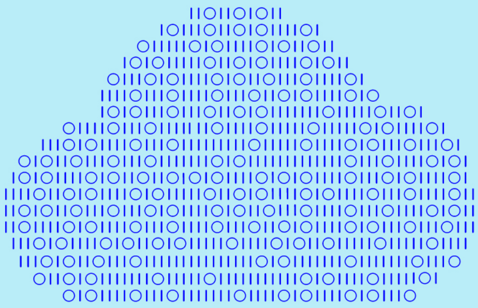
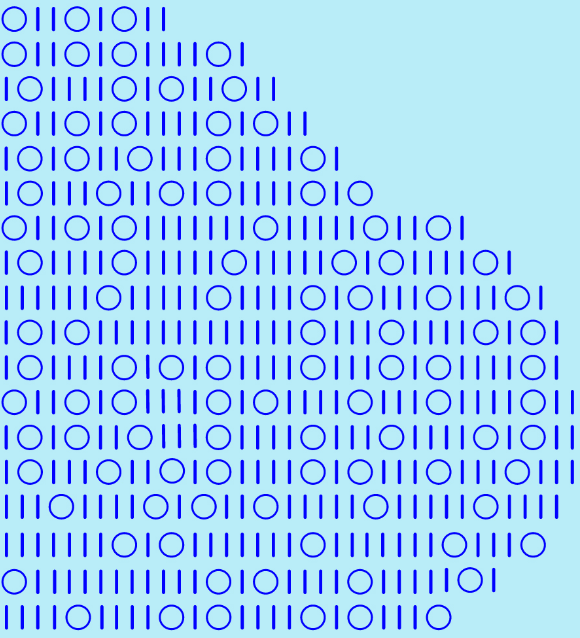
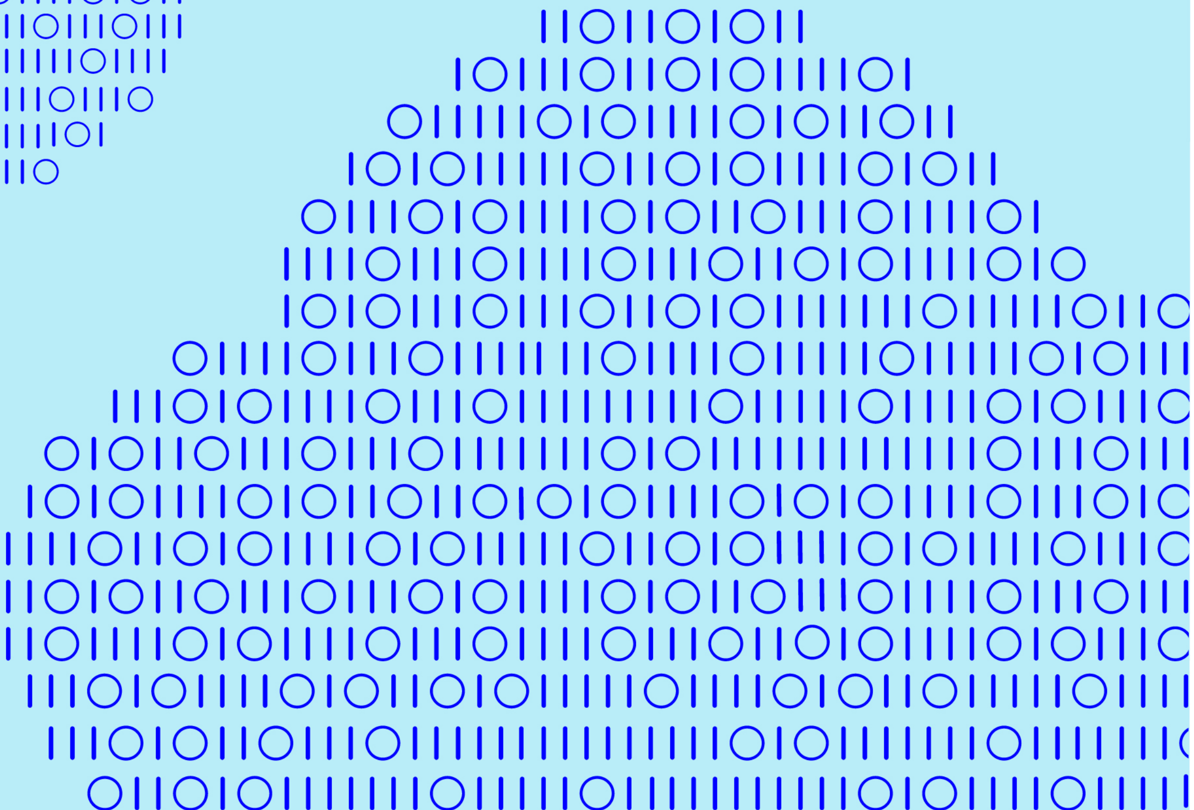
AIDA

Adaptive, Intelligent and Distributed Assurance Platform







Deliverable 1.1

AIDA - Preliminary Studies and Requirements

Date December, 2020

Activity N 1

Version 1.0

Index

1 Scope

The purpose of this document is to provide a preliminary overview of the 5G technology and its impacts in terms of fraud risks, ensuring that addresses all relevant areas of focus which will drive the requirements for a platform that is able to identify and prevent fraud in 5G Non Standalone and Standalone networks.

Unlike previous evolutions, 5G re-thinks and re-architects how the network is built and managed, by introducing emerging use cases and business models, affecting not only consumers but also enterprises and industries. The capabilities to support the divergent use cases proposed in 5G, can only be accomplished in a flexible and efficient architecture with the agility of modular network functions which can be quickly deployed and scaled on demand in a cost effective way, a paradigm which can only be achieved using containers in cloud native (CN) environment in order to create a cloud ready mobile core (MC). In this context, Software Defined Networking (SDN) and Network Function Virtualization (NFV), are the key enablers to achieve the cloudification of the 5G core (5GC) network, two core cloud native principles leverage by the European Telecommunications Standards Institute (ETSI), introducing a Service Based Architecture (SBA) as well as Cloud Native Network Function (CNF) in the reference NFV architecture. Moreover, 5G core is decomposed into Network Functions and Network Services exposed through Application Protocol Interfaces (API), providing the flexibility of breaking the existence functions into micro services, a design philosophy based on decomposition of business requirements into domains and each domain into services that fulfill the business requirements. The term micro is employed as a limitation for the functional scope being provided and not as scale or volume delimitation. By design, each micro service needs to be deployed individually, describing the interfaces and dependencies, conforming to the Representational State Transfer (REST) software architectural style. A RESTful web service, exposing a RESTful API that provides to its peers a constant endpoint for the service being provided.

Micro services in distinct containers provides not only scaling with much higher granularity, but also optimal tuning in terms of resource consumption. Yet, this brings an increase complexity when managing the infrastructure, requiring orchestration solutions to manage the life cycle management of Cloud Native Ecosystems and ensure reliable service-to-service connections that abstract the network lawyer, handles authorization, access control and quotas, and builds a zero trust environment based on service identity.

In the context of 5G, Fraud is an Application container integrated with the multiple network services and functions of the 5G Core and 5G RAN, deploying multiple micro services that report abnormal behavior.

The changes in 5G network will bring significant changes to fraud management, namely by introducing new attack vectors and type of stakeholders involved in fraud scenarios changing the current threat model. Specifically, the use of Artificial Intelligence (AI) to perpetrated fraud and avoid detection (Smart Fraud), massive expected increase in the number of connected devices and the new use cases which will expand 5G capabilities to new industries are the root drivers. It is key to address these new anticipated challenges but also consider that existent fraud will be carried out using different enablers. In order to face the challenges in 5g Fraud, it is essential to have real time analysis using network signaling and real time feeds which can be analyzed by algorithms with high accuracy and learning capabilities, reducing the amount of false positives, a key aspect considering the volume of IoT devices with inconsistent patterns in the network. The majority of fraud occurs from a lack of security, hence collaboration and integration with security is key in order to prevent fraud. This integration occurs at different levels, from information sharing between industries and stakeholders to end to end monitoring controls. The risk for multi-vector attacks in 5G is high, only integrated holistic views can reveal a potential fraud threat.

In the following chapters we will cover the main use cases and technologies that resumes the standard requirements expected to be address in a 5G fraud solution. This is not a static list and it's expected to change as the implementations of 5G network progress.

2 5G Overview

The next-generation of telecom networks (fifth generation or 5G) has started hitting the market at end of 2018 and will continue to expand worldwide.

Beyond speed improvement, 5G is expected to unleash a massive IoT (Internet of Things) ecosystem where networks can serve communication needs for billions of connected devices, with the right trade-offs between speed, latency, and cost. 5G connectivity promises to break traditional paradigms of data delivery by providing network connectivity almost everywhere.

5G technology is driven by 8 specification requirements [8]:

- Up to 10Gbps data rate - > 10 to 100x speed improvement over 4G and 4.5G networks
- 1-millisecond latency
- 1000x bandwidth per unit area
- Up to 100x number of connected devices per unit area (compared with 4G LTE)
- 99.999% availability
- 100% coverage
- 90% reduction in network energy usage

- Up to 10-year battery life for low power IoT devices

5G speed tops out at 10 gigabits per second (Gbps), 10 to x100 faster than 4G. The use of shorter frequencies (millimeter waves between 30GHz and 300GHz) for 5G networks is the basis for 5G high speeds. This high-band 5G spectrum provides the expected boost not only in speed but also in capacity, low latency, and quality. However, 5G download speed may differ widely by area.

5G technology offers an extremely low latency rate, the delay between the sending and receiving of information. From 200 milliseconds for 4G, we go down to 1 millisecond (1ms) with 5G.

The main evolution compared with today's 4G and 4.5G (LTE advanced) is that, beyond data speed improvements, new IoT and critical communication use cases will require a new level of improved performance that can only be catered to with 5G technology.

5G vs. 4G also means at least a 100-fold increase in the number of devices connected. 5G must be able to support 1 million devices for 0.386 square miles or 1 km2.

Also, low power consumptions will allow connected objects to operate for months or years without the need for human assistance.

Unlike current IoT services that make performance trade-offs to get the best from current wireless technologies (e.g. 3G, 4G, WiFi, Bluetooth), 5G networks will be designed to bring the level of performance needed for massive IoT. This will in theory enable a perceived entirely ubiquitous connected world.

5G will support all communication needs from low power Local Area Network (LAN) – like home networks, such as Wide Area Networks (WAN), with the right latency/speed settings.

5G is designed to allow simple virtual network configurations to better align network costs with application needs.

As of November 2020, 146 mobile operators have launched commercial 5G services according to the Ericsson mobility report. The study forecasts 220m subscriptions by end of 2020 and reaching 3.5 billion subscriptions by 2026, making it the fastest generation ever to be rolled out on a global scale.

5G is still a cellular broadband technology and is a network of networks.

Today, security solutions are already a mix of security at the edge (device) and security at the core (network).

Several security frameworks may co-exist in the future, and 5G is likely to re-use existing solutions used today for 4G networks and the cloud (SEs, HSM, certification, Over-The-Air provisioning, and KMS). [9]

Furthermore, based on the knowledge of the current situation it can be assumed that all possible intermediate state scenarios are likely to occur in some place or another during the implementation phase of 5G technology, especially also given the heterogeneous and non-linear nature of the distribution of this type of technology.

Therefore, for the introduction of 5G, 3GPP has specified 5 possible configurations or 'Options' for connecting to an EPC or new 5G core network (6 if the current 4G system is included).

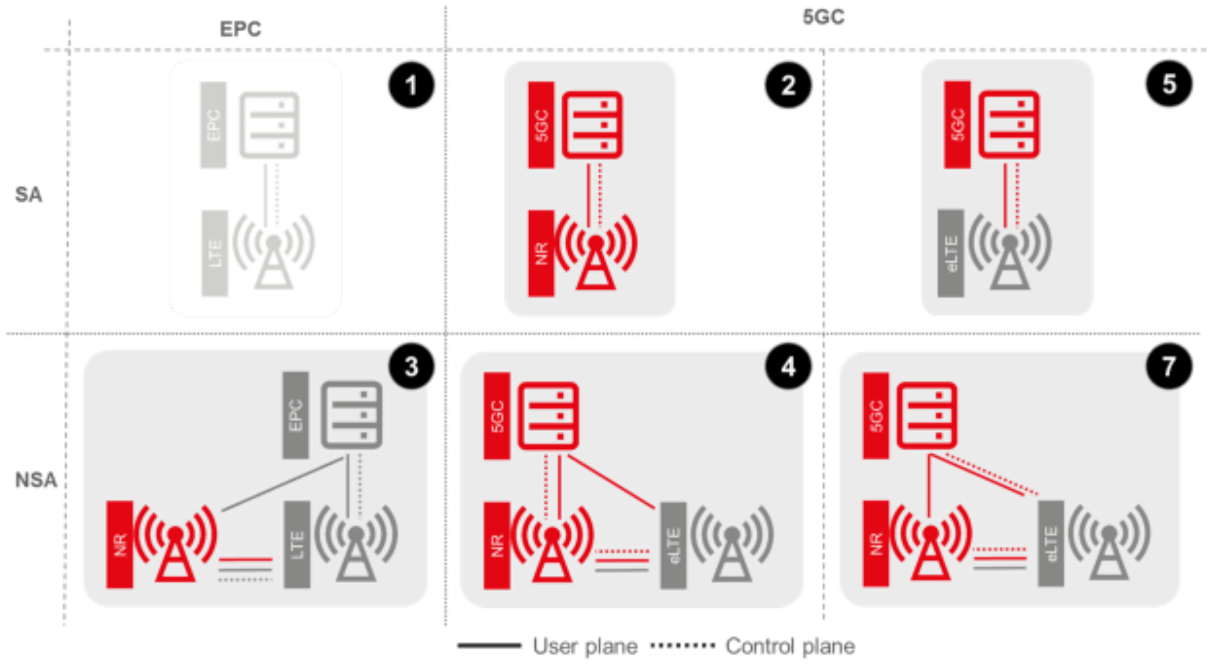


Figure 1 - 3GPP defined options for 5G deployment. [9]

The features that distinguish each Option are:

- Use of Dual Connectivity;
- Radio Access Technology acting as master node
- Core Network used.

The Options using Dual Connectivity are grouped together under the term "Non-standalone" (NSA) to indicate that 5G radio access technology (NR) and LTE are used simultaneously to provide radio access. Options where only one radio access technology is in use are referred to as "Standalone" (SA).

It is widely expected that mobile operators will initially deploy 5G using Option 3 allowing the re-use of existing EPC Core functionality. Option 3 has been fully specified in an early drop of 3GPP Release 15.

The other deployment Options already fully specified by 3GPP are Option 2 and Option 5, both standalone options differing in the type of radio access technology connected to the new mobile core network 5GC. These options were completed in June 2018 (ASN.1 in September 2018).

In addition, 3GPP has finalized the two remaining Options, Option 4 and Option 7, completed in March 2019 (ASN.1 in June 2019).

Operators have planned their network deployment strategies based on what is included in the 3GPP specifications, and therefore reduction of the support of these options after standardization is not feasible.

Support for option 2 will be delivered with the initial industry support for standalone connectivity to the 5G Core. The aim of this paper is to provide the supplier ecosystem with a view of Operators' requirements for the support the 5G Core connectivity options beyond option 2, i.e. options 4, 5 and 7.

To support this GSMA conducted some initial interviews with 20 Operators (and Operator groups) who are planning to launch 5G in the next 2 years to understand which of the 5 Options they are considering for their deployment plans. The results of this initial survey clearly indicated the Operators have a strong requirement for the additional support of Options 4, 5 and 7. In addition Operators require that Option 2 introduction is not delayed.

2.1.1 5G Architecture (NFV, SDN, CDN)

5G technology brings with it the need of a novel architecture built from the ground-up in order to accommodate the various use-cases described in more detail in 3.1.1. This is thus a challenging and demanding task, and one that is still in its initial steps.

Unlike previous evolutions, 5G re-thinks and re-architects how the network is built and managed, by introducing emerging use cases and business models, affecting not only consumers but also enterprises and industries. The capabilities to support the divergent use cases proposed in 5G, can only be accomplished in a flexible and efficient architecture with the agility of modular network functions which can be quickly deployed and scaled on demand in a cost effective way, a paradigm which can only be achieved using containers in cloud native (CN) environment in order to create a cloud ready mobile core (MC). In this context, Software Defined Networking (SDN) and Network Function Virtualization (NFV) are the key enablers to achieve the cloudification of the 5G core (5GC) network. These two core cloud native principles are leveraged by the European Telecommunications Standards Institute (ETSI), introducing a Service Based Architecture (SBA) as well as Cloud Native Network Function (CNF) in the reference NFV architecture. Moreover, 5G core is decomposed into Network Functions and Network Services exposed through Application Protocol Interfaces (API), providing the flexibility of breaking the existence functions into micro services, a design philosophy based on decomposition of business requirements into domains and each domain into services that fulfil the business requirements. The term micro is employed as a limitation for the functional scope being provided and not as scale or volume delimitation. By design, each micro service needs to be deployed individually, describing the interfaces and dependencies, conforming to the Representational State Transfer (REST) software architectural style. A RESTful web service, exposing a RESTful API that provides to its peers a constant endpoint for the service being provided.

Micro services in distinct containers provide not only scaling with much higher granularity, but also optimal tuning in terms of resource consumption. Yet, this brings an increase complexity when managing the infrastructure, requiring orchestration solutions to manage the life cycle management of Cloud Native Ecosystems and ensure reliable service-to-service connections that abstract the network lawyer, handles authorization, access control and quotas, and builds a zero trust environment based on service identity.

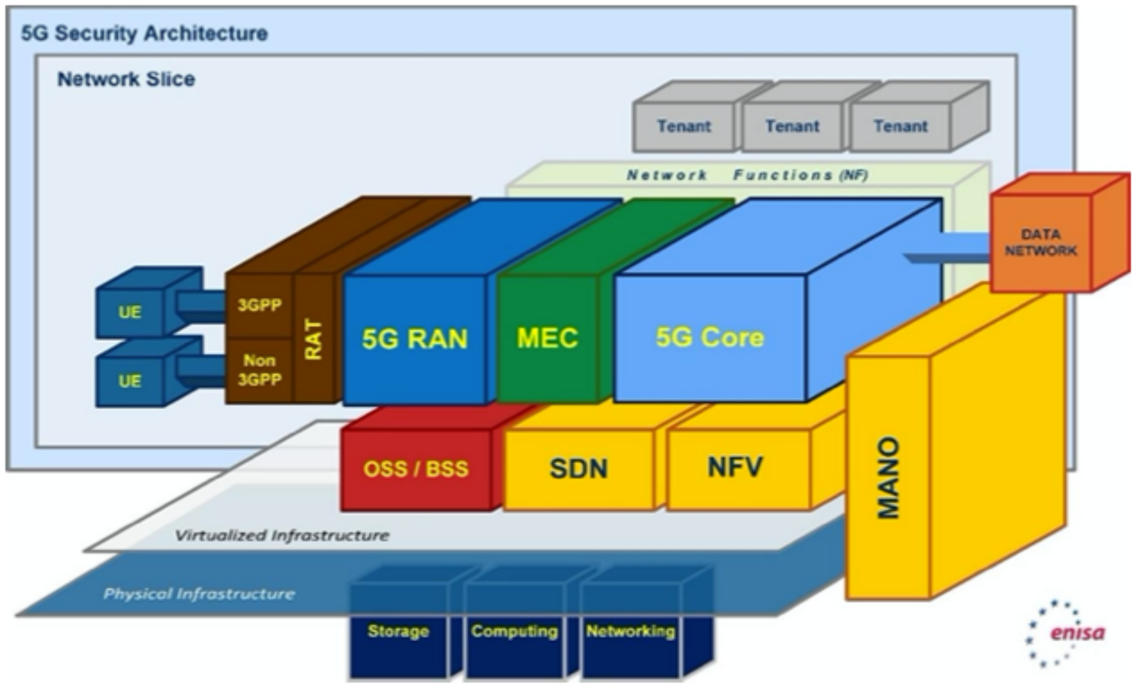


Figure 2. - 5G Architecture. There are many individual pieces involved in 5G technology. Above is a general scheme of the associated service architecture.

Network functions virtualization (NFV)

NFV is a way to virtualize network services, such as routers, firewalls, and load balancers, that have traditionally been run on proprietary hardware. These services are packaged as virtual machines (VMs) on commodity hardware, which allows service providers to run their network on standard servers instead of proprietary ones.

NFV decouples network functions from the hardware. Those network functions are called virtual network functions (VNFs). VNFs run in virtual machines on commercial off-the-shelf (COTS) hardware (typically x86 servers). NFV is used by software-defined networks and allows for scaling of VMs to handle changes in data center traffic, theoretically simplifying network operation.

The different functions — such as firewalls, traffic control, and virtual routing — are called virtual network functions (VNFs). NFV uses these virtualized networking components to support an infrastructure totally independent of hardware. The standard resources of compute, storage, and network functions can all be virtualized and placed on commercial off-the-shelf (COTS) hardware like x86 servers. Having virtualized resources means that VMs can be given portions of the resources available on the x86 server. That way, multiple VMs can run on a single server and scale to consume the remaining free resources. This also means that resources are less often sitting idle and data centers with virtualized infrastructure can be more effectively used. Within the data center and the outside networks, the data plane and control plane can also be virtualized with NFV.

A typical NFV architecture consists of:

- **Virtualized network functions (VNFs)** are software applications that deliver network functions such as file sharing, directory services, and IP configuration.
- **Network functions virtualization infrastructure (NFVi)** consists of the infrastructure components—compute, storage, networking—on a platform to support software, such as a hypervisor like KVM or a container management platform, needed to run network apps.
- **Management, automation and network orchestration (MANO)** provides the framework for managing NFV infrastructure and provisioning new VNFs.

NFV removes the need to have dedicated hardware for each network function. NFV improves scalability and agility by allowing service providers to deliver new network services and applications on demand, without requiring additional hardware resources.

Moreover, with NFV service, providers can run network functions on standard hardware instead of dedicated hardware. Also, because network functions are virtualized, multiple functions can be run on a single server. This means that less physical hardware is needed, which allows for resource consolidation that results in physical space, power, and overall cost reductions. NFV also gives providers the flexibility to run VNFs across different servers or move them around as needed when demand changes. This flexibility lets service providers deliver services and apps faster.

The NFV architecture proposed by the European Telecommunications Standards Institute (ETSI) is helping to define standards for NFV implementation. Each component of the architecture is based on these standards to promote better stability and interoperability.

Software-defined networking (SDN)

SDN separates network forwarding functions from network control functions with the goal of creating a network that is centrally manageable and programmable. NFV abstracts network functions from hardware. NFV supports SDN by providing the infrastructure on which SDN software can run.

An SDN typically has an SDN controller, northbound application program interfaces (APIs), and southbound APIs. The controller allows network administrators to view the network and dictate behaviours and policies to the underlying infrastructure. Southbound APIs take information about the state of the network from that infrastructure and send it back to the controller, which is necessary to keep the network running smoothly. Applications and services use northbound APIs to communicate their resource needs to the controller.

Content Delivery Network (CDN)

A content delivery network (CDN) refers to a geographically distributed group of servers which work together to provide fast delivery of Internet content.

A CDN allows for the quick transfer of assets needed for loading Internet content including HTML pages, javascript files, stylesheets, images, and videos. The popularity of CDN services continues to grow, and today the majority of web traffic is served through CDNs, including traffic from major sites like Facebook, Netflix, and Amazon.

At its core, a CDN is a network of servers linked together with the goal of delivering content as quickly, cheaply, reliably, and securely as possible. In order to improve speed and connectivity, a CDN will place servers at the exchange points between different networks.

These Internet exchange points (IXPs) are the primary locations where different Internet providers connect in order to provide each other access to traffic originating on their different networks. By having a connection to these high speed and highly interconnected locations, a CDN provider is able to reduce costs and transit times in high speed data delivery.

CDNs see a great opportunity to advance edge computing and application delivery capabilities to meet the needs of the new services enabled by URLLC, while tackling additional challenges in content routing, management, purging, and security.

Overall, the definition of CDN will be significantly broadened in the 5G era; not only will the forms of the distributed content, applications and services be greatly expanded, the network structure, business model and the value chain relationship can also be very different from what they are today.

Network Slicing

Network Slicing consists in reserving some defined section of a Telecom Operator network – a Network Slice Instances (NSI) - to a given client, as established in a contractual agreement between the two agents. This allows a client to have its own exclusive network space, thus assuring the necessary network requirements for its service to work successfully according to connection demands.

Network Slice Selection Function (NSSF) is a function is responsible for the management (including lifecycle) of NSIs. It derives network slice subnet related requirements from the network slice related requirements. NSMF communicates with the NSSMF and the CSMF. - The NSSF offers services to the Access and Mobility Management function (AMF) and NSSF in a different public land mobile network (PLMN) via the NSSF service based interface.

MANO

Management and orchestration (MANO) is a key element of the ETSI network functions virtualization (NFV) architecture. It is the ETSI-defined framework for the management and orchestration of all resources in a virtualized data center including compute, networking, storage, and virtual machine (VM) resources. The main focus of NFV MANO is to allow flexible on-boarding, sidestepping the chaos that can be associated with rapid spin-up of network components.

MANO is an architectural framework that coordinates network resources for cloud-based applications and the lifecycle management of virtual network functions (VNFs) and network services. As such, it is crucial for ensuring rapid, reliable NFV deployments at scale. MANO includes the following components: the NFV orchestrator (NFVO), the VNF manager (VNFM), and the virtual infrastructure manager (VIM).

NFV MANO is broken up into three functional blocks:

- **NFV Orchestrator**: Responsible for on boarding of new network services (NS) and virtual network function (VNF) packages; NS lifecycle management; global resource management; validation and authorization of network functions virtualization infrastructure (NFVI) resource requests.

- **VNF Manager**: Oversees lifecycle management of VNF instances; fills the coordination and adaptation role for configuration and event reporting between NFV infrastructure (NFVI) and Element/Network Management Systems.

- **Virtualized Infrastructure Manager (VIM)**: Controls and manages the NFVI compute, storage, and network resources.

RAT

A Radio Access Technology or (RAT) is the underlying physical connection method for a radio based communication network. Many modern mobile phones support several RATs in one device such as Bluetooth, Wi-Fi, and GSM, UMTS, LTE or 5G NR.

5G Core

The 5G Core is known as a Service-Based Architecture (SBA). At a high level, it consists of the User Plane, the Control Plane, and the Shared Data Layer Network Functions. This enables a more resilient core network (CN).

OSS/BSS

OSS/BSS refer to Operations Support System and Business Support System, respectively. The distinction emphasizes a separation of concerns between maintaining network operations and the business around which that network is built. Communications service providers support a broad range of services and functions with their OSS/BSS. BSS primarily consists of order capture, Customer Relationship Management and Telecommunications billing whereas OSS covers Order Management, Network Inventory Management and Network Operations.

5G RAN

In order for a cell phone to connect to a network or the internet, it connects first through a radio access network (RAN). A RAN provides radio access and assists to coordinate network resources across wireless devices. It utilizes radio transceivers to connect users to the cloud. Most base stations (aka transceivers) are primarily connected via fiber backhaul to the mobile core network.

This evolution of RAN for 5G will have a huge impact on wireless technologies, including enabling Mobile Edge Computing (MEC) and network slicing. These RANs of the future will also contribute to the lower latency that makes 5G so powerful.

MEC

Multi-access Edge Computing (MEC) moves the computing of traffic and services from a centralized cloud to the edge of the network and closer to the customer. Instead of sending all data to a cloud for processing, the network edge analyses, processes, and stores the data. Collecting and processing data closer to the customer reduces latency and brings real-time performance to high-bandwidth applications.

MEC characteristics include:

- Proximity
- Ultra-low latency
- High bandwidth
- Virtualization

MEC also offers cloud-computing capabilities and an IT service environment at the edge of the network, as it is typically implemented with data centers that are distributed at the edge. The MEC platform enables distributed edge computing by processing content at the edge using either a server or a CPE.

2.1.2 Micro Services Architecture

Container as a Service (CaaS)

Containers as a service (CaaS) is a cloud service model that allows users to upload, organize, start, stop, scale and otherwise manage containers, applications and clusters. It enables these processes by using either a container-based virtualization, an application programming interface (API) or a web portal interface. CaaS helps users construct security-rich, scalable containerized applications through on-premises data centers or the cloud. Containers and clusters are used as a service with this model and are deployed in the cloud or in onsite data centers.

As already explained, containers are a form of virtualization at the operating-system level. Containers share the running instance of the operating system hosting them. OS-level features provide isolation between containers so that one container cannot affect other running containers.

With CaaS, cloud providers offer a hosted container orchestration engine that is capable of running many containers, as well as maintaining the infrastructure among running containers. A couple of examples of this are Azure Kubernetes Service from Microsoft or Elastic Container Service from Amazon. CaaS offers cloud-based hosting independently of cloud provider. Containers provide a standard unit of software deployment therefore it is possible to deploy a container to a CaaS offering from provider A or from provider B. In either case, the containers are the same.

A container is a package of software that includes all dependencies: code, runtime, configuration, and system libraries so that it can run on any host system. Containers are an executable unit of software in which application code is packaged, along with its libraries and dependencies, in common ways so that it can be run anywhere, whether it be on desktop, traditional IT, or the cloud. CaaS enables software teams to rapidly deploy and scale containerized applications to high availability cloud infrastructures. CaaS differs from Platform as a Service (PaaS) since it relies on the use of containers. PaaS is concerned with explicit 'language stack' deployments like Ruby on Rails, or Node.js, whereas CaaS can deploy multiple stacks per container

CaaS is essentially automated hosting and deployment of containerized software packages. Without CaaS, software development teams need to deploy, manage, and monitor the underlying infrastructure that containers run on. This infrastructure is a collection of cloud machines and network routing systems that requires dedicated DevOps resources to oversee and manage.

Container runtimes offer configuration and virtualization of the operating system, allowing for advanced customization and control. Containers can be critical to the development of highly customized and specialized software.

A container system's primary goal is to avoid and ensure consistent behavior across underlying deployment environments.

Containers and CaaS make it much easier to deploy and compose distributed systems or micro service architectures. During development, a set of containers can manage different responsibilities or different code language ecosystems. The network protocol relationship between containers can be defined and committed for deployment to other environments. The promise of CaaS is that these defined and committed container architectures can be quickly deployed to cloud hosting.

Deploying containerized applications to a CaaS platform enables transparency into the performance of a system through tools like log aggregation and monitoring. CaaS also includes built in functionality for auto scaling and orchestration management. It enables teams to rapidly build high visibility and high availability distributed systems. In addition, CaaS increases team development velocity by enabling rapid deployments. Using containers insures a consistent deployment target while CaaS can lower engineering operating costs by reducing the DevOps resources needed to manage a deployment.

There are benefits to moving to a container-based application deployment. Containers are built up from an immutable image. This means that developers and testers are using the same image that gets deployed to production, without the risk of problems that come from having multiple environments with out-of-sync configurations. And, as mentioned earlier, containers have become a standard unit for software deployment. The fact that there are multiple cloud providers offering CaaS is a testament to this fact.

A CaaS offering brings a lot of advantages, including deployment standardization, provider independence, and improved monitoring. All of these can lead to faster and higher-quality application deployments, facilitating the introduction of new features to the application.

A model with broad application, CaaS helps developers streamline the process of constructing a fully scaled container and applications deployment. The model is a boon for IT departments, providing an enabled container deployment service that has governance control in a security-rich environment. The CaaS model helps enterprises simplify container management within their software-defined infrastructures.

Similar to other cloud computing services, users can choose and only pay for the CaaS resources they want. Some CaaS resource examples are compute instances, scheduling capabilities and load balancing.

An essential quality of CaaS technology is orchestration that automates key IT functions. Google Kubernetes and Docker Swarm are two examples of CaaS orchestration platforms. IBM, Amazon Web Services (AWS) and Google are a few examples of public cloud CaaS providers.

Enterprise clients from all industries are seeing the benefits of CaaS and container technology. Using containers provides increased efficiency and gives these clients the ability to quickly deploy innovative solutions for application modernization and cloud native development with microservices. Containerization helps these clients release software faster and promotes portability between hybrid and multicloud environments, and reduce infrastructure, software licensing and operating costs.

Containerization helps these release software faster and promotes portability between hybrid and multicloud environments, and reduce infrastructure, software licensing and operating costs.

Kubernetes and container orchestration

Kubernetes (K8s) is a container orchestration system for automating application deployment, management and scaling. Originally designed by Google and open sourced in 2014, Kubernetes is maintained by the CNCF. The Kubernetes website describes Kubernetes as a “portable, extensible open-source platform for managing containerized workloads and services that facilitates both declarative configuration and automation”.

The three primary platforms that Kubernetes functions as includes:

- Containers
- Microservices
- Portable Cloud

A container-centric management environment, Kubernetes coordinates computing, networking and storage infrastructure for user workloads. Kubernetes includes the same ease of use as a PaaS along with the malleability of IaaS and the portability across infrastructure providers.

a. ETSI NFV architecture

As a standard specification, ETSI focuses on high-level architecture, development guidelines, and interoperability enabled by open interfaces. Most of the specifications in ETSI Management and Network Orchestration Group Specification (MANO GS) continue serving the purpose when applying to the Cloud Native NFV. Nevertheless, augmentation is needed in some areas because of the differences between the VM based and Cloud Native solutions. [7]

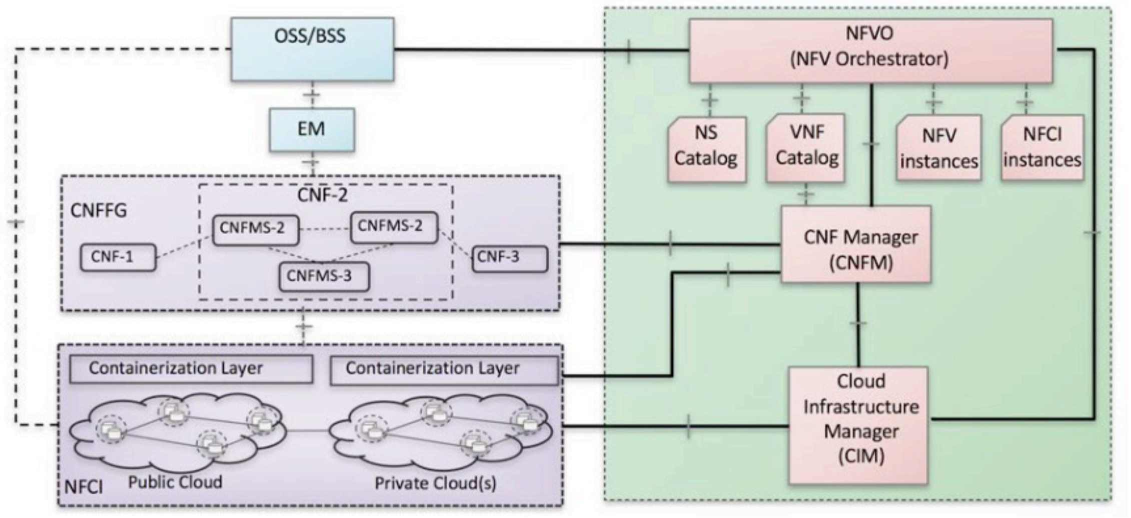


Figure 3 - ETSI MANO Cloud Network Function Orchestration Architecture. [13]

Cloud Network Function MicroService (CNFMS)

CNFMSs are the microservice containers from which a Cloud Network Function is composed.

Cloud Network Function (CNF)

CNFs are the network functions deployed in the cloud as microservices, usually in container format.

Network Function Cloud Infrastructure (NFCI)

NFCI provides the underline physical infrastructure for the network functions. This includes the hardware equipment for the computer, networking, storage, as well as the containerization layer on top of the hardware platform. CNFs are deployed on top of the NFCI.

Cloud Infrastructure Manager (CIM)

CIM is responsible for controlling and managing the NFCI compute, storage and network resources, as well as scheduling the microservice containers in the cloud. It manages the lifecycle of the containers in the cloud.

CIM also collects performance measurements in the infrastructure including container level and makes the data available for other functional blocks for monitoring purposes.

Other responsibilities of CIM include virtual networking control and management, as well as the southbound integration with various network controllers to achieve the physical network control and management capabilities.

Examples of the CIMs available in the market are Kubernetes, AWS ECS, and Docker Swarm.

Cloud Network Function Manager (CNFM)

CNFM focuses on the life cycle management of individual CNF instances. In the Cloud Native architecture, a CNF is usually composed of a set of containers that implement a network function. A CNF manager takes the responsibility of the management of the multiple container instances of the same network function. To control the lifecycle of the CNFs, CNFM works closely with CIM, which manages the lifecycle of the individual container of the CNF.

CNFM also serves as an overall coordination and adaptation role for configuration and event reporting between the CIM and the EM systems of traditional operator architectures.

NFV Orchestrator (NFVO)

The NFVO continues serving the responsibility of on-boarding a new Network Service (NS) composed of multiple CNFs, CNF forwarding graph, Virtual Links, and, as an option, Physical Network Functions (PNFs). The orchestrator also controls the life cycle of the Network Service including instantiation, scale-in/out or up/down, performance measurements, event correlation and termination. Further key operational functions are global resource management, validation and authorization of NFCI resource request, as well as policy management of Network Service instances.

Cloud Network Function Forwarding Graph (CNFFG)

The CNFFG contains a list of CNFs and the virtual links among the CNFs and the physical endpoints.

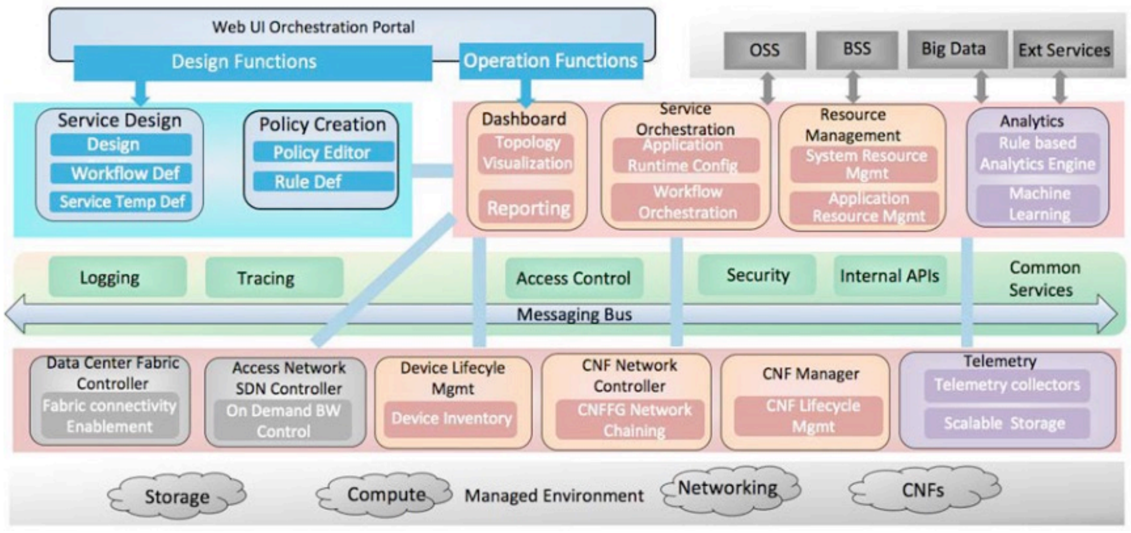


Figure 4 - A pragmatic NFV Software Architecture in the Cloud Native Environment. [13]

ETSI NFV architecture for Cloud Native environment contains the microservices for the NFV Management and Orchestration in both the design phase and the Runtime execution phase. The design phase generates the network services model to be deployed onto the target private or public cloud(s). The Runtime execution phase contains the microservices for the network service deployment, orchestration, lifecycle management, network control, monitoring, and analytics. Besides the functional services, there are a set of common infrastructure services for all the microservices containers in the Cloud Native architecture environment.

2.1.3 ETSI Network Function Virtualization

About ETSI

The **European Telecommunications Standards Institute (ETSI)** is a European Standards Organization (ESO) in the telecoms industry. It is a recognized regional standards body dealing with telecommunications, broadcasting and other electronic communications networks and services. This includes supporting European regulations and legislation through the creation of Harmonised European Standards. Only standards developed by the three ESOs (CEN, CENELEC and ETSI) are recognized as European Standards (ENs).

ETSI has been key in developing standards for information and communications technologies (ICT) in Europe. They took the lead on NFV in 2012, and subsequently created the Industry specification group (ISG) to provide a forum for the industry to collaborate on establishing the required standards and supporting the implementation of network virtualization.

About ETSI ISG NFV

The **ETSI Industry Specification Group for Network Functions Virtualization (ETSI ISG NFV)** is a group charged with developing requirements and architecture for virtualization for various functions within telecoms networks. Founded in November 2012 by seven of the world's leading telecoms network operators, ETSI ISG NFV became the home of Network Functions Virtualization (NFV). [2]

The ISG NFV community has evolved through several phases with its publications having moved from pre-standardization studies to detailed specifications. The early Proof of Concepts (PoCs) efforts have evolved and led to a series of interoperability events (NFV Plugtests). This large community is still working intensely to develop the required standards for NFV transformation incorporating latest technologies, as well as sharing their experiences of NFV implementation and testing in multi-vendor environments.

Building the Software-based Network

Telecoms networks are made up of an array of proprietary hardware devices. Launching a new service often means more devices, which means finding the space and power to accommodate those appliances. However, this has become increasingly difficult. Hardware-based devices now have shorter and shorter life cycles due to rapid innovation, making the return of investment lower and lower when deploying new services, as well as limiting innovation as the industry is driven toward network-centric solutions. Furthermore, modern telecoms networks contain an ever-increasing variety of proprietary hardware. The launch of new services often demands network reconfiguration and on-site installation of new equipment which in turn requires additional floor space, power, and trained maintenance staff.

In the digital world the innovation cycles accelerate and require greater flexibility and dynamism than hardware-based appliances allow. A hard-wired network with single functions boxes is tedious to maintain, slow to evolve, and prevent service providers from offering dynamic services.

In the same way that applications are supported by dynamically configurable and fully automated cloud environments, virtualized network functions (VNFs) allow networks to be agile and capable to respond automatically to the needs of the traffic and services running over it.

Network functions virtualization (NFV) focuses on addressing these problems. By evolving standard IT virtualization technology, NFV implements network functions into software so that it can run on a range of industry-standard server hardware and may easily be moved to various locations within the network as needed. With NFV, the necessity to install new equipment is eliminated. This results in lower CapEx and OpEx, shorter time-to-market deployment of network services, higher return on investment, more flexibility to scale up or scale down, openness to the virtual device network, as well as more opportunity to test and deploy new services with lower risk.

Another key enabling technology for this vision include SDN (Software Defined Networking). SDN and NFV are complementary but increasingly co-dependent. While the former provides the means to dynamically control the network and the provisioning of networks as a service, the latter offers the capability to manage and orchestrate the virtualization of resources for the provisioning of network functions and their composition into higher-layer network services.

ISG NFV has developed over 100 different specifications and reports for the virtualization of network functions, with focus on the management and orchestration of virtualized resources. From an architectural point of view, NFV specifications describe and specify virtualization requirements, NFV architecture framework, functional components and their interfaces, as well as the protocols and the APIs for these interfaces. Another set of NFV specifications define the structure and format of deployment templates and how to package all artefacts which are used by the NFV management and orchestration framework.

ISG NFV also studies VNF performance, reliability, and resiliency matters, analyzes the security challenges linked to virtualization (trust, attestation, regulation) and specifies associated requirements. In support for 5G deployments, the ISG NFV specifications include support for multi-site and multi-domain deployments, as well as network slicing. New virtualization technologies such as support for containerized VNFs and container infrastructure management are tackled in studies and on-going normative specifications work. In addition, The ISG NFV specifies requirements for hardware acceleration, multi-tenancy, autonomous networks, etc.

The ETSI ISG NFV helps by setting requirements and architecture specifications for hardware and software infrastructure needed to make sure virtualized functions are maintained. ETSI ISG NFV also manages guidelines for developing network functions. It routinely publishes its research findings to advance the industry's use of NFV.

a. Service Framework

With NFV the software is able to run on generic hardware, which enables networks to become more flexible and agile, and to be run more cheaply. [1] Traditional networks are comprised of network functions that are statically connected in a particular way so they provide a desired overall function. Virtualization allows connecting the network functions using dynamic methods as well as just static ones. However, multiple touch points for management are required, resulting in a more complex architecture than that of traditional networks. Additionally, future network services could be very different to current ones, as they will be made up of both non-virtualized and/or virtualized network functions. This will require legacy network domains to be able to operate in conjunction with NFV-based network domains.

In order for this to be possible, an architecture needed to be established that supported the virtualization of diverse sets of network functions, and encouraged the standardisation of NFV by addressing interoperability and enabling VNFs to be used in a way that is independent of the vendor supplying them.

As part of their work on defining requirements for NFV, ETSI established the reference architectural framework for NFV. The scope of the framework does not include the aspects of network functions/infrastructure that are common to Physical and Virtualized Functions, such as the specifics of the network functions, the control of the physical network infrastructure, or packet flow and control of the E2E (exchange-to-exchange) network service. Instead it focuses on the changes likely to occur in networks due to the NFV process i.e. the new functional blocks and reference points resulting from the virtualization of the network. Figure 5 shows a high-level view of the NFV framework, as established by ETSI

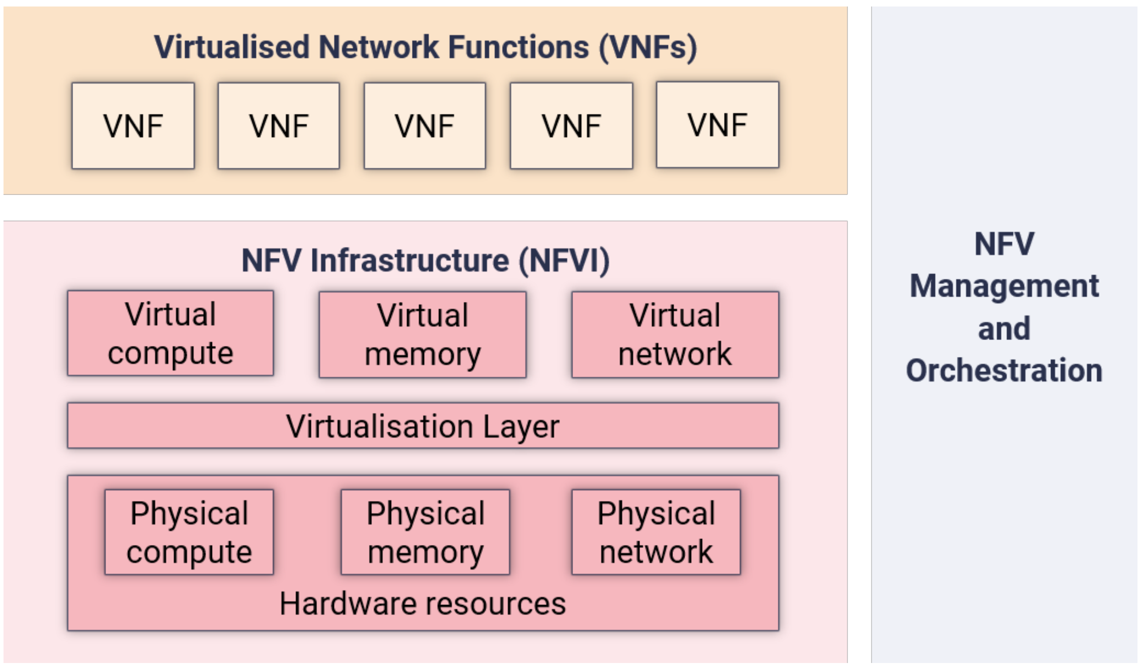


Figure 5. High-level ETSI NFV framework. [11]

ETSI identified three main working domains:

1. **Virtual Network Functions (VNFs)** - these are individual functions of a network that have been virtualized, i.e., software implementations of network functions that can be deployed on a network functions virtualization infrastructure (NFVI). Possibilities are endless: firewalls, Evolved Packet Core, etc.
2. **NFV infrastructure (NFVI)** - the infrastructure required to run the VNFs. This is made up of hardware resources (computing servers and network switches), and virtual resources ("abstractions" of the hardware on which the VNFs run, known as "virtual machines" (VMs). A virtualization layer (the "hypervisor") exists to abstract

between the two. NFVI is the totality of all hardware and software components that build the environment where NFVs are deployed. The NFV infrastructure can span several locations. The network providing connectivity between these locations is considered as part of the NFV infrastructure.

3. **Network functions virtualization management and orchestration architectural framework (NFV-MANO Architectural Framework)** - MANO is the framework for management and orchestration of all the resources in the NFV environment. It is where the management of resources in the infrastructure layer takes place and where resources are created and delegated, and allocation of VNFs is managed. MANO is the collection of all functional blocks, data repositories used by these blocks, and reference points and interfaces through which these functional blocks exchange information for the purpose of managing and orchestrating NFVI and VNFs.

The building block for both the NFVI and the NFV-MANO is the NFV platform. In the NFVI role, it consists of both virtual and physical processing and storage resources, and virtualization software. In its NFV-MANO role it consists of VNF and NFVI managers and virtualization software operating on a hardware controller. The NFV platform implements carrier-grade features used to manage and monitor the platform components, recover from failures and provide effective security – all required for the public carrier network.

Full Architectural Framework

For each of the three higher-level working domains, ETSI established individual functional blocks that each have a specific role. These are shown in the full NFV reference architectural framework in Figure 6. It depicts both the functional blocks and the main reference points between these blocks (shown as solid lines linking the blocks together). The functionalities that are necessary for the virtualization and operation of a network are included in the framework, but it does not stipulate which functions should be virtualized, as this is down to the owner of the network to decide.

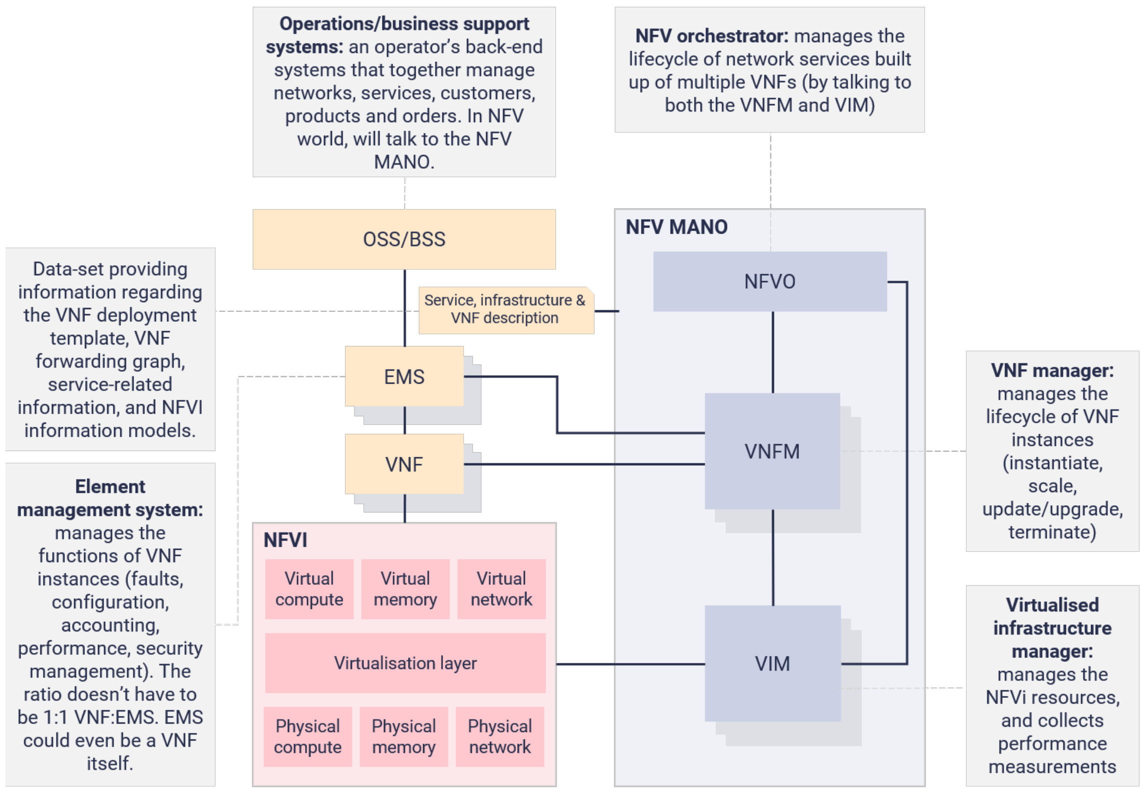

Operations/business support systems: an operator's back-end systems that together manage networks, services, customers, products and orders. In NFV world, will talk to the NFV MANO.
NFV orchestrator: manages the lifecycle of network services built up of multiple VNFs (by talking to both the VNFM and VIM)
OSS/BSS
NFV MANO
NFVO
Data-set providing information regarding the VNF deployment template, VNF forwarding graph, service-related information, and NFVI information models.
Service, infrastructure & VNF description
EMS
VNFM
VNF manager: manages the lifecycle of VNF instances (instantiate, scale, update/upgrade, terminate)
VNF
Element management system: manages the functions of VNF instances (faults, configuration, accounting, performance, security management). The ratio doesn't have to be 1:1 VNF:EMS. EMS could even be a VNF itself.
NFVI
Virtual compute
Virtual memory
Virtual network
Virtualisation layer
Physical compute
Physical memory
Physical network
VIM
Virtualised infrastructure manager: manages the NFVi resources, and collects performance measurements

Figure 6 - ETSI NFV reference architectural framework. [11]

Overall, defining the purpose of each functional block within the ETSI framework has played a key role in creating standards in virtualization, resulting in NFV solutions that are more homogenous and reusable and can take advantage of economies of scale. However there is still work to be done, and the architecture is continually being built upon as NFV continues to develop.

i. Service Based Architecture (5G-PPP)

5G-PPP

The 5G Infrastructure Public Private Partnership (5G PPP) is a joint initiative between the European Commission and European ICT industry (ICT manufacturers, telecommunications operators, service providers, SMEs and researcher Institutions). The goal of 5G PPP is to deliver solutions, architectures, technologies and standards for the ubiquitous next generation communication infrastructures of the coming decade. The challenge for the 5G Public Private Partnership (5G PPP) is to secure Europe's leadership in the particular areas where Europe is strong or where there is potential for creating new markets such as smart cities, e-health, intelligent transport, education or entertainment and media. The 5G PPP initiative will reinforce the European industry to successfully compete on global markets and open new innovation opportunities.

Some key challenges identified so far for the 5G Infrastructure PPP include:

- Providing 1000 times higher wireless area capacity and more varied service capabilities compared to 2010
- Saving up to 90% of energy per service provided. The main focus will be in mobile communication networks where the dominating energy consumption comes from the radio access network
- Reducing the average service creation time cycle from 90 hours to 90 minutes
- Creating a secure, reliable and dependable Internet with a “zero perceived” downtime for services provision
- Facilitating very dense deployments of wireless communication links to connect over 7 trillion wireless devices serving over 7 billion people
- Ensuring for everyone and everywhere the access to a wider panel of services and applications at lower cost

Service Based Architecture

At the core of most modern networks and services is typically a cloud- and virtualization-based platform. This is also the case for 5G networks. These platforms are programmable, and allow many different functions to be built, configured, connected, and deployed at the scale that is needed at the given time. One example of such a platform is OPNFV, an open source project supporting Network Function Virtualization. [4]

But the demand for easily scalable systems that can be tailored for new situations does not stop at the platform. The ability to develop new functions easily, time-to-market, and use of off-the-shelf technology also drive changes in the network functions themselves. The goal is to migrate from telecom-style protocol interfaces to web-based APIs. The 5G core network will thus be based on what is called a “Service-Based Architecture” (SBA), centered around services that can register themselves and subscribe to other services. This enables a more flexible development of new services, as it becomes possible to connect to other components without introducing specific new interfaces. The new system architecture is specified in 3GPP technical specification 23.501.

In addition, the use of APIs based on web-based technology makes development easy, as libraries, development tools, specification tools, code generators, security mechanisms and many other components are broadly available. Not to mention the broad experience many programmers have in using them.

It was decided to employ the new web protocol, HTTP/2 from IETF, design interfaces according to the REpresentational State Transfer (REST) principle where possible, and use JavaScript Object Notation (JSON) as a data format and OpenAPI as the interface definition language.

b. Service Oriented Architecture (SOA)

SOA

Service-oriented architecture (SOA) is a style of software design where services are provided to the other components by application components, through a communication protocol over a network. A SOA service is a discrete unit of functionality that can be accessed remotely and acted upon and updated independently. SOA is also intended to be independent of vendors, products and technologies. [3]

A service has four properties according to one of many definitions of SOA:

- It logically represents a business activity with a specified outcome.
- It is self-contained.
- It is a black box for its consumers, meaning the consumer does not have to be aware of the service's inner workings.
- It may consist of other underlying services.

Different services can be used in conjunction to provide the functionality of a large software application, a principle SOA shares with modular programming. Service-oriented architecture integrates distributed, separately maintained and deployed software components. It is enabled by technologies and standards that facilitate components' communication and cooperation over a network, especially over an IP network.

SOA is related to the idea of an application programming interface (API), an interface or communication protocol between different parts of a computer program intended to simplify the implementation and maintenance of software. An API can be thought of as the service, and the SOA the architecture that allows the service to operate.

Service-Oriented Architecture (SOA) is a style of software design where services are provided to the other components by application components, through a communication protocol over a network. Its principles are independent of vendors and other technologies. In service oriented architecture, a number of services communicate with each other, in one of two ways: through passing data or through two or more services coordinating an activity. This is just one definition of Service-Oriented Architecture.

Characteristics of Service-Oriented Architecture

While the defining concepts of Service-Oriented Architecture vary from company to company, there are six key tenets that overarch the broad concept of Service-Oriented Architecture. These core values include:

- Business value
- Strategic goals
- Intrinsic inter-operability
- Shared services
- Flexibility
- Evolutionary refinement

- Each of these core values can be seen on a continuum from older format distributed computing to Service-Oriented Architecture to cloud computing (something that is often seen as an offshoot of Service-Oriented Architecture).
- Service-Oriented Architecture Patterns
- Image for post

There are three roles in each of the Service-Oriented Architecture building blocks: service provider; service broker, service registry, service repository; and service requester/consumer.

The service provider works in conjunction with the service registry, debating the whys and hows of the services being offered, such as security, availability, what to charge, and more. This role also determines the service category and if there need to be any trading agreements.

The service broker makes information regarding the service available to those requesting it. The scope of the broker is determined by whoever implements it.

The service requester locates entries in the broker registry and then binds them to the service provider. They may or may not be able to access multiple services; that depends on the capability of the service requester.

c. Service Based Interface (SBI)

SBI is the term given to the API based communication that can take place between two VNFs (Virtualized Network Functions) within the 5G SBA (Service Based Architecture). A given VNF can utilise an API call over the SBI in order to invoke a particular service or service operation.

Each Network Function service exposes its functionality through a Service Based Interface (SBI), which employs a well-defined REST interface using HTTP/2. To mitigate issues around TCP head-of-line (HOL) blocking, the Quick UDP Internet Connections (QUIC) protocol may be used in the future.

2. ETSI NFV Reference Architecture for Cloud Native Environment

Cloud Native is a new approach for developing and running applications in a virtualized cloud environment. The basis of this approach is to break down a monolithic application into small microservices and deploy them as containers in the cloud. One of the advantages of doing so is that applications can be tested in an iterative and distributed model, without taking applications offline. In the cloud world, large scale applications have been developed, tested, and deployed with more agility using this distributed model. [12]

Since 2016, several large service providers have publicly embraced the move to a microservices architecture in the telco cloud. There have been announcements from major service providers to use containers to build out their network function virtualization infrastructure. Some key telecommunications equipment suppliers are using microservices to implement some of their software. Open-source initiatives are moving towards microservices and containers. In NFV space, there is a trend of moving from the virtual appliance based solutions to the cloud native approach, which is referred to as the Cloud Native NFV.

The cornerstone principles of cloud native are:

- Applications are "carved up" into smaller units called microservices. Application monoliths are replaced by a constellation of smaller, inter-connected microservices.
- Containers house the microservices and provide a run-time environment. Containers dispense with VM overhead, and instead just package up the application code, binaries and dependencies. Containers share a guest and/or host OS/kernel.
- Orchestration provides complete container lifecycle management including scheduling, placement, start/stop/re-start and visibility. Kubernetes is the cloud native orchestration platform.
- In the Kubernetes environment, one or more containers are grouped into pods that run on physical or virtual hosts. A pod is the smallest unit of work managed by Kubernetes. A collection of hosts and pods managed by Kubernetes is referred to as a cluster.
- In addition, cloud native embraces the notions of open source, 12-factor app, and devops CI/CD pipelines.

These principles are encapsulated in the Cloud Native Compute Foundation (CNCF) definition:

"using open source software stack to be containerized, where each part of the app is packaged in its own container, dynamically orchestrated so each part is actively scheduled and managed to optimize resource utilization, and microservices-oriented to increase the overall agility and maintainability of applications."

2.1.4 ETSI NFV Reference Architecture for Cloud Native Environment

Cloud Native is a new approach for developing and running applications in a virtualized cloud environment. The basis of this approach is to break down a monolithic application into small microservices and deploy them as containers in the cloud. One of the advantages of doing so is that applications can be tested in an iterative and distributed model, without taking applications offline. In the cloud world, large scale applications have been developed, tested, and deployed with more agility using this distributed model.

Since 2016, several large service providers have publicly embraced the move to a microservices architecture in the telco cloud. There have been announcements from major service providers to use containers to build out their network function virtualization infrastructure. Some key telecommunications equipment suppliers are using microservices to implement some of their software. Open-source initiatives are moving towards microservices and containers. In NFV space, there is a trend of moving from the virtual appliance based solutions to the cloud native approach, which is referred to as the Cloud Native NFV.

The cornerstone principles of cloud native are:

- Applications are “carved up” into smaller units called microservices. Application monoliths are replaced by a constellation of smaller, inter-connected microservices.
- Containers house the microservices and provide a run-time environment. Containers dispense with VM overhead, and instead just package up the application code, binaries and dependencies. Containers share a guest and/or host OS/kernel.
- Orchestration provides complete container lifecycle management including scheduling, placement, start/stop/re-start and visibility. Kubernetes is the cloud native orchestration platform.
- In the Kubernetes environment, one or more containers are grouped into pods that run on physical or virtual hosts. A pod is the smallest unit of work managed by Kubernetes. A collection of hosts and pods managed by Kubernetes is referred to as a cluster.
- In addition, cloud native embraces the notions of open source, 12-factor app, and devops CI/CD pipelines.

These principles are encapsulated in the Cloud Native Compute Foundation (CNCF) definition:

“using open source software stack to be containerized, where each part of the app is packaged in its own container, dynamically orchestrated so each part is actively scheduled and managed to optimize resource utilization, and microservices-oriented to increase the overall agility and maintainability of applications.”

2.1.5 Cloud Network Function (CNF)

A cloud native network function (CNF) is a network function designed and implemented to run inside containers. CNFs inherit all cloud native architectural and operational principles including Kubernetes (K8s) lifecycle management, agility, resilience, and observability. A CNF implementation should strive to meet these requirements. [5]

CNF capabilities can be embedded in application pods. A CNF is just another pod, albeit with specialized network functionality. CNFs are versatile and can be implemented with a single purpose in mind, an example being a load balancer. They can also be designed and built for programming and executing specific data plane functions.

Some of the advantages of CNFs include:

- Lifecycle management parity with application containers.

- Stateless Configuration. CNFs operate in pods. Pods are durable, but if something breaks or new functions are needed, then pods can be torn down and a new instance started up.
- Smaller footprint vis-a-vis physical or virtual devices. This reduces resource consumption, with the potential savings allocated to applications and/or infrastructure expansion.
- Rapid development, innovation and immutability.
- Performance. CNF processing in user space, NIC hardware control, multi-core tuning and kernel bypass all contribute to maximum throughput and resource efficiency.
- Agnostic to host environments. Bare-metal and VMs are the most common. Since VMs do not add value to CNFs, bare-metal could become the preferred host of choice.

a. Best Practices

CNFs receive at no cost all existing best practices bestowed to application pods. These include: development environments, toolchains, CI/CD, K8s orchestration and scheduling, 12-factor app, distributed management, logging and telemetry streaming.

The dynamic nature of cloud native environments precludes centralized or stateful CNF configuration management. Instead, a stateless approach is employed where CNFs “watch” for and then process configuration updates stored in an external data store. Another lightweight option uses a service request to trigger a local gRPC call that pushes configuration state into the CNF data plane.

CNF functions should run in user space where it is easier, faster and less risky to develop and deploy new features and innovations. There is no need to interact with the stable and mature Linux kernel used by other components in the environment.

A cloud-native architecture has many benefits. The following sections capture the primary benefits and best practices for applying cloud-native principles to CNFs.

Distributed microservices

Microservices are componentized, reusable software modules enabling a number of benefits for the customer and the development organization. Microservices expose smaller discrete functions to an application through APIs. APIs are maintained and versioned and promote reuse of microservices in other applications.

Lightweight footprint

Containers are a way of virtualizing an application process or set of processes and are inherently lightweight because, unlike a virtual machine, the OS is shared across containers. Significant performance improvements can be realized when starting and upgrading containers during lifecycle operations. Containers may be deployed on bare metal with a basic Linux OS or can be deployed on virtual machines residing on

COMPETE 2020 PORTUGAL 2020 UNIÃO EUROPEIA Fundo Europeu de Desenvolvimento Regional FCT Fundação para a Ciência e a Tecnologia Carnegie Mellon Portugal

top of a hypervisor. Although some of the benefits of containers are limited when running on virtual machines, a majority of the instances do not require the virtual machine to be upgraded for lifecycle events. For example, upgrading the software containers in a lifecycle event do not require the virtual machines to be upgraded.

Service discovery

Service discovery is one of the primary components of the cloud-native stack and is used to provide a real-time service registry for all available services. The service registry enables new services to be dynamically orchestrated into an application. Services are automatically scaled and recovered via Kubernetes if a service becomes unavailable and the service must be restored.

Lifecycle management

One of the primary benefits of moving to containerized microservices is the ability to orchestrate the containers so that separate lifecycle management processes can be applied to each service. This allows for each service to be versioned and upgraded singularly as opposed to upgrading the entire application or virtual machine image. When upgrading an application, the container scheduler determines which individual services have changed and deploys only those specific services into the broader application. When the application is implemented with the appropriate level of state separation, this process allows for fully automated in-service upgrades and rollback of the containers that make up the application.

State separation

One of the most commonly agreed-on design patterns in cloud-native applications is a clean separation of stateful (also known as backing services) and stateless services. Application services containing functional logic (database, file system, or cache) should be separated from stateful services. For example, a service handling a create session request implements the logic for creating the session but stores the session information in a separate stateful service, which physically stores the session to memory or disk. This allows for the stateless application services to be autonomous, lightweight, upgradable, recoverable, and rapidly scalable.

Stateful services are more challenging because of where and how the state is actually stored: for example, on the file system, in memory, or in a cloud storage file system. Stateful services must address the availability, consistency, and portability of state, which typically requires replication across one or more containers while making sure that consistency of the state is maintained.

Availability and resiliency

Cloud-native applications inherently provide support for high availability and resiliency through service discovery and load-balancing transactions across stateless application containers. In addition, because containers are lightweight, recovery times are far less than when recovering a virtual machine, physical box, or application as a whole. This allows for faster and more granular ways of responding to failure events.

High availability cannot be solved by container orchestration alone, and, in most cases, the application itself has resiliency requirements. Stateful services, such as a resilient database, require resiliency beyond the inherent features of a cloud-native architecture, which requires state synchronization and data integrity. Additionally, protocol services require specific failover and availability mechanisms defined at the protocol level.

Operational benefits

Fundamentally, containerized applications running on bare metal perform better than those running on virtual machines because there is not the overhead of a hypervisor. Because of the lightweight footprint of containers, the speed of instantiating or recovering services is optimized. Because virtual machine instantiation includes an underlying OS and disk resources, the provisioning process can take minutes, whereas a container instantiation can take seconds. When containers are deployed on top of virtual machines—for example, in a CNF architecture—and the hypervisor overhead is still present, there are still a number of operational benefits because containers have a separate lifecycle from virtual machines. For example, a software upgrade or recovery might not require the instantiation of a new virtual machine. Therefore, because containers are lightweight, the benefits of starting, recovering, and upgrading services are substantially faster.

Scalability

A containerized architecture enables the ability to scale each microservice independently. Each container is monitored based on KPI metrics, enabling the orchestration scheduler to scale/descale individual containers. As new containers are started up for scaling, they register themselves in the service discovery layer and are automatically orchestrated into the broader application. Load balancing is used to transparently add new container instances without affecting the containers that depend on that container.

i.Design Characteristics

An alliance of telcos and vendors launched the Common NFVi Telco Task Force (CNTT) for the purposes of simplifying NFV standards. The NFVi reference architecture is dependent on OpenStack as containers can run on an OpenStack infrastructure, and vice versa. [6]

As described previously, NFV replaces network hardware appliances with software -- including virtual network functions (VNFs) -- that runs on virtual machines (VMs) running on commodity hardware. Moreover, CNFs are like VNFs, but they run on lighter-weight containers, providing greater agility and ease of deployment compared with VMs. That lightweight implementation is useful throughout the network, but particularly so for edge applications and 5G network slicing.

As described previously, moving network functions to containers seems to be a logical step forward in the evolution of network architecture with Cloud-native Network Functions (CNFs) being the term coined for this technology. CNFs, network functions like routers or firewalls, moved inside Linux containers orchestrated by tools such as Kubernetes. This means that, in a sense, Kubernetes is managing network traffic. Thus, containers need support for high network throughput, security, isolation and other telco needs.

In this approach, software is distributed as a container image and can be managed using container orchestration tools (e.g. Docker images in Kubernetes environment). Examples of CNFs include:

- Dynamic Host Configuration Protocol (DHCP) service is using database (DB) service to keep the state in a DB layer
- VPN as userspace software--secure tunnel between different clusters
- Lightweight Directory Access Protocol (LDAP) service as a source of authorization/authentication for an application, and also for storing its data in a DB layer by using DB service

Furthermore, microservices for control plane are used across a number of different applications. Microservice APIs are typically exposed over a RESTful interface or via a message bus, which allows each service to choose the best technology available for client operations. For example, Java can be used for a control-plane service, and Go can be used for data-plane services. This componentization allows open-source technologies to be more easily integrated into the application or swapped for different technologies as the application evolves.

CNFs, when running inside telecommunications premises, form a private cloud, and the same public cloud principles can be effectively used. CNFs cover all branches of the service provider market, including cable, mobile, video, security, and network infrastructure.

Virtualization and VNFs helped us in getting started in moving toward cloud-native applications. Virtualization offered software models with increased flexibility with hardware dependencies eliminated. However, there are limitations in that VNFs upgrades are slow, restarts take a long time, CLI is still the main interface, software was typically a lift and shift operation, hypervisors such as OpenStack were hard to install, there was little elasticity, and scaling was problematic. Cloud-native applications address these limitations. Cloud-native applications in general have these characteristics:

- Developed using microservices architecture (that is, 12-factor apps)
- Managed by Kubernetes-style orchestration

- Built-in microservice discovery mechanisms
- Supports dynamic elasticity and scale
- Resilient services
- Improved feature velocity
- Smaller footprint with fast restart
- Continuous deployment and automation principles
- Consistent lifecycle management across containers
- Modern health and status telemetry

Primary cloud-native constructs

A cloud-native application utilizes a collection of tools that manage and simplify the orchestration of the services that make up the application. These services, each with its own lifecycle, are connected by APIs and are deployed as containers. These containers are orchestrated by a container scheduler, which manages where and when a container should be provisioned into an application and is responsible for lifecycle management. Cloud-native applications are designed to be portable to different deployment environments: for example, in a public, private, or hybrid cloud. Continuous delivery and DevOps are methods used to automate the process of building, validating, and deploying services into a production network.

Microservices

An architectural style that structures an application as a collection of loosely coupled services to implement business capabilities. Microservices are commonly deployed in containers and enable the continuous delivery and deployment of large, complex applications. Each microservice can be deployed, upgraded, scaled, and restarted independently of other services in the application as part of an automated system, enabling frequent updates to live applications without affecting end customers.

Containers

Containers are another form of virtualization, using Operating System (OS)–level virtualization. A single OS instance is dynamically divided among one or more isolated containers, each with a unique writable file system and resource quota. Containers can be deployed on both bare metal and virtual machines. Containers deployed on bare metal offer performance benefits to virtual machines by eliminating the hypervisor overhead. Although each microservice is commonly deployed in a separate container, multiple microservices may be deployed per container to address application and performance requirements, for example, when colocation of services logically simplifies the design or when services fork multiple processes in a container.

Continuous delivery

Makes an individual application change ready for release as soon as it is ready, without waiting for bundling with other changes into a release or an event such as a maintenance window. Continuous delivery makes releases easy and reliable, so organizations can deliver frequently, at less risk, and with immediate feedback from end users. The way service providers consume software this frequently will revolutionize speed to market. Eventually, deployment becomes an integral part of the business process and enterprise competitiveness, taking advantage of canary and A/B testing in the real world rather than artificial labs.

DevOps

DevOps is the utilization of lean and agile techniques to combine development and operations into a single IT value stream. DevOps enables organizations to build, test, and release software more rapidly and iteratively by applying continuous integration and delivery. For example, DevOps enables the automation of deploying and validating a new software feature in an isolated production environment, which can then be rolled out more broadly into production after it has been proven. To fully realize DevOps, service providers must adopt cloud-native techniques, establish automated continuous integration, and deliver pipelines with its vendors.

Container Networking Requirements

One aspect of K8s networking involves inter-pod connectivity provided by the Kubernetes Container Network Interface (CNI) specification. There are multiple CNI network plugins for establishing different forms of virtual L2/L3 overlay networks between pods. CNI solutions are quite suitable for web traffic workloads. However, they are not ideal for enabling services that leverage VNF-like functionality, process compute-intensive traffic workloads, or exploit the agility and dynamics of cloud native environments.

A new set of requirements emerge for augmenting basic container networking with VNF-like capabilities: NFs adapted for cloud native deployment and operations. This assumes a container (pod) form-factor under K8s orchestration control, or a control/management plane co-existing or compatible with K8s. The latter is useful if pods need to be wired up into a mesh or chained topology.

2.1.6 Cloud Native Network Function (CNFM)

Cloud Native Computing Foundation (CNCF), which is an open source software foundation under the umbrella of Linux Foundation (LF), defines Cloud Native as: Cloud Native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative Application Programming Interfaces (API) exemplify this approach. [7]

CNCF provides structure and constraints for being Cloud Native, requiring that the applications being developed and architected use microservices methodologies to be deployed as Containers. Traditional applications when broken down into small, reusable components are referred as “microservices.” To be properly classified as Cloud Native, microservices need to be “stateless” meaning there has to be separation of processing from the associated data and the storage of the data in Cloud. Dynamic Orchestration is another critical pillar of being Cloud Native. That’s where the CNCF-hosted Open Source container orchestrator “Kubernetes” plays a crucial role in actively scheduling and optimizing resource utilization while providing observability, resiliency and an immutable infrastructure

3 5G Fraud Landscape

3.1.1 5G Use Cases

There are three main use-cases associated with 5G. These are:

1. Ultra-low latency (ultra-reliable low latency (URLLC))

2. Massive implementation (mIoT)

3. Connection speed (enhanced Mobile Broadband (eMBB))

Ultra-low latency

Ultra-low latency in a 5G scenario is considered to be a latency not greater than 1-millisecond. This is a great improvement over previous standards (50-millisecond with 4G).

This is of special importance if we consider the ever increasing number of devices (IoT) which will be connected to a given network at any given time. Moreover, many features and functionalities of such devices require extreme low latencies in order to function properly. A good example are self-driving cars, for which delays in the update of traffic information can easily lead to tragic consequences.

Massive implementation

With the advent of the Internet of Things, one of the consequences of the current digital transformation era, it is expected that the number of devices with an Internet connection will continue to markedly rise within the next coming years. Eventually, we can expect that everything will be connected, everything will be a sensor: collecting, sharing, and importing information from the network and other devices. This means that everywhere there will be a Massive Implementation of new devices and technologies connected to the Network, which will create a lot of strain to Network providers to provide ubiquitous Internet access to all these devices and in a way that guarantees that all requirements are met, be them speed of connection, broadband width, or latency to name a few.

Connection speed

As previously mentioned, 5G will allow for much higher connection speeds as its technological predecessors. It can be noted that 5G will aim to provide 20 times the peak data rate (speed).

In terms of spectrum bands earmarked for deployment of 5G, they can be sub-divided in three macro categories: sub-1GHz, 1-6GHz and above 6GHz. Sub-1GHz bands are suitable to support IoT services and extend mobile broadband coverage from urban to suburban and rural areas. This is because the propagation properties of the signal at these frequencies enable 5G to create very large coverage areas and deep in-building penetration. The 1-6GHz bands offer a reasonable mixture of coverage and capacity for 5G services. There is a reasonable amount of existing mobile broadband spectrum identified with this range which could be used for initial 5G deployments. Spectrum bands above 6GHz provide significant capacity thanks to the very large bandwidth that can be allocated to mobile communications and thus enable enhanced mobile broadband applications.

3.1.2 AI/ML Fraud Detection (Federated Edge Learning)

A distributed approach is needed to address the real time and massive volumes of information expected in 5G, additionally privacy and regulation prevents some of the data to exit the edge where it was collected and be processed in a central cloud. Collaborative Machine learning addresses technical and lawful restrictions. Using federated learning, it is possible for an edge to download or update a model and process the data locally. The model trained locally can be sent to a central server where it will be calibrated and consolidated, improving the global model, which is sent back to edge devices. A major issue in distributed training is the limited communication bandwidth between contributing nodes or prohibitive communication cost in general. The system must employ methods for reducing the communication cost for distributed training, including techniques of communication delay and gradient sparsification as well as optimal weight update encoding.

3.1.3 Fraud and Security Synergy

The building blocks of 5G are based in virtualization, Edge Computing and Multi Access Edge Computing (MEC). There are several threats that fraudster can explore directly on the infrastructure, hence collaboration with the security area is essential to minimize it. A risk factor unique to virtual environments is the hypervisor, which is the software and/or firmware responsible for hosting and managing Virtual Machines (VMs). It provides a single point of access into the virtual environment for hackers and/or fraudsters to target and it represents a potential single point of failure. A misconfigured hypervisor can result in a single point of compromise of the security of all its hosted components. The compromise of a single point of access presents fraudsters with an opportunity to obtain vital information. The likelihood and potential for these large-scale attacks vastly increases the need for these risks to be fully considered in fraud threat mapping activities. Edge computing is the performance of data processing and analytics operations on, or near, the devices that produce or consume that data. The data processing occurs at the “edge” of the cloud, rather than on central cloud servers, minimising the need for data transfers and optimising performance associated with the processing of large amounts of data. However, edge computing does involve some risks. It is anticipated there could be an increased risk of physical theft and direct attacks on interfaces due to the increased number of third parties in 5G, which will see more technology being placed in remote locations. Fraudsters can harvest data through unauthorised malicious third party applications. Several threats can be expected on the core network due to the 5G deployment models. As the equipment is (broadly) in the non-telco space, it is, from an MNO point of view, an untrusted environment where various security attacks could happen.

3.1.4 5G Fraud Threat Model and Fraud Use Cases

The 5G use cases, together with technology enabling them are the key inputs for the 5G Fraud Threat Model. According to the 5G Fraud Risks Guide FS39 from GSMA, “5G Fraud is perpetrated where a process, control or technical weakness, primarily relating to 5G network or service deployment, is intentionally exploited, resulting in a financial, other loss or damage”. 5G will bring significant changes in the fraud domain, at a high level, it is expected that those core changes are will be:

- The fraud types will be the same, yet new fraud methods/enablers will be used to accomplish them. Current systems will fail to detect new fraud enablers or variants that are dependent on 5G technologies.
- Fraud based on Artificial intelligence to simulate human behaviour or explore statistical blind spots avoiding detection, e.g.: low and slow increases.
- Massive increase in number of connected devices and consequently large volume of data with real time detection requirements will force fraud systems to be more based on prevention than detection. Cooperation with other areas like security and adoption of signalling as the main data source for fraud detection will be essential.
- Stakeholders and partners related to IoT and industries that are embracing the new 5G capabilities and use cases (healthcare, automotive, manufacturing, …) will be part of the fraud management process requiring new identification methods, controls, monitoring and fraud risk management framework.

The attack surface, risk vectors and new stakeholders of the 5G network is considerably higher than the previous generations forcing MNO’s to review the current fraud threat mapping. In a network where all network elements are virtualized, exploring a virtual layer vulnerability can create a single point of failure and an opportunity to succeed for fraudulent activities. Additionally, in 5G networks, the data processing occurs at the “edge” of the cloud, rather than on central cloud servers, minimising the need for data transfers and optimising performance associated with the processing of large amounts of data. However, edge computing does involve some risks. It is expected an increased risk of physical theft and direct attacks on interfaces due to the increased number of third parties in 5G. With several deployment models, 5GS technology is being placed in multiple remote locations outside the control of the MNOs. Several threats can be expected on the core network due to the 5G deployment models.

In a context of IoT solutions, Insecure Provision, Single Identity for subscription aggregation, device life cycle management together with unauthorized modification of device identities and exploitation of IoT devices weakness for billing fraud purposes, represent the higher risks to be addressed. With this landscape in mind, the following use cases for a 5G fraud solution were identified.

- Service Disruption Detection
- Platform Service Abuse
- OTT Service Abuse
- Platform Capacity Planning
- Customer Experience Segmentation
- OTT Services Retention
- Platform Services Retention

A general overview for each of the identified use-cases is provided next.

Use Case 1 - Service Disruption Detection

A service disruption may have different causes, in particular when a CSP provides the platform and a partner provides the OTT services. Therefore it is important to monitor the overall system, to ensure the platform is working as expected, that the usage evolution is also under the expected parameters, and that no capacity planning will happen in the short term.

Capacity, which in terms of platform means the availability to connect to the service, and the availability of the contracted bandwidth, latency and resilience.

On the other hand it is important to monitor the availability and supply of the OTT service in accordance with the contracted parameters with the partner, especially in the event the service is not hosted in the CSP's infrastructure. It is also important to ensure the interface points are working as expected (e.g. the credential validation interface).

Use Case 2 - Platform Service Abuse

An abusive usage of a service means that a CSP is not getting the value for the service being provided. It can occur due to different reasons, such as design problems in the offer, processes, or in the platform infrastructure, that supports the service. Regardless of the subscriber acting intentionally or not, it is key to detect these situations and act on them as soon as possible.

There is the concern that in a distributed platform like 5G EDGE this risk can be even higher than in traditional technological platforms. Moreover, as it is impossible to foresee all the scenarios, it is important to have a process that can monitor usage to detect this type of situations and alert the proper areas for them to act. In many situations, some pre-defined actions can be executed immediately to prevent losses or service disruption.

Due to the distributed architecture we foresee a bigger challenge as the required data to detect these situations may reside only at the edge.

Use Case 3 - OTT Service Abuse

An improper or abusive usage of an OTT Service, even if a partner provides the service, is most likely to cause losses to both the CSP and the OTT Service provider. In addition, even if the OTT Service provider has controls for these types of situations there is information at the platform level that can allow detecting scenarios that are difficult or even impossible to detect by the OTT Service provider itself. It is therefore key to put in place controls at the CSP side that can monitor specific OTT Services for improper usage, either because someone is accessing the service without intention to pay or without having the service contracted, or because, due to a flaw in the process or supporting services architecture, platform subscribers are able to access the service in such a way they shouldn't be able, or using it in an improper manner, like service reselling or redistribution, content extraction, etc.

It is therefore crucial to have a monitoring service to prevent these type of situations.

Use Case 4 - Platform Capacity Planning

To ensure a proper service delivery it is important to have a solid, data driven and continuously updated capacity planning for the Platform Infrastructure. CSPs have different technologies (3G, 4G, 5G, 5G Edge) able to deliver OTT Services and need to have a clear view on the existing capacity in each of them, especially in a transition phase where these technologies should be used to serve subscribers, according with their contracted terms, devices, and the technological platform existing capacity. It does not make sense to migrate a subscriber to a 5G Edge platform if its device does not support it or if there is no capacity in his regional area. On the other hand, it does not make sense to keep a congested 4G platform if investments were already done and there is 5G capacity available in the region.

It is therefore important to have a service in the Assurance Architecture that can collect data about the existing capacity in each of the existing technological platforms, as well as the corresponding levels and patterns of use. With this information, a capacity forecast should be continuously updated, with predicted or existing pain points adequately identified. A recommendation should be made on Subscribers and OTT services that are good candidates to be migrated to a different platform.

Use Case 5 - Customer Experience Segmentation

A proper customer experience segmentation that can classify a subscriber according with his Service Experience - based on past behavior, usage and paying data - is key to manage Customer loyalty and prevent the risk of churn while measuring the satisfaction of the subscriber community served by a specific network segment or specific service.

With an EDGE architecture we foresee the need to have “local” lightweight segmentation algorithms that can group subscribers according with their past behavior, taking into account what is being done at the moment in order to be able to serve the subscribers the best way possible, including preparing the infrastructure capacity according with the split of segments in a particular location.

It is important to bear in mind that sensible information may exist at the edge that does not exist in the centralized server, therefore the importance of a hybrid approach to determine the subscriber segment.

Experience Segmentation estimates the overall experience level that the subscriber is having.

Use Case 6 and 7 - OTT and platform Services Retention

OTT Services Churn is the event of a subscriber stop using a particular OTT service. An example is a subscriber using Amazon Prime or Netflix canceling the service. There is also the particular case of abandoning the buying process in a pay per view service. There are many reasons behind such a decision and in many of them the event can be predicted, based on the usage pattern change, experience issues, etc.

Knowing the event in advance allows retention measures to be taken. They can be point measures like reminding the subscriber he has some credits expiring, a task that can be fully automated, or a retention campaign involving a Customer service.

Especially for the first type of situations it is key to be able to monitor in real time the content browsing behavior and the subscriber’s cart, in a similar way to any E-Commerce service. This type of monitoring, especially if involving sensible information, should ideally be performed at the EDGE.

In order to successfully cover these use-cases in the new RAID platform there are functional requirements that need to be met. Below we describe the requirements identified for the use-cases described before.

3.1.5 5G Fraud Use Case Requirements

Regarding the Use Cases presented in the previous section, several associated requirements have been identified so far. It should be noted that this is not a static list and shall evolve together with the development course of this project.

Use Case 1 - Service Disruption Detection

For this use case there is the need for developing a service disruption detection that can work as a micro service and is able to monitor the overall health of both Platform and OTT Service ability to provide the service according with the contracted terms.

Is also able to monitor usage patterns and identify anomalies or short term trends that can put at risk the availability to deliver the service.

Additionally, there are several data sources required for addressing this use case:

- Service usage data (mandatory)
- Platform usage data (if available)
- CRM data (if available)

Use Case 2 - Platform Service Abuse

Development of a service abuse detection at the platform level, that can work as a micro service and is able to monitor the monitor subscriber service connection and usage patterns, together with the service contract data, in order to detect situations of abnormal usage that don't comply with the contracted service terms or have the risk to have the CSP incurring in financial or reputational losses. If such a case is detected, relevant data for the analysis should be collected and an alarm generated. Automated actions may be executed for certain scenarios.

Data sources required:

- Service usage data (mandatory)
- Platform usage data (if available)
- CRM data (if available)
- Billing data (if available)

Use Case 3 - OTT Service Abuse

Development of an OTT service abuse detection, at the platform level, that can make use of an wealthier set of data not at the reach of the OTT Service provider, and that can work as a micro service in order to monitor the connections to the OTT service and usage patterns, together with the service contract data, and spot situations of abnormal service usage that don't comply with the contracted service terms or have the risk to generate losses to either the CSP or the OTT Service provider. If such a case is detected, relevant data for the analysis should be collected and an alarm generated. Automated actions may be executed for certain scenarios.

Data sources required:

- Service usage data (mandatory)
- Platform usage data (if available)

- CRM data (if available); Billing data (if available)

Use Case 4 - Platform Capacity Planning

Development of a capacity planning micro-service, able to collect and process the current capacity indicators for the different technological platform together with the usage pattern, for each site or regional area, and with that information, to forecast the capacity levels and the ability to cope with the predicted usage levels for those sites and regions. Base on this, it should be able to pinpoint problematic areas / technologies / services (if there is a latency problem, only some services are affected). It should also recommend subscribers and services to be migrated, or investment measures to be made to cope with the growing needs.

Data sources required:

- Service usage data (mandatory)
- Platform usage data (if available)
- CRM data (if available)

Use Case 5 - Customer Experience Segmentation

A ML based Customer Experience Segmentation approach, that is able to run centrally based on the heavy historical data but with the ability to be adjusted at the Edge with the local usage and infrastructure data stored at the EDGE.

Data sources required:

- Service usage data (mandatory)
- Platform usage data (mandatory)
- CRM data (if available)
- Complaints data (if available)

Use Case 6 and 7 - OTT and platform Services Retention

Development of OTT Services Churn prediction models able to use relevant information stored in the central servers or stored at regional servers or even at the edge, where sensible data that can’t be moved to central location may reside, either in near real time or in batch.

Models should be adaptive as much as possible to changing patterns, without or with minimal human intervention.

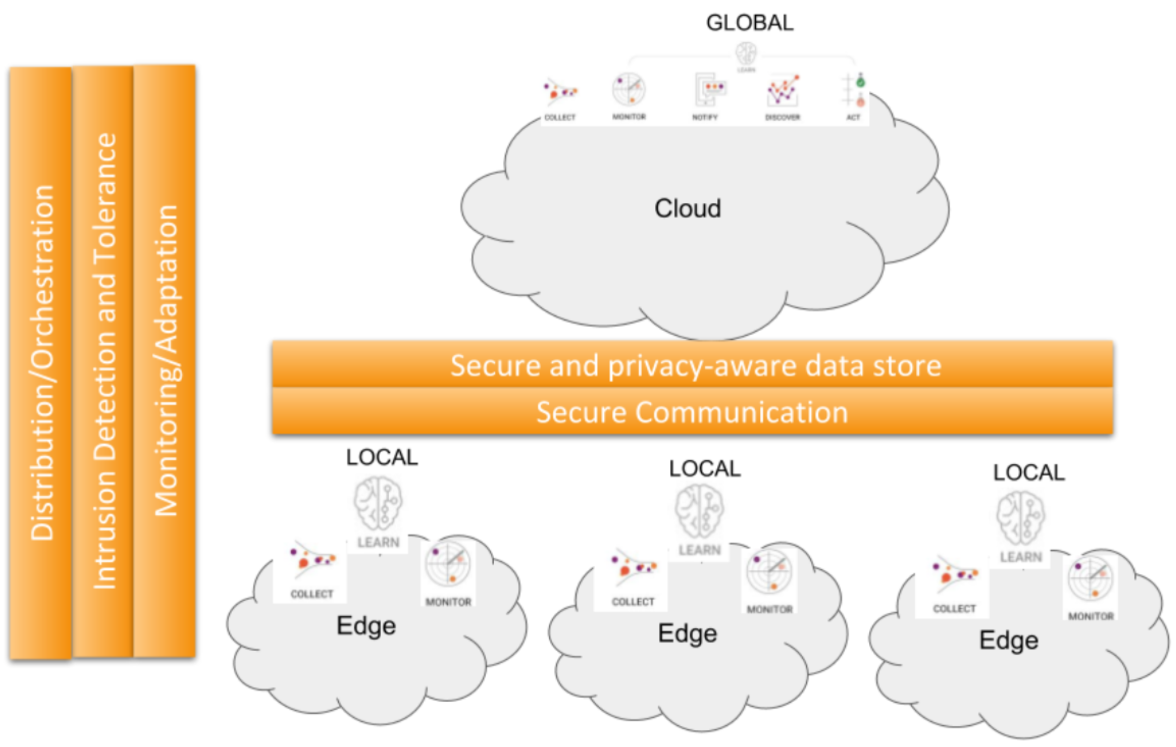
Data sources required:

- Service usage data (mandatory)
- Platform usage data (if available)
- CRM data (if available); Billing data (if available)

4 5G Security and Fraud Architecture Overview

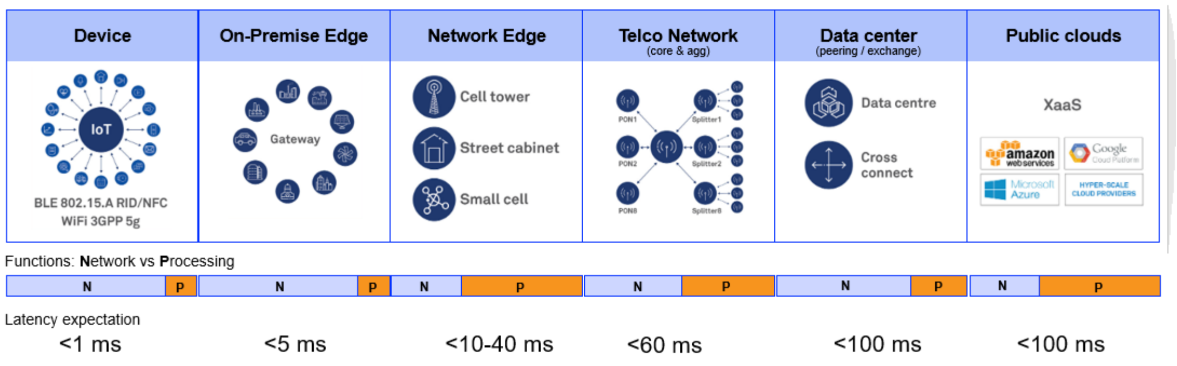
4.1.1 High Level Service Architecture

The end goal of MEC is to provide an optimized and low latency computing infrastructure with deployment agility that can scale horizontally or vertically based on requirements. With MEC we can move services and content closer to end-users and get more QoE, QoS while reducing backhaul congestion and optimizing gateway interconnectivity costs. MEC offers to developers and content providers the ability to run their applications that require high computational power and real time responses at edge of the network, while less demanding and non-real time response services can run on a central cloud. A first draft is provided in figure 7, where local edge connects to the multiple network layers, identify abnormal behaviours patterns based on model uploaded by a central service, and at the same time contributes with model training for federated learning and provides control and report capabilities.

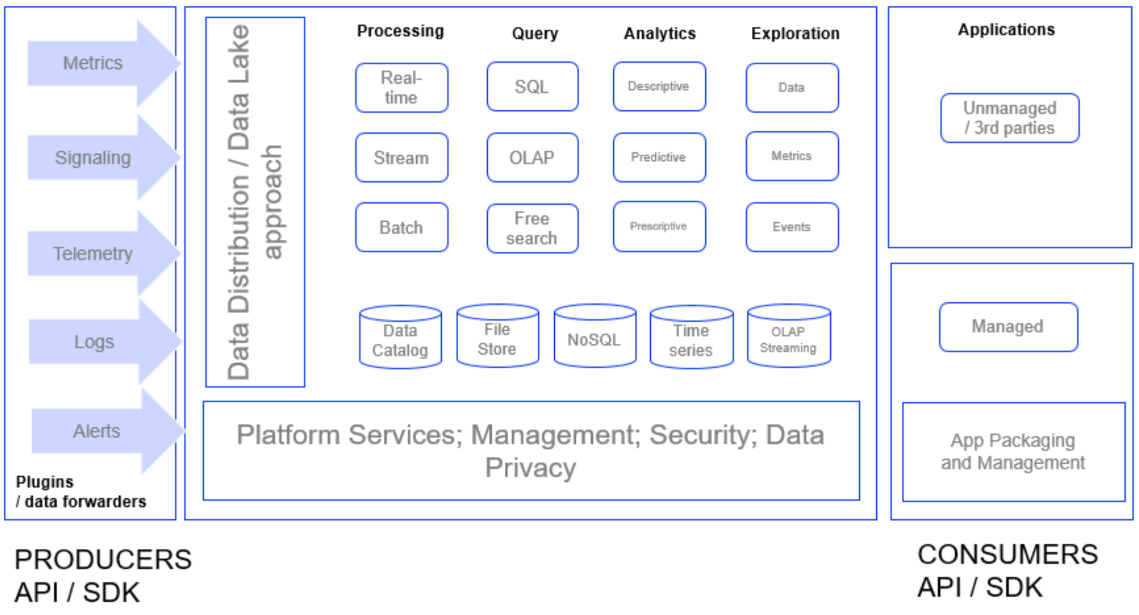


NETWORK ARCHITECTURE

https://www.etsi.org/committee/1425-mec

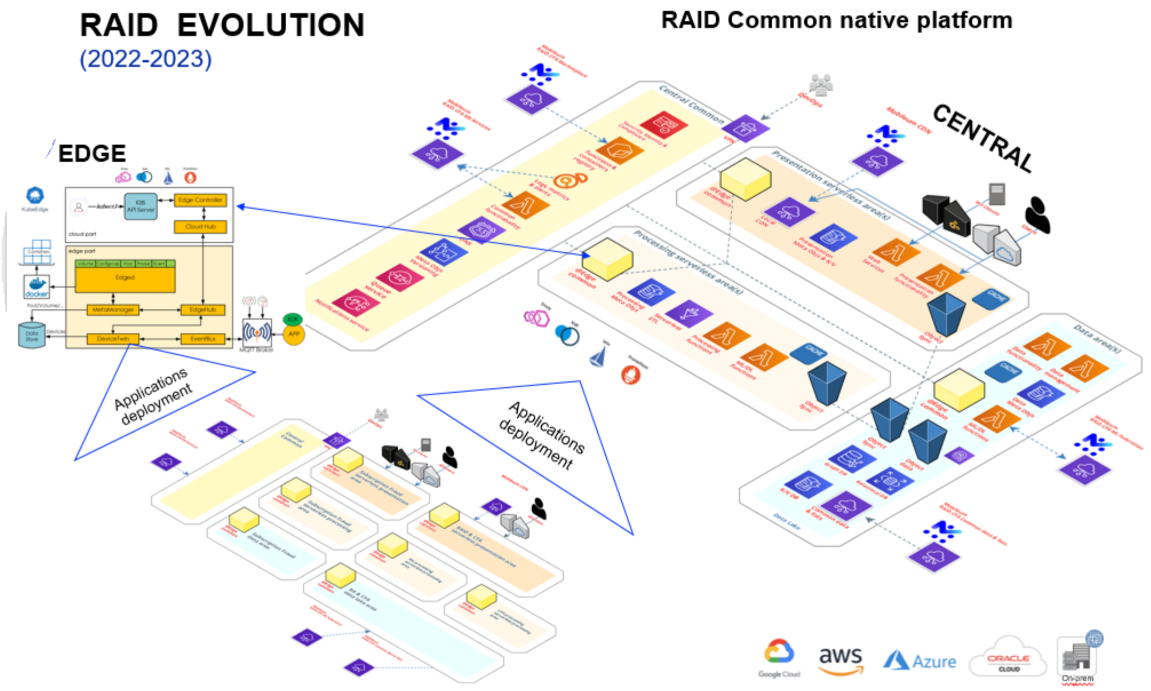


DATA ARCHITECTURE

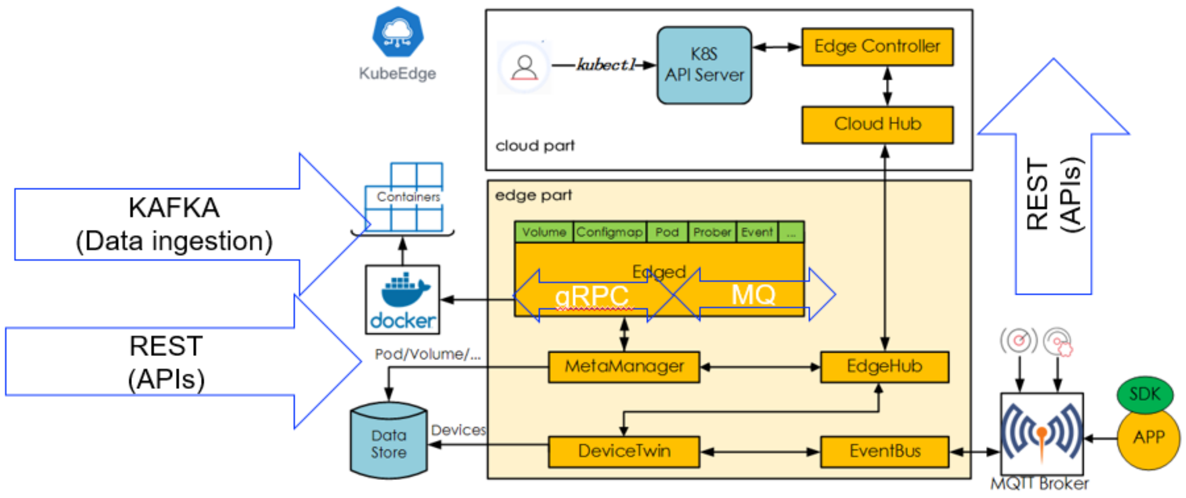


RAID EVOLUTION

(2022-2023)



EDGE COMMUNICATIONS

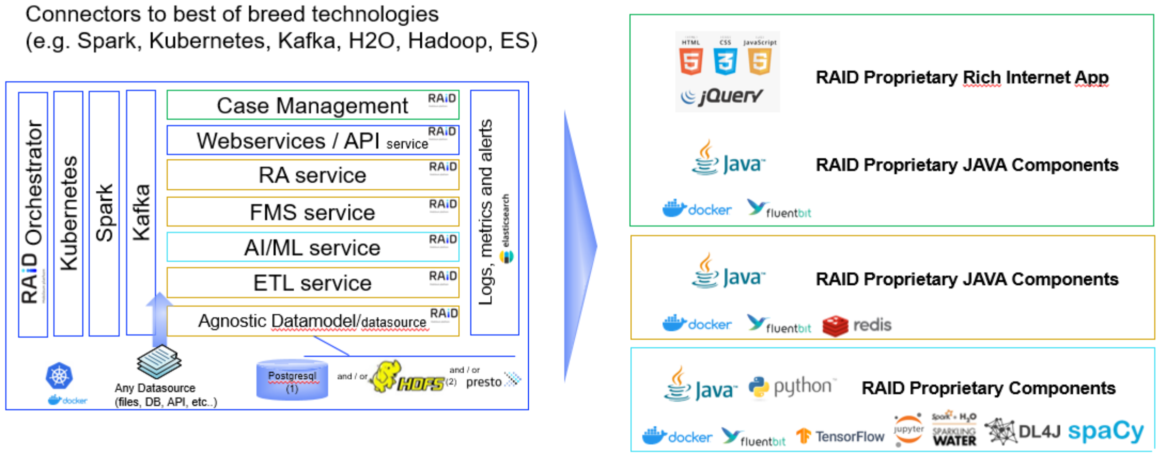


COMPETE 2020
PORTUGAL 2020
UNIÃO EUROPEIA
Fundo Europeu de Desenvolvimento Regional
FCT Fundação para a Ciência e a Tecnologia
Carnegie Mellon Portugal

TECHNOLOGIES

End to End RAID Proprietary components – no 3rd parties at the core

Connectors to best of breed technologies
(e.g. Spark, Kubernetes, Kafka, H2O, Hadoop, ES)



Cloud native agnosticity

LOGICAL VIEW OF PROPOSED PLATFORM | GCP COMPONENTS

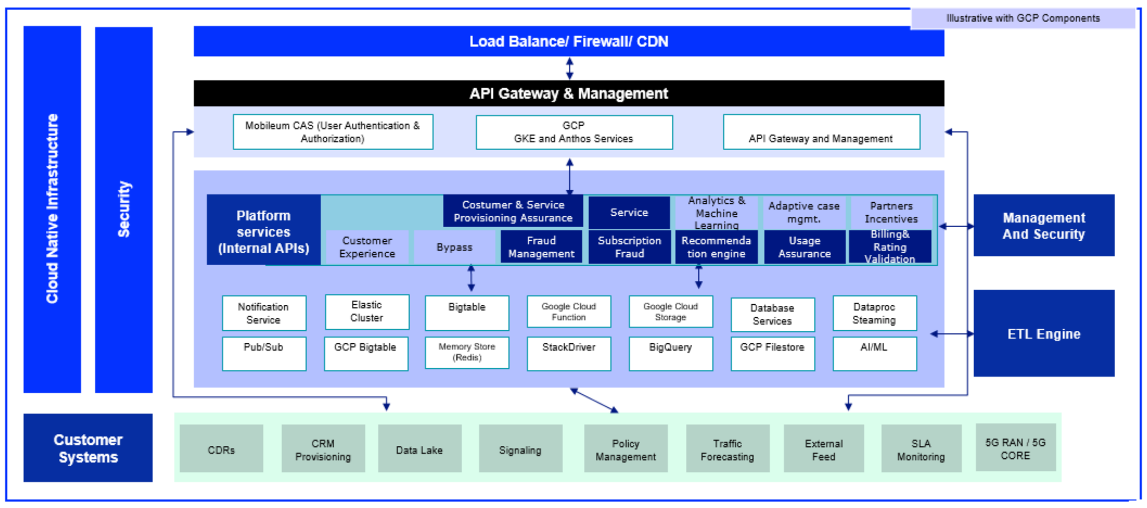


Figure 7. Adaptive, Intelligent and Distributed Adaptive Platform (AIDA) High Level Architecture Diagrams. [10]

The scheme depicted in fig. 7 exhibits the edge-centric perspective around which this architecture is built upon. It displays the reliance upon edge computing, where a big part of the processes will be running. All this must be accomplished while ensuring a secure communication and data store privacy standards between the edges and the cloud.

This distributed framework is orchestrated with full autonomy, scaling and adapting seamlessly to current needs, whilst providing live monitoring of the network in real time. Concurrently, the platform must be built with effective mechanisms for the detection of all sorts of intrusive challenges and constrains that might affect the performance, security or reliability of the network, and to be able to efficiently address and overcome them with minor disruptions at the consumer end.

Some implicit major contributes from this design are:

- Analytics: Edge technology collects a significant amount of data from all its networks that cannot tolerate latency

- Compliance: Compliance issues diverging from copyright enforcement to geographical data placement and GDPR

- Security: Edge computing allows for applications such as DDOS and cybersecurity to prevent these types of attacks and moves the security perimeter closer to the source.

- NFV: Network Functional Virtualization (NFV) access virtual network functions (VNFs) such as vRAN, C-RAN, vCMTS, vOLT need to run at the same location as edge computing.

5 Challenges and Issues

While the previous sections highlights the current best practices of Cloud Native applications to CNF (and for that matter VNF), it does not articulate the immaturity that this technology has with respect to its application in cellular networks. Additionally, **Microservices architecture** and **5G** technology benefit each other, yet there are a few challenges:

- The degree of orchestration that **Microservices** require is pretty high. This means that microservices need a lot of coordination and management. Therefore, decomposing an already existing system might be a little difficult.

- The same goes for 5G. Reengineering an existing system for suiting a 5G environment can be challenging and time-consuming.

- Even the microservices architecture, if not properly strategized, can create problems with latency resulting in poor performance of 5G applications and services.

6 Conclusions

The Adaptive, Intelligent and Distributed Assurance Platform, AIDA, project aims to deliver an end-to-end 5G-ready fraud management platform that is based on the latest advances in machine learning, edge computing, and hybrid cloud architectures to protect networks for 5G and beyond. AIDA address 5G's scalability and privacy challenges by enhancing RAID's engines and expanding its automation capabilities. In particular, the project will be focusing on the following goals:

- Leverage edge computing and 5G - to distribute RAID platform components to delegate processing to the edge or use central servers, according to the nature of the computation and the type and localization of monitoring and reference data.
- Explore emergent federated machine learning techniques - to learn from local data and push incremental model updates to coordinator nodes that maintain global models based on the contribution of edge nodes and other relevant data sources.
- Test resilience to intrusion or tampering - by requiring the research and application of intrusion detection techniques at multiple levels of the architecture, with the goal of enabling system-wide intrusion tolerance.
- Protect data privacy and confidentiality – by maintaining the confidentiality of the operational data being monitored, analysed, and protecting the privacy of the entities to whom the data refers.

In this document we provided an overview of the 5G Architecture which Raid will interface both at the core and radio access network as well as the infrastructure and technologies that Raid will use to detect 5G fraud.

An overview of the 5G fraud landscape was provided, yet this is not a final list, since 5G implementation is in an early stage and most use cases that define them are yet to be live outside of proof of concepts, labs or pilot users. The amount of new stakeholders in the fraud landscape will bring new types of fraud that are difficult to identify now. The use of AI and abnormal behaviour algorithms addressing smart fraud and statistical blind spots are key to address unknown scenarios as well as the integration of data feeds that traditionally are not considered in fraud management systems. This is one of the reasons edge and multi access edge computing are mandatory. 5G massive machine-type communication (MMTC) supports the IoT use cases. Billions of devices introducing multiple new attack vectors threating the core network as well as end users. A central solution to deal with such volume would be impractical, in terms of data transfer as well as data processing. A micro service approach seems ideal to deploy the independence, fault isolation and scaling necessary to support the multiple fraud threats targeting 5G.

As future work, we will move to the conception of the algorithms that support the identified use cases running according to the principles highlighted in this document, and define an architecture that allows the implementation of the use cases.

7 References

[1] , "Network Functions Virtualisation (NFV)". [Online]. Available: https://www.etsi.org/technologies/nfvf. [Last access Jan 2021].

[2] , "Industry Specification Group (ISG) Network Functions (NFV)". [Online]. Available: https://www.etsi.org/committee/nfv. [Last access Jan 2021].

[3] , "What Is Service-Oriented Architecture?". [Online]. Available: https://medium.com/@SoftwareDevelopmentCommunity/what-is-service-oriented-architecture-fa894d11a7ec. [Last access Jan 2021].

[4] , "What is the 5G Service-Based Architecture (SBA)?". [Online]. Available: https://www.metaswitch.com/knowledge-center/reference/what-is-the-5g-service-based-architecture-sba. [Last access Jan 2021].

[5] , "What is a CNF?". [Online]. Available: https://ligato.io/cnf/cnf-def/. [Last access Jan 2021].

[6] , "Cloud-Native Network Functions". [Online]. Available: https://www.cisco.com/c/en/us/solutions/service-provider/industry/cable/cloud-native-network-functions.html. [Last access Jan 2021].

[7] , "5G and the Cloud: a 5G Americas White Paper". [2019].

[8] , "Road to 5G: Introduction and Migration". [April 2018].

[9] , "Operator Requirements for 5G Core Connectivity Options". [Online]. Available: https://www.gsma.com/futurenetworks/wiki/operator-requirements-for-5g-core-connectivity-options/. [Last access Jan 2021].

[10] , "AIDA Presentation for the CMU Portugal External Review Committee". [January 2021]. Available: AIDA_CMU Presentation_v6.pptx.

[11] , "NFV Architectural Framework: The ETSI architectural framework explained". Available: https://stlpartners.com/telcocloud/nfv-architectural-framework/. [Last access Jan 2021].

[12] , "Bridging the Gap Between ETSI-NFV and Cloud Native Architecture". [2017]. Available: https://www.nctatechnicalpapers.com/Paper/2017/2017-bridging-the-gap-between-etsi-nfv-and-cloud-native-architecture/download.

[13] , "Network Functions Virtualisation (NFV)". Available: https://www.etsi.org/technologies/nfv. [Last access Jan 2021].