EASE: EPC as a Service to Ease Mobile Core Network Deployment over Cloud

Tarik Taleb, Marius Corici, Carlos Parada, Almerima Jamakovic, Simone Ruffino, Georgios Karagiannis, and Thomas Magedanz

Abstract

The objective of this article is to demonstrate the feasibility of on-demand creation of cloud-based elastic mobile core networks, along with their lifecycle management. For this purpose the article describes the key elements to realize the architectural vision of EPC as a Service, an implementation option of the Evolved Packet Core, as specified by 3GPP, which can be deployed in cloud environments. To meet several challenging requirements associated with the implementation of EPC over a cloud infrastructure and providing it "as a Service," this article presents a number of different options, each with different characteristics, advantages, and disadvantages. A thorough analysis comparing the different implementation options is also presented.

Mobile data traffic has been growing at an unprecedented rate over the last few years. According to forecasts from Cisco, mobile data traffic will grow at high compound annual growth rates (CAGR) [1]. New mobile applications, high-end devices, and the almost ubiquitous high-bandwidth coverage will further drive this significant growth. Mobile operators are struggling to cope with these increasing data demands coming from an ever-increasing number of tablets and smartphones. Indeed, new technologies, such as Long Term Evolution (LTE), are helping to increase the capacity of radio access networks (RAN). However, mobile operators are not taking full advantage of higher transmission rates provided by the new radio interfaces, placing even greater demands on core networks to support many diverse devices and applications. This shortfall may be attributable to highly centralized core networks and strong dependency on custom hardware components. In addition, the current core network components are not designed with elasticity in mind. Indeed, traffic patterns can be highly dynamic and unpredictable, and operators are forced to overprovision components considering peak hours, keeping them unused during off-peak periods. In this situation it is difficult to balance active sessions with other less loaded components in real-time and even more difficult to increase the components' capacity based on the real-time demands. On the other hand, it is also difficult to optimize resources during non-peak hours. Effectively, this may also lead to resources being wasted in terms of energy, processing, and network (e.g. energy and bandwidth) capacity. The waste of resources is costly and influences directly the Operational Expenditures (OPEX).

Nowadays, the network of a mobile operator is typically built using a variety of network appliances. These include the network entities for control and data planes, the surrounding platforms used to filter, control, and charge the services offered to the end-users, and management functions, as well as the infrastructure needed to deliver services going beyond pure connectivity (e.g. application servers or IP multimedia subsystem (IMS)). All these entities are typically based on custom hardware and need to be statically deployed and configured. Consequently, this whole environment does not support any elasticity or on-demand features. The network is typically dimensioned based on the load foreseen for the next three to five years and the peak hours. For example, to increase the network capacity, it requires the deployment of new entities in specific network sites. Hence, the operation of such a static network is a costly, cumbersome, and time-consuming process. Furthermore, in the current LTE/EPC or any other mobile architecture, any network equipment failure causes tremendous strain on mobile operators, as it may lead to temporary service outage. Indeed, due to the failure of a network node, all active services in that specific node may be interrupted. Consequently, end-users may experience poor quality of the delivered service, such as voice call interruptions and video glitches. Moreover, the session re-establishment of impacted users may generate a significant amount of signaling as new connections with the end users must be established. This may create a signaling storm at the mobility management level and may, ultimately, cause total network breakdown.

The solution to the above problems could be found in the convergence of mobile networks and cloud. That convergence creates the capability for mobile operators to use network function virtualization (NFV) concepts [2], in order to virtualize and decentralize their core network to be able to cope with the upcoming traffic demands. In this way operators can

Tarik Taleb is with the School of Electrical Engineering, Aalto University, Finland (work done while at NEC Europe).
Marius Corici and Thomas Magedanz are with Fraunhofer FOKUS Institute.
Carlos Parada is with Portugal Telecom Inovao.
Almerima Jamakovic is with University of Bern.
Simone Ruffino is with Telecom Italia.
Georgios Karagiannis is with Huawei Technologies.

0890-8044/15/$25.00 © 2015 IEEE

create, scale, and deploy network components whenever they are needed, all in accordance with the particular real-time traffic conditions. In contrast, cloud platforms are built using commercial off-the-shelf (COTS) hardware, primarily to maximize cost-efficiency of the provided services. This type of hardware is cheaper (i.e. as it benefits from large-scale selling), homogeneous, and easy to procure, manage, and maintain. It has different characteristics from the high-end, highly specialized hardware upon which the current generation of mobile core networks (EPC) is built. General purpose hardware does not provide the same level of redundancy of every physical element and is not equipped with dedicated processors, for example, for data packet forwarding or for optimizing processing latencies. The convergence between mobile networks and cloud is challenging, but can be efficiently realized by building these features on top of COTS.

Moreover, cloud services run a virtualized environment in which the physical hardware resources are shared. CPU, memory, storage, and networking are provided in abstract slices to a set of virtual machines (VMs). VMs represent the most granular manageable entity, being the basic building block of every cloud service. Virtualization also provides a number of benefits, including workload consolidation and scaling, that is, to allocate more resources to a VM (i.e. vertical scaling) or to allocate more VMs to a service (i.e. horizontal scaling), both providing in the end more computational power. All of these new features pose new requirements on the design of an EPC for implementation in a cloud computing environment and its offering "as a Service" (aaS). These requirements lead to numerous challenges that need to be tackled in the design of EPCaaS, as detailed later. To cope with these challenges, this article presents a number of different implementation options for EPCaaS, each with different characteristics, advantages, and disadvantages. It is worth pointing out that EPCaaS is an implementation of the 3GPP's EPC, and as such will implement all the standardized reference points and functions, so that, from the perspective of another legacy EPC or individual EPC legacy nodes, it will behave transparently as a non-cloudified implementation of EPC, with full compliance with the 3GPP standards [3]. In fact, combined solutions (cloud-based and traditional/legacy) are some of the most interesting use cases.

The remainder of this article is structured as follows. First a brief overview of the Evolved Packet System (EPS) is provided, highlighting as well some related research work. The main EPCaaS requirements and their associated challenges are then discussed. The following section provides a list of possible EPCaaS use scenarios, and the next section describes and compares an imperative number of EPCaaS implementation options. Then we discuss the management and orchestration of EPCaaS. The article is concluded in the final section.

Related Work

Before delving into recent initiatives that are relevant to EPCaaS, we first give a brief overview of the EPS architecture. EPS includes both the RAN and EPC networks. EPC has been designed as a flat all-IP architecture supporting only packet-based services. EPC aims at providing seamless Internet connectivity between user equipment (UE) and correspondent applications. EPC separates between the control and the user data planes, enabling LTE connectivity using the native RAN — the e-UTRAN (Evolved Universal Terrestrial Radio Access Network) — or legacy technologies (e.g. UTRAN/GERAN). E-UTRAN's main component is eNodeBs (eNBs), which have two interfaces: the X2 interface used to interconnect eNBs, and the S1 interface connecting eNBs to EPC (i.e. S1-MME for control traffic and S1-U for user data traffic). EPC has a simplified flatter structure, consisting of only a few main entities: mobility management entity (MME), serving gateway (SGW), and packet data network gateway (PGW). These entities are responsible for forwarding the user traffic to and from the network, by creating one or more channels with the end user, namely, bearers. The QoS (Quality of Service) level of each bearer is decided by PGWs. To efficiently deliver packets in the core network, the UE states, associated with each EPS bearer (i.e. the signaling radio bearer and the data radio bearer), are maintained in the volatile storage (e.g. memory) of MME, SGW, and PGW (and also in UE). The content details of the EPS bearer can be found in [3]. The GPRS Tunneling Protocol (GTP) [4] and Proxy Mobile IP (PMIP) are among the main communication protocols within the EPC architecture. GTP supports two main components: the GTP-U (GTP user plane), which is used to transfer user traffic in separated tunnels for mobility management over S1 and S5 interfaces; and the GTP-C (GTP control plane), which is used to establish, update, and maintain the GTP-U tunnels. Signaling exchanged over the S11 and S5 interfaces are based on GTP-C. MME uses the S1-AP protocol over the S1-MME interface to transfer radio and GTP tunneling parameters to eNBs. PMIP is an alternative solution for GTP in S5 and S2b interfaces; the S2a interface also uses PMIP.

Network functions virtualization (NFV) has emerged as an important topic of inquiry among all major players in the networking domain [2]. It aims at offering network services using network functions implemented in software and deployed in an on-demand and elastic manner on the cloud. We will mention several recent works on network virtualization enabled by the concept of Software Defined Networking (SDN) so as to virtualize mobile network function over an OpenFlow [5] network.

In several pioneering research papers (e.g. [6, 7]), one sees from-design-to-implementation ideas of how to move toward software-defined mobile networks. For instance, the authors in [6] describe an advancement of EPC utilizing SDN that allows only the control plane to be moved into a datacenter. In detail, they extend the OpenFlow to allow the GTP control plane to be implemented as an application on top of OpenFlow, in this way separating out the GTP control plane and eventually enabling the entire GTP control plane to be implemented in a datacenter. In [7] the authors go one step further and introduce the software-defined mobile network architecture that builds on the full decoupling of data and control into the mobile network user plane and a new control stratum. The key enablers in their architecture consist of the MobileFlow forwarding engine (MFFE) responsible for the user plane, and MobileFlow controller (MFC) responsible for the control plane. Forwarding in MFFE can be fully defined in software, while the control software can flexibly steer user traffic to different service enablers that can be distributed throughout the mobile network. This design is different from an OpenFlow-based network, as MFFEs are not switch-level equipment and must support carrier-grade functionality, such as network layer (i.e. L3) tunneling and flexible charging.

In three recent research works [8–10], virtualization of network functions is furthermore explored. In [8] the authors present a proof-of-concept implementation of the routing network function over an OpenFlow-enabled network, by externalizing routing decisions from the actual equipment. The authors use different scenarios, that is, the separation between IPv4 and IPv6 routing and offering inter-domain routing under the OpenFlow network to demonstrate the applicability of the NFV-powered implementation proposed into actual production environments. In [9] the authors focus on the vir-

Requirements	On-demand creation and lifecycle mgmt.	• Geo-distribution • Auto-scaling • Monitoring (i.e. UE, service instance component, EPC) • Migration
	Interoperability	• 3GPP standards compliant • Backward-compatibility
	Improve O&M	• Easier operations • Explore other features: incremental release upgrade without downtime
	Cloud-native	• A pool of compute, storage, and networking • VM, as the smallest entity that can be be managed • A software platform that controls/ orchestrates VMs
Challenges	High availability (99.999 is a must)	• One cannot assume it is provided by the hardware (COTS) • Fault tolerance and resiliency: design issue?
	Scaling (horizontal, vertical)	• New components added to a running instance must not require complex and costly configuration operations, especially on other running/legacy components
	Load balancing	• Inter-VM, intra-DC, inter-DC • Improvements on standard-defined static and DNS-based mechanisms
	Performance	• Latency for control plane components • Throughput for user plane components
	Management (i.e. O&M, OSS)	• Complexity hiding • What and how to virtualize?

Table 1. EPCaaS requirements and challenges.

tualization of EPC. In particular, nodes such as MME, HSS (home subscriber server), SGW, and PGW, are mapped according to their functions on four alternative deployment frameworks based on SDN and OpenFlow. Their finding points to an easier deployment of those EPC nodes that involves high data packet processing such as tunneling on the data-plane network element, that is, realized by an OpenFlow switch. Hence, they argue for an enhanced OpenFlow network element, referred to as NE+, which contains additional network functions next to the basic OpenFlow protocol. The work presented in [10] devises an entire framework for the creation of end-to-end mobile services, including mobile transport networks on the cloud, via a joint or separate virtualization of the EPC and RAN. Besides these classical research studies, research work presented in [11] addresses the problem of designing and building experimental facilities to exploit SDN and NFV concepts in the wireless networking domain. In [11] the authors present EmPOWER, an experimental testbed offered as an open platform on top of which novel concepts can be tested at scale.

EPCaaS: Requirements and Challenges

This section introduces the most important requirements associated with the implementation of an EPC mobile core network over a cloud infrastructure. As any EPCaaS implementation shall be compliant with the original 3GPP standards [2, 12], EPCaaS shall be designed to provide the same levels of availability, resiliency, service quality, service continuity support, and fault tolerance that are provided today by traditional solutions, while at the same time leveraging the benefits of the cloud, in terms of dynamicity, optimized usage of physical resources, on-demand provisioning of resources on a fine-grained, self-service, and near real-time basis. As such, EPCaaS shall be designed in a way that facilitates:

- Scaling (horizontal and/or vertical).
- Load balancing among VMs, across servers, and across datacenters (DCs) to meet a variable workload in near-real time.
- VM migration and access to a shared backend database (persistent/non-persistent data).
- High availability of the components to provide fault tolerance and resiliency.

EPCaaS shall also support compatibility with legacy systems, supporting a smooth migration path. Most importantly, it shall incur no additional latency to the traffic of both user data and control planes. EPCaaS shall also support the dynamic adaptability of EPC function instances to topological changes in near-real time, monitoring their location, workload, and stability [13, 14]. It shall also support incremental software/release upgrades on EPC function instances without any downtime in service.

Given the above requirements, several fundamental design issues and challenges shall be taken into account when designing EPCaaS. These give rise to the following fundamental questions:

- What are the EPC components that can be virtualized (i.e. deals with complexity of EPC)?
- How are these components virtualized, that is, what are the characteristics and number of VMs on which each virtualized EPC functional component can be deployed?
- How to interwork, that is, not to break interoperability with legacy EPCs and other legacy elements, in particular NMSs (network management systems)?
- How to manage and orchestrate EPCaaS as a whole and in an end-to-end fashion covering both the service layer (i.e. packet data network (PDN)) and RAN [15]?

Table 1 summarizes and categorizes the most important requirements of EPCaaS and associated challenges.

EPCaaS: Scenarios

This section describes a set of important use scenarios of EPCaaS. The primary use case consists of a cloud-enabled mobile virtual network operator (MVNO). MVNO usually refers to the provision of mobile communication services without the need to own a mobile network infrastructure. It is an important player in the mobile telecommunication industry. MVNOs need to perform different functions that can either be handled in-house by an MVNO itself or outsourced to a mobile network operator (MNO), meaning that MVNOs can adopt different operating models. This combination of different functions varies from a simple reselling communication services model to a more complex model, which combines the full range of functions of an MNO, except RAN. In the cloud-based context, a cloud-enabled MVNO means addressing the issue of running services provided by various stakeholders on top of a cloud infrastructure, with the aim of incorporating a minimal function of maintaining a business relationship with the end-users, or a more extensive function of building a specific MVNO type by integrating services provided by (one or all) of the stakeholders involved.

In contrast to an MVNO, an MNO owns a mobile network infrastructure, which includes implementation of all relevant network entities, from EPC itself, through the surrounding platforms used to manage and charge the services offered to the end-users, to the infrastructure needed to deliver services going beyond pure connectivity. The cloudification of EPC and of the aforementioned surrounding platforms creates the opportunity for an MNO to move to a completely different network paradigm, where the network functions (e.g. MME, PGW, SGW, and HSS) that are used to be implemented on physical boxes and deployed on specific points of presence (PoPs), become workloads running on top of a cloud infrastructure. The workloads, instantiated in the cloud, are not statically bound to a specific location, but can be transparently moved between physical servers located in the same datacenter or across datacenters, without causing any service disruption to end-users. In addition to the obvious advantage of reducing the heterogeneity of deployed hardware, this approach allows the evolution from today's mostly static deployments to highly dynamic network implementations, where the network topology, configuration, and dimensioning can be changed over time depending on a variety of factors. It also offers advantages for end-users in terms of improved quality of experience by introducing the concept of dynamic workload placement. Nevertheless, the most significant end-user advantage should be the reduction in the service fees.

An important use scenario furthermore is an end-to-end (e2e) service, which is a composed service that comprises sub-services that are combined together with additional logic and configuration (integration) in order to make a new end-to-end service offering. Typically, in mobile networks there are three main service domains: wireless access service or RAN domain; EPC service or core network domain; and packet data network domain. These service domains can also be offered as compute and storage services as part of a datacenter domain. It is possible to offer these services in an integrated, orchestrated, and uniform fashion, to a stakeholder (e.g. mobile application developer) that offers application services, referred to as the application services provider (ASP), directly to its service consumers (or individual end-users). These services can be offered with additional support services, such as an operations support system (OSS) and business support system (BSS).

EPCaaS Implementation Options

This section describes the development of an efficient EPCaaS architecture to be deployed, in an on-demand and elastic manner, on the cloud. It also provides an overview of an overall supporting architecture. For this purpose, a set of considerations were identified regarding the suitability for running the specific 3GPP EPC functions in a cloud infrastructure, resulting to two high-level virtualization models (as shown in Fig. 1):

- Full virtualization: all control plane (CP) and user plane (UP) functional entities are implemented in VMs. User data must "traverse" hardware, hypervisor, and operating system of VMs. Routing and processing of UP is fully "contained" in the cloud, controlled and managed by the management and orchestration framework described later.
- Partial virtualization: only control plane functional entities are implemented in VMs, while user traffic is forwarded and handled by high performance hardware switches. The orchestration framework, described later [15], manages CP VMs and controls the forwarding of the user plane on the hardware switches, for example using SDN, for example, OpenFlow.

The two approaches stem from the original design of EPC whereby the intention was to clearly separate user data processing functional entities (e.g. SGW and PGW) from control plane functional entities (e.g. MME, HSS, and policy and charging rules function (PCRF)), that is, those responsible for user mobility management, location update, security, and data session set-up. Control and user data plane entities usually have different architectures to cope with different requirements. Currently, both types of entities are implemented in hardware-based devices, but specialized for the different types of processing. In particular, S/PGWs are designed to achieve high throughput of user packets, while performing traffic analysis and applying filtering policies. MME and other control plane entities usually have fewer requirements in terms of throughput capacity, but stricter requirements in terms of processing latency and computation. For this reason, control plane entities can be considered as being more suitable to be virtualized, leveraging the high availability of computing resources in the cloud. Other drawbacks of the virtualization of the user data plane are related to the large amount of traffic coursed and the routing performance advantages of specialized hardware in comparison to COTS. However, this does not mean that user data plane entities are not suitable to be virtualized in specific scenarios.

In the remainder of this article the focus will be on the full virtualization approach, in order to demonstrate that both throughput-demanding services and control/latency-sensitive/computational-intensive services can be supported over cloudified mobile core networks.

As stated earlier, EPCaaS is an implementation of 3GPP EPC. We do not envision adding any new functional entity to the 3GPP EPC standard architecture, nor do we envision modifying interfaces between EPC nodes. We only describe how EPC can be implemented and provided "as a service" in a cloud computing environment, providing compute, storage, and networking as atomic services. We note that EPCaaS is an example of a composed type of service as its entities, used to represent aspects of service orientation, are a service, a service instance (SI), and a service instance component (SIC). Therefore, we define as an implementation option a "mapping" between EPC functional entities and service instance components (SIC), which are eventually mapped on physical/virtual resources. In the following, we will detail the architecture reference models of four implementation options envisioned for EPCaaS, as shown in Fig. 2, highlighting some observations we made about our initial practical implementations [16].

1:1 Mapping — Each 3GPP EPC functionality is mapped 1:1 to a running VM (Fig. 2a), which implements all functions and 3GPP interfaces of that specific entity. For example, one MME is implemented in one running VM, and one PGW on another running VM. This implementation version is especially beneficial for an immediate translation of an EPC to a cloudified version, provided indeed that the legacy software can be deployed on top of a VM. The functional elements for this architecture directly follow the 3GPP standards. The reference points within an SIC and between an SIC and the external entities are the ones defined by 3GPP, thus not requiring any adaptation in terms of communication protocols or state machines.

In this architecture each running VM is state-full in regards to the individual service consumers; that is, it stores the data of the session state for the users, including the UE mobility context, the bearer context, and the security context. The data is valid only for the duration of the user session, not being permanently stored. It is worth noting that some sessions, such as basic connectivity, may have a long duration, for example,

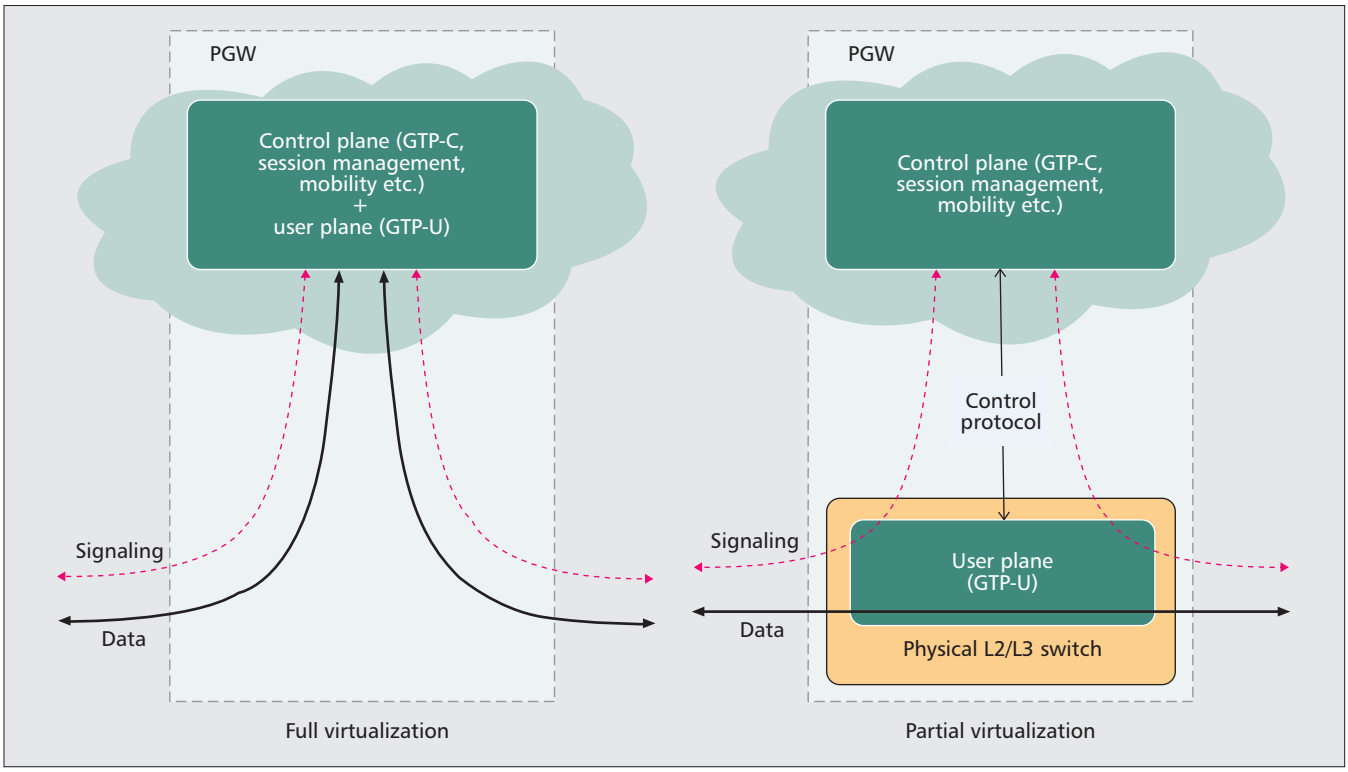

Figure 1. Virtualization approaches: Full vs. Partial EPC Virtualization.

for as long as a device is connected to the network, which is currently increasing with the always-connected mobile devices.

Every running VM requires a specific configuration of EPC-related parameters (e.g. GPRS timers, access point name (APN) definitions, and pre-emption policies), a specific configuration of networking parameters (i.e. IP addresses on the virtual interfaces), and interfaces toward the O&M systems.

To support SIC elasticity, a new component of the same type as the previous ones may be started, moved, or stopped. The introduction of the new component in the system can follow an automated version of the legacy hardware deployment procedures, not requiring modifications on the legacy element management systems (EMSs).

The 1:1 mapping architectural option is conceptually simple, as it requires no major changes to the current deployment model of EPC (one function–one hardware device), and is also the first one we have practically realized [16]. However, it should be noted that this architectural option has some disadvantages. First, the automatic configuration of virtual EPC components presents a scalability challenge. Each virtual component must be configured with a different set of 3GPP-specific parameters, and when the SIC needs more capacity and more virtual components are instantiated, scalability problems may arise in managing and maintaining the configuration of a high number of VMs. In the 1:1 implementation described in [16], a complete EMS system had to be developed specifically for this type of operation. Moreover, there are also impacts on external nodes (real or virtual). For instance, when adding a new MME to an MME pool, a configuration change is needed on all eNBs managed in that pool and this, in turn, poses unexpected requirements on EMSs of eNBs, which today have static behaviors. Specifically, eNBs should be able to recognize dynamic IP address allocations to MME.

Second, instantiating new virtual components to increase capacity is a relatively straightforward operation (instantiate a new virtual component and configure it), but decreasing capacity by turning them off (as the load decreases) is a complex operation, because virtual components are state-full; that is, each of them contains the state of active user sessions, and the virtual components cannot simply be shut down without impacting ongoing user sessions. Transferring this user state to other virtual components is theoretically possible. For example, 3GPP has defined a MME relocation procedure, but these procedures imply additional signaling overhead.

Finally, each virtual component should be managed, as any other core network node in any operator network, by one (or many) external NMS. During scaling, when the number of components increases, there can be additional scalability requirements on the O&M itself, to be able to scale at the same speed. From our implementation perspective [16], NMS and the management and orchestration have comparable complexity in terms of controlling the system deployment and configuration.

1:N Mapping — Each 3GPP EPC network function is decomposed into multiple elements of the following three types, which collectively build a virtual component pool (Fig. 2b):

- The *front end* (FE), which takes care of the communication interfaces toward other entities. It terminates specific interfaces and protocol state communication between entities (e.g. it terminates GTP protocol).
- A stateless virtual component, named *worker* (W), which actually implements the logic of that specific EPC functional entity. Each worker is logically connected to a storage named "operational data storage," which stores log files and possibly other information needed for the basic operations and troubleshooting of the service.
- The *state database* (*SDB*) contains user session state. This is the central point where the state information handled by the workers (W) is stored. This feature makes the workers (W) stateless.

This implementation model, as shown in Fig. 2b, follows the multi-tiered web services deployment model for cloud-based applications. Each network function instance of EPC is therefore implemented by instantiating multiple virtual component pools, each one corresponding to a former 3GPP EPC

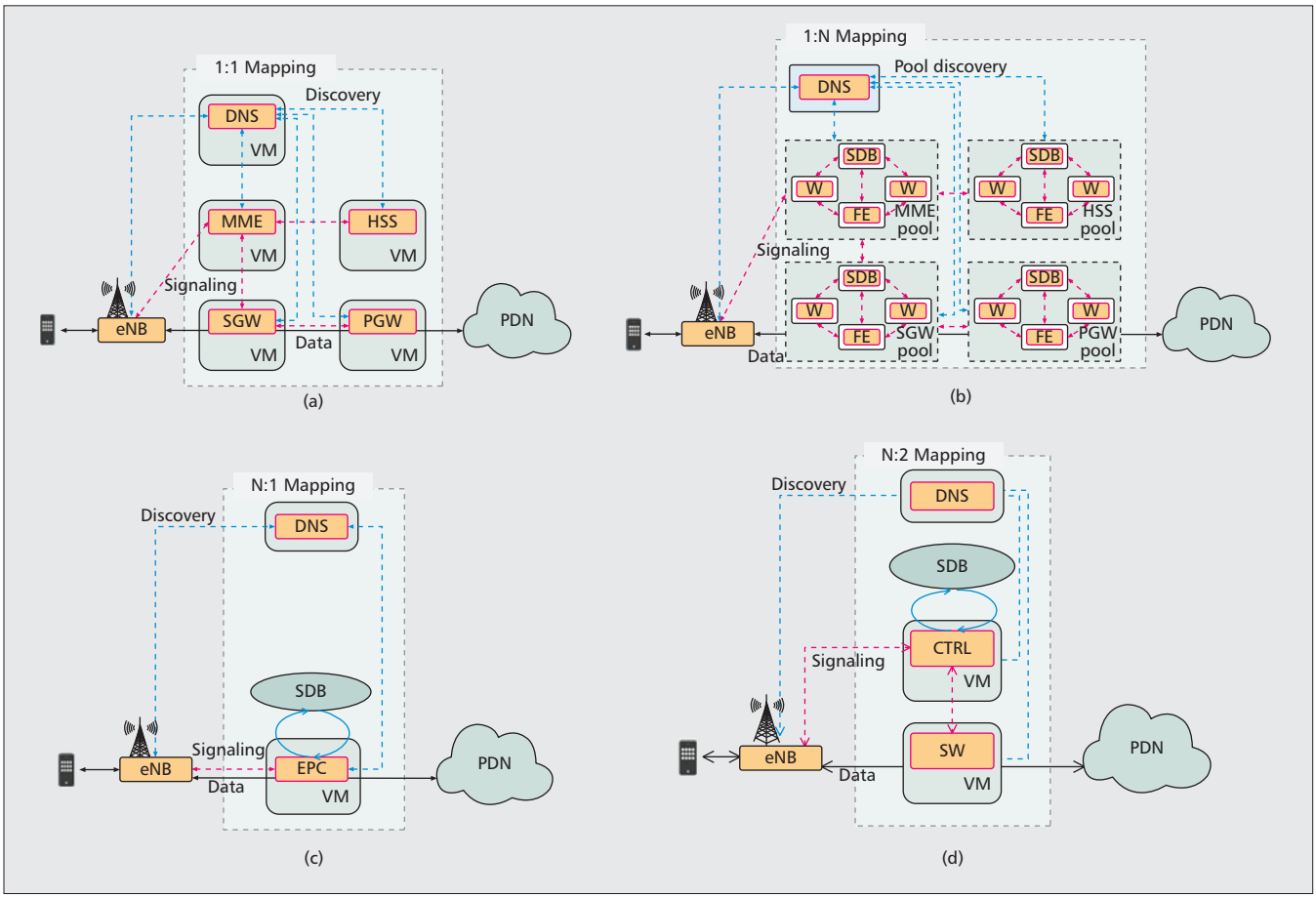

Figure 2. Architecture reference models of four implementation options envisioned for EPCaaS. a) 1:1 mapping; b) 1:N mapping; c) N:1 mapping; d) N:2 mapping.

functional entity. Each component pool is seen by an external entity as a single EPC node, terminating all the 3GPP standard interfaces of that specific entity. As a side effect, only one configuration of EPC-specific parameters is needed per pool. This design simplifies O&M from external nodes, lowering the number of logical management connections compared to the 1:1 mapping.

Workers share the same EPC-specific node configuration and are state-less. As such, they can scale out/in without impacting external connected peers and independently from other VMs, while all the state information is stored and provided by SDB. Therefore, auto-scaling rules can be set for each pool independently from other pools and horizontal scaling.

The scaling of SDB, which is a state-full component, is possible although it requires careful design, for example, for the choice of the database (e.g. SQL or NoSQL). Moreover, load balancing can be implemented in a more granular way (with respect to 3GPP mechanisms, which are based on DNS), exploiting all available workers. Additionally, multiple front ends for the same workers may be placed in the network scaling based on current 3GPP mechanisms.

A final important aspect of this architecture is that it enables high availability of EPCaaS as a whole, because a state-less component can fail without causing disruption of the ongoing user sessions. This provides the required robustness and resiliency to the failure of a VM as well as physical servers (as long as the mapping between physical and virtual machines is cautiously performed). However, there are some possible disadvantages to this implementation option. First, there may be possible synchronization issues between the different virtual components, which will result in serialization of the access to SDB, deterring the overall system performance, especially when SDB is acting as a distributed storage. Additionally, the processing of a specific control or data plane message has to pass through multiple nodes — at least one FE, one worker, and one SDB, which increases the design complexity since it requires a specific level of synchronization between these components and may introduce longer processing delays.

Due to the high complexity of the design, expected high delays as well as due to the lack of readiness to support such features in the current available software, we have decided to implement such a 1:N architecture only for specific dedicated EPC components which relate to EPCaaS support to non-3GPP access networks, not presented in this article [16].

N:1 Mapping — In the N:1 mapping option of EPCaaS (Fig. 2c), all the functional entities of EPC are collapsed into one virtual component, named merged-EPC, which fully implements all EPC components. It has the functionality of all control and user plane functional entities of EPC — MME, SGW, PGW, HSS, and PCRF — conceptually acting as a single application server providing the whole connectivity service.

Each subscriber (or a group of subscribers, depending on the maximum capacity of an SIC) is served by one merged-EPC containing all the specific user session state information. Since all interfaces between EPC functional entities serving the same subscriber are contained in one virtualized component, it is straightforward to optimize internal processing of control and user plane messages for that particular subscriber. Specifically, for each 3GPP EPC interface, the messages can be directly forwarded between processes implementing the

Mapping option	1:1	1:N	N:1	N:2
Basic concept	Each EPC component is mapped to a VM	Each EPC component becomes a pool of resources	Processing for an end-device happens in a single component	Same as N:1 with control-data plane split
Rationale	Maintaining the complete consistency with legacy	Based on the web services paradigm	Low delay, high parallelization using uniform functions	Low delay, high parallelization, control-user plane split
Advantages	Simple	• Stateless design facilitating –Scaling –High availability –Load balancing • Complexity hiding • Easy backward compatibility	• Minimal number of interfaces • Highly low delay processing • High parallelization of the same component	• Control and user data planes split • Stateless design • Fewer interfaces
Disadvantages	• Scaling is not easy: automatic configuration and O&M are complex • Design is not fault-tolerant: if a VM fails, all the sessions are gone.	• Possible synchronization issues between components • Possible larger processing delay due to passing through multiple nodes	• Less granular scaling • Mixture of control and data planes	• Scaling is less granular than 1:N • Scaling of user plane component is not trivial

Table 2. Comparison among the four envisioned EPCaaS mapping options.

logic of the different EPC components, thus making redundant at both interface ends the following functionality:

- The reference point state machine (e.g. S11, S5, and S6a) synchronizing the end user state.
- The communication protocol state machine (e.g. diameter, GTP, and S1-AP) enabling consistent communication between the two entities.
- The encoding and decoding of the communication messages into the specific protocol format, enabling the transfer of binary self-designed data structures.

In a more radical format, the message queue, which enables the forwarding between the different worker processes implementing the different EPC components, may be further merged into a single worker procedure, thus not requiring any internal synchronization at all.

Each merged-EPC has a separate address space for subscribers and separate storage for subscriber data and session data. It is worth noting that only local storage is needed in this option, and not a fully-fledged database system, unlike the 1:N option, which places higher performance and resiliency requirements on the database.

Having all EPC interfaces internal to the single merged-EPC, the only remaining external interfaces are the ones toward the PDN to which the user is connected (e.g. Internet and SGi), RANs (e.g. eNBs and S1), and toward the O&M and billing system of the cloudified EPC service provider.

However, it is worth noting that this option, while achieving the highest possible optimization as a software realization of EPC, presents management problems similar to those highlighted for the 1:1 mapping option. In particular, management of a potentially high number of virtual components can pose scalability challenges on both management entities and external/legacy nodes and networks, although this issue can be alleviated by having all components of the same type (absolute uniformity of the network). Moreover, subscriber data management is highly complicated in this option, as the (before centralized) HSS is actually split into smaller databases, each containing information about a small number of subscribers. This indeed poses hard scalability requirements on the (external) provisioning platform. Another important limitation consists in the fact that interfaces between EPC components become internal, making it impossible for an operator to use multi-vendor solutions. Each EPC has to be provided by a single vendor.

In order to alleviate these limitations, one implementation option is to combine the N:1 mapping with a 1:N design through the introduction of a load balancer and a user state database for the single merged-EPC component, and through this to use the advantages of the 1:N design for scalability and state sharing, and the advantages of the N:1 design for processing simplicity, management uniformity, and short service delay. For the sake of simplicity, this hybrid implementation model is not further presented.

N:2 Mapping — The N:2 implementation architecture model, shown in Fig. 2.d, is a refinement of the previous N:1 mapping wherein the control and data planes are split. A split may be necessary due to the different processing needs of the two types of components. For example, for each message received, the control components fetch the state of the relevant UE from the subscriber state repository, analyze the message, make different access control and policy based decisions, and then update the specific state. Instead, for the data path components, the state should not be modified and the data packets should be immediately forwarded. Additionally, from the UE perspective, the control procedures may have longer delays compared to the user data path forwarding.

For the N:2 implementation option, the following virtualized components are defined:

- The *state database* (*SDB*) has the same functionality as SDB of the 1:N option.
- A *control* (*CTRL*) component encompassing all the control functions of EPC. In particular, the CTRL functions include the functions of MME, PCRF, HSS, and the control plane relevant functions of SGW and PGW.
- A *switch* (*SW*) component handling only user data packet processing and policy enforcement. Thus, it functions as SGW-U, PGW-U (i.e. user plane functions of S/PGW), or a combination of the two.

The switch component could remain non-virtualized and reside outside the cloud; this would result in an implementation of the partial virtualization approach presented earlier (Fig. 1).

This implementation architecture enables a large part of the advantages of the N:1 option, including short delay pro-

cessing, a small number of interfaces, and high uniformity, with the ability to separately handle control and user plane messages. However, it shares a large part of the limitations of the N:1 option, especially the scaling of the user plane components.

This implementation option was practically realized by using the Fraunhofer OpenEPC toolkit and the adapted management and orchestration of the 1:1 option [16]. Although less flexible than the 1:1 option, due to the very low delay of the control and user planes, the N:2 option presented large advantages in terms of complexity management, thus making it more suitable for fast deployments of customized core networks.

For the sake of illustration and readability, Table 2 compares the four envisioned implementation options, briefly describing the basic concept of each and the rationale behind it, highlighting the advantages and pitfalls of each. It should also be noted that the implementation options described above may be grouped into other hybrid versions, wherein for each subscriber one of the parallel running implementations is selected, for example, depending on its subscriber profile. Additional implementation options may be further considered, especially in regard to the transfer of part of the functionality to what is currently considered as the radio access network and to its virtualization.

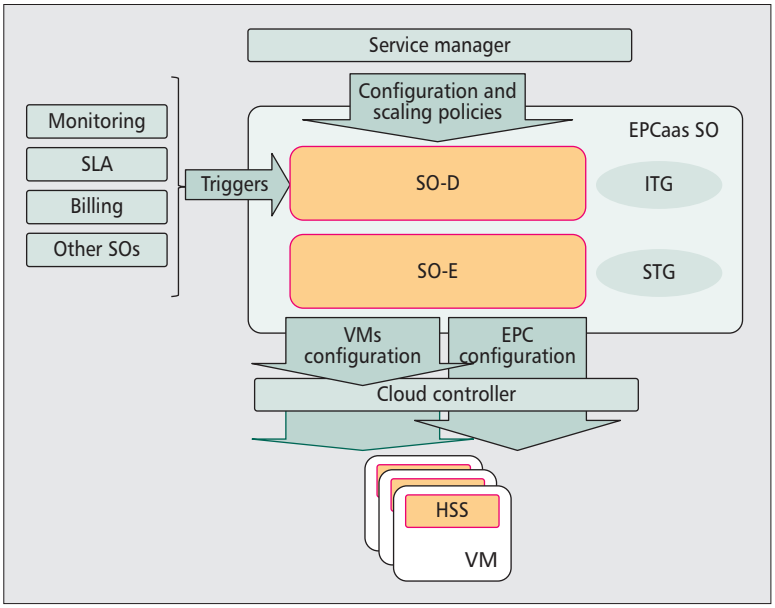

Figure 3. High level diagram of the EPCaaS service orchestrator's architecture.

Management and Orchestration of EPCaaS

The entire lifecycle management of EPCaaS is performed by the service orchestrator (SO) as part of the architecture described in [15] and schematically depicted in Fig. 3. In this architecture, all functional elements are implemented as services, implemented, in turn, by resources that operate service-related entities. These entities are managed in a common consistent fashion regardless of their category by three key architectural entities. The service manager (SM), which offers to enterprise end users (EEUs), for example, MNOs, multi-tenant capable services, and an external interface, both programmatic and visual. Moreover, the SM is used in two dimensions: the business dimension, which encodes business agreements, and the technical dimension, which manages the different SOs of a particular tenant. The SO embodies how a service is actually implemented and is controlling the complete end-to-end orchestration of a service instance, including creation and scaling. The SO monitors metrics related to the service instance. The third key architecture entity is the cloud controller (CC), which is used to deploy, provision, and dispose SOs. Moreover, the CC is responsible for cloud infrastructure resource provisioning and deployment, together with the management of virtual resource coordinating with the SO. The technical lifecycle of a service is controlled by the management and orchestration framework associated with the particular service, such as EPCaaS.

Indeed, for provisioning, orchestrating, and managing EPCaaS, interactions among the SM, SO, and CC must be considered, which are developed as part of the cloudified service enablement platform and contextualized specifically for EPC; the role of each is detailed below. EPCaaS follows a common lifecycle model based on the lifecycle of telecom services as standardized by TMF (TeleManagement Forum), including the following stages:

- Design: at this stage the service's technical design is carried out.
- Implement: with a service design, the service is implemented. This entails the implementation of the SM and SO.
- Deploy: in order for the SM to take requests to create new service instances, the SO needs to be deployed using the CC.
- Provision: this phase is when the SO is instantiated and begins to create the services necessary to satisfy the SO's needs.
- Runtime and operation: the SO has completed its job of providing the corresponding tenant with the requested service instance and is now monitoring and managing the service instance. It is during this step when scaling in and out of components is carried out. Scaling in occurs when a component is releasing resources; scaling out occurs when new resources are allocated to a component.
- Disposal: the service instance's sub-components are destroyed and deleted.

The SO makes decisions and manages the initial provisioning and scaling of EPCaaS. A high-level diagram of its architecture is shown in Fig. 3. Details of its initial implementation are provided in [16]. The SO includes the following high-level functional blocks:

- SO decision (SO-D): functional entity deciding on the EPCaaS's virtual component deployment during the initial EPC provisioning and the runtime phases. SO-D includes a policy engine that is able to customize the specific EPCaaS service instance according to the requirements of the EPC customer (e.g. MVNO).
- SO execution (SO-E): functional entity handling the EPCaaS deployment or disposal management. This entity communicates to the CC, for the new EPCaaS SIC deployment, as it needs to transmit configurations to the CC for running or stopping a specific SIC, including the additional information that is required at the start-up of SIC. Additional to the provisioning and the configuration of the virtual machines running the EPCaaS components, SO-E has the role to configure the EPCaaS software running on these virtual machines. For this purpose, it directly communicates with the EMS collocated within the same virtual machine with the EPC components.
- Infrastructure template graph (ITG): the infrastructure graph is a data structure that contains the topology of the hosts underlying the EPCaaS instances. ITG has the role to provide information on the substrate for the received parameters and for the decisions contextualizing SO-D to a specific instance.

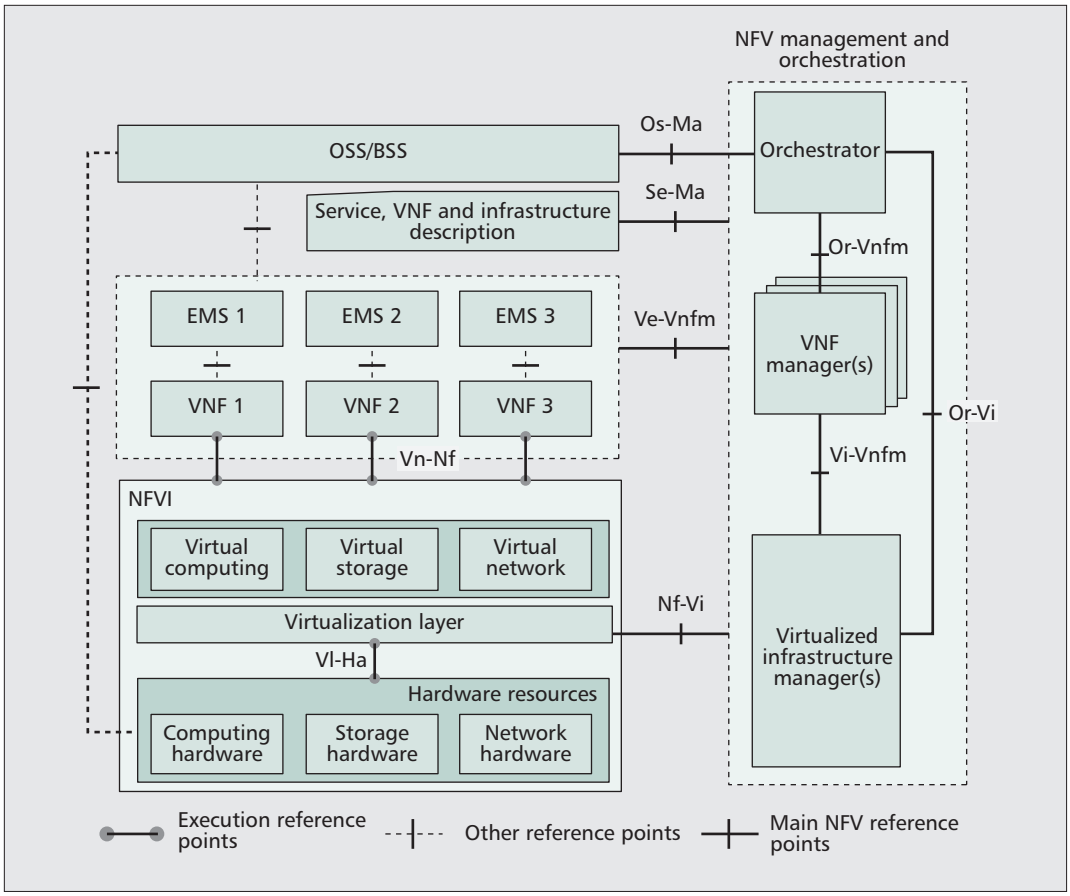

Figure 4. ETSI NFV architecture (copied from [17]).

- Service template graph (STG): STG is a data structure that describes the current EPCaaS SIC topology as deployed on top of the infrastructure, including the network connections, the interfaces, and the resources consumed.

As stated earlier, the EPCaaS SO makes decisions based on various inputs, that is, triggers, STG, and ITG information, as well as on internal policies for the specific EPCaaS of EEUs. The triggers that the EPCaaS SO receives may include the following:

- Explicit requests from the SM: the SM may request a new EEU service topology or may change policies and parameters for instantiated and automated runtime service.
- Triggers from other SOs: other SOs transmit information on STG modifications of other services cooperating with the EPCaaS services (e.g. RAN as a service (RANaaS)) provided by the EPCaaS SO.
- Triggers from supporting services: a monitoring service may notify the SO on passing the load thresholds, both high and low. The SO may be notified about the impossibility to fulfill the service level agreement (SLA) of running EPCaaS. Additionally, the SO may receive explicit notifications on the current resource availability to be consumed by an EPCaaS instance, optionally, along with billing information.

The CC supports the deployment, provisioning, and disposal of SOs. For this purpose, it is able to instantiate virtual networks, to instantiate virtual machines, to connect them to the virtual networks, and to dynamically allocate to them IP addresses. The CC functionality acts as a wrapper toward the specific service deployments of the northbound interface of OpenStack, the de-facto open source standard for cloud computing research activities [15].

The ETSI NFV Industry Specification Group (ISG) was formed to analyze the issues of the virtualization of traditional network functions. The working group has released the first version of their reference architecture [17], along with a terminology document [18]. Although the definition of both the architectural functional entities and the reference points is an ongoing effort, it is worth analyzing here the current ETSI NFV management and orchestration (MANO) framework and comparing it to the architecture described above. Figure 4 represents the ETSI NFV architecture: a complete description is out of the scope of this article, but can be found in [17]. The architecture, depicted in Fig. 3, follows and extends the current MANO framework. Indeed, compared to MANO, the described architecture is based on a more granular approach to functionality, enabling a large set of heterogeneous services to be composed into a single virtual network deployment, enabling turn-key solutions for end-to-end communications. Effectively, ETSI NFV does not explicitly address service-oriented concepts, although a VNF could be considered as the instantiation of a service (i.e. a service instance in the architecture described herein), composed of a single VM or of multiple VMs, whose interconnection is defined by a service graph. Table 3 details a possible mapping between our envisioned architectural elements and the ETSI NFV elements. The aggregation of the new core network functions, namely the SM, SO, and CC, and the cloud-based systems, such as OpenStack, has the exact functional role as the ETSI MANO framework. For instance, the EPCaaS SO described above represents the MANO virtual network functions manager (VNF Manager), being one of the initial practical implementations of such a solution, proving that the NFV MANO architecture is feasible for further production. Additionally, our envisioned functions

Proposed architecture	ETSI NFV architecture
Service instance	VNF.
Service instance component	VNF component (if the VNF is composed by multiple VMs, otherwise there is a one-to-one mapping between SI/SIC-VNF/VNF component).
Service orchestrator	VNF manager, but SO also has some functions in common with the ETSI NFV orchestrator.
Service manager	No corresponding entity; SM is actually a higher level entity, which EEUs interact with (also possibly through a GUI). Such interaction is not in the scope of ETSI NFV. However, some SM functions can be covered in the orchestrator (namely the realization of "network services" mentioned in its definition).
Cloud controller	No corresponding entity; CC is designed to provide an environment where SOs run with an SDK for SO implementation. It acts basically as an "actuator" of the SO logic, but, besides atomic services, it also manages support services and network connectivity. Its functions could be covered by an interaction of VNF manager, virtualized infrastructure manager, and orchestrator[1] together.
OpenStack,[2] with extensions for metering	Virtualized infrastructure manager.
SO bundle	Service, VNF and infrastructure description; ETSI NFV definition of this information model is vague, but it contains data similar to the ones contained in SO bundle (apart from the SO code, which is CC specific). It is worth noting that STG and ITG seem to be a super-set of the VNF-FG, which only deals with network connectivity.

[1] ETSI NFV orchestrator has in fact an interface towards virtualized infrastructure manager (Or-Vi); envisioned CC has also an interface towards "the Cloud."

[2] OpenStack is not an element of the envisioned architecture, but rather a reference technology. It has been included in the table since it may be an example of technical realization of an ETSI virtual infrastructure manager. Other relevant examples could be VMware vCloud director.

Table 3. Comparison among the four envisioned EPCaaS mapping options.

add a northbound interface through the SM toward the end-users of cloud services that is not in the scope of ETSI, as currently NFV mainly addresses single-operator environments. In this vein it is worth noting that the SM provides an easy means to comprehend and to adapt the EPCaaS structure requiring medium skill level for administration. Since a full, precise mapping of the two architectures is not possible, we are not analyzing here the mapping of the reference points. However, it can be noted that some interfaces could be mapped on our envisioned reference points. For example, Vi-Vnfm corresponds roughly to the southbound interface of the CC, and Ve-Vnfm corresponds to the service-specific interfaces between SICs of EPCaaS services and SO.

Conclusion

Mobile operators are in a highly dominant and advantageous position as they "own" the network and thus can provide guarantees to services that no other cloud service provider can do so far. However, they are not exploiting the situation to their advantage. Indeed, they need to leverage the cloud and understand how to adapt to cloud technologies. Understanding this, the cloudification of mobile core networks may be the first step in unlocking this potential. In this vein, this article discussed the feasibility of "cloudifying" mobile core networks, in particular EPC as specified by 3GPP, highlighting the requirements and challenges associated with this vision of EPC as a service. For that, the architectural composition of EPCaaS has been demonstrated along with a number of different implementation options. Importantly, each implementation option is thoroughly motivated, each with different characteristics, advantages, and disadvantages, because they need to cope with different challenges associated with an on-demand creation of a cloud-based elastic EPC service. Furthermore, the article introduced the EPCaaS management and orchestration platform that provides a way to manage network functions very similarly to IT application services, that is, following a service-centric approach, facilitating automation, control, and management of the service itself. In particular, it enables the EPCaaS composition and orchestration as well as orchestration across multiple domains and service types. While this article provides a concise overview on EPCaaS and its challenges, further details can be found in [16, 19].

Acknowledgments

The research work presented in this article is conducted as part of the Mobile Cloud Networking project, funded by the European Union Seventh Framework Program under grant agreement number [318109].

References

[1] "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 20132018," White Paper, Feb. 2014.
[2] Authored by network operators, "Network Functions Virtualization: An Introduction, Benefits, Enablers, Challenges, & Call for Action," Oct. 2012.
[3] 3GPP TS 23.401, "General Packet Radio Service (GPRS) Enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access."
[4] 3GPP TS 29.060, "General Packet Radio Service (GPRS); GPRS Tunnelling Protocol (GTP) across the Gn and Gp interface, Rel. 12, March 2014.
[5] N. McKeown *et al.*, "OpenFlow: Enabling Innovation in Campus Networks," *SIGCOMM CCR*, vol. 38, no. 2. ACM, 2008, pp. 69–74.
[6] J. Kempf *et al.*, "Moving the Mobile Evolved Packet Core to the Cloud," *Proc. WiMob*, Barcelona, Spain, 2012.
[7] K. Pentikousis, Y. Wang, and W. Hu, "MobileFlow: Toward Software-Defined Mobile Networks," *IEEE Commun. Mag.*, vol. 51, no. 7, Jul. 2013, pp. 44–53.
[8] J. Batalle *et al.*, "On the Implementation of NFV over an OpenFlow Infrastructure: Routing Function Virtualization," *Proc. 2013 IEEE SDN for Future Networks and Services* (SDN4FNS), Trento, Italy, Nov. 2013.
[9] A. Basta *et al.*, "A Virtual SDN-Enabled LTE EPC Architecture: A Case Study for S-/P-Gateways Functions," *Proc. 2013 IEEE SDN for Future Networks and Services* (SDN4FNS), Trento, Italy, Nov. 2013.

[10] T. Taleb, "Towards Carrier Cloud: Potential, Challenges, & Solutions," *Proc. IEEE Wireless Commun. Mag.*, vol. 21, no. 3, Jun. 2014. pp. 80–91.
[11] R. Riggio, T. Rasheed, and F. Granelli, "EmPOWER: A Testbed for Network Function Virtualization Research and Experimentation," *Proc. 2013 IEEE SDN for Future Networks and Services* (SDN4FNS), Trento, Italy, Nov. 2013.
[12] 3GPP TS 22.101, "Service Aspects; Service Principles," Rel. 13, Dec. 2013
[13] T. Taleb and A. Ksentini, "Gateway Relocation Avoidance-Aware Network Function Placement in Carrier Cloud," *Proc. ACM MSWIM 2013*, Barcelona, Spain, Nov. 2013.
[14] M. Bagaa, T. Taleb, and A. Ksentini, "Service-Aware Network Function Placement for Efficient Traffic Handling in Carrier Cloud," *Proc. IEEE WCNC '14*, Istanbul, Turkey, Apr. 2014.
[15] MCN D2.2, "Overall Architecture Definition, Release 1," EU MCN Deliverable D2.2, Oct. 2013
[16] MCN D4.2, "First Mobile Network Cloud Software Components," EU MCN Deliverable D4.2, May 2014.
[17] Network Functions Virtualisation (NFV); Architectural Framework. GS NFV 002: http://www.etsi.org/technologies-clusters/technologies/nfv
[18] Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV. GS NFV 003: http://www.etsi.org/technologies-clusters/technologies/nfv
[19] MCN D4.1, "Mobile Network Cloud Component Design," EU MCN Deliverable D4.1, Oct. 2013.

Biographies

TARIK TALEB (S'04, M'05, SM'10) 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. Dr. Taleb is a professor at the school of electrical engineering, Aalto University, Finland. He has been a senior researcher and Third- Generation Partnership Project Standardization Expert with NEC Europe Ltd., Heidelberg, Germany. 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 assistant professor at the Graduate School of Information Sciences, Tohoku University. His current research interests include architectural enhancements to mobile core networks (particularly 3GPP), 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 & 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.

MARIUS CORICI has been a senior researcher in Fraunhofer FOKUS's Next Generation Network Infrastructures (NGNI) department for 10 years, currently leading the Reliable Network Infrastructure team in charge of research and innovation in the areas of 5G, NFV, and SDN, and the development of the correspondent software toolkits: Open5GCore (www.open5Gcore.net) for wireless ecosystem developments, and OpenSDNCore (www.opensdncore.org) for network service enablement based on virtualization technology, sustaining the industry and academia R&D to obtain and to demonstrate meaningful results with high impact toward standardization.

CARLOS PARADA has a degree in informatics and systems engineering from the University of Minho (Portugal) and a master's degree from Carnegie Mellon University (CMU), United States. He joined Portugal Telecom Inovao in 2000, and since then has been working in the networks area, both for research projects and for industry projects, as a consultant. He has also worked on the development of products for mobile networks. Recently he has participated in projects related to cloud, NFV, and SDN, especially from a telco perspective.

ALMERIMA JAMAKOVIC holds MSc and Ph.D. degrees in electrical engineering from the faculty of Electrical Engineering, Mathematics and Computer Science at Delft University of Technology (TU Delft), the Netherlands. During her Ph.D. work from 2004 to 2008 she performed research in the field of complex networks, focusing on quantitative characterization and its relevance to the robustness analysis of large-scale real-world structures. After obtaining her Ph.D. in 2008 Almerima joined TNO – The Netherlands Institute for Applied Research, where she worked on quantitative analysis and modeling of network performance, and network optimization. Since January 2012 she has been at the Communication and Distributed Systems CDS research group of the University of Bern (UniBE), Switzerland. Besides her main task of leading parts of research in the fields of mobile and wireless (mesh) networking and cloud computing, Almerima's further areas of interest include characterization and modeling of complex networks, network topology analysis, and network performance analysis in general. She has been a technical steering and technical program committee member for a number of international conferences, and she also participated in Dutch, Swiss, and European projects. Almerima is currently participating in the EU FP7 IP project "Mobile Cloud Networking," among others.

SIMONE RUFFINO graduated in computer engineering from the University of Genova and joined Telecom Italia in 1998. He is currently involved in R&D projects on network function virtualization and software defined networking, and his focus is on cloud technologies, applied to operators' fixed/mobile networks. He is responsible for assessing and testing new commercial and open source SDN/NFV solutions. He has been participating in several EU-funded integrated projects in the future Internet area. In the past he followed the conformance and performance testing of innovative products for the Mobile Core Network (GPRS/EPC). He was involved in several standardization activities in the WiMAX Forum and IETF, in the area of mobile ad-hoc networks and IP mobility.

GEORGIOS KARAGIANNIS holds a Ph.D. degree and a M.Sc. degree in electrical engineering from the University of Twente, the Netherlands. From 1994 to 1998 he worked as a researcher at the University of Twente. In 1998 he joined the Wireless Multimedia Research unit of Ericsson Eurolab Netherlands in Enschede, Netherlands, where he stayed until April 2003. From April 2003 to August 2014 he worked as an assistant professor in the Design and Analysis of Communication Systems (DACS) group of the University of Twente. In September 2014 he joined Huawei Technologies Dsseldorf GmbH. His research interests are in the fields of fixed, mobile and wireless (inter)networking, SDN, NFV, cloud computing, end-to-end QoS signalling and provisioning, mobility and routing in communication networks, and performance evaluation. He was a technical steering and technical program committee member for a number of international conferences, and he is currently active within the IETF. He has also participated in Dutch and European RACE, ACTS, IST, and FP7 projects.

THOMAS MAGEDANZ has been actively performing R&D in the field of converging networks and ICT for more than 25 years. Since 2000 he has been director of the "Next Generation Network Infrastructures" division of the Fraunhofer Institute FOKUS, where he is responsible for the development of the globally recognized OpenXXX testbed toolkits that enable industry and academia to prototype the newest network and service platform technologies. Since 2004 he has also been a full professor on the Electrical Engineering and Computer Sciences faculty at the Technische Universitt Berlin, Germany, leading the chair for Next Generation Networks and educating master and Ph.D. students.