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

# From Webscale to Telco, the Cloud Native Journey

**July 2018**

Date: 2018-07-04 Version: 1.0

## Abstract

The 5G Software Network Working Group as part of the 5G-PPP Initiative prepared this white paper (WP) to provide first insights and trigger discussions about 5G cloud-native design. The shift from the cloud ready to cloud native, from VM to container, from MANO to Kubernetes, etc is a must have to avoid the risk that 5G remains a niche connectivity gap-filler largely ignored by cloud applications and services boom.
The WP highlights that we must ensure ascending compatibility with cloud environment in order to design a cloud-native 5G system and benefit from the best technologies from the cloud industry, essentially today the open source community, including micro-services the de-facto cloud industrial standard, and more general the service based architecture, and the most crucial technical assets from 5G-PPP Phase 1[1] and Phase 2[2] projects.

[1] 5GPPP Phase 1 Projects - https://5GPPP.eu/5GPPP-phase-1-projects/

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

# Table of Contents

**1 Introduction ..... 6**

**2 Cloud Native: a Webscale View ..... 7**
2.1 Microservice Architecture ..... 8
2.2 Twelve Factors ..... 10
2.3 Technology for Cloud Native ..... 11

**3 Telco Requirements ..... 13**
3.1 Service Requirement ..... 14
3.2 Application Type ..... 14
3.3 Deployment Environment ..... 15
3.4 Operability ..... 15
3.5 Business Environment ..... 15
3.6 Human Factors ..... 16

**4 Toward Telco Grade Webscale Frameworks ..... 16**
4.1 Need for Telco Grade features ..... 16
4.2 Twelve Factors in Telco Domain ..... 18
4.3 Telco-Grade Features ..... 19

**5 Technology Enablers – Open source and Standards ..... 20**
5.1 Standards ..... 20
5.1.1 ETSI NFV ..... 20
5.1.2 TM Forum ..... 21
5.1.3 Open Networking Foundation ..... 21
5.2 Open Source Software ..... 22

**6 Conclusion ..... 23**

**7 References ..... 23**

**8 List of Contributors ..... 25**

# List of Acronyms and Abbreviations

| | |
|---|---|
| 3GPP | 3rd Generation Partnership Project |
| 5G-PPP | 5G Infrastructure Public Private Partnership |
| API | Application Programming Interface |
| COE | Container Orchestration Engine |
| CSP | Cloud Service Provider |
| DPDK | Data Plane Development Kit |
| EC | European Commission |
| EU | European Union |
| eNB | Evolved Node B |
| ETSI | European Telecommunications Standards Institute |
| IoT | Internet of Thing |
| KPI | Key Performance Indicator |
| LCM | Life Cycle Management |
| LTE | Long Term Evolution |
| MEC | Multi-Access Edge Computing |
| NFV | Network Function Virtualization |
| NUMA | Non-Uniform Memory Access |
| PaaS | Platform as a Service |
| OSS | Operations Support System |
| KPI | Key Performance Indicator |
| RAN | Radio Access Network |
| SBA | Service Based Architecture |
| SFC | Service Function Chaining |
| SLA | Service Level Agreement |
| SOA | Service Oriented Architecture |
| VNF | Virtual Network Function |
| VNFFG | Virtual Network Function Forwarding Graph |
| VNFC | Virtual Network Function Component |

| | |
|---|---|
| VM | Virtual Machine |
| UE | User Equipment |
| WG | Work Group |
| WP | White Paper |
| ZOOM | Zero touch Orchestration Operation and Management |

# 1 Introduction

The objective of this white paper is to provide first insights and trigger discussions about 5G cloud-native design. 5G standard emerges at particular time in technology history when cloud transforms deeply almost all industries and services: it becomes obvious that innovations have to be made cloud-native for being successful. Cloud has had however, so far, only limited benefits on Telco infrastructure with a risk for 5G to remain a niche connectivity gap-filler largely ignored by cloud applications and services boom if another better model is not adopted.

**Software in 5G-PPP so far**: During Phase 1[3], the 5G-PPP Software Network WG focused on demystifying the important role of Software Defined Networks (SDN) and Network Function Virtualization (NFV) as a pillar in the 5G transformation. SDN and NFV could be seen as different expressions of an overall transformation trend, which is deeply impacting Telecom and IT industries. The white paper published in January 2017 provides a first conceptual architecture seamlessly and flexibly combining SDN and NFV technologies for 5G [14].

The expression "softwarization" is often referred to as a general paradigm shift in telecom architecture from "boxes" to "functions", and from "protocols" to "APIs". The softwarization is one of the feature of a more powerful transformation named cloudification.

**Cloud impact in general**: Cloud has disrupted the established order in many sectors as companies have been able to reduce investment in their internal data centers in favor of the unlimited computing resources available on-demand and billed for use from cloud providers. From now on, companies' competitiveness depends directly on their ability to quickly deliver new ideas. Start-ups understand this well, so they rely on cloud-native approaches to disrupt traditional sectors. It becomes obvious that innovations should be made cloud-native for being successful. Cloud-native is an approach to build and run applications that fully exploit the benefits of the cloud computing model. It includes services architectures, infrastructure-as-code, automation, continuous integration/delivery pipelines, monitoring tools, etc. Applications built and deployed with the cloud-native pattern have some common characteristics: first, they are composed of microservices, i.e. each application is a collection of small services that can be operated independently of one another. Microservices are often owned by individual development teams that operates on their own schedule to develop, deploy, scale and upgrade services. Second, cloud-native applications are packaged in containers, aiming to provide isolation contexts for microservices. Containers are highly accessible, scalable, easily portable from one environment to another and fast to create or teardown. This flexibility makes them ideal for building and running applications composed of microservices. Third, cloud-native applications are built and run on a continuous delivery model supporting fast cycles of build, test, deploy, release, and develop. This helps software service developers and infrastructure IT operations teams to collaborate with one another for building, testing and releasing software updates as soon as they are ready without affecting end-users or developers of other teams. Such a model encourages the adoption of DevOps principles fostering collaboration between software service developers and infrastructure IT operations and creating behavioral culture where building, testing and releasing services happens more rapidly, frequently and consistently. Cloud-native applications are dynamically managed, often built and run on modern platforms such as Kubernetes or Pivotal Cloud Foundry which offer hardware decoupling critical in terms of deployment automation, scaling and management. Of course, the shift to cloud-native mind-set, especially for the telecom sector, is not be easy and takes a phased approach to rap full benefits. When successfully implemented, this transformation can bring unprecedented speed, agility and resilience in service development and management process [1]. It greatly increases developer's productivity and

[3] 5GPPP Phase 1 Projects - https://5GPPP.eu/5GPPP-phase-1-projects/

simplifies operations. New application features and services can be pushed live for customer use whenever they are ready with no need to worry about other teams work and zero impact on the end user experiences.

**Cloud impact in telecom ecosystem in particular:** To analyze cloud-native impact in telecom ecosystem, the WG first studied service-based architecture principles, covering Service-Oriented Architecture (SOA), Microservices Architecture (MSA) and Service-Based Architecture (SBA), the latter being adopted in next generation CORE [15]. The WG then focused on the twelve factors methodology [16], a cloud-native developer best-practice compendium created by cloud service provider pioneer Heroku [17]. Then the WG performed a review of cloud-native and open-source enabling technologies for deployment, orchestration, networking, continuous integration/continuous delivery, monitoring, etc... Finally, the WG analyzed interactions between cloud-native technologies and telecom standard such as ETSI, by collecting feedback from relevant participating 5G-PPP projects on following topics:

- *What are the specific requirements between Webscale and telco players?*
- *What gaps do we have in container management eco-system to achieve telco-grade performance?*
- *Which segments are critical from latency perspective*?
- *What features are required to support micro-services architecture in all the segments?*
- *Which factors are difficult to implement in telco (from https://12factor.net)?*

The different inputs have been categorized and used to build a section where we analyze the telco-grade enhancements that should be added in frameworks like Kubernetes, monitoring, stateless design, etc...

**White-paper organization:** The remaining part of this paper is organized as follows:

- Section 2 introduces the service-based architecture, the twelve factors and the required technology for networking, virtualization, orchestration, etc.
- In Section 3 the telco requirements are listed.
- In section 4 we describe the telco-grade enhancements.
- In section 5 the technology enabler, open-sources and standards are briefly summarized.
- Finally, conclusions are summarized in Section 0.

# 2 Cloud Native: a Webscale View

Recently, under SA2 [18] , 3GPP proposed a service-based architecture of the 5G core system. The architecture shown in figure 1 is composed of network functions (NFs) and reference points that connects NFs. The User Equipment (UE) is connected to either RAN or Access Network (AN). RAN represents a base station using new Radio Access Technology (RAT) and evolved LTE [2]. However, AN is a general base station including non-3GPP access such as Wi-Fi. At the same time, UE is connected to Access and Mobility Function (AMF). Other next generation core NFs are Session Management Function (SMF), Policy Control Function (PCF), Application Function (AF), Authentication Server Function (AUSF), User Plane Function (UPF), User Data Management (UDM) and Network Slice Selection Function (NSSF). Based on a general agreement on 3GPP SA2, the mentioned functions form two separate planes in the next generation architecture, i.e. the user plane and the control plane. User plane carries only user traffic while control plane is dedicated to network signalling. NFs are interacting one another via predefined interfaces such as NG1, NG2, and NG3. All these interfaces are candidates to be REST APIs [19] and in this sense, the architecture reference points can be simply replaced by a “message bus” in a logical diagram.

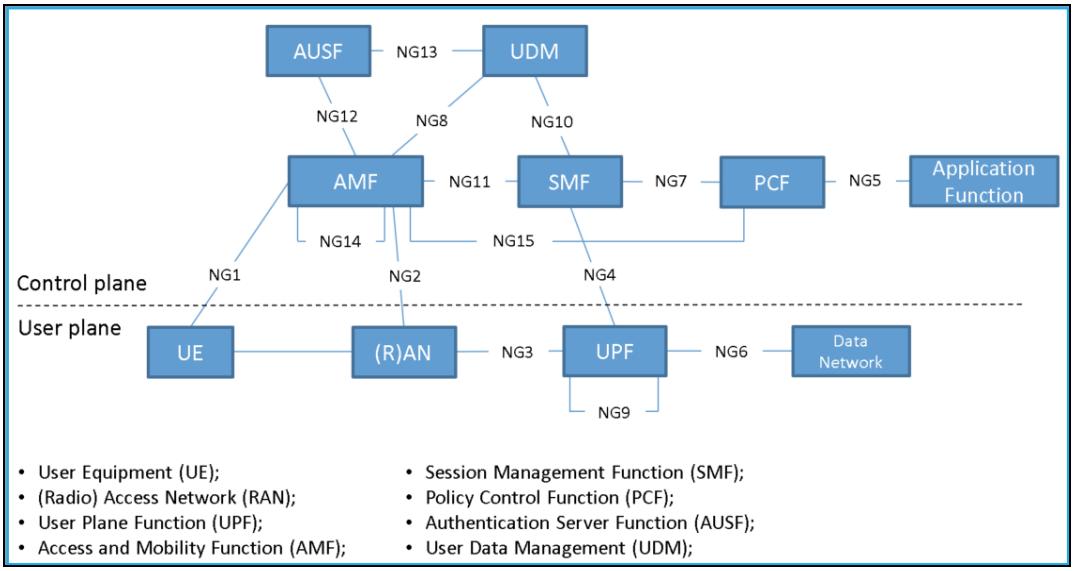


**Figure 1: 3GPP proposed architecture and reference points for 5G networks.**

One can observe that 5G core network is now based on what is called "Service-Based Architecture" (SBA), centered on services that can register themselves and subscribe to other services. In practice, there are three possible ways to implement NFs: either as a network element on a dedicated hardware, or as a software instance running on a dedicated hardware, or as a virtualized / cloud-native function instantiated on an appropriate platform, e.g., a cloud infrastructure. Leveraging cloud software technologies, 3GPP NF architecture leads to higher flexibility, programmability, automation and significant cost/energy reduction.

With the direction taken by 3GPP, it is expected that 5G core NFs become cloud- and virtualization-based applications. 5G core platform will be more programmable and allow many different functions to be built, configured, connected and deployed at needed scale. For that, we propose to start this section by reviewing the Service-based Architecture characteristics, the twelve factors and the required technology.

## 2.1 Microservice Architecture

With the promise of offering ultra-reliable, low-latency communications, and high speed, 5G is expected to introduce a golden digital age of remote healthcare, autonomous cars and advanced robotics use-cases. For that a high level of flexibility in the architecture is needed, which can be summarized as the following:

- Support of different requirements to the network, such as network capacity, data rate, transmission delay, information security;
- Support the exposing of network capabilities to operator's services and 3rd party applications
- Support easy deployment and maintenance. Each functionality shall be able to upgrade according to new requirements and scale-out according to system capability, without affecting other functionalities.

Based on these requirements, virtualization and service based mechanisms are a significant industry trend especially relevant for the 5G ecosystem.

Comparing to the Webscale companies, one can notice that the challenges faced by CSPs are like those faced by the Webscale companies ten years ago. Webscale companies started by building their businesses on single monolithic applications. Over the time, they developed so-called microservice architectures. By deconstructing an application into smaller components which can

be reused for other applications – and restructuring the IT organization into fully accountable microservice teams – companies can create more flexible, scalable and dynamic software development capabilities.

Microservices is a powerful architectural design pattern where the system is composed of small granularity, highly cohesive and loosely-coupled services. Each of these services fulfil a specific functionality and is self-contained. Interactions between services implement standard light-weight interfaces (e.g. RESTful principles etc.). More generally, three variety service-based of architectural styles, including microservices, service-oriented architecture (SOA [21]), and service-based hybrids types (see [22]) are proposed based on domain driven design framework [23].

At the service granularity level, microservices are small. They contain typically one, two or three modules focusing on one purpose. They have a bounded context where the service components are bounded to own data and to own implementation. At the other extreme, we find SOA evolving big services, an imperative and procedural programming style and a heavy messaging (e.g. ESB). Between these two extremes, the SBA solves a lot of practical problems. The idea is to keep the same architecture style of microservices but to increase the service granularity implemented. We can qualify that SBA is a hybrid approach between microservices and SOA.

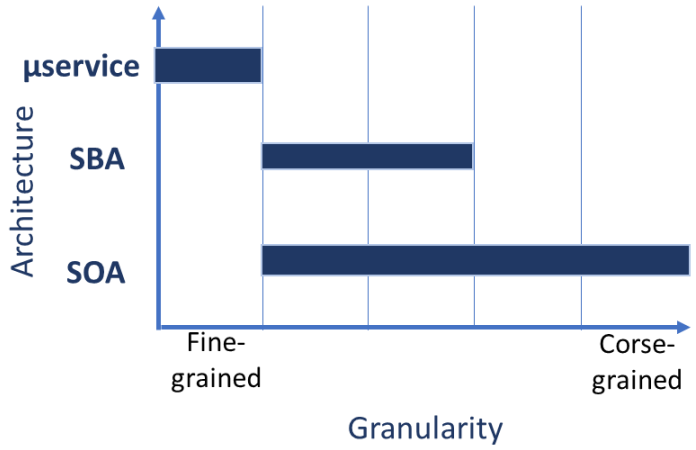


**Figure 2: Service Granularity of the three architectures, SOA, SBA and µservice.**

We invite the reader to the following reference [24] where a complete analysis could be found.

| Architecture / Criterion | SOA | µ-service architecture | SBA | Monolithic |
|---|---|---|---|---|
| **Agility** | Low | High | Medium | Low |
| **Deployment** | Low | High | Medium | Low |
| **Testability** | Low | High | Medium | Medium |
| **Scalability** | Medium | High | Medium | Low |
| **Performance** | Low | Medium | Medium | High |
| **Simplicity** | Low | Medium | Medium | High |

Source:[24]

**Table 1: Comparison of different architecture following a set of criteria (Agility, Deployment, Testability, Scalability, Performance, Simplicity)**

It is clear from the table that the intermediate approach is the Service-based architecture which brings following benefits to 5G:

- Easy update of Network:
  - Finer granularity allows individual services to be upgraded with minimal impact to other services.
  - Facilitated continuous integration reduces the time-to-market for installing bug fixes and rolling out new network features and operator applications.
- Extensibility:
  - Light-weighted service-based interfaces are needed to communicate across services.
- Modularity & Reusability:
  - The network is composed of modularized services reflecting the network capabilities and provides support to key 5G features such as network slicing.
  - A service can be easily invoked by other services (with appropriate authorization), enabling each service to be reused as much as possible.
- Openness
  - Together with some management and control functions (i.e. authentication, authorization, accounting), the information about a 5G network can be easily exposed to external users such as 3rd parties through a specific service without complicated protocol conversion

## 2.2 Twelve Factors

We examine the idea that cloud-native applications can act as a precursor of transitioning web-scale computing paradigm to telco domain. Such a shift should enhance internal telco needs and enable vertical solutions. It is worth examining whether properties of cloud-native apps - commonly referred as the Twelve Factors [16][17] - can be adopted by telecom industry with tangible benefits in terms of reduced Capex/Opex, east-west and north-south scalability, faster time-to-market service development and deployment. In this section, the twelve factors are briefly listed. In section **Error! Reference source not found.**, mapping of the 12 factors to the telco domain is reported by participating 5G-PPP phase 1 and phase 2 projects. This account is accompanied by necessary adjustments justified by the specific telco requirements and restrictions as well as the gaps identified at the present state of innovation.

The Twelve Factors4 are:

I. **Codebase**: A single codebase properly versioned should be associated with varying application deployment strategies (recreate, rolling-update, blue/green, canary, A/B testing, shadow).

II. **Dependencies**: Dependencies (either system or between apps) should be explicitly declared in a dependency manifest and isolated in environments during both development and production stages.

III. **Configuration**: The configuration that varies between deployments should be stored in the deployment's environment separately from the code.

IV. **Backing services**: Backing services are services the app consumes over the network (databases, messaging queues, caches etc.). Those must be considered as loosely-coupled resources attached to the app.

V. **Build, release, run**: Code is transformed to a deployment in three separate stages: **build**, during which a codebase is transformed to an executable with its appropriate

4 https://12factor.net/

dependencies, **release**, which combines the build with the environment configuration, and the **runtime** stage which launches the app's processes. Such stages should be air-tight.

VI. **Processes**: The app is executed as stateless, share-nothing processes. Persistence is offloaded in a stateful backing service (IV). Cached data is considered volatile and not stateful. This is a factor that can break scalability of a cloud-native app.

VII. **Port binding**: Services are exposed to other services through specified ports. This aspect gives rise to the service discovery role of an app orchestrator mechanism.

VIII. **Concurrency**: To scale out the app follows the process model, i.e. by adding more of the same processes to enable concurrency. This scalability enabler is complemented by the shared-nothing, stateless property of the app.

IX. **Disposability**: Processes should require minimum startup time (to quickly fire up new processes and support elasticity and rapid deployment). Graceful shutdown is also a goal; a process should accommodate standing requests or computation before responding to a SIGTERM signal.

X. **Dev/prod parity**: A cloud-native app's development environment should mimic as much as possible the production environment. More specifically the time, personnel and toolset gaps between the deployment areas should be as small as possible to increase the effectiveness of Continuous Delivery (Deployment) of the app. This is facilitated by the usage of containerized environments and declarative provisioning and configuration tools aiming exactly at shortening the time and toolset gaps

XI. **Logs**: Logging should be treated as a stream of events available for monitoring all separate services of the app. The streams should be aggregated together to better understand the whole picture of the state of the app. However, the app is not involved with actual log analysis; this is offloaded to separate external tools.

XII. **Admin processes**: Run admin/management tasks as one-off processes that are preferably declaratively stated against a release, using the same codebase and configuration. These tasks are to be shipped with application code under its environment.

## 2.3 Technology for Cloud Native

The Cloud-native Computing Foundation's Cloud-native Landscape project suggests a roadmap for the cloud-native journey [25]. This roadmap encompasses technologies, tools, and patterns that help implementing cloud-native development and their deployment environment, and categorizes them:

1. **Containerization**: Containers are a lightweight virtualization alternative to VMs [26]. They leverage two Linux kernel features: namespace for enabling isolation of an application by limiting its view of the Operating System environment like process trees, networking resources (e.g. interfaces, IP addresses, routing table, etc.), and cgroups for limiting and prioritizing system resources like CPU, memory, I/O, etc. While VMs package an entire operating system along with the function/application, containers only package the application with its application-specific OS dependencies because running containers rely on the host kernel. Containers provides many advantages over VMs [45]. Docker is one of the most widely used container engines.
2. **CI/CD**: Continuous Integration/Continuous Delivery is a pipeline process that automates the steps involved between compiling a source code and having it deployed in production environment [27]. CI is a practice that encourages developers to integrate their code into a main branch of a shared repository early and often. CD is an extension of continuous integration. It focuses on automating the software delivery process so that teams can easily and confidently deploy their code to production at any time. Therefore CI/CD can take the

code pushed into a repository, compile it, pass it through a test suite, and if tests were successful build a container image and deliver it/deploy it appropriately. Jenkins is one of famous CI/CD tools [28].

3. **Orchestration**: Orchestration is involved when things need to take place in a certain order. (micro)service orchestration refers to the coordination of multiple services through a centralized mediator such as a service consumer or an integration hub. Kubernetes [29] is widely deployed at scale and used both in testing and production environment.
4. **Observability & Analysis**: Monitoring, logging and tracing are three crucial requirements of any service deployment in general, and particularly for VNFs. They enable observability and allow the analysis of a service state. A whitepaper [30] identified these three features among others as required for telco-grade PaaS support.
5. **Service Mesh & Discovery**: Complex services are made up of a high number of microservices. At a large scale, manually configuring and providing infrastructure services (e.g. routing, security policy, load balancing, etc.) to these microservices is practically not possible. In [46] it is explained how Service Mesh provides a configurable infrastructure layer integrated from within the compute cluster itself through APIs that doesn't require any additional appliances [31]. Linkerd [32], Envoy [33], and Istio are commonly used for mesh architecture. Another feature that is needed when a high number of microservices need to communicate is service discover. This is similar to the DNS system in concept, but is designed to cope with the dynamic nature of microservices. Services are registered in a service registry, that is a database of available services, and a service's client queries the registry to determine the service's location. Examples of a service discovery engine are CoreDNS [34], Consul [35], or etcd [36].
6. **Networking**: Traditional IP routing-like networking cannot cope with the requirements of cloud-native systems where instances of (micro)services are created, moved, and stopped very quickly. There are currently two networking models which provides specifications for container networking solutions: Container Network Model (CNM) and Container Network Interface (CNI). CNM is developed and being only used by Docker, while CNI is becoming the de facto standard of networking and is extensively used by other container management systems, particularly Kubernetes. Some implementations of CNI are calico [38], flannel, and WeaveNet.
7. **Distributed Database**: The cloud is distributed by nature. Data Centers and clusters span multiple geographical locations. A distributed database provides scalability, resiliency, and performance required for this kind of environment, and which cannot be found in centralized database. Cassandra is a well-known distributed NoSQL database. CNCF suggests Vitess for MySQL database.
8. **Messaging:** By design, microservices are not deployed in isolation. They communicate and exchange message with each other very often. For Web application, REST is usually used and provides the required performance. However, this is not always the case, especially for telco services and VNFs. gRPC is a high performance Remote Procedure Call system that was initially developed by Google and is now open-source. Another messaging system is NATS.
9. **Container Runtime:** This is the ecosystem that enables building images and running them. There are many container runtime engines with different level of maturity and features. Linux Container LXC comes built-in in the Linux kernel, but it is not supported by orchestration engines. Other feature-rich container runtimes include docker, rkt and containerd.
10. **Software Distribution:** When installing any piece of software on any system, its origin must be authenticated and its integrity must be verified to avoid any compromise of the system by malware or other means. Therefore, the software provider must use and provide tools that enable the users to verify its identity and be sure that the software has not been modified by

someone else. Notary is client-server based system that relies on TLS and digital signature for secure communication and verification of the software's integrity.

# 3 Telco Requirements

ITU-R Recommendation M.2083-0 [4] has defined the following 5G Usage Scenarios for International Mobile Telecommunications (IMT) for 2020 and beyond, which are often depicted by the triangle shown in Figure 3 below:

- **eMBB** – Enhanced Mobile Broadband: Mobile Broadband addresses the human-centric use cases for access to multi-media content, services and data. The demand for mobile broadband will continue to increase, leading to the need for enhanced Mobile Broadband connectivity. Such an enhanced Mobile Broadband will enable new applications which requires improved performance and increased seamless user experience on top of what is offered today. The eMBB usage scenarios cover a range of cases, including wide-area coverage and hotspots - each has its own requirements. For the hotspot case, i.e. an area with high user density, very high traffic capacity is needed, whereas the requirement for mobility is low. The user data rate in the hotspot case is higher than that of wide area coverage. For the wide area coverage case, seamless coverage and medium to high mobility are desired, with much improved user data rate compared to existing data rates. However, as stated above the data rate requirement may be relaxed compared to hotspot.
- **mMTC** – Massive Machine Type Communications: This use case is characterized by a very large number of connected devices typically transmitting a relatively low volume of non-delay-sensitive data. Devices are required to be low cost, and have a very long battery life.
- **URLLC** –Ultra-Reliable and Low Latency Communications: This use case has stringent requirements for capabilities such as throughput, latency and availability. Some examples include wireless control of industrial manufacturing or production processes, remote medical surgery, distribution automation in a smart grid, transportation safety, etc.

Note that the term "5G Usage Scenarios" referring to eMBB, mMTC, URLLC corresponds to the terminology employed in the ITU-R Recommendation M.2083-0 [4] . This is different from the terminology employed in 3GPP TR 38.811 [5] where eMBB, mMTC, URLLC are referred to as "5G Service Enablers". Nevertheless, the IMT-2020 5G Triangle illustrated in

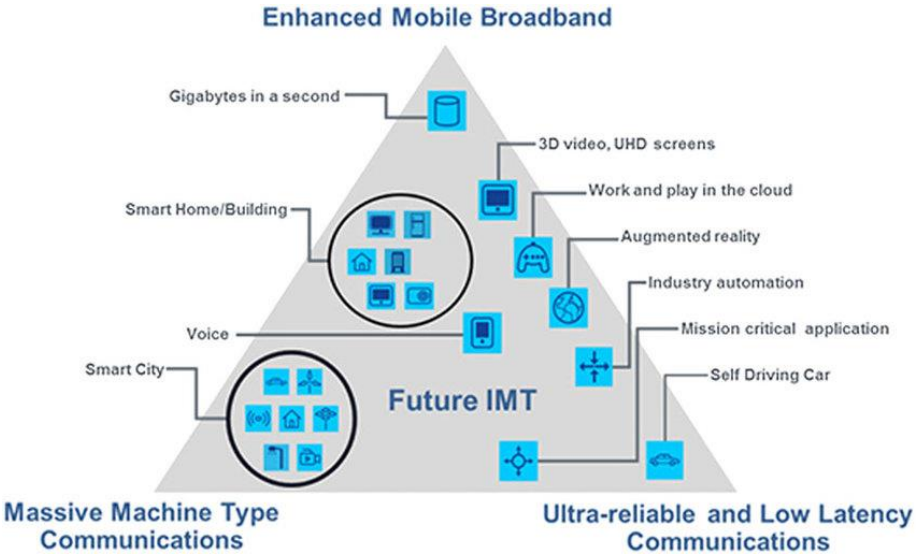


**Figure 3: Usage scenarios of IMT-2020 and beyond (Source: ITU-R [4])**

In addition to the high level ITU recommendation, the Software Network WG ran a survey to collect telco requirements from different 5G-PPP Phase II projects following their targeted use-cases. As second step, a data categorization has been done by the WG. It results the following categories described in this section:

- **Service requirement**: Requirements related to the service to be provided, most prominent example is latency.
- **Application type**: Technical characteristics of applications, such as used protocol stacks.
- **Deployment environment**: Requirements related to the environment within which the telco system is deployed.
- **Operability**: Requirements related to operability of telco systems, e.g. different timescales of creating logs.
- **Business environment**: Requirements related to business aspects, e.g. multi-vendor environments or support of legacy systems.
- **Human factors**: Characteristics related to mindsets of human stakeholders.

## 3.1 Service Requirement

Most important services a telco system provides to end-users are communication services. Communication services may be of different types like support of voice calls and data calls. The communication service has to be provided also to users on the move, the system has to keep track of the location of users as well as to handover active calls among different base stations. The different communication services impose different requirements on the telco system in terms of latency, jitter, throughput, and packet loss. If not satisfied, the quality of experience for the end-users drops. With 5G, these requirements become more stringent: Peak throughput for enhanced Mobile Broadband can reach 10Gbps, latency between UE and corresponding application in the network might be as small as 1 ms for ultra-reliable, low-latency communication.

Equally, procedures such as call establishment, handover, etc., are timebound. If not successful within the allowed time, corresponding calls might be terminated by the system. In addition to successful termination of procedures, a telco system as a whole has to be highly reliable. It has to provide carrier-grade or 5 nines availability, i.e. it has to be up for 99,999% of time, corresponding to a downtime of about 5 minutes per year. Such high availability is critical as telco systems are used for emergency calls.

A telco system provides communication services to a huge number of devices; in developed countries there is already more than one phone per inhabitant. With IoT the number of connected devices will increase even further. Smartphones, even when not in active use, establish data calls every now and then for background data. This results in a huge amount of simultaneous procedures to be handled by a telco system.

## 3.2 Application Type

In general, establishing communication services implies to forward high amount of data with short latency. Except for deep packet inspection, the forwarding considers L2 or L3 headers. The forwarding generates a high I/O load in the user plane, whereas a high signaling load in the control plane exists. The user plane traffic is handled either by dedicated hardware or by software. When the user plane is handled in (virtualized) software it usually requires a 1:1 relation among logical and physical cores to satisfy the latency, jitter, and packet loss requirements. I.e. overbooking of processing resources in the user plane is difficult.

In the control plane of telco systems, many operations follow a synchronous communication pattern, id est control plane procedures are a conducted sequence of operations with an ordered orchestration of the output and input of successive steps.

With multi-access edge computing (MEC[5] [6]) applications can be deployed close to the user at the mobile edge, achieving low latency between the user and the application. When the user moves, the application might have to be migrated to another MEC datacenter or user specific context and data has to be handed over to another instance of the same application.

## 3.3 Deployment Environment

Although parts of telco systems such as core network functionality or customer relationship management are well suited for deployment in data centers, this does not hold for a telco system as a whole. There is specific physical equipment whose functionality cannot be virtualized and migrated to a datacenter, e.g. base stations in the radio access network and optical line terminals in passive optical networks for fixed-line access. Nevertheless, this equipment is an integral part of the telco system and needs to be controlled and managed.

This physical equipment is spread over a large geographical area, it cannot be placed in a few central locations. Correspondingly, even the parts which can be virtualized, have to be deployed in a large number of central offices, one or a few per city. MEC will increase the number of datacenters, to which the microservices have to be deployed, even further.

Both the distributed nature of telco systems and the use of physical equipment imply that substantial investments have been made already by operators. Redeploying such a system as a whole with softwarized components might not be feasible for an operator. A typical deployment scenario for a softwarized telco system would be a brownfield scenario, where parts of the whole system have been migrated already, whereas other parts of the system are still operated as a ‘classical’ telco system.

## 3.4 Operability

An operator is using (software) components of multiple vendors. To prevent damage to the overall system, operators require authentication of the software images. Telco systems provide services to a vast number of users, especially they provide critical services such as emergency calls. Therefore, operators require that new components are assessed before deployment to ensure the proper operations of the end-to-end services. Such pre-assessed or certified software loads should be authenticated and only such authenticated components are allowed to be deployed on a wide scale.

After deployment, there is a large number of physical devices and deployed microservices, which have to be monitored. The corresponding logs need to be retrieved, stored, and analyzed. Due to the variety of devices, the log events are generated on different time scales, requiring the corresponding flexibility from the monitoring system.

## 3.5 Business Environment

A telco operator might develop and deploy some parts of its telco system on its own or using FOSS, but large parts of the system will be developed by different vendors. There are at least 3

[5] http://www.etsi.org/technologies-clusters/technologies/multi-access-edge-computing

types of stakeholders for a telco system. the vendors of physical equipment and VNFs, the telco operators using this equipment to provide connectivity services to their users and the end-users consuming the telco services. Additional stake holders are service companies to operate whole networks or authorities requiring and supervising certain regulations.

Telco operators try to avoid a lock-in with single vendors, therefore in different regions of a network the same kind of equipment or VNF might be deployment by different vendors. Vice versa, from the vendor perspective, their equipment or VNFs will be used by several operators, each of them potentially using a different version. In this multi-vendor environment different management interfaces and procedures are used. Often, equipment-specific integrations to an operations support system (OSS) are needed to allow uniform operability of a telco system.

When services of verticals such as entertainment, automotive, or eHealth are connected to a telco system or even integrated into it, the number of stakeholders and heterogeneity of the system increases even further.

A telco operator is somehow limited in the services it can offer. In general, the services are communication services and the whole environment is strongly regulated, and adherence to standards in many parts of the system is required. This reduces the flexibility in the services an operator can offer and might as well discourage innovation in service development. On the one hand, this environment allowed to create a highly competitive environment of telco equipment vendors and operators. On the other hand, this creates a large cost pressure on the operators in deploying and operating the infrastructure. Despite the competitiveness and the cost pressure, the required investments for a telco system are high.

## 3.6 Human Factors

Due to large CAPEX investments, telco equipment must be operated for extended periods of time. This may cause difficulty or slow adoption of new trends in system development as the already-deployed part of the network needs to be maintained and operated, such that radically new ways of building and operating networks are not embraced really.

In addition, there is a lack of DevOps mentality due to the issues of services criticality, multi-vendor integration, etc... There is also a lack of holistic training in the integration of software and network technologies. No system engineering specialty covering these aspects are available at large yet, but the situation starts to change slowly.

# 4 Toward Telco Grade Webscale Frameworks

## 4.1 Need for Telco Grade features

In the context of emulating the successful best-practices of cloud giants like Google and Facebook, a networking approach leveraging web-scale IT principles seems promising. Telco-grade webscale frameworks will need to adhere to following principles:

- **Open and modular software**, i.e. delivering application at a fast pace, going hand in hand with DevOps.
- **Intelligence in software**, meaning decoupling hardware and software, offering vendor-agnostic solutions.
- **Scalable and efficient**, i.e. building modular networks providing efficient scaling with respect to Telco-domain requirements.

The rapidly emerging service-based architecture paradigm in cloud-native distributed applications advocates for "smaller" services, i.e. microservices, to compose complex applications. These microservices are independent, small-scale processes that communicate through pre-defined APIs [39]. Container technologies, including Docker, Kubernetes, Mesosphere etc., are in fact ideal for cloud-native applications as containers offer a lightweight atomic unit of computing [26].

Microservices and containers allow developers to focus on the actual business logic, disassociating the development process from concerns on deployment, testing, underlying infrastructure and so on. They offer better scalability and rapid development cycles.

However, as with any distributed computing approach, towards telco-grade webscale frameworks it is especially necessary to avoid the assumptions widely known as "the eight fallacies of distributed computing" [40]:

1. The network is reliable.
2. Latency is zero.
3. Bandwidth is infinite.
4. The network is secure.
5. Topology doesn't change.
6. There is one administrator.
7. Transport cost is zero.
8. The network is homogeneous.

Based on the aforementioned fallacies and given the different requirements of the telco domain, it is no surprise that in order to apply cloud-native principles to networking, there are gaps in container management related to telco features that need to be addressed. Typically, telco key performance indicators taken into account include latency and bandwidth, availability and security. For instance, since each microservice contributes in the overall latency, it is challenging to predict the latency of a specific service, especially in cases of large and distributed systems. Additionally, container management technologies (Kubernetes, Mesos, Swarm) are using an overlay network (Layer 4) which is not optimized for low latency requirements, e.g. in the cloud RAN. Another aspect is network availability, as 99.999% availability is commonly set by the operators. Currently, cloud-native applications are not built to handle bandwidth, latency and network availability requirements as this is not the original type of workload they were designed for, although some new applications set latency demands (e.g. critical IoT services). In addition, the nature of the workload (e.g. need for support of specific network protocols) and the actual network structure, dealing with applications designed with large numbers of components that need to communicate with each other may be required. To ensure workload deployments while at the same time not negatively impacting performance, east-west communication for end-to-end services across different domains may be necessary. Similarly, e.g. for applications following users on the mobile edge, lack of support for migration of containers and for connecting containers to multiple networks, could be a challenging issue.

In terms of security, role-based access is provided, however additional solutions are needed for data encryption. Other gaps in container management include the lack of a dynamic carrier-grade orchestration, the management of containers across different sites and data centers and the lack of centralized controllers for networking aspects with respect to provisioning, resiliency, management/coordination, automatic network configuration and visibility of traffic from container-to-container across networks -whether physical or virtual.

To summarize, for telco service providers to embrace the cloud-native approach some adjustments must be made to support inherent network requirements and offer features related to orchestration, monitoring and alerts, load balancing & high availability. The challenge for telco-grade web-scale

platforms is to automate network configuration for rapid container deployment, monitor container instances and provide a consistent network operational experience, ensuring KPIs are met.

In order for the above to be met the following are necessary:

- Automation of operations, upgrade, backup and restore as well as restart capabilities
- Monitoring, performance counter, alarms and logging/trace support
- Security and tenant isolation.

## 4.2 Twelve Factors in Telco Domain

It is expected from telcos to adopt Cloud-native design principles to support vertical services offerings and their own network management. The ecosystem of microservices become richer as result of the pressure from big web players (e.g. Google Cloud Platform, Amazon Web Services, Microsoft Azure Stack) that have a first mover's advantage on container (Docker), orchestrator (Kubernetes, Mesosphere, Docker Swarm) and microservices (e.g. Istio, Envoy, Linkerd) technologies and their investment on specialized human capital. This guided transition toward adopting and thinking Cloud-native needs to address the "cloud-native" nature of application development, configuration and deployment, i.e. a way to reconcile with the Twelve Factors guidelines mentioned in section 2.2.

Participants from 5G-PPP Phase 1 and Phase 2 projects have diagnosed these needs and moved towards the goal of such adaptation. Telco-grade requirements and challenges have been identified to map the road ahead towards telco-grade webscale computing. One can separate these concerns in two groups, one that have to do more with the development stage of applications and one that refers to the deployment and post-deployment aspects of telco-grade cloud-native applications/services/functions.

On the application development side, the whole pipeline needs to be rethought. To begin with, Build/Release/Run (12.V) is not a part of ETSI standards. Furthermore, software design patterns cannot directly be translated from the IT/webscale world to the telco one. Microservices are well established in the webscale but new and enhanced architecture patterns have been bred to accommodate for network connectivity between application components (e.g. the service mesh pattern). This has also to do with the fact that the proposed telco service orchestrators are richer than current container orchestrators (i.e. Kubernetes) and their functionality that carries over to the network plane has to be also taken into account by the developer.

In addition, dependency declaration and management (12.II) should be of particular concern partly because of the different hardware and subdomain due to vendor lock-in, brownfield setups on L2/L4. This concern can carry over to the codebase (12.I) and Dev/Prod parity (12.X) which should accommodate various deployments strategies over a potentially multi-cluster multi-vendor environment. This will be addressed gradually by new breeds of Developers/DevOps and System Engineers who will be trained holistically in hardware and software technologies and by a standards or best-practice convergence in near future.

However, factors 12.VIII (Concurrency) and 12.VI (Processes) 12.IX (Disposability) and 12.XI (Logs) pose the greatest challenges ahead for the participating 5G projects. Concurrency, i.e. the inherent ability to scale out expects stateless and shared-nothing functions. This is hardly the case for all telco VNFs at the current state and demands either a ground-up rethinking of the NFs implementation or well thought out scalable abstractions for VNFs, NFPs and VNFFGs of thereof. Logs (12.XI) speed, heterogeneity and variability of origin will force a readjustment of current telco methodologies. Nevertheless, telcos can take lessons from IT grade solutions and systems that integrate heterogeneous logs in real-time, such as pub/sub streaming tools. Disposability (12.IX), i.e. small start-up times and graceful termination, is major concern for five nines availability systems, especially for emergency services and in relation to QoS constraints.

In addition, the factor on administrative processes (12.XII) seems a key aspect, and especially critical since it has less to do with technology evolution.

## 4.3 Telco-Grade Features

Despite the undeniable convergence of cloud and network, there are still key differences between IT and NFV (Network Function Virtualization) environments. Such fundamental differences impact the way the telco applications need to be managed. Indeed, NFV ecosystem requires novel management and orchestration techniques to achieve the required carrier grade performances. In this context, micro-services and containers are appealing candidates for implementing telco applications due to their small footprint and fast start times. The architecture of containers enables an efficient deployment of micro-services. In fact, an application is deployed as a suite of small services where each one is running in its own container and communicating with lightweight mechanisms.

Several container management frameworks exist such as: i) Kubernetes, ii) Mesos, iii) Swarm, etc. However, all of them have been designed to suit IT applications. Consequently, they lack a number of mandatory features for the management and performance guarantee of Telco applications. To deal with such limitations, these platforms need to be customized to close the gap with regard to NFV requirements. In this context, several features must be integrated to the aforementioned frameworks:

1) *Multi-network interfaces*: contrary to IT applications, Telco workloads may require sophisticated network models to support multi-homing with various QoS as specified by ETSI MANO. Indeed, a VNF Component (VNFC) can be connected to different networks. The latter can belong to different planes: control, management and data. Also, they may be associated with different QoS requirements and different network isolation domains.
2) *Service function chaining*: in a Cloud environment, IT applications are generally discrete workloads. With an NFV environment, telco applications must be configured together as a service through which traffic needs to be correctly steered. This Service Function Chaining (SFC) requires an explicit definition of the network topology and a distinct ingress and egress network ports specification when necessary. In this context, the support of multiple networks per micro-service is a prerequisite for SFC and could allow chain of containers to be managed through extra network interfaces.
3) *Data plane acceleration*: existing container's network models are suitable for IT applications as they provide easy, reliable networking and generally good throughput. However, for Telco applications, additional network properties are mandatory to ensure deterministic performances. Indeed, we need to ensure i) low packet processing time to reach very low latency, ii) predictability and stability of packet processing delay to minimize jitter, iii) guaranteed and prioritized bandwidth for each Telco workload to prevent network contention, iv) low CPU consumption for network tasks to allow low energy usage, and v) native network performance for Telco workloads. To respond to the aforementioned requirements, two complementary data plane acceleration techniques can be used. The first technique permits to bypass the kernel network stack and move the data processing in user space (e.g., DPDK, VPP). The second technique bypasses the hypervisor and the virtual switch and gives multiple VNFs direct access to the same physical network interface controller (e.g., Single Root – I/O Virtualization). It is worth noting that these acceleration techniques may be used within the NFV infrastructure (e.g., virtual switch or router) or within the Telco workload themselves (e.g., DPDK-aware virtual network function with poll-mode driver).
4) *Enhanced platform awareness*: container management frameworks need to acquire a greater awareness of the underlying platform's capabilities. Such a feature will address issues related to resource abstraction which can negatively impact the performance of SLA critical Telco applications. By doing so, Telco workloads can benefit from the access

to a certain platform features to improve their performance or to increase the stability and the predictability of their behaviour. In this context, various compute related capacities can be cited: i) CPU pinning to avoid unpredictable latency and host CPU overcommit by dedicating CPUs to the Telco containers, ii) Numa awareness to improve the utilization of compute resources for Telco containers that need to avoid cross NUMA node memory access, and iii) huge pages to accelerate the memory management by using larger page sizes.

5) *Environment aware scheduling*: current container schedulers, which are responsible for the placement of containers, lack environment awareness. Indeed, no contextual knowledge about the application requirements, environment properties and network topology is provided. To deal with such a limitation, platform capabilities need to be detected first through i) discovery, ii) tracking, and iii) reporting of enhanced features and then communicated to the container orchestrator. Besides, to ensure a network aware scheduling, the scheduler needs to acquire a global view of the network. In this context, new placement constraints need to be supported such as: i) affinity and anti-affinity inter-VNFC constraints, ii) network SLA (bandwidth, latency, etc.) with regard to network links between components, and iii) energy consumption.
6) *Multi-site support*: with IT environment, applications are generally hosted in centralized datacentres. However, with NFV environment, some Telco workloads must be distributed to the network edge in order to be nearer to the users, and hence minimize latency. Consequently, container management frameworks need to be extended to support the deployment and the management of Telco applications across multiple datacentres, geographical locations and administrative domains while maintaining a unique control plane.

# 5 Technology Enablers – Open source and Standards

Cloud native applications can be built using a large number of open source projects. There are also a couple of standard developing organizations creating standards and reference platforms relevant to the development of cloud native applications or to virtual applications in general. In this section we summarize these standards and open source projects.

## 5.1 Standards

### 5.1.1 ETSI NFV

ETSI NFV (Network Function Virtualization) [7] has defined a framework to provide network functions in a virtualized way. This framework is applicable in general to cloud-ready applications, but it can be applied to cloud-native applications as well. More specifically, ETSI NFV is studying a classification [8] of cloud-native Virtual Network Functions (VNF) and proposes architectural enhancements of the NFV framework [9].

The classification of cloud-native VNFs [8] uses a number of non-functional parameters:

- Resiliency: Service failures can be repaired by reconfiguring the service to use new instances of VNF components (VNFC).
- Scaling: Scale in/out should be supported.
- Composition: VNFs are composed from VNFCs, each performing a single capability.
- VNF design for location independence: VNFC functionality should be independent of resource location. This independence might be constrained by availability of hardware

accelerators, affinity constraints to achieve required performance, anti-affinity constraints to improve resiliency, and regulatory constraints.

- VNF state handling: Stateless applications are preferred.
- Use of services for VNF architecture: Cloud-native systems should be micro-services oriented. Benefits of micro-service based systems are seen in the areas of independent and fast deployments, elasticity, reduced side-effects among services. Drawbacks are seen regarding the size and number of units and the transactions among them.
- Use of containers: VNFCs or micro-services should run in containers due to their lightweight characteristics.
- Zero-touch management: VNFs should be automatically or self-configured. This applies to self-healing and self-optimization as well.
- Load-balancing: It is expected that a load-balancer distributes the load among VNFs or VNFCs.

[9] describes the potential impact on the NFV architecture of providing Platform-as-a-Service (PaaS) type capabilities and supporting cloud-native VNFs. PaaS type capabilities could be relevant to provide backing services such as service registries as part of the platform. PaaS services may be deployed as VNFs themselves, which can be accessed from the actual applications, they can be deployed as a new kind of resources in the NFV infrastructure, or they can even be deployed as a new type of object with its own management and orchestration.

### 5.1.2 TM Forum

The TM Forum investigates how virtualized applications can be operated [10]. With the introduction of VNFs, not only the application itself needs to be managed but also the underlying cloud infrastructure hosting it. Traditionally, these are managed separately but have to work together. The dynamic aspects of virtualization being able to instantiate and terminate VNFs increases the need to reduce the effort for managing VNFs. Here, TM Forum is investigating zero-touch orchestration, operation, and management (ZOOM, [11]). This project has been developing best-practices to support technology and business transformation due to the introduction of network function virtualization and software-defined networks. Although this work is driven by the introduction of NFV, the project develops a hybrid management platform for both physical and virtual infrastructure. Regarding the lifecycle management (LCM) of network resources, it investigates how to automate the complete end-to-end lifecycle as well as license management of virtual resources.

### 5.1.3 Open Networking Foundation

The Open Networking Foundation (ONF) defined a reference platform for telco central offices: Central Office Re-architected as a Datacenter (CORD, [12]). This project aims to transform the edge of the operators' networks into agile service delivery platforms. The platform is based on SDN, NFV, and cloud technologies, it integrates several open-source projects to create a complete operational edge datacenter using cloud-native design principles.

The platform allows to deploy both virtual machines and containers. More specifically, CORD contains a platform named XOS to assemble and compose services [13], and implements a service control plane. XOS itself is organized as a collection of micro-services. The XOS core includes a language for specifying both models and invariants on those models and a toolchain that generates code to enforce those models and invariants. XOS can be considered a cloud-native implementation of the control plane of services, which themselves may be implemented in a cloud-native way.

## 5.2 Open Source Software

The open-source concept accelerates innovation by enabling any person with internet connection to contribute in the development of a project or a product. Open-Source Software is one of the fields where Schumpeter's creative destruction paradigm can be easily observed due to the amount of disrupting initiatives. Cloud service providers like Amazon or Microsoft Azure usually use proprietary software to manage their cloud. However, the obvious added-value of open-source pushed key players like Microsoft to start adopting and contributing to Open-source Software (OSS). For instance, Microsoft figures in the top three open-source contributors on github according to a recent article of InfoWorld (ref: https://www.infoworld.com/article/3253948/open-source-tools/who-really-contributes-to-open-source.html). While the CNCF's trail map point out some OSS that can be leveraged in the journey to cloud-native, it attempts to categorize a large collection of projects that vary in scope and size as depicted in the below figure. Telcos need to participate in the open source process to be sure cloud-native technologies meet their needs, which are different from the needs of cloud-native companies as it is highlighted in Section 0.

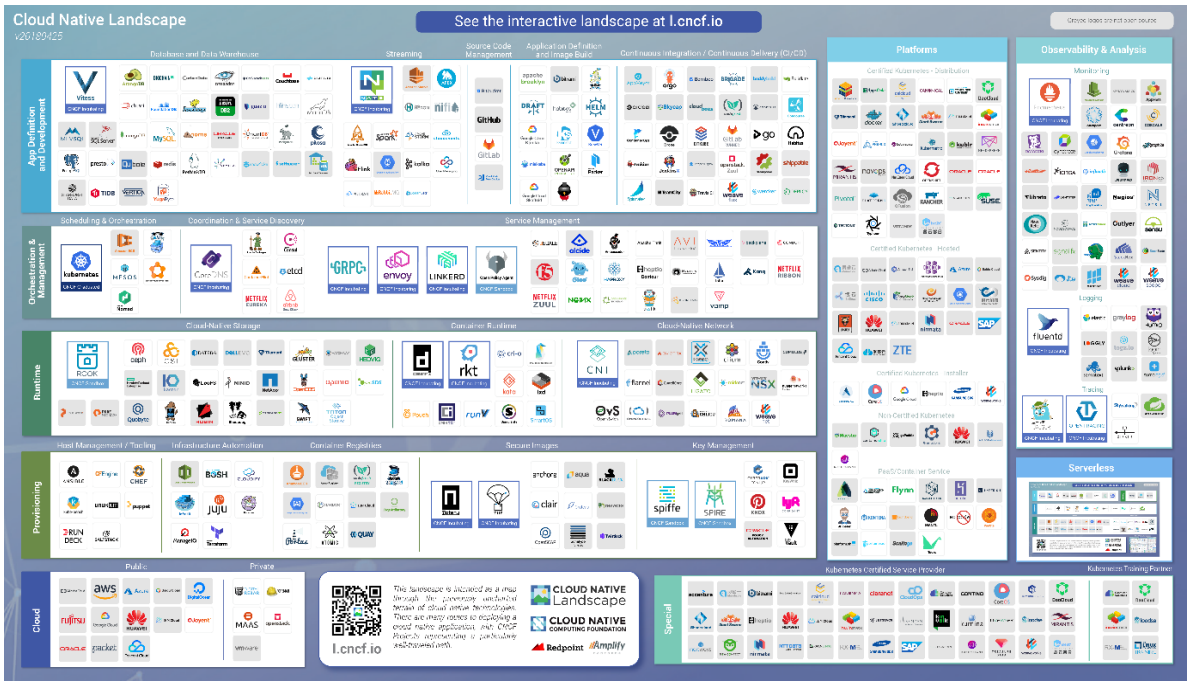


**Figure 4: Cloud Native landscape available online,**
https://raw.githubusercontent.com/cncf/landscape/master/landscape/CloudNativeLandscape_latest.png

A significant number of projects enriches the containers ecosystem, which encourages the implementation of microservice pattern. Moreover, MANO frameworks such as Linux Foundation's ONAP [41], ONF XOS [42], Cloudify [43] and ETSI OSM [44] are starting to evolve towards this approach for microservices. VNFs might be deployed not only as virtual machines on top of a hypervisor, but directly on containers, making use of orchestration mechanisms such as Kubernetes.

However, the weak isolation and security vulnerabilities attached with containers pose an impediment for its use in multi-tenant environments in 5G. The security vulnerabilities can be overcome but with a heavy compromise on performance. Unikernels seems to offer an attractive solution as compared to inherently insecure and not-so-efficient containers and expensive (in terms of resources) virtual machines (VMs). Unikernels are lightweight VMs that contain only the minimum required functionality to achieve the targeted task. In Unikernels, the application is not running on top of any conventional operating system, rather the application is compiled against minimalistic OS functionality which results in a tiny App+OS bundle. One big

disadvantage and barrier for large scale adoption of Unikernels is the manual process for their creation which requires highly skilled resources. The main goal of Unikraft, an open-source Xen incubation project under the auspices of the Linux Foundation, is to automatically build Unikernels targeting specific applications.

fogØ5 is designed to address the requirements characteristics of fog and multi-access edge computing. fogØ5 defines a set of abstractions to unify the compute, storage and communication fabric end-to-end and thus allow applications to be managed, monitored and orchestrated across the cloud-to-thing continuum. The key abstraction that it uses in order to provision generic applications, is the entity. An entity is either an atomic entity, which is a single deployable unit (like a VM or a Container), or a Directed Acyclic Graph (DAG) of entities, an entity maps to an ME application, that can be composed by only one or by more components. fogØ5 is part of Ecplise IoT OpenSource community and aims to provide an open-source platform for Fog Computing.

SDN controllers for software defined networks can be considered mature, with two key representatives: Linux Foundation's OpenDayLight and ONF ONOS. They both have been demonstrated in real networks, and they are currently being adapted to support microservices networking.

# 6 Conclusion

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

This paper aims at demystifying the cloud native for the telcos and demonstrating that the shift to cloud native era is a more a psychological effect than a technological one.

Cloud-native requires a "mindset shift". The way the VNFs are designed, architected and implemented is different. The VNFs are now broken into pieces, support continuous integration and continuous development (CI/CD) and interact using service meshes. The paper started by introducing the key element of this transformation which is the microservices architecture. The three architectures inspired from the domain driven design, the Service Oriented Architecture, the Service Based Architecture and the Microservice Architecture are recalled and compared.

Once the new VNFs, microservice, are designed, the paper introduced a set of technology used to deploy, orchestrate, network and monitor them. Most of the technology is available as Open Source which is another face of this transformation.

Open source helps a lot telcos to implement this transformation and being different. However, telcos can use community-developed open source for technologies that don't differentiate, and focus on filling in the gaps. This is called telco grade features.

The paper identifies the telco requirements that are needed to be supported in community-developed open source. It is clear that telcos need to participate in the open source process to be sure cloud-native technologies meet their needs, which are different from the needs of webscale companies.

# 7 References

[1] Steven Van Rossem, et al., "A Vision for the Next Generation Platform-as-a-Service," in Proceedings of IEEE 5G World Forum, July 2018.

[2] Katsalis et al., "Architectural Design Patterns for the RAN," in IEEE ICC, 3rd International Workshop on 5G Architecture, 2016.

[3] F. Ahmed et al., “Distributed Graph Coloring for Self-Organization in LTE Networks,” Journal of Electrical and Computer Engineering, 2010.
[4] ITU-R, “IMT Vision – Framework and overall objectives of the future deployment of IMT for 2020 and beyond,” [Online]. Available: https://www.itu.int/dms_pubrec/itu-r/rec/m/R-REC-M.2083-0-201509-I!!PDF-E.pdf
[5] 3GPP RAN, 3GPP TR 38.811, V0.4.0 (2018-03), "Study on New Radio (NR) to support Non Terrestrial Networks (Release 15)", 2017.
[6] ETSI MEC, http://www.etsi.org/technologies-clusters/technologies/multi-access-edge-computing
[7] ETSI NFV, http://www.etsi.org/technologies-clusters/technologies/nfv
[8] ETSI GS NVF-EVE 011 V0.0.11, Specification of the Classification of Cloud Native VNF Implementations, work in progress, 2018.
[9] ETSI GR NFV-IFA 029 V0.8.0, Report on the Enhancements of the NFV architecture towards “Cloud-native” and "PaaS", work in progress, 2018
[10] TM Forum, https://www.tmforum.org/virtualization/
[11] TM Forum ZOOM, https://www.tmforum.org/collaboration/zoom-project/
[12] ONF CORD, https://www.opennetworking.org/projects/cord/
[13] ONF XOS, https://www.opennetworking.org/projects/xos/
[14] 5G-PPP Software Network WG, ‘Vision on Software Networks and 5G’, White Paper, January 2017.
[15] 3GPP TS 23.501 V1.0.0 (2017-06), 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; System Architecture for the 5G System; Stage 2, (Release 15)
[16] [online] https://12factor.net/
[17] [online] https://www.heroku.com/
[18] 3GPP TR 23.799, Study on Architecture for Next Generation System, V14, Dec 2016.
[19] [online] http://www.restapitutorial.com/
[20] Sam Newman, Building microservices, Oreilly Edition, February 2015
[21] Thomas Erl, ‘Service-Oriented Architecture (SOA): Concepts, Technology, and Design’, Prentice Hall, 2005
[22] Neal Ford, ‘Comparing Service based Architectures’, Available online: http://nealford.com/downloads/Comparing_Service-based_Architectures_by_Neal_Ford.pdf
[23] Eric Evans, ‘Domain-Driven Design: Tackling Complexity in the Heart of Software’, Book, 20 August 2003
[24] Neal Ford, Mark Richards, ‘Service-Based Architectures: Structure, Engineering Practices, and Migration’, O’Reilly Media, July 2015.
[25] [online] https://github.com/cncf/landscape
[26] White paper, ‘The next step in server virtualization: How containers are changing the cloud and application landscape’, by NuageNetworks, Nokia, 2017, available online: file:///C:/Users/bsayadi/Downloads/PR1512017019EN_NN_Docker_Integration_StraWhitePaper.pdf
[27] Derek Rangel, ‘DevOps: Learn One of the Most Powerful Software Development Methodologies FAST AND EASY!’, Book.
[28] [online] https://jenkins.io/
[29] [online] https://kubernetes.io/
[30] Ericsson Technology Review, ‘Paving the way to telco-grade PaaS’, 2016, available online: https://www.ericsson.com/en/ericsson-technology-review/archive/2016/paving-the-way-to-telco-grade-paas
[31] [online] S. Rajagopalan, L. Ryan, Istio: a modern service mesh’, 2017, https://istio.io/talks/istio_talk_gluecon_2017.pdf
[32] [online] https://linkerd.io/
[33] [online] https://www.envoyproxy.io/
[34] [online] https://coredns.io/
[35] [online] https://www.consul.io/
[36] [online] https://coreos.com/etcd/
[37] [online] http://events17.linuxfoundation.org/sites/events/files/slides/Container%20Networking%20Deep%20Dive.pdf
[38] https://github.com/projectcalico/cni-plugin

[39] M. Hofmann, E. Schnabel, K. Stanley, 'Microservices Best Practices for Java', RedBooks, IBM, Dec 2016
[40] [online] http://www.rgoarchitects.com/Files/fallacies.pdf
[41] [online] https://wiki.onap.org/display/DW/ONAP+on+Kubernetes
[42] [online] https://guide.opencord.org/controlkube_scenario.html
[43] [online] https://cloudify.co/kubernetes
[44] [online] https://osm.etsi.org/gerrit/#/c/5837/5/Release4/K8_Support.md
[45] [online] https://medium.com/flow-ci/introduction-to-containers-concept-pros-and-cons-orchestration-docker-and-other-alternatives-9a2f1b61132c
[46] [online] https://avinetworks.com/what-are-microservices-and-containers/

# 8 List of Contributors

| Name | Company / Institute / University | Country |
|---|---|---|
| **Editorial Team** | | |
| ***Overall Editor*** | | |
| Bessem Sayadi | NOKIA Bell Labs<br>5G-PPP Software Network WG Chairman | France |
| ***Contributors/Reviewers*** | | |
| Nikos Stasinopoulos | Incelligent | Greece |
| Bilal Al Jammal | NOKIA Bell Labs | France |
| Thomas Deiss | NOKIA | Germany |
| Athina Ropodi | Incelligent | Greece |
| Ilhem Fajjari | ORANGE | France |
| Cristian Patachia | ORANGE<br>5G-PPP Software Network WG, Co-Chair | Romania |
| Bessem Sayadi | NOKIA Bell-Labs | France |
| David Griffin | UCL | UK |
| David Breitgand | IBM Research | Israel |
| Josep Martrat | ATOS | Spain |
| Ricard Vilalta | CTTC | Spain |
| Shuaib Siddiqui | I2CAT Foundation | Spain |
| Gabriele Baldoni | ADLINK Technology | France |

**Contact**: bessem.sayadi@nokia-bell-labs.com