deployment and the more agile

development with microservices, containers are used as virtualization technique. The 5G-TRANSFORMER system [28] consists of three major components, the vertical slicer (5GT-VS) to define services and control their instances, the service orchestrator (5GT-SO), and the mobile communication and transport platform (5GT-MTP) to manage resources in different technical domains such as radio, compute, and transport. Inside the 5GT-SO, a service manager (SM) has the end-to-end view of a service instance and is also responsible for connecting multiple PoPs and wide-area networks (WAN) if used for a service instance. Different MANO platforms can be integrated to the 5GT-SO, the one used for this use case is cloudify [26]. Whereas the SM is agnostic whether a function is deployed as VM or as a container, the MANO platform actually has to be aware of this difference and would call different plugins towards the 5GT-MTP.

Different technological domains in the 5GT-MTP would allow to instantiate services where some functions are deployed as VMs and other ones as containers, even if such a mixed deployment is not used in this specific use case. Here, Kubernetes is used as lower level orchestration engine, see Figure 17.

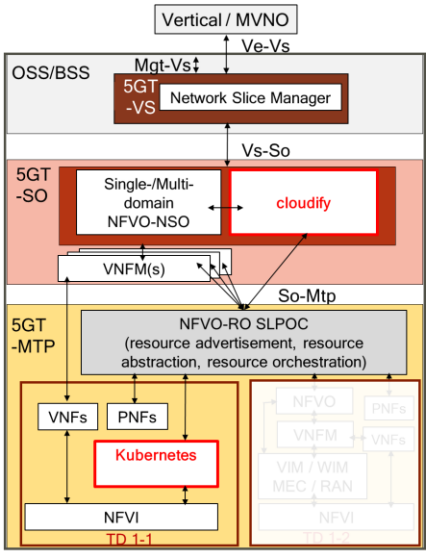


**Figure 17: 5G-TRANSFORMER MANO over Kubernetes**

In this use case, cloudify is used as resource orchestrator in the 5GT-SO. Note, the selection of cloudify or OSM in the 5GT-SO and Openstack or Kubernetes in the 5GT-MTP are independent of each other. The single logical point of contact (SLPOC) component in the 5GT-MTP provides a common abstraction to the resource orchestrator in the 5GT-SO for both Kubernetes and Openstack. This common abstraction is based on ETSI NFV IFA005 [20].

In summary, end-to-end service orchestration in 5G-TRANSFORMER is independent of using VMs, containers, or both. Lower level orchestrators actually deploy the functions with the matching virtualization approach.

## 3.3.6 Mobile Backpack Unit for Real-Time Transmission

### 3.3.6.1 Use-case description

Almost all TV stations are using what in popular language is called “backpack unit” for video transmission in remote areas. The backpack unit bundles multiple 4G connections together to transmit the video signal back to the TV station for further processing. However, this technology presents a considerable set of challenges. More precisely:

- It is impossible to use in crowded events, due to the spectrum congestion.
- It is impossible to use in bandwidth-restricted areas.

- It usually requires long time for communication establishment.
- The quality of the transmission fluctuates due to unmanaged bandwidth changes

Currently TV broadcasters are using satellite technology in such situations, with all the constrains that this technology has, namely very low flexibility (the satellite segments must be booked in advance) and the high cost of the satellite time. The rollout of this Use Case within the 5GCity system will address such challenges by:

- Enabling the increase of bandwidth used for live connections
- Provisioning specific slices with a guaranteed QoS
- Enabling edge computing processing capabilities for production of video contents at the edge.
- Enabling required video processing at the very edge instead of TV station's datacenter, thus reducing the high production cost of multi-camera events.

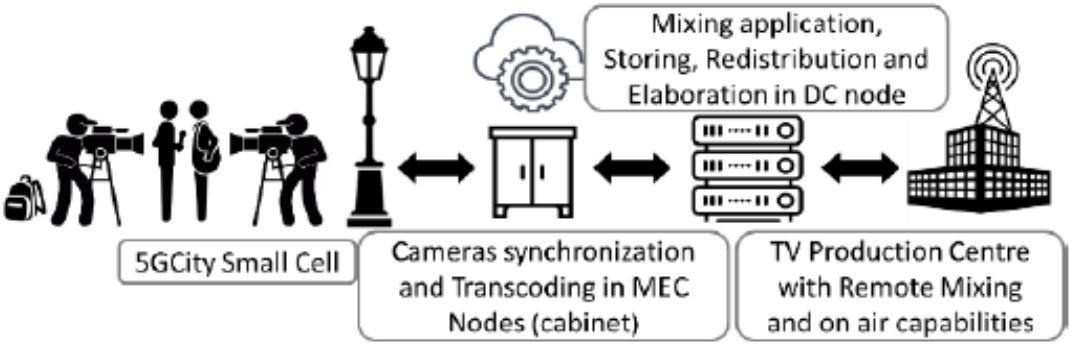


**Figure 18: Mobile backpack real-time transmission Use Case**

A summary of the Use Case as it will be trialed in real-life scenarios in the City of Barcelona by BTV is illustrated in Figure 18. Dedicated end-to-end slices across access, edge, and core, will be setup and adjusted on demand to tackle the fluctuation issues.

### 3.3.6.2 Key technologies description

To design and deploy this use-case, a software suite which allows the complete lifecycle management of end-to-end network slices and services dynamically created over heterogeneous city infrastructures is needed, including the ability to handle Cloud-Native technologies such as containers and fast/dynamic CI/CD for the maintenance of (softwarized) slices. The 5GCity platform is composed of the following main elements:

- 5GCity orchestrator is the core entity of the platform, which handles the complete life-cycle management of the network services, providing the suitable degree of abstraction between resources as exposed by infrastructure layer and services as intended by end-users.
- 5GCity dashboard is the main operational interface allowing 5GCity users (vertical or service providers) to deploy network services over an owned set of network slices.
- 5GCity Virtual Interface Managements (VIMs) are the entities which acts as intermediary between the orchestrator and the physical resources located at various levels (Core VIM for datacenter level, Edge VIM for network edge level, and Extended Edge for devices level).

- 5GCity NFVIs are thin layers deployed over the pool of hardware resources and acting as agent for the configuration of the physical resource management (on the three levels, similarly to the VIMs).
- SDN controllers are the functional entities which provide suitable control plane operations.

Figure 19 presents the 5GCity platform.

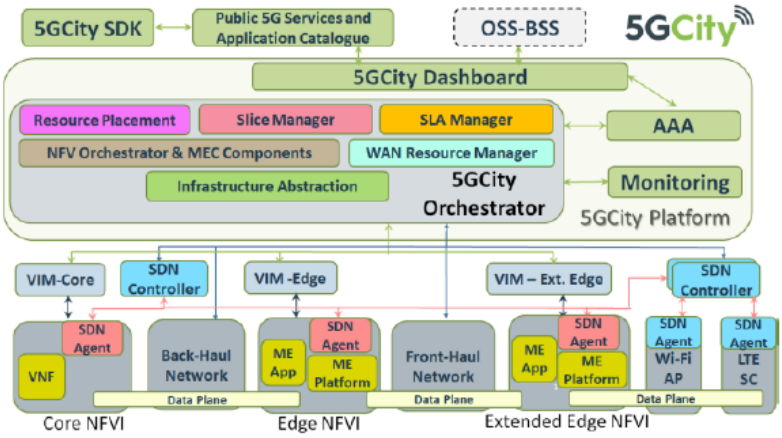


**Figure 19: The 5GCity platform used to design and deploy the Use Cases**

With the platform of Figure 19 in place it is possible to design the "Mobile Backpack Use Case" as a network service as shown in the upper part of Figure 20 and deploy it depending on the conditions and the requirements in a way similar to the one shown in the lower part of Figure 20. This design and deployment present the following main characteristics:

- With the video caching and the synchronization functions (cf. VM1) "containerized" and dynamically deployed to the edge, potentially by using the Cloud-Native parts/enablers of the 5GCity platform (e.g., the container-technology-supporting MEC components, implemented by using the open-source fog05 software), delay- and bandwidth-related aspects of the media transmission can be tackled.
- By using the same Slice Manager to individually chunk all resources assigned to this service (access resources, network link, Cloud and edge compute resources), and combine them as needed, slices that are fit to eliminate QoS fluctuation can be easily in place.

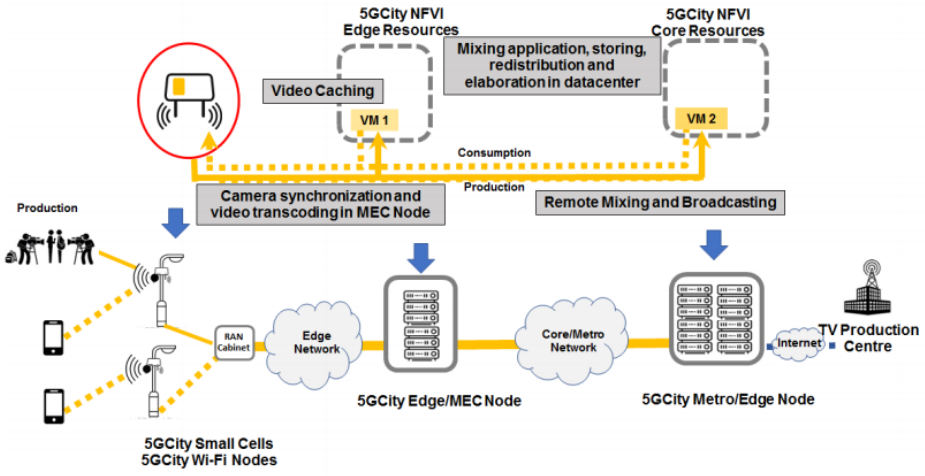


**Figure 20: Mobile Backpack use-case design and deployment**

# 3.4 Smart City

## 3.4.1 Smart City Intelligent Lighting

### 3.4.1.1 Use-case description

The intelligent lighting system is a smart city application that runs over a LoRaWAN network and enhances the lighting company in the city to easily manage and maintain all the lighting poles in a city, while achieving energy consumption reduction up to 70% when combined with full LED lamps. The application considered in this case is divided in three parts: lighting controllers, middleware platform and management interface. The lighting controllers are hardware parts based on LoRaWAN technology, which are installed on lighting poles assuring the capability of remotely controlling them. The middleware platform collects, stores and secures the data from the controllers to be used later through open APIs by any other application. The management interface is divided in two parts: administration of the lamps' schedules and the alert and ticketing system. The administration part gives administrators the potential to remote control and access lamps' data, but also to set a schedule on groups of lamps (turn on after sunset, dim to 70% after 1AM, turn off after sunrise, etc.). The alert and ticketing system provides administrators a way to inform in no time the repair team and keep track of every service required by all of the poles. Lighting controllers will be converted to be able to access the 5G network, potentially taking advantage of roaming functions, but also increasing the lifetime of the devices' battery and coverage in restricted areas.

The application graph of the Smart City Intelligent Lighting System is shown below (Figure 21) and it contains six components.

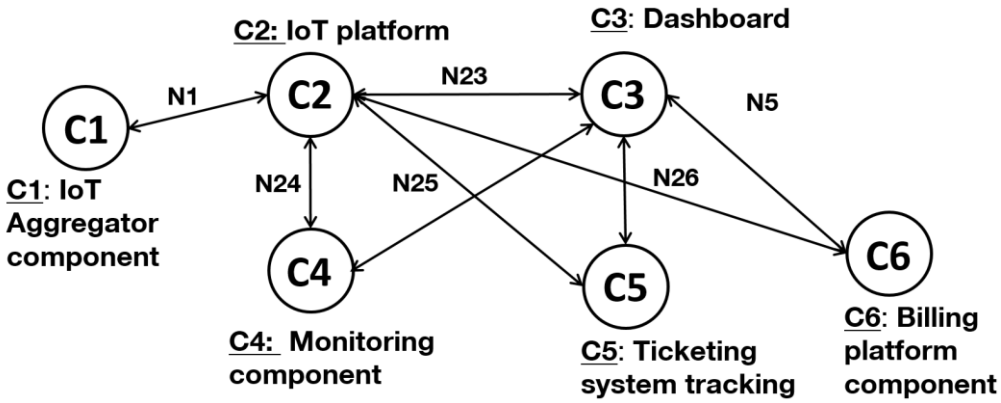


**Figure 21: Application Graph for Smart Lighting Use Case**

- **C1 (Component No. 1)**: known as IoT Aggregator component, is the part of infrastructure connected to the end devices (smart lighting lamps). The IoT aggregator stands a proxy between the smart lightning lamps and C2 (IoT platform). Therefore, it aggregates all the messages from the smart lighting lamps and passes them forward towards C2. The communication between C1 and C2 uses MQTT protocol enabled by python scripting.

- **C2 (Component No. 2):** is the main component of the use case and it is used to store and process the data (telemetry, alarms, monitoring, etc) and also to control devices (send different types of commands: on, off, dim). The IoT platform is an open-source version (community edition) of Thingsboard.IO platform that is connected to a PostreSQL Database or Apache Cassandra NoSQL Database. The platform has available several connectivity protocols and APIs, such as MQTT/MQTTs, CoAP, HTTP/HTTPS, SigFox,

LoRa (only with specific network servers). In this use case all integrations have been made using MQTT and HTTP protocols.

- **C3 (Component No. 3):** the component number 3, dashboard administration is a centralized front-end application with multiple functions that can offer to user/admin an overall view and general control of the entire system.

- **C4 (Component No. 4):** Monitoring system to check the status (including alarms) of the infrastructure and services.

- **C5 (Component No. 5):** ticketing system for tracking and resolving possible issues. Can offer information regarding the alarms/incidents in the smart lighting system (infrastructure or services), root cause and tenants involved.

- **C6 (Component No. 6)**: is the billing application that is connected to C2 (IoT platform) and collects all information about tenants and users (e.g. number of devices connected, number of telemetry messages, type of devices, etc) of the Smart Lighting System. Based on a commercial offer regarding number of devices connected and other additional services, a bill for every tenant of the platform shall be generated.

Three main layers are embedded in the application architecture: (1) the Development Environment and Marketplace to support all pre-deployment steps of a 5G-enabled application; (2) Vertical Application Orchestrator (VAO) layer supporting in-life slice intent and adaptation of the application to its service requirements by using the optimisation schemes and intelligent algorithms to provide the needed resources across the programmable infrastructure; (3) the Network virtualized infrastructure layer responsible for setup and management, deployment and operation of vertical application. The Smart City slicing architecture is created by developing an end-to-end operational service layer covering the design, development and orchestration and the network virtualized infrastructure layer.

As depicted in Figure 22 the integration between layers, first developed in MATILDA and the second one in SliceNet is performed through a One-Stop API (OSA) solution.

The role of the dedicated interface One-Stop API (OSA) is to permit control data transfer (requirements and answers) between Vertical Application Orchestrator and Telco Infrastructure. In MATILDA project this interface is called the Telco Provider Northbound API.

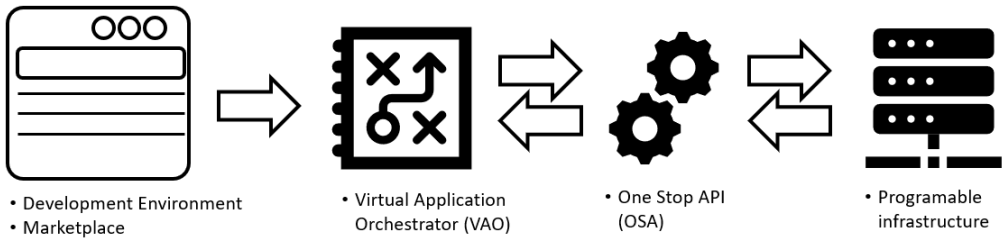


**Figure 22: Smart City application layers**

### 3.4.1.2 Key technologies description

The End-to-End architecture (Figure 23) is built based on the developed and integrated components from the two projects, MATILDA and SliceNet considering the following principles:

- the service design and service composition is defined within the Marketplace developed within MATILDA project

- the service is described by the Smart Lighting use case requirements through the Slice Intent Model from MATILDA project, with respect of the element components needed to be instantiated
- the communication with the resource network layer, as an adaptation point of communication between the projects, done through the One-Stop-API developed within SliceNet project
- the service is instantiated, configured and deployed over the test bed network infrastructure using virtualized components, as NFV/VNF implementation or fixed physical components as PNFs leveraging slicing capabilities.

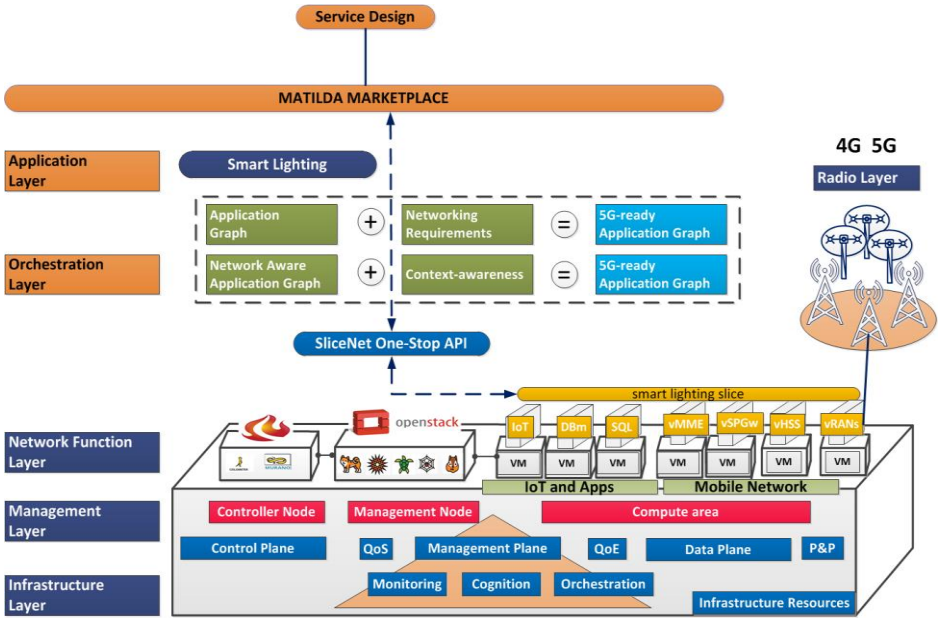


**Figure 23: 5GPPP MATILDA SLICENET Collaboration Infrastructure Application**

The principal components of the E2E smart lighting system are divided in two main sections: Infrastructure and Application.

### 3.4.1.2.1 Smart City Infrastructure components

The testbed infrastructure, used application, software and hardware components:

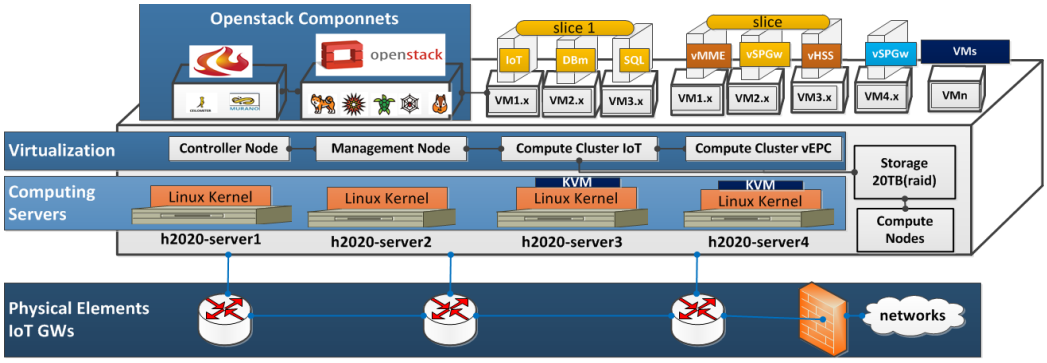


**Figure 24: Overall Testbed infrastructure**

a. Hardware components
   i. the entire suite of physical network elements (routers, switches, firewalls), hardware servers and storage, radio units
b. Software components for testbed virtualization instantiation
   i. Openstack framework
c. Software components for NFV/VNF instantiation

i. the entire OpenAirInterface (OAI) suite for a fully mobile access and core network for OAI-RAN, OAI-Core and HSS

d. Software components for Orchestration
   i. ETSI OpenMano or similar orchestrator and the on-boarded VNFs templates
e. Software component for monitoring: Ceilometer, Gnocchi, ElasticMon, FlexRAN, and E2EQOE which is a software tool developed by Orange Romania measuring and collecting E2E slice performance (RTT, Jitter and Packet Loss).

#### 3.4.1.2.2 Smart City Application components

Smart City application components (Figure 21) are described by several developed and integrated apps, as the MATILDA Marketplace, the application layer for 5G ready application graph, SliceNet One-Stop-API (OSA), the software component acting as the integration and adaptation point between the VAO and the testbed infrastructure and use case application components, described in the previous section.

All Smart City Use Case components are based on a virtualized testbed with containerization technologies that allows fast deployment and easy process of management of the entire system and applications life cycle.

As containerization technologies, Orange Romania has implemented the two next approaches:

1. Docker containers on VM: run Docker Engine over KVM hypervisor,
2. Docker containers on bare metal: run Docker Engine over single physical node,
3. Kubernetes installed in 2 VMs cluster (master and slave node): pods running over KVM hypervisor.

Every component represented in Figure 21 (Application Graph for Smart Lighting Use Case) is composed of one or more containers (subcomponent) that helps building entire system.

## 3.4.2 Unauthorized Waste Dumping Prevention

### 3.4.2.1 Use-case description

One of 5GCity'suse cases is focused on unauthorized waste dumping prevention. This use case is focused on monitoring, by means of video surveillance cameras, urban areas under the risk of environmental abuse because of unauthorized dumping of waste materials. Media content produced by these cameras is analyzed using machine-learning based image recognition to automatically detect illegal waste dumping. In case of illegal action detection, an automatic alarm is triggered by software hosted at these edge resources and an alarm sent to the nearest Municipal Police station containing a selection of media demonstrating the potential illegal action.


**Figure 25: Unauthorized waste dumping prevention Use Case**

The use case will be deployed in a real-life scenario in the City of Lucca nearby Lucca's ancient city walls inside the historic town, as shown in Figure 25. More concretely, street cabinets that can host sensors-related or access network-related equipment and edge services exist close to the monitored areas, while municipal data centers can host the core services. By deploying the current use case, Lucca expects the following impact:

- Lowering costs. The automatic video analytics application can significantly shorten the time dedicated to manually analyze video recording to identify relevant waste disposal abuse scenarios.
- Shortening the response time of Municipal Police after the abuse has been performed, to ease the identification of the culprit.
- Enhancing the urban environment.

### 3.4.2.2 Key technologies description

Equally to the previously presented "Mobile Backpack" media use-case, it takes a platform with Cloud-Native capabilities (such as the described 5GCity platform) in order to efficiently design and deploy the unauthorized waste dumping prevention Use Case in a way that leads to achieving the goals of the involved cities.

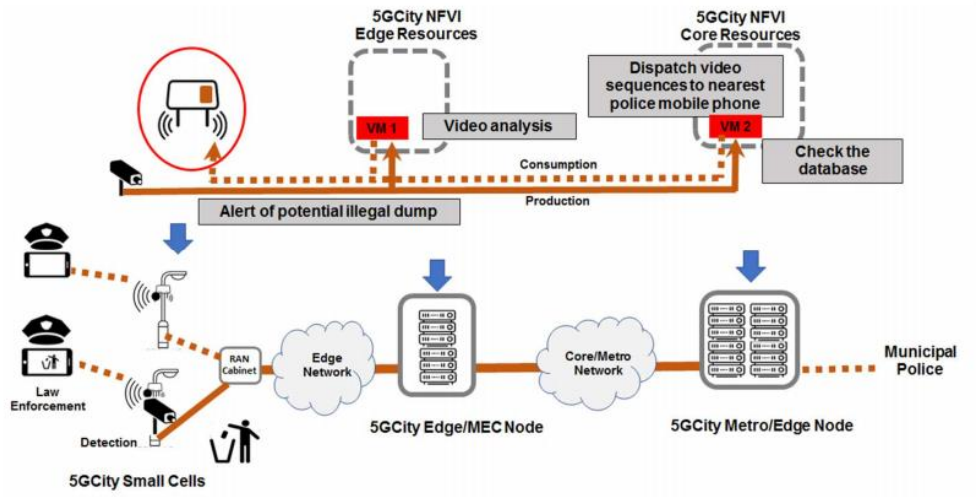


**Figure 26: Unauthorized waste dumping prevention Use Case design and deployment**

With the capabilities of the 5GCity platform in place (please refer to Figure 19 and the analysis of the "Mobile Backpack" Use Case for details), the unauthorized waste dumping prevention Use Case would be designed and deployed as shown in Figure 26:

- Usage of distributed database technologies can be used so that the ML logic of the Video Analysis (cf. VM1) might have fast and efficient access, when required, to partly the same data that police software would check against.
- The Video Analysis function, including machine-learning tasks, is either containerized or implemented as a Unikernel because of ML-related instantiation performance requirements, but still orchestrated in an NFV-compatible way in line with the rest of the system.

## 3.5 Automotive

### 3.5.1.1 Use-case description

This use case [28] implements a road safety application named EVS (Extended Virtual Sensing) designed to alert drivers about the presence of unseen vehicles or other unexpected obstacles at intersections. Vehicles are providing information on their speed and trajectory with cooperative awareness messages (CAM). These messages are used to update a cooperative information manager database (CIM). A car manufacturer specific algorithm analyses the data in the CIM and reports critical observations via decentralized environmental notification messages (DENM) back to the vehicles.

In this use case the CIM and the EVS algorithm are deployed as VNFs or MEC applications and they should be deployed at the mobile edge to reduce latencies. The interaction among the components is shown in Figure 27.

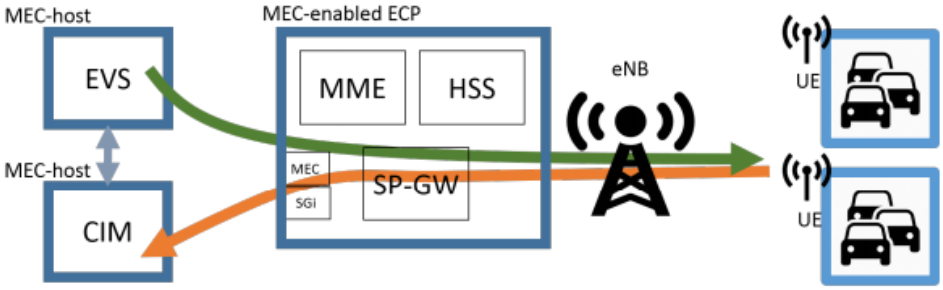


**Figure 27: Interaction of the EVS Application Building Blocks**

Additionally, in this use case videos are streamed to the passengers of vehicles. This is done to demonstrate the handling of services with different priorities in case of resource shortage. Also, the video streaming related functionalities are deployed as VNFs.

In case the EVS must be scaled out based on monitored CPU load, additional VNF instances are created. Each of the VNF instances of the EVS is responsible for a specific area. After the scaling, the VNF instances have to rearrange the areas they are responsible for. Especially the VNF instances existing before the scaling decision will reduce the area they are supervising, and the new instances will take over these areas. Note, this implies active coordination of instances in the context of scaling decisions.

### 3.5.1.2 Key technologies description

The 5G-TRANSFORMER architecture [28] allows both a MANO Centric deployment and a deployment using Kubernetes. Figure 28 shows the MANO-centric deployment of the 5G-TRANSFORMER system, which is used in the automotive use case.

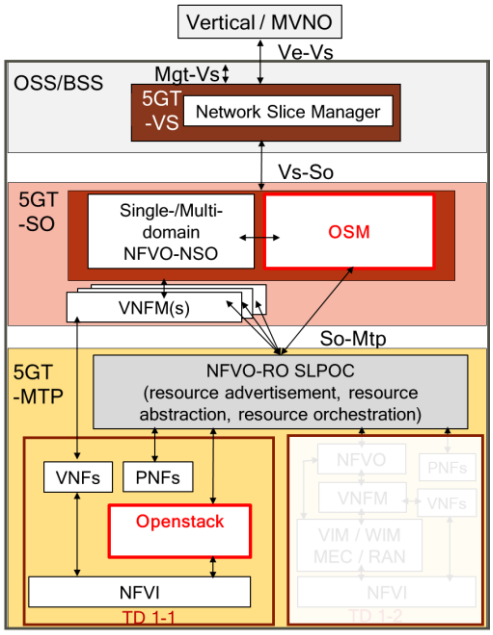


**Figure 28: 5G-TRANSFORMER MANO-centric**

In the 5G-TRANSFORMER service orchestrator (5GT-SO) there is a network service orchestrator (NFVO-NSO) and a resource orchestrator component. In this use case, OSM [27] is used as resource orchestrator. This controls the compute resources in the infrastructure via Openstack. I.e. the VNFs are deployed as virtual machines by Openstack, which is orchestrated by OSM.

# 3.6 Smart Grid

### 3.6.1.1 Use-case description

The use of drone in the Predictive Maintenance as a service uses the automatic mode pilot to make periodic predefined maintenance given a certain pre-defined path. In this periodic mission the drone performs: process alarm, intrusion, thickness measurement, landscape movements. It periodically sends some information to the application logic on the central cloud but also offloading in the mobile edge part of the computation-intensive and data-intensive tasks. The entire service is partitioned in different VNFs and these functions are connected and deployed also in the edge cloud.

In case of an alert, the use of drones operating in manual mode or semi-automatic way should allow for the incident localization, support of the active team on their activities in the alerted zone. The drone is directed in the area 1: the application platform must create the service that send the drone in the given point and have to perform some computation functionalities in the edge and report results to the application logic in the central cloud.

In Figure 29, we illustrate the special case in which the area in which the EDGE 1 is based don't have the capability to process and realize video analysis. In this case the application logic can instantiate in the second edge cloud the requested VNFs and allow the accomplishment of the mission performing the given data analysis.

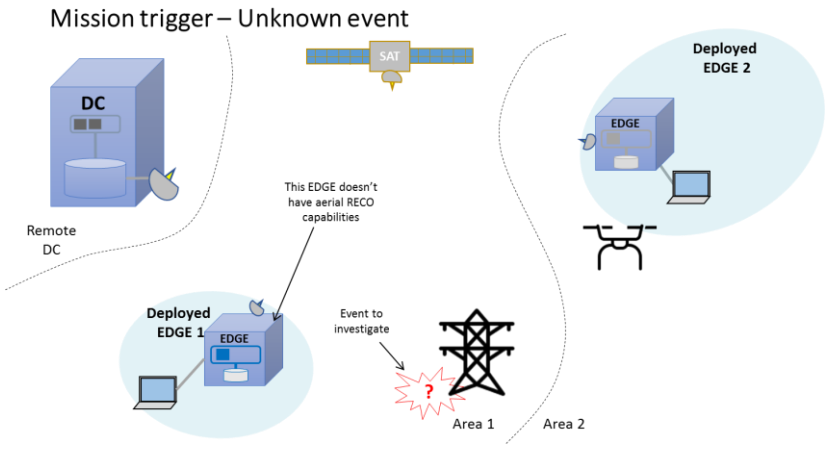


**Figure 29: Preventive Maintenance UC**

### 3.6.1.2 Key technologies description

The architectural scheme of NRG-5 [78] is based on the interaction between the OSM framework for management of NFVs and the application requirements related to the three families of 5G UCs that address Massive broadband, Massive and Critical machine-type communication.

The NRG-5 core functional decomposition is based on VNFs that are instantiated to create Network Services (NS), which are hosted by a VIM, such as Open Stack or Kubernetes. The VIM provides monitoring information to the MANO framework (e.g. Open Source MANO, OSM), which in turn publishes the information to the pub/sub bus (PSB) (e.g. Kafka).

The flexible adaptation of the entire NRG-5 framework is governed by the MAPE-K. This component is composed of different engines as in Figure 30.

The MAPE-K component will trigger and provide dynamic adaptation to the different application requirements is operating in synergy with the MANO for the intelligent placing and reconfiguration of the network. MAPE-K monitors and analyses the network environment and provides input to the planning engine. In parallel, the Application Logic poses a number of Requirements, which drive the Critical Infrastructure SLA (CI-SLAs) parameters.

Having both real-time information about the network conditions and the applications CI-SLA requirements, the planning engine in MAPE-K may employ analysis techniques (e.g. complex event processing, machine learning etc.) to propose the optimal VNF placing. VNF placing information is forwarded to the MANO framework via the execution engine and the PSB, while an autonomic loop provides continuous optimization. For the fulfilment of the application requirements MAPE – K component triggers and provides dynamic adaptation to the different application requirements operating in synergy with the OSM for the intelligent placing and reconfiguration of the network based on automatic network management solution based on

machine learning, hence components and workflow were designed to support the Monitor, Analyze, Plan, and Execute parts that share Knowledge (MAPE-K) autonomic loop.

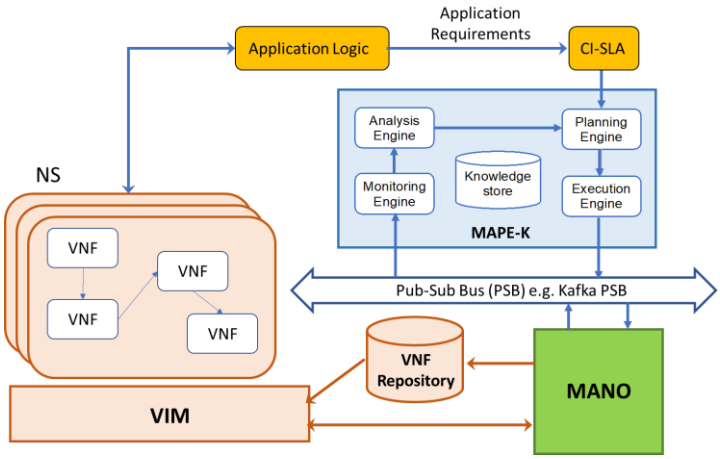


**Figure 30: MAPE – K**

# 3.7 Connectivity Services to Passengers

### 3.7.1.1 Use-case description

The selected satellite use cases in 5G elaborated in this section correspond to specific satellite use cases for eMBB which have been selected to be further investigated in the SaT5G project [7]. With focus on the eMBB usage scenario for 5G and by following the methodology presented in [64]. In this section we focus on communications on the move as presented in Figure 31.

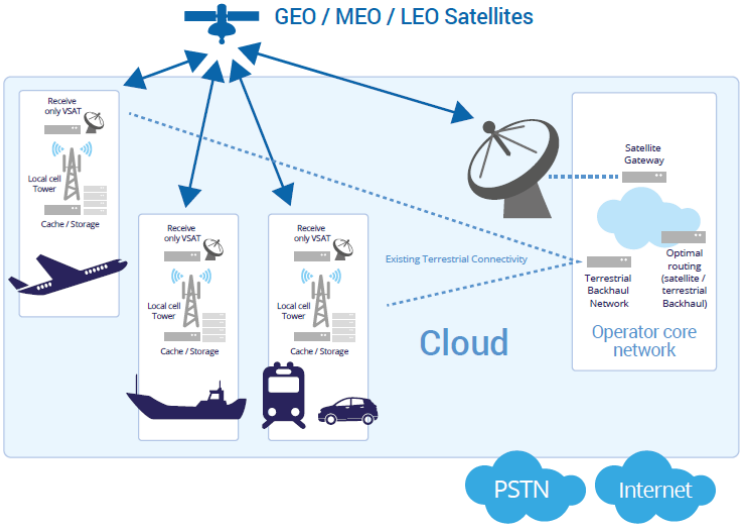


**Figure 31: 5G connectivity services to passengers**

One of the promises of 5G is high-speed backhaul connectivity to individual in-motion terminals on airplanes, vehicles, trains, and vessels (including cruise ships and other passenger vessels), with the ability to multicast the same content [e.g. video, HD/UHDTV, Firmware and Software Over the Air (FOTA/SOTA), as well as other non-video data] across a large coverage area (e.g. for local storage and consumption). The same capability also allows for the efficient backhauling of aggregated IoT traffic from these moving platforms. A very-high-speed, multicast-enabled, satellite link (up to Gbps speed), direct to the airplane, vehicles, train, or vessel, from geostationary and/or non-geostationary satellites will complement existing terrestrial

connectivity, where available (such as, in airports, harbors, train stations, and connected cars). Moreover, the satellite user links are either bidirectional and/or unidirectional since, depending on the case, broadband (i.e. unicast, thus VSAT satellite terminals) and/or broadcast/multicast (thus receive only satellite terminals) communications are supported by this category. Within the framework of SaT5G project, the project demonstrates an in-flight connectivity scenario as explained below.

Only a few years ago, a single movie could be played simultaneously for all passengers during a flight. Technology evolution delivered further options such as on-demand viewing platforms with built-in screens on the back of the seats, allowing viewers to choose from a variety of TV shows, movies and games. To enhance the passenger's experience of the in-flight entertainment system, the aviation industry targets to have live media contents on board. With the help of 5G technology integrated with the satellite systems, now it is possible to deliver live and refreshable contents onboard using multicasting video content delivery over the satellite systems.

#### 3.7.1.2 Key technologies description

TALENT is a coordination solution introduced within the framework of H2020 SaT5G project. It supports end-to-end services composed of satellite, radio access, cloud and mobile edge computing resources. TALENT in this use case automate the provisioning of resources (satellite and terrestrial connectivity) as well as computational resources at main datacenter and inflight edge. TALENT features important aspects are:

- Not vendor-locked and can support satellite and radio elements of different vendors.
- NFVO (e.g. OSM, ONAP) and VIM (e.g. OpenStack) agnostic.
- An end-to-end service management over cloud and edge computational resources.
- A single and easy to use point of interaction for all stakeholders evolved in the ecosystem, e.g. terrestrial and satellite operators as well as different 5G verticals.

The high-level design and internal architecture of TALENT, focused on being a top-layer solution, modular and extensible. As the solution has to be completely agnostic form the underlying layers, the architecture has to offer a high grade of dynamism when it comes supporting different frameworks, tools and proprietary solutions. Figure 32 presents TALENT's high-level architecture, while the following lines elaborates on the different components and modules that compose TALENT:

- *Northbound REST API*: This is the main entry point for different actors such as operators and verticals. It also provides an abstraction layer, which exposes a well-defined set of functions serving different needs of operators and verticals (e.g. service instantiation, check service status, QoS monitoring, to name a few possibilities). TALENT API is a REST-based API, which feed the TALENT Graphical User Interface - GUI.
- *MANO Selector Plugin*: This module is a reference point where all the required information such as IP address, data model format, vendor and interaction procedures of the supported NFVOs (OSM, ONAP, etc.) and VIMs (OpenStack, OpenVIM, etc.) are kept. "MANO Selector Plugin" is responsible to register all supported NFVOs and VIMs at the bootstrapping phase. This release supports OSM release 4 and 5, and OpenStack Queen. In the next releases, other NFVO (e.g. ONAP) and VIM (e.g. OpenVIM) solutions will be added to the TALENT.
- *Multi-MANO Lifecycle Manager*: This module is responsible to support NFVO and VIM clients, including their actual service and life-cycle management at the run-time phase. It uses a template file to produce request with required attributes to the cloud/edge domains that registered to system. To manage all received requests, "Multi-MANO Lifecycle Manager" synchronizes with "MANO Selector Plugin" to load the required dependencies and interact with underlying NFVO and VIM solutions.
- *Domain Component Plugin*: It is a reference point to keep all the required information such as IP address, data model format and configuration settings of the supported satellite

components. “Domain Component Plugin” is responsible to register all supported satellite components at the bootstrapping phase. TALENT will be loaded required dependencies and libraries to establish communication with underlying satellite solutions. Release 1 supports TotalNMS of Gilat. Later, more vendors will be added to the TALENT.

- *Domain Configuration Module*: This module is responsible to support satellite clients including configuration and life-cycle management of supported solution at the run-time phase. Similar to “Multi-MANO Lifecycle Manager”, it use a template file to produce request with required attributes to the satellite domains that registered to the system. To manage received requests, “Domain Configuration Module” synchronizes with the “Domain Component Plugin” in order to load the required dependencies and interact with underlying satellite solutions.

For more details, we invite the reader to go through this reference [76] where the main operational phases (bootstrapping and run-time) of TALENT are also described.

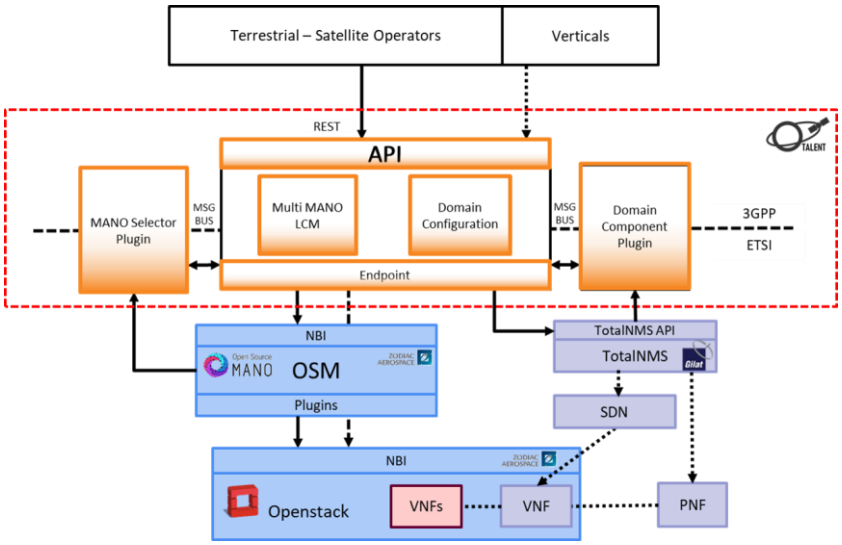


**Figure 32 High-level design**

# 4 Technological Analysis: Evolving from VNFs to CNFs

PNFs and VNFs are likely to be with us for at least another decade. The only feasible approach for Cloud-Native telecom is to offer an evolution of PNFs and VNFs to become CNFs (Container Network Function). This mirrors how enterprises are moving their monoliths to Kubernetes and then (often slowly) refactoring them into microservices. For this to be economic, there need to be incremental gains in resiliency, bin packing, and development velocity as more network functions become Cloud-Native.

Moving from network functionality from physical hardware to encapsulating the software in a virtual machine is generally easier than containerizing the software. In fact, many network function virtualization VMs rely on kernel hacks or otherwise do not restrict themselves to Linux kernel users-pace. They also often need to use DPDK or SR-IOV to achieve enough performance. However, containers provide nearly direct access to the hardware with little or no virtualization overhead. But they expect containerized applications to use the stable user-space Linux kernel, not to bypass it.

There’s been a similar journey on the software stack deployed currently in telco eco-system and in 5G-PPP level. Most of the prototypes and the project realization moved from a pure OpenStack

ecosystem, derived by ETSI MANO, to include the capability to run Kubernetes on top of either bare metal or any cloud where the intelligence is still centralized in VNFM like box. Whereas in the future version, most functions will be CNFs running on Kubernetes (denoted in short K8s). If some VNFs haven't been ported to CNFs yet, they can use technologies like KubeVirt and Virtlet to be managed by Kubernetes. This transformation has been captured in Figure 33 below.

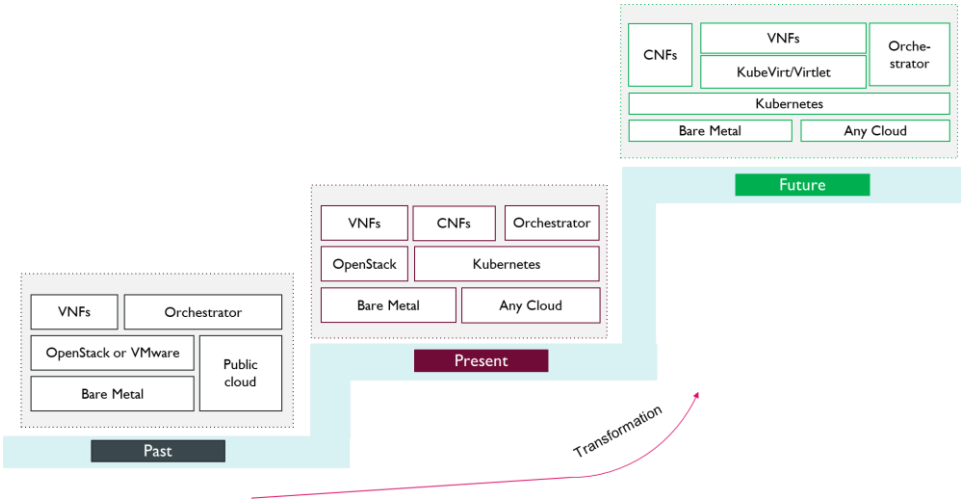


**Figure 33: Evolution of the VNFs toward container and Cloud-Native network functions**

## 4.1 Past: MANO-centric

Network function virtualization (NFV) decouples software NFs, such as MME, from proprietary hardware appliances to transform them into building blocks that can be flexibly combined to build network services.

A network service is defined as a composition of network functions and defined by its functional and behavioral specification. VNFs can be chained with other VNFs and/or Physical Network Functions (PNFs) to realize a network service. The NF Forwarding Graph (NFFG) describes the topology of the network service or a portion of the Network Service by referencing VNFs and PNFs and (virtual) links that connect them [18].

An end-to-end network service (e.g. mobile voice/data, Internet access, a virtual private network) can be described by one or multiple NF Forwarding Graph(s) of interconnected Network Functions and end points. The end points correspond to devices, applications, and/or physical server applications. ETSI NFV MANO Architecture is here defined.

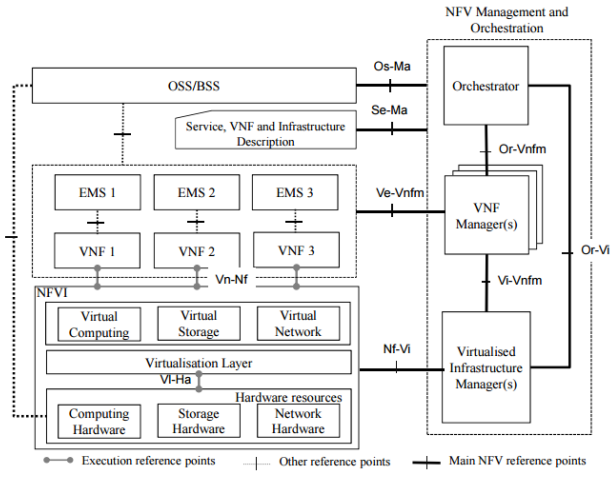


**Figure 34: ETSI NFV MANO Architecture**

The NFV MANO architectural framework represented in Figure 34 shows the functional blocks. Multiple functional blocks may be merged and the reference point amongst them can be readily internalized. Each of the functional blocks has a well-defined set of responsibilities and operates on well-defined entities, using management and orchestration as applicable within the functional block, as well as leveraging services offered by other functional blocks.

In the NFV MANO architectural framework different NFV MANO functional blocks are identified:

- Virtualized Infrastructure Manager (VIM),
- NFV Orchestrator (NFVO),
- VNF Manager (VNFM).

Additionally, we have Element Management (EM), Virtualized Network Function (VNF), Operation System Support (OSS) and Business System Support functions (BSS), NFV Infrastructure (NFVI), which share reference points with NFV MANO.

Referring to a recent analysis published in [78], some concerns around MANO are pointed out. These concerns are less about architecture, and more about implementation:

- *Multi-domain orchestration*: In many cases, multi-VIM or multi-clouds features is needed to support diverse use-case case e.g. data plane in one edge cloud and control plane in public cloud. Having an OpenStack across many clouds is very hard. However, Kubernetes can be run across different clouds having consistent orchestration between clouds.
- *Manage multiple vendors' virtualized network functions (VNFs) in a consistent way*: While multi-vendor orchestration is supported, most VNFMs are offered by the same compagnies that supplies the VNF recreating the hard dependency 1:1 between hardware and Element Managed System that exist in physical world.
- *Build-to-order MANO*: At architecture level, it is possible to combine the different blocks from different suppliers. But, at implementation level, it depends on the use-case and on the orchestration, system making hard to combine different blocks from different vendors.
- *Impact of Hyperscale public cloud providers*: This is translated by the recent support of container in Open source communities such as Open Network Automation Platform (ONAP) and Open Source MANO (OSM).

These concerns could be easily resolved if MANO had been defined as a platform-as-a-service in the sense of a framework of APIs into which all VNFs would integrate. Tom Nolle said in his recent post[5]: "*The PaaS would enable operators to create, contract for, or require as a condition of use, a standardized "adapter" that would expose all control, parametric, and management APIs and data in a common way*".

## 4.2 Present: MANO enhancements towards "Cloud-native" and "PaaS"

- **VNFManager as the central orchestrator:**

  As Cloud-Native transformation unfolds, it becomes evident that at least for the transition period (and perhaps beyond that), both traditional PNFs, VM based VNFs and container based VNFs will have to co-exist seamlessly within the context of the same network services. Other types of VIM like Kubernetes or Function as a Service (FaaS) are added in the architecture

[5] https://blog.cimicorp.com/?p=3821

via VIM type plugins while keeping the orchestration intelligence centralized in a VNFManager as it is depicted in Figure 35.

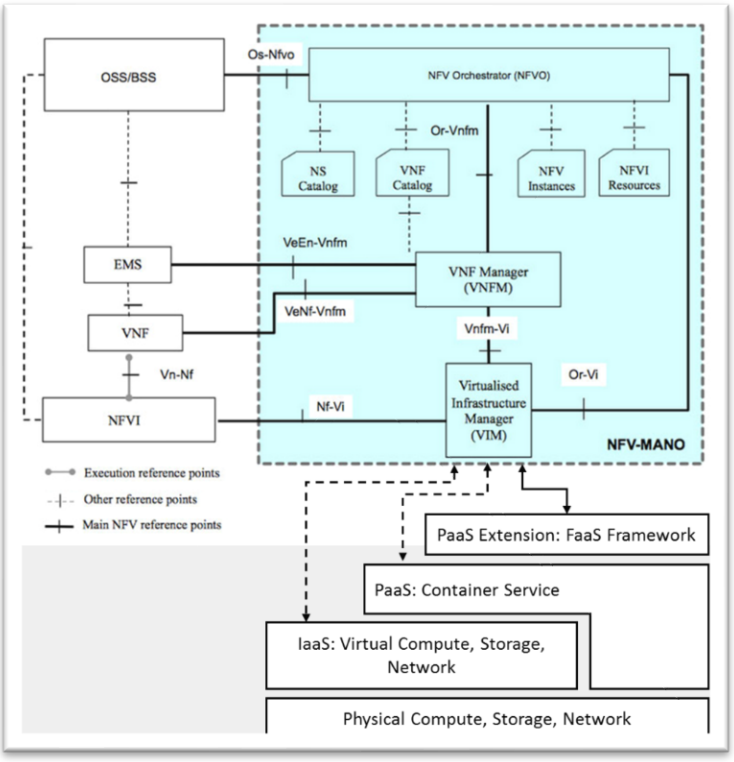


**Figure 35: Extension of ETSI MANO architecture to support other type of virtualization**

- **FaaS/OpenWhisk:**

One of the design tenets of integrating the FaaS framework into the MANO architecture is the minimal changes to current ETSI MANO flows. In fact, except for the scale-up/down flow, there are no changes at all to the standard ETSI MANO flows. The reason for this is as follows.

One of the main advantages of the FaaS approach is the built-in support for scale out. Therefore, when a new VNF instance should be started, it just amounts to invoking another instance of the OpenWhisk action (via the FaaS VIM Plugin) that implements this VNF. Moreover, since the OpenWhisk actions have a limited life time for execution, scale-in – which is a considerably more complex operation than scale-out – is supported out of the box as well.

- **Containers:**

Containers are supported via the usage of Kubernetes VIM Plugin. In that case, Kubernetes is considered only as a container infrastructure service, where all the intelligence is kept at the VNFManager.

- **ETSI IFA029 [79]:**

These enhancements towards “Cloud-native” and "PaaS" are discussed in ETSI IFA029 draft where the concept of VNF common and dedicated services have been introduced. These VNFs are instantiated inside the PaaS and both expose capabilities that are consumed by the Network Services (composed by consumer VNFs) that runs over the PaaS:

- *VNF Common Service*: common services or functions for multiple consumers. Instantiated independently of any consumer. For example, a generic monitoring service framework.

- *VNF Dedicated Service*: required by a limited set of consumers with a specific scope. Are instantiated dependently of their consumers, so it is instantiated when required by a consumer and destroyed when no relation exists with any consumer.

ETSI IFA029 describes a set of use cases for the deployment of consumer and common/dedicated VNF services using container-based virtualization. These use cases express the possible deployments in different cases:

- NFVI that only supports hypervisor-based virtualization. In this case, the container service is offered nested in a virtual machine. Thus, the Container Infrastructure Service Management is inside the VNF and is not visible from the MANO.
- Deploying the VNF Component (VNFC) in container on bare metal. NFVI is responsible to provide the container runtime environment on all nodes, so should provide the OS and the virtual network for the node's communication and NFV-MANO provides the management of the VNFC, (assuming that the Container Infrastructure Service Manager is part of MANO) allocating resources and providing and provisioning the containers. Should detect in the descriptors the usage of containers.
- NFVI provides containers on bare metal and VM at the same time. The Container Infrastructure Service Manager is at VIM level and can manage several Container infrastructure service instances.

The concept of PaaS service catalogue is introduced to enable their re-use for instantiating multiple VNFs. This catalogue is a set of PaaS service descriptors for individual PaaS services. Before a consumer VNF is instantiated, if the VNF Descriptor (VNFD) has references or requirements of a PaaS service, the VNF Manager (VNFM) request the discovery of the PaaS service descriptor inside the catalogue and creates a temporary VNFD with the original consumer VNFD and with the PaaS service descriptor allowing the instantiation of the services that compose the PaaS.

## 4.3 (Possible) future: Kubernetes-centric

- **Looking into the future is challenging!**

Going forward, a legitimate question to ask is whether a mature container orchestrator engine (e.g., Kubernetes) itself can become a fully-fledge VNF orchestrator? Obviously, if a VNF is implemented using a CRI-O (Container Runtime Interface) and deployed on Kubernetes, an intent-driven orchestrator such as Kubernetes is a natural choice for VNFM. In principle, Kubernetes can manage resources across the board via its innate Custom Resource Definition (CDR) mechanism. We posit that at this stage it is difficult to predict the exact adoption path of the Cloud-Native technology. It is reasonable to assume that Cloud-Native capabilities will be *explored* and *leveraged* via technologies such as Kubernetes while legacy will continue playing an important role. Hereunder are several topics that need to be explored.

- **Kubernetes as 5G default orchestrator?**

To ensure virtualization technology neutrality and portability across different use cases and future environments, Kubernetes is used as an industry de facto standard container orchestrator. Kubernetes can be either deployed on the bare metal servers or on top of some virtualization technology. Some predict Kubernetes can become the operating system for 5G networks. Indeed, thanks to its flexibility, such an orchestration engine will accelerate the adoption of container network functions (CNF) by operators.

- **Carrier-grade capabilities**

Kubernetes already offers a myriad of options for networking and with the adoption of Network Service Mesh as a CNCF sandbox project, is poised to support nearly all other use cases:

- The CNCF-hosted project Container Network Interface (CNI) supports over a dozen networking technologies, including Multus and DANM, which has seen use in operator applications
- Network Service Mesh (a new sandbox project) flexibly creates layer 2 or 3 network endpoints ("virtual tunnels") similarly to how Envoy/Istio work with TCP and HTTP
- Kubernetes implements cluster functionality as a Custom Resource Definition (CRD)
- Kubernetes supports several carrier-grades features such as bridging VPNs, high performance vSwitch, and connecting to PNFs
- Kubernetes supports IPv6 with dual-stack support coming this summer

- **Orchestrator scalability**

Telco applications have stringent requirements especially in terms of latency. Hence, the deployment of some CNFs at the network edge will be mandatory to meet these requirements. To achieve this goal, operators need to deploy Kubernetes at *large scale* with hundreds of thousands of instances at the edge. However, this distributed cloud architecture imposes challenges in terms of resource management and application orchestration. In this perspective, k3s a lightweight K8s is put forward by Rancher to address the increasing demand for small, easy to manage Kubernetes clusters running in resource-constrained environments such as edge. It is straightforward to see that k3s will enable the rolling out of new 5G services relying on multi-access Edge Computing (MEC) deployments.

As detailed in Figure 36, k3s relies on the following Kubernetes components:

- *kube-apiserver*: It acts as the gatekeeper through which all operations are passed to perform on the cluster. It is responsible for exposing different APIs. To do so, it maintains RESTful services to perform operations, hence allows the configuration and validation of data related to k3s objects including pods, services, etc. Note that the aforementioned objects will be detailed later.
- *kube-manager*: It is responsible for the overall coordination and health checking of the entire cluster. It acts as the conductor and the coordinator which ensures that the nodes are up and running and the pods are behaving the right way and the desired state of the configuration is continually maintained
- *kube-scheduler*: It is responsible for physically scheduling artifacts which could be containers or pods across multiple nodes. Depending on the specified constraints in terms of CPU, memory, disk, affinity/anti affinity, etc., the scheduler selects the appropriate nodes that meet the criteria and schedules then the pod appropriately.
- *kubelet*: It is an agent which runs on the node to ensure the monitoring of the pods which are composed of containers running on the node, restarting them if required to keep the replication level. To do so, it watches for pod specs via the Kubernetes API server.
- *kube-proxy*: This is a network proxy which runs on the node to ensure TCP, UDP forwarding. It is used to reach services. Specifically, it reflects the services as defined in the Kubernetes API. It refers to the API server to build a bunch of iptables rules and reference the portal IP.

All these components are bundles into combined processes that are presented as a simple server and agent model which will facilitate their deployment in the edge environment.

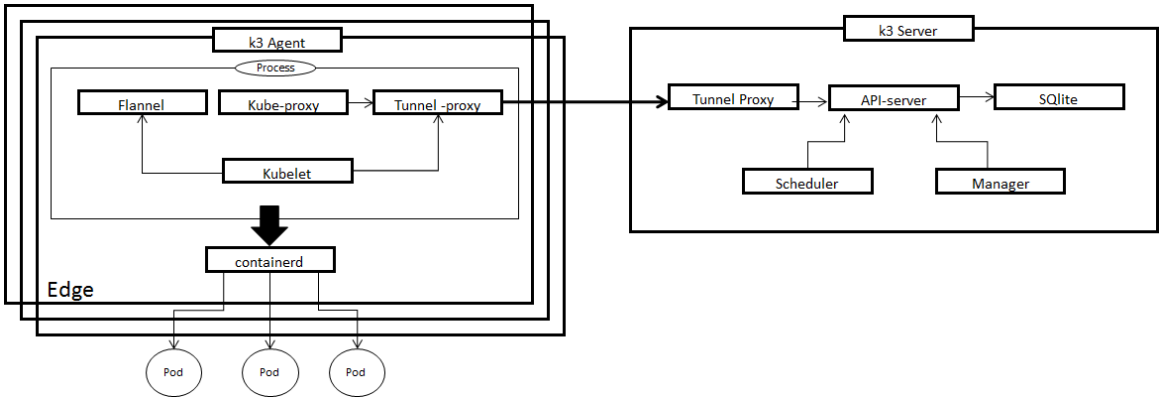


**Figure 36: k3s architecture**

## 4.4 Evolution of Open-Source platforms

The same trend highlighted in the previous sections: from MANO centric, to MANO enhancements toward "Cloud Native ", toward Kubernetes-centric, is observed in the latest releases of open-source platforms, like: CORD, OSM, ONAP and SONATA.

### 4.4.1 CORD

#### 4.4.1.1 Scope and architecture

Central Office Re-architected as a Datacenter (CORD) [24] is an edge network data center platform that allows Telco operators to virtualize their Central Office (CO) infrastructure. This is achieved by leveraging the Software Defined Networking (SDN), Network Function Virtualization (NFV) and Cloud technologies, hence allowing for greater scalability and programmability when compared to a traditional CO.

To provide this functionality, CORD comprises three functional elements: an orchestrator, an SDN Controller (SDNC), and a Virtual Infrastructure Manager (VIM). The role of the orchestrator is fulfilled by XOS [67], the role of the SDNC is fulfilled by ONOS [68] and finally the VIM can be OpenStack [69], Kubernetes [70] or both. OpenStack provides support for Virtual Machine (VM) based Virtual Network Functions (VNFs), while Kubernetes provides support for containerized VNFs.

#### 4.4.1.2 CORD's evolution towards a Cloud-Native design

Since its initial release in 2015, CORD has gone through multiple releases, which have dramatically increased its capabilities, stability and overall performance. The biggest shift in the design of CORD came with the release of version 6.0. Prior to this version, CORD was deployed using a static and error prone workflow without support for either component versioning or onboarding of new VNFs. Additionally, only OpenStack VIM was supported, meaning that containerized VNFs could not be deployed. This monolithic design and unreliability in deployment, acted as a barrier to the adoption of CORD by Telco Operators and the Open Source community. A simplified view of the pre-6.0 CORD architecture is illustrated in Figure 37. It should be noted that, both XOS and OpenStack comprise of sets of LXD containers and are not singular components.

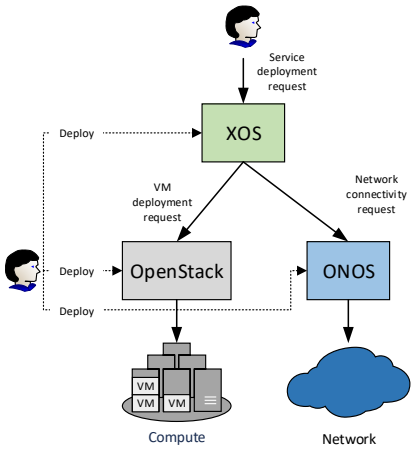


**Figure 37: Pre-CORD 6.0 architecture**

These majority of considerations raised by the Telco industry and the Open Source community were addressed with the release of CORD version 6.0 [66]. There two major shifts were featured, both related to the introduction of the Kubernetes VIM. With regards to the complexity of the deployment process, Kubernetes allowed for a much higher degree of orchestration and control thus making the deployment process simpler, more flexible and less error prone. Additionally, due to the loosely-coupled and containerized architecture, both component versioning as well as ad-hoc deployment of new components was introduced. Finally, the considerations about the lack of containerized VNFs, were also resolved using Kubernetes as a VIM for VNF deployment and management. This provided faster service deployment with reduced computational overhead. The new architecture of CORD is illustrated in Figure 38, again in a simplified format. While this release has not been characterized as Cloud-Native, the new architecture of CORD, is pointing towards this direction.

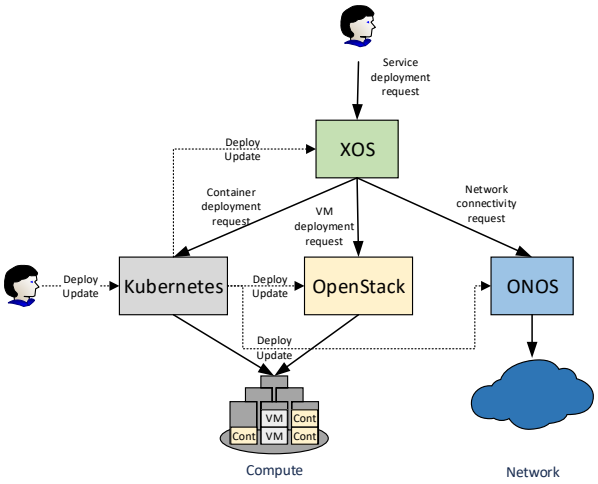


**Figure 38: CORD 6.0 architecture**

However, even with the introduction of Kubernetes the CORD architecture remains relatively complex. To address this issue and thus move closer towards cloud nativity, ONF is currently trying to further decouple the associations between the different components of CORD. This is done by decomposing the CORD architecture into smaller projects of a more specific scope. Some examples are the SEBA [71] and Trellis [72] platforms. SEBA is a variant of the initial Residential CORD (RCORD) release, tailored for residential access networks. Trellis is an SDN-based L2/3 switch fabric, designed for use in datacenters.

### 4.4.2 OSM

ETSI OpenSource MANO (OSM) release 5 does not currently provide support for Cloud-Native network functions.

Future work on OSM will include support for Kubernetes as a VIM. To provide this support, the OSM Information Model is expected to include K8s-cluster assignments and descriptions. Several options can be considered for exploration, for example the option that a Kubernetes cluster might be deployed dynamically through a VIM plugin. Another future needed feature is the ability of OSM to run VNF as containers.

### 4.4.3 ONAP

Open Network Automation Platform (ONAP) is the product that enables capabilities for design, creation and lifecycle management of the network services, providing the vendor-agnostic framework for future 5G network and services deployments. The ONAP community defines blueprints for 5G use cases, focused on automating 5G using today SDN/NFV technologies and future Cloud-Native applications.

The current ONAP release – Casablanca - has no support for Cloud-Native network functions. The next one named – Dublin – recently announced (June 2019), is expected to include support for microservices-based architectures, orchestration and Cloud-Native deployments (public and private). ONAP release 4 Dublin is enabling the deploying of virtualized and containerized networking functions in K8s based on Cloud regions as functional requirements.

### 4.4.4 SONATA

SONATA NFV platform, [77], is the NFV MANO framework contributed from H2020 SONATA and 5GTANGO projects. Its upcoming release 5.0 (which will be published in July 2019) will provide support to Cloud-Native network functions (CNF). In order to implement this support novel features have been added:

- Support for Cloud-Native deployment units within VNFDs: CNF support. The updated VNFD of release 5.0 support Cloud-Native deployment units (CDUs) as alternatives to typical virtual deployment units (VDUs) or physical deployment units (PDUs). The new CDUs allow to clearly specify container image, connection points, and environmental variables as required.

Support for Kubernetes as a VIM: A plugin for Infrastructure Abstraction component has been created. It is the component of SONATA framework that is responsible to manage the interface between MANO and deployed functions on top of Kubernetes.

# 5 Remaining barriers toward a truly and fully Cloud-Native Telco system

There is a larger and growing demand for Cloud-Native solutions in the telecommunications arena. This was highlighted by the CTO of Telecom Italia at the TM Forum's Digital Transformation World event in May 2019, with the comment that "vendors are not yet delivering Cloud-Native software for us. Time is running out."[6] Technology is maturing and solutions for serverless and Cloud-Native telco core functions and applications are emerging as discussed in

[6] https://www.lightreading.com/business-employment/business-transformation/nice-turns-nasty-as-vendors-face-fire-over-cloud-native-claims/d/d-id/751460?

this white paper. However, when it comes to virtualized telecommunications functions and applications there is still much work to be done and it is not just simply a matter on converting current research results into products.

In the Software Network WG, we analyzed how 5G-PPP projects interpret Cloud-Native design patterns and identified their adoption barriers.

- First, Cloud-Native is **relatively new in comparison to other approaches** and technologies that have been widely used to develop and create virtualized systems.
    - From a business viewpoint, ready-to-pick services that have been available for long time provides shorter time-to-market and therefore generate a faster revenue. These long existing, stable services are not currently Cloud-Native. The time of transforming them to Cloud-Native would introduce a delay in time-to-market, which in turn reduce the market share if other competitors arrive first to the market. So, for certain verticals, the abundance and maturity of non-Cloud-Native solutions are the main drivers of this deliberateness.
    - From an organization viewpoint, cloud native, e.g. microservice, bring cultural change as well. Change in the way of working. Moving away from siloed teams to cross functional teams. The teams will be formed as per the functionality or the module. And not based on the skillset. Each team, called pizza team, will have a mix of people having different skills. Two pizza teams are autonomous, independent and have all the freedom for the services they are responsible for. Telco companies are still far from this model.
    - Additional research and development, prototype development and testing in several areas of Cloud-Native solutions for telecommunications functions and applications. We can cite for example:
        - architectures for the hybrid operation mixing serverless orchestration logic with container virtualization;
        - federation protocols and optimization algorithms to enable seamless operation across geographically distributed clusters and cloud nodes, including the support of Cloud-Native service function chaining's;
        - mechanisms for the establishment and management of service chains across services composed of a mixture of traditional VNFs and FaaS-based VNFs.

- Second, another factor is the **lack of standards in Cloud-Native technologies**.
    - On one hand, standards like ETSI GS NFV-IFA 013 gives a precise guidance and definitions on the elements of architecture, interfaces, and technologies. On the other hand, Cloud-Native is mainly driven by opensource software. The opensource gives openness and adaptability. It allows anyone to contribute and use a software, and it advances faster because of the high number of community members contributing. Unfortunately, however, it is not as strict and well precise as the standards. For example, the Service Function Chaining and VNF Forwarding Graphs (VNFFGs) is well defined in [73] [74] for MANO based architecture. However, the equivalent feature does not yet exist in the Cloud-Native eco-system as such. But the open-source community and the different actors in Cloud-Native identified this gap and proposed framework like service mesh and network service mesh that can offer you these features.
    - Consequence: This lack of standardization raises compliance and interoperability questions. Two verticals providing the same service cannot be sure if their services would communicate with each other because there is no standard to comply with. This is a seen as a disadvantage of Cloud-Nativeness which is no more valid if we know that TM-Forum for example is solving this issue using the Open API framework. It is expected though that more attention is going to be

given on this point as the cloud is gaining momentum and showing measurable advantages.

- Third, Cloud-Native transformation is a long journey in Telco eco-system due **to non-technical reasons like investments, mindset,** etc. This point was addressed in the 1st white paper.

# 6 Conclusion

It becomes obvious that communications service providers need to adopt Cloud-Native architectures on their networks to meet customer demands for new services. This transformation is unavoidable if the telco wants to successfully and competitively compete on the market.

This paper is a follow-up of 2018 Cloud-Native transformation white paper published in July 2018, where we analyze how Cloud-Native design patterns are being interpreted within 5G-PPP projects and initiatives and identifies facts on Cloud-Native adoption barriers. Needs for standard and market maturity are the main barriers identified during the analysis phase of the input collection. Despite this "reluctance", it was acknowledged that large and small organizations are poised to develop functions primarily and natively designed for a cloud environment rather than re-packaged as an after-thought.

# 7 References

[1] http://ngpaas.eu/

[2] http://www.5gmedia.eu/

[3] http://5g-transformer.eu/

[4] https://www.5gcity.eu/

[5] https://www.5g-picture-project.eu/

[6] https://www.5gtango.eu/

[7] https://www.sat5g-project.eu/
[8] http://www.matilda-5g.eu/
[9] https://slicenet.eu/
[10] https://metro-haul.eu/
[11] https://5genesis.eu/
[12] Steven Van Rossem, et al., “A Vision for the Next Generation Platform-as-a-Service,” in Proceedings of IEEE 5G World Forum, July 2018.
[13] Katsalis et al., “Architectural Design Patterns for the RAN,” in IEEE ICC, 3rd International Workshop on 5G Architecture, 2016.
[14] F. Ahmed et al., “Distributed Graph Coloring for Self-Organization in LTE Networks,” Journal of Electrical and Computer Engineering, 2010.
[15] ITU-R, “IMT Vision – Framework and overall objectives of the future deployment of IMT for 2020 and beyond,” [Online]. Available: https://www.itu.int/dms_pubrec/itu-r/rec/m/R-REC-M.2083-0-201509-I!!PDF-E.pdf
[16] 3GPP RAN, 3GPP TR 38.811, V0.4.0 (2018-03), "Study on New Radio (NR) to support Non Terrestrial Networks (Release 15)", 2017.
[17] ETSI MEC, http://www.etsi.org/technologies-clusters/technologies/multi-access-edge-computing
[18] ETSI NFV, http://www.etsi.org/technologies-clusters/technologies/nfv
[19] ETSI GS NVF-EVE 011 V0.0.11, Specification of the Classification of Cloud-Native VNF Implementations, work in progress, 2018.
[20] ETSI GS NFV-IFA 005, Network Functions Virtualization (NFV) Release 3; Management and Orchestration; Or-Vi reference point – Interface and Information Model Specification, 2019.

[21] ETSI GR NFV-IFA 029 V0.8.0, Report on the Enhancements of the NFV architecture towards "Cloud-Native" and "PaaS", work in progress, 2018
[22] TM Forum, https://www.tmforum.org/virtualization/
[23] TM Forum ZOOM, https://www.tmforum.org/collaboration/zoom-project/
[24] ONF CORD, https://www.opennetworking.org/projects/cord/
[25] ONF XOS, https://www.opennetworking.org/projects/xos/
[26] Cloudify, https://cloudify.co
[27] OSM, https://osm.etsi.org/
[28] 5G-TRANSFORMER, D1.3, Refined Architecture, 2019, http://5g-transformer.eu/wp-content/uploads/2019/05/D1.3_5G-TRANSFORMER_Refined_Architecture.pdf
[29] 5G-PPP Software Network WG, 'Vision on Software Networks and 5G', White Paper, January 2017.
[30] 3GPP TS 23.501 V1.0.0 (2017-06), 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; System Architecture for the 5G System; Stage 2, (Release 15)
[31] [online] https://12factor.net/
[32] [online] https://www.heroku.com/
[33] 3GPP TR 23.799, Study on Architecture for Next Generation System, V14, Dec 2016.
[34] [online] http://www.restapitutorial.com/
[35] Sam Newman, Building microservices, Oreilly Edition, February 2015
[36] Thomas Erl, 'Service-Oriented Architecture (SOA): Concepts, Technology, and Design', Prentice Hall, 2005
[37] Neal Ford, 'Comparing Service based Architectures', Available online: http://nealford.com/downloads/Comparing_Service-based_Architectures_by_Neal_Ford.pdf
[38] Eric Evans, 'Domain-Driven Design: Tackling Complexity in the Heart of Software', Book, 20 August 2003
[39] Neal Ford, Mark Richards, 'Service-Based Architectures: Structure, Engineering Practices, and Migration', O'Reilly Media, July 2015.
[40] [online] https://github.com/cncf/landscape
[41] White paper, 'The next step in server virtualization: How containers are changing the cloud and application landscape', by Nuage-Networks, Nokia, 2017, available online: http://www.nuagenetworks.net/wp-content/uploads/2015/12/PR1512017019EN_NN_Docker_Integration_StraWhitePaper.pdf
[42] Derek Rangel, 'DevOps: Learn One of the Most Powerful Software Development Methodologies FAST AND EASY!', Book.
[43] [online] https://jenkins.io/
[44] [online] https://kubernetes.io/
[45] [online] https://cri-o.io/
[46] Ericsson Technology Review, 'Paving the way to telco-grade PaaS', 2016, available online: https://www.ericsson.com/en/ericsson-technology-review/archive/2016/paving-the-way-to-telco-grade-paas
[47] [online] S. Rajagopalan, L. Ryan, Istio: a modern service mesh', 2017, https://istio.io/talks/istio_talk_gluecon_2017.pdf
[48] [online] https://linkerd.io/
[49] [online] https://www.envoyproxy.io/
[50] [online] https://istio.io/
[51] [online] https://coredns.io/
[52] [online] https://www.consul.io/
[53] [online] https://coreos.com/etcd/
[54] [online] http://events17.linuxfoundation.org/sites/events/files/slides/Container%20Networking%20Deep%20Dive.pdf
[55] https://github.com/projectcalico/cni-plugin
[56] M. Hofmann, E. Schnabel, K. Stanley, 'Microservices Best Practices for Java', RedBooks, IBM, Dec 2016
[57] [online] http://www.rgoarchitects.com/Files/fallacies.pdf
[58] [online] https://wiki.onap.org/display/DW/ONAP+on+Kubernetes
[59] [online] https://guide.opencord.org/controlkube_scenario.html
[60] [online] https://cloudify.co/kubernetes
[61] [online] https://osm.etsi.org/gerrit/#/c/5837/5/Release4/K8_Support.md

[62] [online] https://medium.com/flow-ci/introduction-to-containers-concept-pros-and-cons-orchestration-docker-and-other-alternatives-9a2f1b61132c
[63] [online] https://avinetworks.com/what-are-microservices-and-containers/
[64] Shree Krishna, et al., "Satellite Communications in the 5G Era", IET The Institution of Engineering and Technology, 2018.
[65] [online] https://openwhisk.apache.org/
[66] [online] https://opencord.org/: "The CORD platform website"
[67] [online] https://www.opennetworking.org/xos/:"The XOS orchestrator",
[68] [online] https://onosproject.org/: "The ONOS SDNC website"
[69] [online] https://www.openstack.org/: "The Openstack website"
[70] [online] https://kubernetes.io/: "The Kubernetes website"
[71] [online] https://www.opennetworking.org/seba/
[72] [online] https://www.opennetworking.org/trellis/
[73] Halpern, J., Ed., and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <https://www.rfc-editor.org/info/rfc7665>.
[74] Bh, Deval & Jain, Raj & Samaka, Mohammed & Erbad, Aiman. (2016). A Survey on Service Function Chaining. Journal of Network and Computer Applications. 75. 10.1016/j.jnca.2016.09.001.
[75] [Online] https://5genesis.eu/malaga-platform/
[76] Pouria Sayyad Khodashenas, Hamzeh Khalili, Daniel Guija, Shuaib Siddiqui, "TALENT: Towards Integration of Satellite and Terrestrial Networks", European Conference on Networks and Communications (EuCNC), EuCNC 2019, Valencia.
[77] [online] https://sonata-nfv.github.io/
[78] Roz Roseboro, "MANO & the Future of CSP Automation", Heavy Reading Reports, August 2017 (https://www.giiresearch.com/report/heav549643-mano-future-csp-automation.html)
[79] Network Functions Virtualisation (NFV) Release 3; Architecture; Report on the Enhancements of the NFV architecture towards "Cloud-native" and "PaaS", https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?WKI_ID=50985
[80] [online] http://www.nrg5.eu/

# 8 List of Contributors

| Name | Company / Institute / University | Country | Project |
|---|---|---|---|
| **Editorial Team** | | | |
| ***Overall Editor*** | | | |
| Bessem Sayadi | NOKIA Bell-Labs<br>5G-PPP Software Network WG Chairman | France | |
| ***Contributors*** | | | |
| Thomas Deiss | NOKIA | Germany | 5GTRANSFORMER |
| Ilhem Fajjari | ORANGE Labs | France | NGPaaS |
| Bilal Al-Jammal | NOKIA Bell-Labs | France | NGPaaS |
| Cristian Patachia | ORANGE Labs | Romania | Matilda/Slicenet |
| Marius Iordache | ORANGE Labs | Romania | Matilda/Slicenet |
| David Griffin | UCL | UK | 5G-MEDIA |
| David Breitgand | IBM Research | Israel | 5G-MEDIA |
| Josep Martrat | ATOS | Spain | 5GTANGO |
| Ricard Vilalta | CTTC | Spain | 5GTANGO |
| Ricardo Preto | Ubiwhere | Spain | 5G-CITY |
| Apostolos Papageorgiou | I2CAT Foundation | Sapin | 5G-CITY |

| | | | |
|---|---|---|---|
| Daniel King | Old Dog Consulting | UK | METRO-HAUL |
| Ramon Casellas | CTTC | Spain | METRO-HAUL |
| Antonello corsi | Engineering Ingegneria Informatica | Italy | NRG5 |
| Jara Suárez de Puga García | UPV | Spain | 5Genesis |
| David ARTUÑEDO Guillen | Telefonica | ES Spain | 5Genesis |
| Hamzeh Khalili | i2CAT Foundation | Spain | SAT5G |
| Pouria Sayyad Khodashenas | I2CAT Foundation | Spain | SAT5G |
| Paris Flegkas | University of Thessaly | Greece | 5G-PICTURE |
| Azahar Machwe | Zeetta Networks | UK | 5G-PICTURE |

## Acknowledgment

We would like to thank all the project contributors that are indirectly involved in this White Paper and not cited directly in the list above.

## Contact

For any additional information, please contact: bessem.sayadi@nokia-bell-labs.com