Taleb, Tarik; Ksentini, Adlen; Jäntti, Riku

"Anything as a Service" for 5G Mobile Systems

Published in:
IEEE NETWORK

DOI:
10.1109/MNET.2016.1500244RP

Published: 01/12/2016

Document Version
Peer reviewed version

"Anything as a Service" for 5G Mobile Systems

Tarik Taleb
Sejong Univesity and
Aalto University, Finland
talebtarik@ieee.org

Adlen Ksentini
EURECOM, Sophia-Antipolis, France
adlen.ksentini@eurecom.fr

Riku Jäntti
Aalto University, Finland
riku.jantti@aalto.fi

Abstract – 5G network architecture and its functions are yet to be defined. However, it is generally agreed that cloud computing, Network Function Virtualization (NFV) and Software Defined Networking (SDN) will be key enabling technologies for 5G. Indeed, putting all these technologies together ensures several advantages in terms of network configuration flexibility, scalability, and elasticity, which are highly needed to fulfill the numerous requirements of 5G. Furthermore, 5G network management procedures should be as simple as possible; allowing network operators to orchestrate and manage the lifecycle of their Virtual Network Infrastructures (VNIs) and the corresponding Virtual Network Functions (VNFs), in a cognitive and programmable fashion. To this end, we introduce the concept of "Anything as a Service" (ANYaaS), which allows a network operator to create and orchestrate 5G services on demand and in a dynamic way. ANYaaS relies on the reference ETSI NFV architecture to orchestrate and manage important services such as mobile Content Delivery Network as a Service (CDNaaS), Traffic Offload as a Service (TOFaaS), and Machine Type Communications as a Service (MTCaaS). Ultimately, ANYaaS aims for enabling dynamic creation and management of mobile services through agile approaches that handle 5G network resources and services.

Keywords: 5G, mobile cloud networking, NFV, federated networked cloud, carrier cloud, OpenStack, CDN, and traffic offload.

1. Introduction

5th generation mobile networks (5G), also referred to as beyond 2020 mobile communications systems, represent the next major phase of the mobile telecom industry, going beyond the current Long Term Evolution (LTE)[1] and IMT-advanced systems. In addition to increased peak bit rates, higher spectrum spectral efficiency, better coverage, and the support of potential numbers of diverse connectable devices, including Machine Type Communications (MTC) devices, 5G systems are required to be cost-efficient, flexibly deployable, elastic, and above all programmable. The need for lowering the mobile infrastructure costs and rendering their deployment flexible and elastic has become critical for the sustainability of mobile operators worldwide, mainly in light of the ever-growing mobile data traffic on one hand and the stagnant (rather falling) Average Revenue per User (ARPU) on the other hand. With current mobile network designs, such required flexibility and elasticity are all but impossible to realize particularly due to the traditional usage of specific-purpose networking equipment that can neither dynamically scale with mobile traffic nor be easily upgraded with new functions.

Along with recent and ongoing advances in cloud computing and their support of virtualized services, it has become promising to design flexible, scalable, and elastic 5G systems benefiting from advanced virtualization techniques of cloud computing and exploiting recent advances relevant to Network Function Virtualization

[1] For the sake of readability, abbreviations used in this paper are listed in Table 1.

(NFV). Indeed, with regard to the latter, and thanks to the numerous advantages it offers in terms of network configuration flexibility, scalability, and elasticity, NFV has emerged as one of the key visions of the future 5G architecture. Researchers in both industry and academia have been designing new architectures for elastically composing and operating a virtual end-to-end network platform, on demand, on top of fragmented physical infrastructures provided by federated networked cloud [1]. Software-Defined Networking (SDN) techniques have been seen as promising enablers for this vision of carrier cloud, which will likely play a crucial role in the design of 5G wireless networks.

Indeed, the 5G architectural design should be as flexible as possible by relying on NFV, SDN and cloud computing. This flexibility consists in building, on-demand and within very short times [2], instances of Virtual Network Infrastructures (VNIs) along with their composing Virtual Network Functions (VNFs), e.g. Packet Data Network Gateway – PGW and Serving Gateway – SGW, i.e., to scale up to sudden traffic growth. Meanwhile, the trend towards the decentralization/distribution of the cloud architecture, in the format of cloudlet or the so-called federated networked cloud (i.e., edge or Fog computing), should allow the placement of VNFs nearby end-users. This shall reduce the end-to-end communication path between mobile users and corresponding servers, which will definitively improve users' Quality of Experience (QoE). The 5G network system should also allow mobile operators to create and orchestrate, in an easy and dynamic way, network services targeting a specific optimization of network resources, e.g. caching, traffic offload, or optimization of a specific application. In this vein, we devise in this paper the concept of ANYthing as a Service (ANYaaS), wherein on-demand creation and orchestration of 5G services are specified and enabled exploiting the benefits of both cloud computing and NFV. ANYaaS allows the instantiation of one or multiple mobile network-related VNIs, along with their integration, in order to build a specific 5G service targeting a specific network resource optimization. Among the service instances that ANYaaS may launch are the following:

- Mobile CDN as a Service (CDNaaS): It creates and allows the management of a service instance of a virtual CDN whereby CDN VNF instances (e.g., IPTV, Video on Demand, Video caches) are instantiated and strategically placed over the federated networked cloud nearby users and supported by smart caching strategies based on, among others, content popularity and viewers' geographical distribution and mobility patterns. The latter may also serve as a trigger for the instantiation of new CDN VNFs so that video content of interest are cached at nearby cache VNFs following the mobility of an individual or a group of viewers, similar in spirit to the concept of Follow Me Cloud (FMC) [3].
- Traffic Offload as a Service (TOFaaS): It creates and allows the management of a service instance of a VNI, consisting of Local Gateway (L-GW) VNF instances (e.g., alternatively VNF instances of small-scale PGW), over the underlying federated networked cloud. L-GW VNFs are assumed to have the ability to selectively offload IP traffic towards specific networks as determined by the policies of mobile operators. Exploiting the benefits of a distributed cloud architecture, L-GW VNFs may be instantiated at cloudlets nearby the Radio Access Network (RAN) to achieve efficient utilization of network resources, similar in spirit to the Mobile Edge Computing (MEC) concept [11].

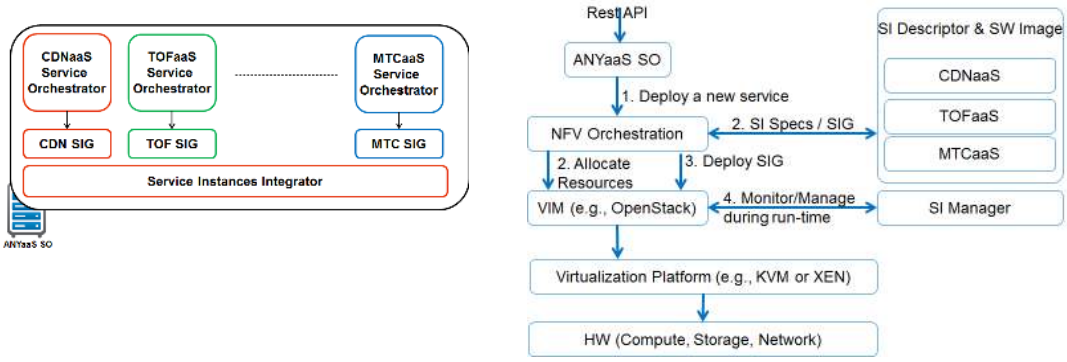
The envisioned ANYaaS concept may create one single service instance operating individually or multiple correlated service instances ensuring efficient integration between them. As an example of multiple services, CDNaaS and TOFaaS can be created, in an efficiently integrated fashion, to complement the role of each other. Further details on this use case will be given in Section 4.

This paper is organized as follows. Section 2 details the ANYaaS concept and describes its orchestration system and how it is integrated within the ETSI NFV reference architecture [4]. Section 3 takes the offering of CDNaaS and TOFaaS as a use case to describe in detail the working of ANYaaS. Section 4 discusses technological trends that allow the creation of a Proof of Concept (PoC) of ANYaaS. Using CDNaaS as an example, the section also investigates which virtualization approaches (Virtual Machine or Container) is appropriate to CDN VNF. Finally, the paper concludes in Section 5.

Table 1: Glossary.

Abbreviation	Name
ANYaaS	Anything as a Service
ARPU	Average Revenue Per User
CDNaaS	Content Distribution Network as a Service
DC	Data Center
EM	Element Manager
EPC	Evolved Packet Core
FMC	Follow Me Cloud
HSD	Horizontal Serial Deployment
L-GW	Local Gateway
LIPA	Local IP Access
LTE	Long Term Evolution
MANO	Management and Orchestration
MME	Mobility Management Entity
MTCaaS	Machine Type Communications as a Service
NFV	Network Function Virtualization
NFVI	NFV Infrastructure
PM	Physical Machine
PoC	Proof of Concept
QoE	Quality of Experience
RAN	Radio Access Network
SA	Service Area
SDN	Software Defined Networking
SI	Service Instance
SIC	Service Instance Components
SIG	Service Instance Graph
SIPTO	Selective IP Traffic Offload
SLA	Service Level Agreement
SO	Service Orchestrator
TOFaaS	Traffic Offload as a Service
VIM	Virtual Infrastructure Manager
VM	Virtual Machine
VNF	Virtualized Network Function
VNFC	Virtualized Network Function Component
VNI	Virtual Network Infrastructure
VSD	Vertical Serial Deployment

2. Anything as a Service: Concept Description



(a) Abstract level architecture of the ANYaaS service orchestrator

(b) Management lifecycle of Service Instances

Fig. 1. ANYaaS concept.

Carrier cloud reveals a new way to ease the management of mobile networks, adding to them flexibility, elasticity, and programmability features, provisioned as potential 5G services. Carrier cloud is about enabling the on-demand edification of a carrier-grade mobile network on the cloud in an elastic way. In other words, the aim is to enlarge the service domain of cloud computing from the traditional way, whereby storage and computing are provided as a service at data centers, to the provision of mobile connectivity (as well) as a service. In this vein, we introduce the concept of ANYaaS, wherein solutions to optimize 5G resources are presented to mobile operators through a unique framework, allowing the on-demand instantiation and orchestration of services such as dynamic caching of video data (CDNaaS), offloading specific traffic (TOFaaS) or building lightweight EPC to handle the traffic of specific applications (e.g. MTC) [2]. An abstract level architecture of the ANYaaS Service Orchestrator (SO) is depicted in Fig. 1(a). ANYaaS SO comprises a number of specific-purpose SOs that create graphs for their offered Service Instances (SI), namely Service Instance Graph (SIG). A SI is a single instance of a certain service type (e.g., CDN, MTC, and TOF). It may consist of multiple Service Instance Components (SIC), e.g., cache VNF in case of CDN and L-GW VNF in case of TOF. SO of each SI gives as output the corresponding SIG, which is a network map of nodes (and links between them) hosting the SICs of a SI, over the underlying federated networked cloud. SI Graphs are formed based on different VNF placement algorithms [12] and taking different metrics relevant to the SI (e.g., caching strategies in case of CDN, L-GW VNF placement in case of TOF considering the relative geographical distances between L-GWs and end-users on one hand, and L-GWs and Caches of associated CDN on another hand). For instance, TOFaaS SO provides a TOFaaS SIG that indicates where to place L-GW VNFs over the underlying federated cloud along with appropriate specifications of the VMs that will be running the L-GW VNFs in order to efficiently offload traffic of a specific application. ANYaaS SO receives monitoring information about the performance of each instantiated VNF to decide on the corresponding SIG. For example, it may instantiate more lightweight vEPC VMs to cope with a sudden increase in MTC traffic. It may also release or migrate VMs when demand for a specific service changes – to reduce the number of cache VNFs of a CDN SI when the cached content is no more popular or users interested in the cached content have moved to another location. The ANYaaS SO may be controlled by a specific high-level web interface or by providing a REST northbound API. The provided REST API should allow an abstraction of the configuration of the ANYaaS service. Through the Service Instances Integrator, the ANYaaS SO orchestrates several SIs, considering synergies between them and updating, when

necessary, their corresponding SIGs. Whilst Fig. 1(a) shows the case of specific service orchestration (i.e., CDNaaS, TOFaaS and MTCaaS), ANYaaS would integrate several other services that aim to optimize network and cloud resources.

The management lifecycle of SIs is performed by ANYaaS SO through interaction with a NFV orchestration system as schematically depicted in Fig. 1(b). A user (i.e., individual as well as enterprise user) first communicates with ANYaaS SO through a web-based interface using the provided REST API. Upon request for the creation of a service, the ANYaaS SO forwards the request including the needed configuration parameters to the NFV orchestrator to create that service instance. Based on the service profile available at the ANYaaS SI portfolio, which includes the configuration and descriptor of each available SI, the NFV SO asks the Virtual Infrastructure Manager (VIM) to allocate resources and deploy the service on the underlying federated cloud. It should be noted that the descriptor and configuration files are service specific. The deployment will be based on the SIG formed by the corresponding SI SO using adequate VNF placement algorithms. The service descriptor also includes the image/software repository of VNFs to be launched. The VIM represents a cloud controller platform, such as OpenStack. The VIM is in charge of the provisioning and deployment of cloud resources, together with the management and monitoring of the deployed SI and its SICs. The SI manager is responsible for the VNF lifecycle, in terms of software update, and scaling and monitoring VNFs throughout the service lifecycle. Monitoring information are made available to the ANYaaS SO and NFV orchestrator through standardized interface (e.g., Or-VNFM interface as depicted in Fig. 2) or through a specific API interface to the ANYaaS SO. Based on these monitoring information, the ANYaaS SO may take decisions on whether to instantiate new SI/SICs or to release resources. It is worth noting that the envisioned architecture of the ANYaaS SO is fully compliant with the ETSI NFV reference architecture as defined by the Management and Orchestration (MANO) working group [4]. Fig. 2 shows how the ANYaaS SO can be integrated within the ETSI NFV reference architecture. In the envisioned architecture, the SI manager includes both the SI VNF manager and the SI Element Manager (EM; i.e., monitoring). Unlike the MANO recommendation, in the proposed architecture, the SI VNF manager is not responsible for scaling up or down a service. It is rather the role of the ANYaaS SO that takes this decision and communicates the determined action to implement (i.e., scale up or down) to the VNF manager. The communication between the ANYaaS SO and SI VNF manager may be carried out through a proprietary interface. The NFV Service Catalogue is similar to the descriptor of the ANYaaS SIs. One SI VNF manager is created per each ANYaaS SI. The associated SIG is communicated to the VIM through the Nf-Vi interface. Note that all the reference interfaces are the same as those defined in the reference architecture. Table 2 summarizes the functional blocks and interfaces used in Fig. 2 as per the MANO definition.

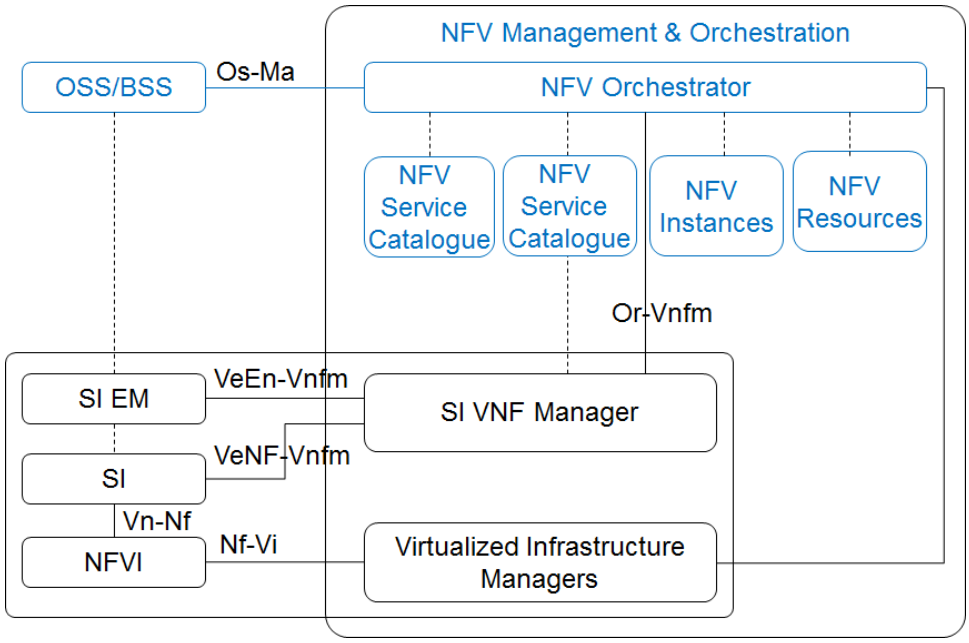
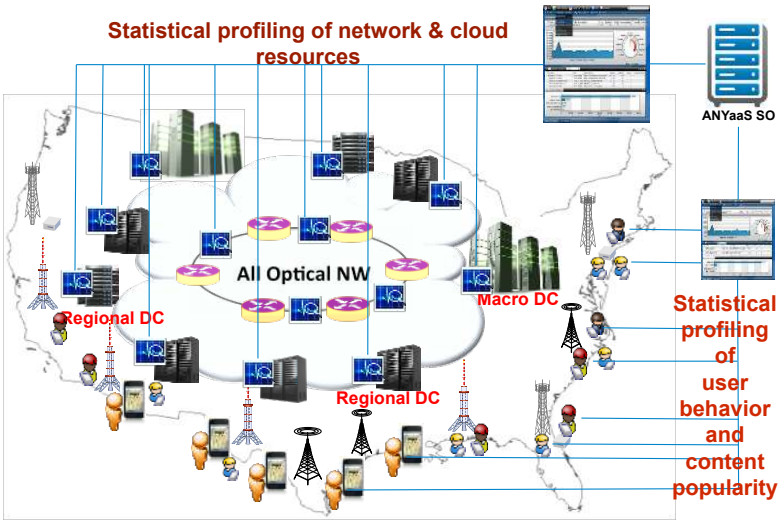


Fig 2. Possible integration of the proposed ANYaaS concept within the reference ETSI NFV architecture.

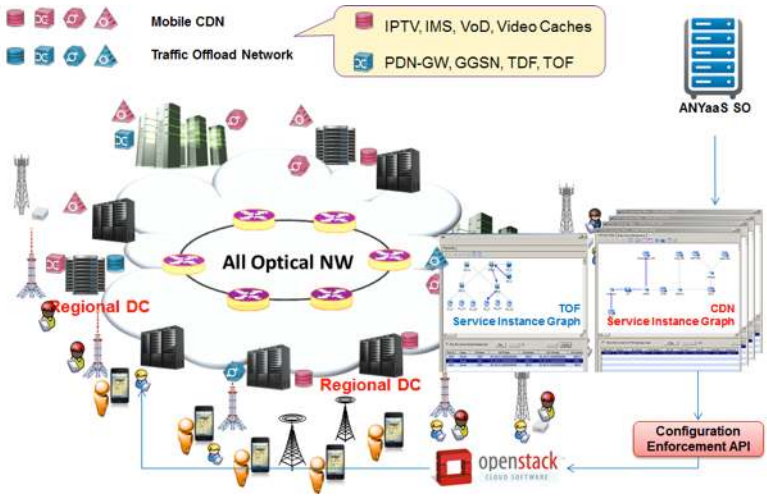
Table 2: Functional blocks and interfaces used in Fig. 2 as per the MANO definition.

Name	Type	Function
NFV Orchestrator	Functional Block	Orchestrates NFVI resources across multiple VIMs and does lifecycle management of Network Services
NFV Service Catalogue	Functional Block	Represents the repository of all VNF packages
NFV Instance Repository	Functional Block	Holds information about available/reserved/allocated NFVIs
VNF Manager	Functional Block	Responsible for the lifecycle management of VNF instances
Virtualized Infrastructure Manager	Functional Block	Control and manage the NFVI compute, storage and network resources
Os-MA	Interface	Used for data exchange between OSS/BSS and NFV Orchestrator (e.g., SI lifecycle management)
Or-Vnfm	Interface	Used for data exchange between NFV orchestrator and VNF Manager (e.g., VNF instantiation and NFVI resource allocation for a VNF)
Or-Vi	Interface	Used for data exchange between the NFV orchestrator and VIM (e.g., NFVI resource reservation/release, VNF software image addition/deletion/update)
VeEn-VFm	Interface	Used for data exchange between Element Management (EM) and VNF Manager (e.g., VNF instantiation and VNF instance query)
VeNf-VFm	Interface	Used for data exchange between VNF and VNF Manager (e.g., VNF instantiation and VNF instance query)
Vn-Nf	Interface	Used for data exchange between VNF SI and the NFV Infrastructure (e.g., southbound API)
Nf-Vi	Interface	Used for data exchange between VIM and NFV Infrastructure (e.g., allocation of VMs with specific compute/storage resources and VM migration)

3. ANYaaS: TOFaaS and CDNaaS Use Case



(a) The monitoring step.



(b) The SIG enforcement and deployment step.

Fig 3. Monitoring-based CDN and TOF SIG creation and enforcement in the envisioned ANYaaS SO.

To describe the functionalities of ANYaaS SO and the lifecycle management of its offered service instances, we consider the use case whereby a mobile operator needs to cope with increasing amounts of video traffic. For this purpose, the mobile operator intends creating a cost-efficient mobile CDN whereby popular videos are strategically cached (i.e., CDNaaS). The mobile operator also wants to have requests to popular videos handled by the created mobile CDN and the corresponding video traffic to be offloaded at traffic offload points nearby its RAN (TOFaaS) and that is not to congest the scarce resources of its core network. For video caching, several techniques have been proposed to place popular video content at several locations in the mobile network, i.e. RAN (eNB) or P-GW [6][8]. Depending on their strategies for video placement in the network, caching techniques could considerably reduce the load on the mobile core network. More details on improving mobile network performance through content caching could be found in [8]. Throughout the remainder of this section, we rather focus on how the envisioned CDN and TOF service instances would be orchestrated and managed using the ANYaaS SO.

Fig. 3 illustrates how the ANYaaS SO instantiates/deploys a CDN and a traffic offload network as services. Whilst Fig. 3(a) shows the step of monitoring the resources of the underlying cloud infrastructure and the end-user's geographical distribution and behavior in terms of video viewership, Fig. 3(b) illustrates the determination of the two CDN and TOF SIGs and their enforcement/orchestration through adequate cloud management platforms (e.g., OpenStack). The ANYaaS SO is assumed to receive feedback from monitoring tools that gather data information about the performance and resource availability of the underlying cloud infrastructure along with the behavior of mobile users. Information on available resources and users' behavioral patterns are used to create profiles on users and cloud performance. If a specific traffic, consuming large network resources, is detected, the ANYaaS SO may request the NFV orchestrator the instantiation of a TOFaaS service instance. This request may be also a follow-up to an explicit request from the mobile operator through the REST API or based on policies pre-configured in the ANYaaS SO. The trigger may also include the traffic pattern, the users' profiles and their locations over the network. The TOFaaS configuration profile may include, among many parameters, the description of the virtual container or the VM description (i.e. the L-GW VNF), the number of TOFaaS instances to deploy, the specifications of VMs to run the different SICs forming the TOFaaS instances, their placement over the underlying federated cloud, the allocated IP ranges, and the S1-MME interface to enable users to connect to the MME. It is worth noting that (i) the S1-MME interface allows connection to the MME to handle users mobility and (ii) the VNF placement in the network is carried out using specific algorithms such as those introduced in [8][9]. Finally, the VIM takes over and deploys the VMs in the selected DCs according to the SIG provided in the previous step. The VMs are connected to the network through SDN by creating appropriate routing rules through the southbound API (e.g., OpenFlow). A TOFaaS manager is also created, which is in charge of the lifecycle and the monitoring of the deployed VNF. The TOFaaS informs the ANYaaS SO of any changes in the VNF state in order to scale up or down. Moreover, the monitoring information should help to detect the case of VNF failure. The TOFaaS manager is connected to the VIM monitoring tool such as Ceilometer available in OpenStack. The ANYaaS SO can also trigger the migration of a L-GW VNF from one DC to another, in case traffic has moved to another location, e.g. users are moving from one location to another while connecting to the same service.

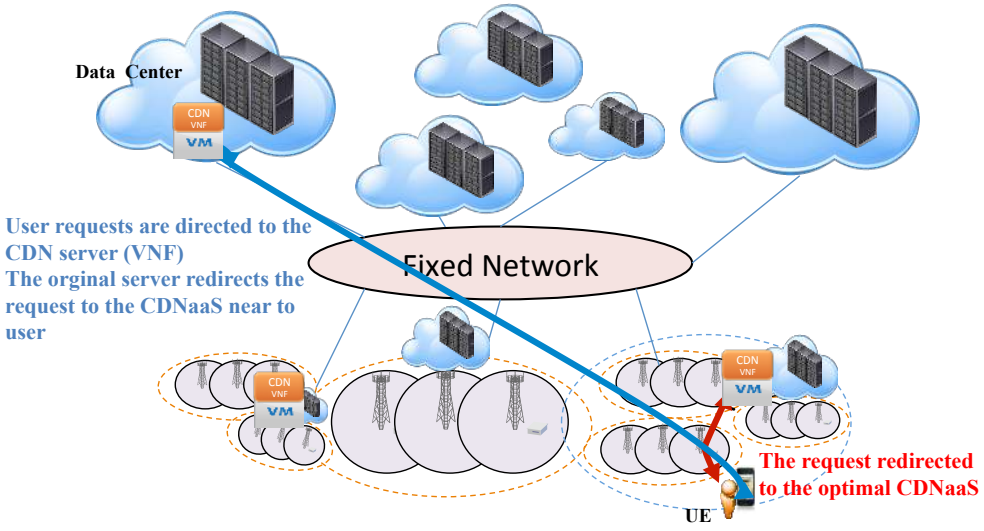


Fig 4. CDNaaS in action.

The deployment of a mobile CDN and its caches depend on the network operator policies and its business relationship with the content providers. Regarding the latter, a content provider may install a CDN server in the

mobile core network, e.g. standalone server or a virtual instance hosted in the cloud (as depicted in Fig. 3). Through DNS (Domain Name System) redirection, all users' requests are redirected to the local CDN server hosted in the mobile core network. If the server has the requested content, it will subsequently forward the content to the user. Otherwise, it will send the request to the original server and cache the response for future requests. To create a CDN SI, ANYaaS OS follows the same procedure as for the creation of TOF service instance, making some important considerations. For example, the storage size is more important in CDNaaS than in TOFaaS since it has an impact on the number of video content that a CDNaaS SIC can cache. Fig. 4 shows an example CDN service instance, whereby users' requests are redirected to the closest CDNaaS SIC.

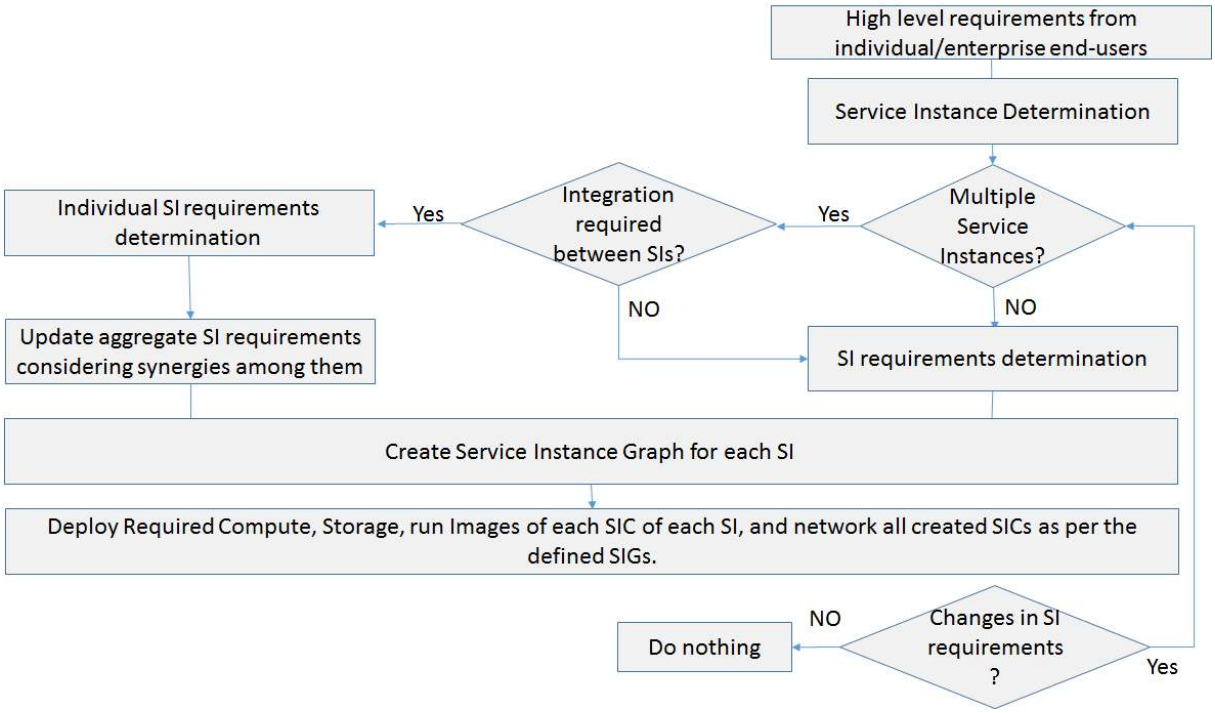
Based on content popularity and its geographical distribution, the ANYaaS OS may specify the locations where CDNaaS SICs have to be deployed. The NFV orchestrator then builds the CDNaaS SI using the configuration descriptor. Based on the number of locations concerned by the CDNaaS SI to deploy, the CDNaaS descriptor specifies the number of VNFs to deploy, their configuration and their placement over the federated cloud. The descriptor also indicates the CDN software to be used for CDNaaS VNFs. Notable examples of such CDN software are Squid[2] and NGINX[3]. The VIM then instantiates the VMs and creates the CDNaaS manager to handle the different CDNaaS SICs. The manager agent monitors the CDNaaS SICs and makes this information available to the ANYaaS SO in order that the latter reacts (scale up or down) to any change in the network traffic or to the failure of a VM. Finally, the VMs are interconnected together through a suitable SDN technology (e.g., OpenFlow) building a hierarchical CDN, i.e. if one CDNaaS SIC does not have the content it can ask another CDNaaS SIC before asking the central CDN server. Once the CDNaaS SICs are active, the CDN server is updated in order to take the new CDNaaS SICs when serving future users requests as illustrated in Fig. 4.

So far, we have shown the situation where ANYaaS SIs are separately built; independently of each other. However, there are cases where two or more SIs may be correlated and their respective performances may improve if their respective SIGs are jointly formed rather than being separately formed independently of each other. Throughout the remainder of this section, we show how the Service Instances Integrator of the envisioned ANYaaS SO can improve the performance of the network when integrating SIGs of both CDNaaS and TOFaaS SIs. For this purpose, we consider the case whereby a mobile operator receives a request to stream a live event over a specific region targeting a specific population of mobile users. Such use-case could represent a local event, e.g. a music concert not broadcast on TV. This use case is different from the CDNaaS use-case, presented in the previous section, as there is no agreement being established with a content provider and accordingly no central CDN server is needed. Furthermore, caches should be carefully placed over the federated cloud. As stated in [7], the cost in terms of the communication path in the mobile network is not inversely proportional to the distance of the cache location to the eNodeB. The cost is rather a convex function of the relative distance of the locations of caches to eNodeBs. For instance, according to the presented results, placing a cache at two hops distance from an eNB and associating it to a L-GW represents the best solution. Accordingly, this use-case requires forming CDNaaS and TOFaaS SIGs together to optimize the cache placement. One of the main challenges in this use-case relates to the orchestration of both service instances, i.e., which service instance should be built first or whether the orchestration should create both service instances in an atomic way. In the following, we illustrate how the ANYaaS SO addresses this challenge.

---

[2] www.squid-cache.org

[3] http://wiki.nginx.org/Nginx



Algorithm 1. ANYaaS SO pseudo-algorithm to decide whether to jointly or separately form SIGs.

Upon receiving a request from a content provider, the ANYaaS SO sends a notification to the NFV orchestrator to build the respective CDN and TOF SIs. The ANYaaS SO indicates that multiple SIs are needed and integration between these SIs is required. The SI integration is deemed by the ANYaaS according to some logic or explicitly indicated by the content provider or mobile operator. The NFV orchestrator then uses the service descriptor to indicate the creation of integrated services, CDNaaS and TOFaaS. The most critical step is done at the service descriptor and configuration where SIC placement in the federated cloud is decided. The SIC placement algorithm begins by determining where to place the TOFaaS SIC that maximizes the coverage of users, which would be interested in the content. Furthermore, L-GWs should be, in general, at a distance from RAN nodes that optimizes the network as stated in [7]. Upon the instantiation of the TOFaaS SI, the CDNaaS SICs are placed near to the L-GW SICs. After that, the NFV orchestrator builds the SIG map and sends a request to the VIM in order to deploy the SI according to this map. The VIM deploys the VMs to host the different SICs of the two TOFaaS and CDNaaS SIs creates the SI managers in charge of the lifecycle of the two SIs. The SI manager sends the monitoring information to the ANYaaS SO, which can ask to change the SI requirement if the SI placement is not efficient. For instance, if users' QoE is lower due to the fact that SIG is not optimal, i.e. SIs are not placed at the optimal positions or the users have changed location. Algorithm 1 summarizes the pseudo-algorithm used to decide whether to jointly or separately form SIGs.

4. ANYaaS: VNF performance challenges and solutions

4.1. Implementation issues

As illustrated in Fig. 2, ANYaaS SO's functional blocks are compliant with the ETSI NFV reference architecture. The envisioned ANYaaS SO uses mainly the same standardized interfaces to communicate with the referenced architectural blocks, e.g., the NFV orchestrator and the VNF manager. The Open NFV[4] (or OP-NFV) initiative (under Linux Foundation) is seeking to create integration projects to have a reference implementation

[4] https://www.opnfv.org

of the NFV as per the ETSI requirement and architecture. The OP-NFV initiative combines upstream codes from projects such as Openstack (i.e., VIM), Opendaylight (i.e., SDN controller) and the Linux Kernel, while carrying out extensive testing, custom configuration and possibly upstream code patches. However, the main challenge remains on the performance of VNFs and its comparison to the performance of purpose-built network nodes. Although ETSI has specified the design patterns of VNFs ([5]), the VNF performance improvement is still an open issue. Indeed, the VNF implementation has to mainly deal with *network Input/Output* performance in the virtualized environment. This introduces non-trivial challenges when hosting the VNF on top of a hypervisor, as many low-level details, such as memory access patterns, cache locality, task allocation across different CPU cores, and synchronization primitives, may have a dramatic impact on the overall performance. An alternative to VMs for running VNFs is the container-based virtualization technology. Containers have emerged as a new technology to isolate and run applications on top of an operating system. In this approach, the operating system's kernel runs on the hardware node with different isolated guest VMs installed on top of it, wherein the isolated guests are called containers. With container-based virtualization, no overhead is associated with having each guest running on a completely installed operating system. Whilst this approach could improve the performance, since just one operating system takes care of hardware calls, one disadvantage is that each guest should use the same operating system the host uses.

One recent initiative, namely ClickOS [10], aims at building a technology enabler for NFV based on open source tools. ClickOS is a minimal OS (Operating System) based on XEN software platform optimized for middlebox processing, whereby middlebox refers to all hardware-based network appliances used to run a specific network function (e.g., firewall, Intrusion Detection System – IDS, and Network Address Translation –NAT). ClickOS includes the software modular router, Click, in order to process packets and acts as a router or firewall. As one of the challenges of NFV is the ability to process packets as fast as hardware-based solutions, ClickOS leverages the XEN I/O subsystem by changing the back-end switch, virtual net devices, and back/front end-drivers. The results presented in [4] show that ClickOS is capable to forward packets at around 30Gbps, proving that NFV could achieve the same performance as hardware-based solutions. Furthermore, ClickOS is able to boot in only few seconds. Another emerging and well-established technology that facilitates the development of network I/O intensive applications and hence facilitates the VNF deployment is the Intel Data Plane Development Kit (DPDK). DPDK is a software framework that proposes a set of primitives that ease the creation of efficient VNFs on x86 platforms, in particular high-speed data plane applications. DPDK supposes that processes operate in pull mode in order to become more efficient and decrease the time spent by a packet traveling the server. Indeed, this requires that each process needs to occupy one full CPU core. In other words, DPDK processes are attached to a specific CPU core for optimization reasons.

Therefore, putting all these features together, ClickOS and DPDK represent a highly relevant platform to implement ANYaaS services. Indeed, all mentioned functionalities required by ANYaaS services, such as assigning IP addresses, instantiating cache VNFs, and establishing connections to remote MTC servers, can be easily implemented and run on top of ClickOS and DPDK thanks to their ability to manage and forward high number of packets. Moreover, in order to scale up, ANYaaS services have to be instantiated instantaneously, which could be achieved by ClickOS VMs as they boot in only few seconds.

4.2. Virtual CDN benchmark results

In this part, we shed lights on some results that compare the VNF implementation of a virtual CDN, running a NGINX HTTP server, on top of a container and a VM. Here, the choice of NGINX is motivated by its wide adoption in CDN industries. For instance, Netflix platform is based on NGINX software to deliver their video content based on HTTP streaming. On the other hand, the container technology is based on Docker[5] (with Ubuntu 14.04 image), while the VM OS is Ubuntu 14.04, running on a KVM-Qemu hypervisor. The host OS is using Ubuntu OS, running on top of an Intel Xeon (4 CPU core, 2.8 Ghz) with 8 Gbytes of RAM. The conducted experiments consist in emulating a CDN system based on HTTP streaming, whereby several requests are sent to the NGINX HTTP server that hosts the content. Although this experiment does not reflect the whole ANYaaS system, it allows obtaining a clear indication on the performance of the VNF hosting the virtual CDN instance. We emulated two scenarios: (i) the NGINX server is run on top of Docker containers and (ii) the NGINX server is run on top of a VM. The shown results focus on two important performance metrics for CDN services: response time and the number of requests handled per second. Whilst the former gives an indication on the quality perceived by a user, as high values indicate high latency to access the service, the latter indicates for the service operator the number of services one VNF can handle.

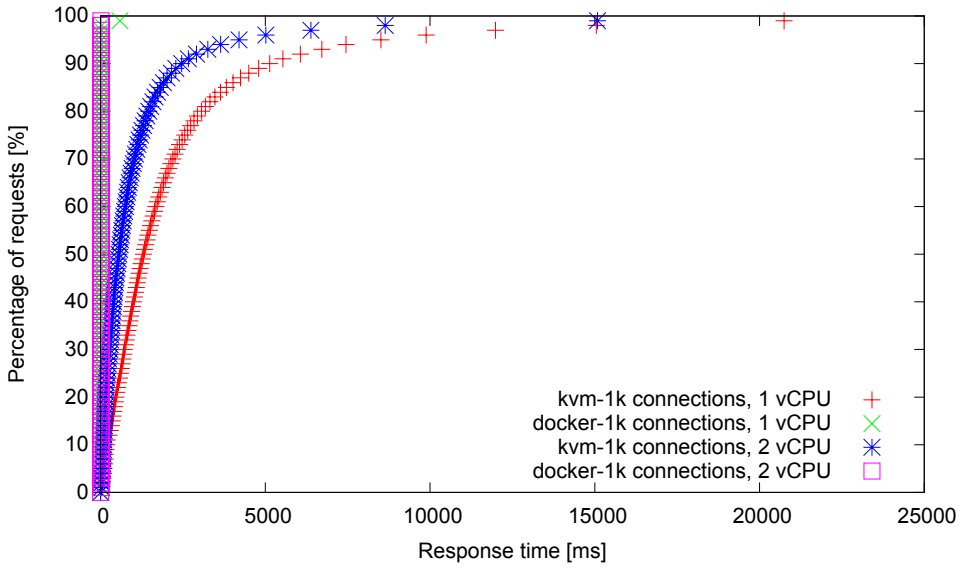


Fig. 5. Empirical CDF of the response time.

Fig. 5 shows the empirical Cumulative Distribution Function (CDF) of response time; each point represents the percentage of requests which were served in a time less than or equal to a specific value on the x-axis. The number of concurrent connections to the server (VNF) was set to 1000, and the results are obtained when allocating one vCPU and two vCPUs to the VNF; one vCPU represents one single CPU core. The HTTP NGINX server hosts a low-quality video stream, wherein the chunk size is equal to 70 KBytes[6]. We observe that Docker outperforms KVM, and achieves high responsiveness regardless the number of vCPUs used. Docker maintains short response times, merely10 ms and 20 ms, for the case of using two vCPUs and one vCPU,

[5] www.docker.com

[6] With Dynamic Adaptive Streaming over HTTP (DASH) technologies, multiple quality/bitrate representations of the same video are stored on a plain HTTP server. A video is segmented in chunks, and a Media Presentation Description (MPD) file describes chunk information. The client receives the MPD file and proceeds by retrieving the video chunk-by-chunk, potentially switching among available qualities.

respectively. However, in case of KVM, 40% and 70% of the connection requests experienced a response time longer than 1s when using two vCPUs and one vCPU, respectively. It reaches for some connections 15 s, which may be inacceptable for a CDN service. These results are attributable to the fact that Docker incurs less overhead for interacting with the operating system and also for network I/O (i.e., especially given our host-mode configuration, which gives it native access to the host's networking stack).

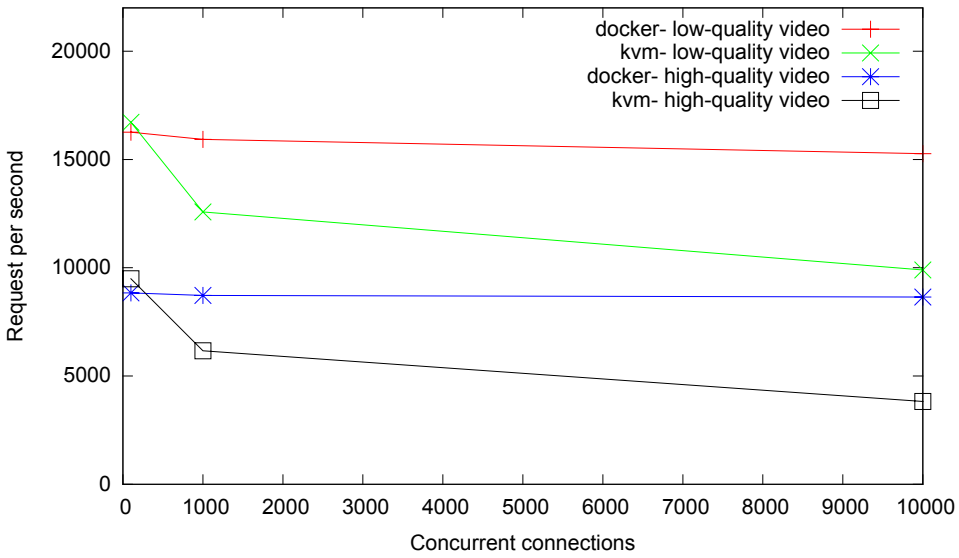


Fig. 6. Number of requests handled per second vs concurrent connections.

Fig. 6 illustrates the performance of KVM and Docker in terms of the number of user requests handled per second, for different numbers of concurrent connections; high numbers of concurrent connections mean highly loaded server. Besides serving a low-quality video, we considered in this test serving also a high-quality video (i.e. chunk size is equal to 450 Kbytes). Moreover, only one single CPU (1 vCPU) is dedicated to the VNF. As for the response time, Docker outperforms KVM as it handles higher numbers of user requests per second for both video qualities. We observe that Docker accommodates more than 3000 requests per second in comparison to KVM for both qualities. Furthermore, it maintains approximately the same number of handled requests regardless the number of concurrent connections.

Clearly, from these preliminary results, it is highly recommended to use Docker as a virtualization approach to implement VNF for CDNaaS. We explain this by two factors: (i) Docker is more lightweight and introduces less overhead when communicating with the host OS; (ii) Docker is highly suitable to application-based services (e.g., CDN), where there is no need for all low level components of an OS. We believe that we might find different results if the tested service requires low level system components such as routing, which is for instance the case of TOFaaS.

5. Conclusion

In this paper, we introduced the ANYaaS concept for the upcoming 5G mobile systems. ANYaaS relies on cloud computing and NFV to ease the management of mobile services that aim for optimizing usage of network resources. ANYaaS allows a mobile operator, in conformance with the ETSI NFV reference architecture, to orchestrate and manage one or multiple service instances on-demand and in a dynamic way. As a potential use case, we demonstrated how the ANYaaS orchestration system enables the creation and deployment of two important correlating services, namely CDNaaS and TOFaaS. Finally, we discussed key technologies that would enable the development of the ANYaaS proof of concept, and provided some experiment results focusing on the

performance of CDNaaS. Particularly, we have found that it is highly recommended to use Docker virtualization approach to build the CDNaaS VNFs.

Acknowledgement

This work is supported in part by the TAKE 5 project funded by the Finnish Funding Agency for Technology and Innovation (TEKES) and in part by the Finnish Ministry of Employment and the Economy.

References

[1] T. Taleb, "Towards Carrier Cloud: Potential, Challenges, & Solutions," in IEEE Wireless Communications Magazine, Vol. 21, No. 3, Jun. 2014. pp. 80-91.

[2] T. Taleb, A. Ksentini, and A. Kobbane, "Lightweight mobile core networks for machine type communications", IEEE Access, Vol. 2, Special section on 5G wireless technologies, 2014.

[3] T. Taleb and A. Ksentini, "Follow Me Cloud: Interworking Distributed Clouds & Distributed Mobile Networks", in IEEE Network Mag., Sep. 2013.

[4] ETSI GS NFV-MAN 001, Network Function Virtualization (NFV) Management and Orchestration (MANO), V. 1.1.1,12-2014.

[5] ETSI GS NFV-SWA 001, "Network Function Virtualization (NFV) Virtual Network Function Architecture", V. 1.1.1, 12 – 2014.

[6] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V.C.M. Leung, "Cache in the air: exploiting content caching and delivery techniques for 5G systems", in IEEE Communications Magazine, Vol. 52, No. 2, Feb. 2014. pp. 131-139.

[7] S. Spagna, M. Liebsh, K. Ozawa, J. Awano, "Design Principles of an Operator-Owned Highly Distributed Content Delivery Network", in IEEE Communications Magazine, April 2013.

[8] M. Bagaa, T. Taleb, and A. Ksentini, "Service-Aware Network Function Placement for Efficient Traffic Handling in Carrier Cloud", in Proc. of IEEE WCNC 2014, Istanbul, 2014.

[9] T. Taleb, M. Bagaa, A. Ksentini, "User Mobility-Aware Virtual Network Function Placement for Virtual Mobile Network Infrastructure", accepted in IEEE ICC 2015, London, UK.

[10] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, F. Huici, "ClickOS and the Art of Network Function Virtualization", in Proc. of USENIX NSDI '14, April. 2014, Seattle, Washington, USA.

[11] Mobile-Edge Computing, Mobile-Edge Computing – Introductory Technical White Paper, available at https://portal.etsi.org/Portals/0/TBpages/MEC/Docs/Mobile-edge_Computing_-_Introductory_Technical_White_Paper_V1%2018-09-14.pdf, last access 26/11/2015.

[12] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes. "Online assignment and placement of cloud task requests with heterogeneous requirements", In Proc. of IEEE Int'l Global Communications Conference (Globecom), Dec 2015

Biographies

Tarik Taleb [S'04, M'05, SM'10] (tarik.taleb@aalto.fi) received the B.E. degree (with distinction) in information engineering and the M.Sc. and Ph.D. degrees in information science from Tohoku University, Sendai, Japan, in 2001, 2003, and 2005, respectively. Prof. Taleb is a professor at the school of electrical engineering, Aalto University, Finland. He was a senior researcher and 3GPP standardization expert with NEC Europe Ltd. He was then leading the NEC Europe Labs Team, working on research and development projects on carrier cloud platforms. Prior to his work at NEC, he worked as an assistant professor at the Graduate School of Information Sciences, Tohoku University. His current research interests include architectural enhancements to mobile core networks, mobile cloud networking, mobile multimedia streaming, and social media networking. He has also been directly engaged in the development and standardization of the evolved packet system as a member of 3GPP's System Architecture Working Group. He is an IEEE Communications Society (ComSoc) Distinguished Lecturer. He is a board member of the IEEE ComSoc Standardization Program Development Board. He is serving as the Chair of the Wireless Communications Technical Committee, the largest in IEEE ComSoC. He founded and has been the General Chair of the IEEE Workshop on Telecommunications Standards: From Research to Standards, which is a successful event that received the "Best Workshop Award" from IEEE ComSoC. He is/ was on the editorial board of IEEE Transactions on Wireless Communications, IEEE Wireless Communications Magazine, IEEE Transactions on Vehicular Technology, IEEE Communications Surveys and Tutorials, and a number of Wiley journals. He has received many awards, including the IEEE ComSoc Asia Pacific Best Young Researcher award in June 2009. Some of his research work has also received Best Paper Awards at prestigious conferences.

Adlen Ksentini (SM'14) received the M.Sc. degree in telecommunication and multimedia networking from the University of Versailles Saint-Quentin-en-Yvelines, and the Ph.D. degree in computer science from the University of Cergy-Pontoise, in 2005, with a dissertation on QoS provisioning in the IEEE 802.11-based networks. From 2006 to 2015, He worked at the University of Rennes 1, as an Associate Professor. During this period, He was a member of the Dionysos Team with INRIA,

Rennes. Recently, He joined the Mobile and wireless networking department EURECOM Institute as an Associate Professor. Adlen Ksentini has been involved in several national and European projects on QoS and QoE support in future wireless, network virtualization, cloud networking and mobile networks. He has co-authored over 100 technical journal and international conference papers. He received the best paper award from the IEEE ICC 2012 and ACM MSWiM 2005. He has been acting as TPC symposium chair for IEEE ICC 2016 and 2017. He was a Guest Editor of IEEE Wireless Com. Magazine IEEE Com. Magazine, and two ComSoc MMTC letters. He has been on the Technical Program Committee of major IEEE ComSoc, ICC/Globecom, ICME, WCNC, and PIMRC conferences.

RIKU JÄNTTI [M'02, SM'07] (riku.jantti@aalto.fi) is an associate professor (tenured) in Communications Engineering and the head of the Department of Communications and Networking at Aalto University's School of Electrical Engineering, Finland. He received his M.Sc (with distinction) in electrical engineering in 1997, and the D.Sc. (with distinction) in automation and systems technology in 2001, both from Helsinki University of Technology (TKK). Prior to joining Aalto (formerly known as TKK) in August 2006, he was professor pro tem in the Department of Computer Science, University of Vaasa. He is an associate editor of IEEE Transactions on Vehicular Technology. The research interests of Prof. Jäntti include radio resource control and optimization for machine type communications, cloud based radio access networks, spectrum and coexistence management, and RF inference.