2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering

Efficient Exploitation of Mobile Edge Computing for Virtualized 5G in EPC Architectures

Eleonora Cau[‡], Marius Corici[‡], Paolo Bellavista[*], Luca Foschini[*],
Giuseppe Carella[†], Andy Edmonds[§], Thomas Michael Bohnert[§]
[*]University of Bologna, Italy, {paolo.bellavista, luca.foschini}unibo.it,
[†]Technische Universität Berlin, giuseppe.a.carella@tu-berlin.de,
[‡]Fraunhofer FOKUS, Germany, {eleonora.cau, marius-iulian.corici}@fokus.fraunhofer.de,
[§]Zurich University for Applied Sciences, Switzerland, {andrew.edmonds, thomas.bohnert}@zhaw.ch

***Abstract*—To pave the way towards disclosing the full potential of 5G networking, emerging Mobile Edge Computing techniques are gaining momentum in both academic and industrial research as a means to enhance infrastructure scalability and reliability by moving control functions close to the edge of the network. After the promising results under achievement within the EU Mobile Cloud Networking project, we claim the suitability of deploying Evolved Packet Core (EPC) support solutions as a Service (EPCaaS) over a uniform edge cloud infrastructure of Edge Nodes, by following the concepts of Network Function Virtualization (NFV). This paper originally focuses on the support needed for efficient elasticity provisioning of EPCaaS stateful components, by proposing novel solutions for effective subscribers state management in quality-constrained 5G scenarios. In particular, to favor flexibility and high-availability against network function failures, we have developed a state sharing mechanism across different data centers even in presence of firewall/network encapsulation. In addition, our solution can dynamically select which state portions should be shared and to which Edge Nodes. The reported experimental results, measured over the widely recognized Open5GCore testbed, demonstrate the feasibility and effectiveness of the approach, as well as its capability to satisfy carrier-grade quality requirements while ensuring good elasticity and scalability.**

I. Introduction

It is manifest that mobile and wireless connectivity has been growing exponentially during the last decade; in addition, this growth trend is not only going to continue but even to accelerate. As a consequence, today's networks are experiencing a continuous and relevant increase in traffic, also with differentiated characteristics and requirements, due to the ever-expanding number of connected devices, not only including smart personal devices but also a large plethora of Internet-of-Things smart objects. The next generation of mobile networks (i.e., 5G), therefore, will have to meet a wide range of requirements derived from the inevitably rise of qualitative/quantitative demands from both users and advanced services. In this perspective and context, flexibility and operational scalability are widely considered as key enablers for rapid innovation, short time to market in particular towards rapid service deployment, and speedy adaptation to changing requirements of evolving industries and enterprises. In short, if compared with previous generation, 5G pushes not only a novel radio-access technology, but a Core Network revolution that will be capable of delivering connectivity to billion of devices and of enabling new business models and continuously evolving advanced services.

To address the above challenges, Mobile Edge Computing (MEC) is one of the most promising technical approaches, towards the direction of reducing end-to-end service delay and network congestion. The basic idea behind MEC approaches is that service providers may take advantage of moving cloud-computing capabilities in proximity of the served users/devices [1], [2], for instance by personalizing services according to local context of a user with no need of global and centralized coordination [1]. In addition, localizing a relevant quota of traffic at the edge, rather than more deeply into the "old concept" of core network, allows a significant reduction of data volume that has to be moved and enhancements in terms of Quality of Service (QoS), end-to-end latency, lower congestion probability, because of proximity to User Equipment (UE) [2]. Hence, these new possibilities will contribute to eliminate the need to route data traffic through the central Evolved Packet Core (EPC), so further improving overall scalability, which is widely recognized as one of the hardest and most crucial technical challenge for 5G. MEC techniques can also contribute to scalability and QoS improvements by properly moving computing and storage features to the edge, possibly at service provisioning time, thus implementing a highly distributed, decentralized, and when possible loosely-coupled infrastructure.

In particular, in our vision, also to accelerate industrial adoption and practical feasibility, most EPC functions will have to be deployed on top of a uniform edge cloud infrastructure - Edge Nodes - following ETSI NFV and ETSI MEC concepts. Here, the elasticity feature that is traditionally provided by cloud environments becomes a binding constraint in such edge-dense infrastructure; in other words, we claim the need for innovative and effective solutions for enabling dynamic network function and self-adaptation to mobility. For instance, scaling Edge Nodes on demand and preventing network failures with hot-standby replication techniques require that 5G infrastructure control entities exchange subscribers state, so that the processing of requests can be dynamically transferred in different network areas. These new and challenging requirements have motivated several ongoing efforts to enable a new generation of cloud-based infrastructures tailored for the

978-1-5090-1754-6/16 $31.00 © 2016 IEEE
DOI 10.1109/MobileCloud.2016.24

specific and stringent needs of telecom services, primarily in terms of QoS, such as low latencies in the [0, 150]ms range for telephony, to be granted with high reliability [3].

The large EU project called Mobile Cloud Networking (MCN)[1] started in 2013 to address the above issues, by involving key EU academic/industrial players in the field: the general vision is to exploit and extend cloud computing techniques at best in order to ease the deployment and operations of future mobile telecom services through self-management, self-maintenance, on-premise design and operations control functions, but also to leverage these functions in order to enable the creation of new value-added innovative services that can dynamically compose traditional telecom and Internet-based Over-The-Top services into new End-to-End (E2E) services with guaranteed QoS. While in our previous work, we have already addressed the design guidelines to realize the EPC as a Service (EPCaaS) in MCN and generally demonstrated the ability of properly coordinated cloud resources to respect the requested QoS constraints [4], the present work originally focuses on the support needed for elasticity provisioning of stateful components in the EPCaaS. In particular, EPC is strongly based on stateful components, where subscriber state is bound to the network function that embodies it, thus limiting the flexibility of the infrastructure and the potential of leveraging MEC techniques. For instance, in state-of-the-art literature, scaling a network function to another data center is only possible through complex signalling procedures in which the subscriber is relocated to another stateful component. In other words, MEC Core Networks need an appropriate and efficient mechanism, capable of exploiting and integrating at best with state-of-the-art solutions for cloud-based virtualized EPC infrastructures, in order to synchronize subscribers state between data centers. The paper will describe the design and implementation of an original solution to that, by showing that the proposal can allow all the network functions pertaining to a pool to have access to the needed subscribers state with low-delay end-to-end operations, in a very dynamic and scalable way.

In particular, the proposal enables state sharing for the sake of fault-tolerance and scalability by exhibiting the following relevant original features. First, in order to favor flexibility and high-availability in case of network function failures, we developed a state sharing mechanism across different data centers. The newly designed Edge Synchronization Protocol (ESP), based on Abstract Syntax Notation.1 (ASN.1), allows subscribers state transfer between different data centers independently of firewall/network encapsulation. Second, our original solution can select which state information to share and where (e.g., to which Edge Nodes) to propagate it. In other words and in short, the original paper focus will be on addressing implicit state synchronization functions for the successful integration of EPC and emerging MEC principles. Let us note that, also given the industry-oriented objectives of the MCN project, the proposal has been efficiently implemented and integrated into the Fraunhofer FOKUS Open5GCore toolkit, widely recognized as a comprehensive solution for providing a practical implementation mirroring the research advancements towards 5G networks.

[1]MCN Home Page: http://www.mobile-cloud-networking.eu

II. MCN Project

To facilitate the full understanding of our original proposal described in the next sections and to make this paper self-contained as much as possible, here we give a rapid overview of the general mobile cloud networking architecture developed within the MCN project, by focusing specifically on its service orchestration functionality, which is central to our proposal. For additional technical details about the general MCN architecture interested readers can refer to [5].

The MCN architecture encompasses and integrates (converges) the three major stakeholder domains of interest for this project, i.e., i) Radio Access Network (RAN), with virtualization as a service of the subscriber wireless access (virtualized base station); ii) Mobile Core, with servitization and cloudification of EPC infrastructure of components; and iii) Data Centers, with additional virtualized services for compute and storage to be used efficiently within mobile and telecom environments. The principal services exposed, cloudified by the MCN approach, are RAN, EPC, IP Multimedia Subsystem (IMS), Content Distribution and Digital Signage System all delivered as a Service. These services are managed by the MCN Service Management Framework, which enables and affords the means to compose them. It is this composition and orchestration across multiple domains and service types that MCN refers to as End-to-End (E2E) composition. Not only does MCN enable usage and extension through cloud computing services within these three domains, but MCN will support and enable developers to build upon MCN services so they can compose and orchestrate their own services delivering much needed additional value and new revenue streams. Let us note that the MCN project started as a parallel activity to NFV and it covers what NFV does, by going beyond NFV in some key areas such as end-to-end service composition as well as exposure of scalability effects when needed between the different services [6]. Careful attention has been given such that MCN maps at best to existing industry standards and best practices such as TMForum, OCCI and ETSI. The MCN architecture is delivered and implemented as Hurtle [7].

One of the most relevantly original aspects of MCN is that its orchestration framework ensures the creation and management of not only the foundational resources required to operate the target service logic but also the so-called external requirements, i.e., the external service resources, possibly not networking-related, needed to compose the final E2E service. These dependencies are executed upon specifically by the Service Orchestrator (SO) ”Resolver” component, as detailed in the following. The MCN architecture has two key aspects, the lifecycle and the architectural entities. The technical phase of the lifecycle includes all activities from technical design all the way through to technical disposal of a service:

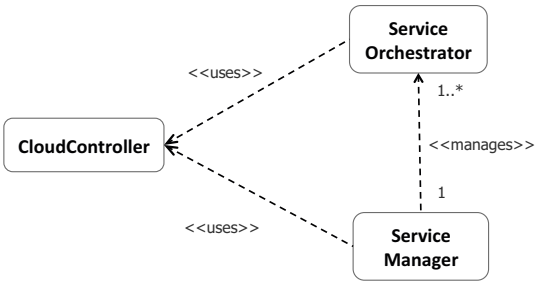

Fig. 1. Key entities of the MCN Architecture

- *Design*: Design of the architecture, implementation, deployment, provisioning and operation solutions. Supports Service Owner to "design" their service.
- *Implementation*: of the designed architecture, functions, interfaces, controllers, APIs, etc.
- *Deployment*: Deployment of the implemented elements, e.g., data centers, cloud controllers, etc.
- Provisioning: Provisioning of the service environment (e.g., network functions, interfaces, etc.).
- *Operation and Runtime Management*: in this stage the service instance is ready and running. Activities such as scaling, reconfiguration of Service Instance Components (SICs) are carried out here.
- *Disposal*: Release of SICs and the service instance itself is carried out here.

The complete MCN lifecycle governs the key architecture entities, which are shown in Fig. 1.

By sketching the primary role of the key MCN entities, the Service Manager (SM) provides an external interface to the Enterprise End-user (EEU) and is responsible for managing SOs. It adheres and implements the MCN lifecycle. The SM programmatic interface (northbound interface, NBI) is designed so it can provide either a CLI and/or a UI. Through the NBI, the SM gives the EEU or SO, both classed as tenant, capabilities to create, list, detail, update and delete (EEU) tenant service instance(s). Its Service Catalogue contains a list of the available services offered by the provider. Its Service Repository is the component that provides the functionality to access the Service Catalogue.

The SO Management (SOM) component has the task of receiving requests from the NBI and overseeing, initially, the deployment and provisioning of the service instance. Once the instantiation of a service is complete, the SOM component can oversee tasks related to runtime of the service instance and also disposal of the service instance. Service instances are tracked in the SO Registry component.

The Cloud Controller (CC) abstracts from specific technologies that are used in the technical reference implementation. Having such a logical component allows other providers to use another technology by simply implementing the relevant CloudController component.

The Service Orchestrator (SO) is a self-contained tenant process that runs within a container, managed by the CC. Its primary responsibility is to manage (according to the MCN lifecycle) resources and external services required to deliver the tenant's service instance. The SO is a collection of imperative and declarative code. This collection is known as a SO bundle and is what the SM manages. The imperative code is responsible for creating and managing (e.g. scaling in and out) the resource and service instances. The declarative code is embodied by representations of two types of related graphs:

- *Service Template Graph (STG)*: defines how services can and should be composed together. For example, the IMSaaS can have a requirement on a monitoring and DNS service. This requirement is hence represented as a dependency. The STG interface can be queried through the Service Managers NBI
- *Infrastructure Template Graph (ITG)*: defines how resources should be composed to be able to host SICs. For example, an Analytics service requires two virtual machines: one to handle compute execution and one to handle the storage backend, both of which are connected through a network. Within the MCN framework ITGs are handled by template documents, which can be placed upon different infrastructure service providers such as CloudSigma, Amazon EC2, as well as OpenStack and Joyent Triton. Finally, the SO enables multi-region/zone deployments: it can either hold multiple ITGs in the SO bundle or compute them on the fly.

In particular, the SO consists of three key components: the SO Execution (SOE) that is responsible for procedural phases, the SO Decision (SOD) that is responsible for runtime decisions related to scaling out or in and SO Resolver that is responsible for composition. In order to operate, the Resolver component requires understanding the service dependencies a specific service requires. These requirements are logically described by the SO bundle's STG.

III. Elastic EPCaaS Provisioning in MCN

This section first provides a brief overview of the EPC architecture with the goal to underline some of its limitations, especially with respect to the possibility to elastically scale its edge components as in virtualized 5G MEC-enabled scenarios. Then, it focuses on our original proposal based on the definition of the MCN EPCaaS service, and on the introduction of a new state sharing function for EPCaaS.

A. Evolved Packet Core

EPC is standardized by 3GPP to enable integration of multiple and heterogeneous access networks including: new E-UTRAN access technology, namely, LTE and LTE Advanced; 3GPP legacy systems, namely, 2G and 3G air interfaces; and non-3GPP systems - WiFi, WiMAX.

MEC shifts the focus from the Radio Access Network to the Core Network: a virtualized EPC will run on top of cloud infrastructure at the edge of cellular networks. EPC is a mobile aggregation network standardized by the 3rd Generation Partnership Project (3GPP) [8]. It provides all-IP

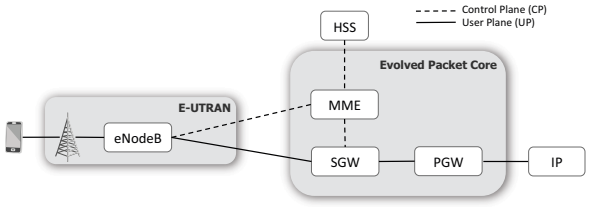

Fig. 2. Evolved Packet Core Architecture

connectivity for 4G networks and maintains interoperability features for 2G and 3G mobile services.

Without any pretence of being exhaustive, in the following we introduce the main EPC functional components; for more details, we refer to [8]:

- *HSS (Home Subscriber Server)* is a subscriber repository that contains all the user subscription information. It provides support functions for mobility management, call and session setup, user authentication and access authorization.
- *MME (Mobility Management Entity)* is responsible of all the Control Plane functions related to subscriber and session management. It is also in charge of subscriber related radio control procedures. It also handles Inter-Radio Access Technology (RAT) handovers.
- *SGW (Serving Gateway)* is the access gateway of the 3GPP network and the endpoint of the packet data interface towards E-UTRAN. It forwards the uplink data traffic from the RAN to the PGW and serves as anchor point for intra and inter RAT handovers (in case of handover between eNodeBs or between LTE and other 3GPP accesses).
- *PGW (Packet Data Network Gateway)* provides connectivity to external packet networks and is the point of interconnection between the EPC and the PDN. It performs multiple functions such as policy enforcement, packet filtering, charging and IP address/IP prefix allocation for the UE, which is maintained for the duration of its active communication, independently from the network location.

All these components communicate through IP-based communication protocols, such as GPRS Tunnelling Protocol (GTP) and Diameter.

Expanding the current EPC infrastructure to support MEC requirements and functionalities, highlights the need to move the EPC next to the edge of cellular networks. However, EPC was designed for centralized network architectures, where MME, SGW and PGW control components are part of the Core Network and the eNodeB is the only component close to the UE, as shown in Fig. 2.

The introduction of a virtualized environment and cloud computing paradigms, such as SDN and NFV, paves the way to new flexible on demand deployment possibilities at the level of access network, next to the user. Leveraging these new functions and according to MEC design principles, we propose to evolve the traditional EPC architecture through the definition of an Edge Node (EN) acting as a controller entity and composed by MME, SGW and PGW control functions. However, the standard EPC architecture shows some structural limitations that hinder the introduction of EN without evolving it.

First, handling subscriber mobility and communication between control components when the UE moves from the current EN to the target EN, is difficult to obtain. In fact, standard handover procedures, in which location management is completely under the control of the network, always require an anchor point (usually either SGW or PGW) that typically enables subscriber state transfer from current MME (or SGW in case PGW acts as anchor point) to the target one. This mechanism cannot be considered as a feasible mobility solution because, according to MEC principles, SGW and PGW are co-located with MME, and thus the anchor point would be missing.

Second, the increased number of deployed ENs will lead to a consistent growth in terms of both signalling and misrouted traffic because, due to the densification of ENs into the network, users will switch much more often between nearby EN instances, performing expensive handover procedures and at the same time increasing the delay.

Third, at the current stage there is no standard coordination procedure among control components to proactively transfer subscriber state; in fact, existing procedures to move subscriber state (e.g., from old to new MME) are typically reactively triggered by EU, for instance due to attachment to a new eNB during the handover procedure. That limits the possibility of elastically de-/activating ENs.

Finally, even assuming such procedure exists, some operations and especially the *scaling-in*, namely, the disposal of a lightly-loaded EN, has to be carefully managed because moving subscribers state from current to target EN requires granting state consistency along the process and in spite of possible network faults.

B. Elastic EPCaaS Provisioning through State Sharing

By considering the above aspects, the standard EPC architecture does not allow (as it is) to create an extended system, following MEC principles. Fig. 3 shows the new proposed architecture which is composed by ENs and a Central Node (CN). CN acts as traditional (non MEC-enabled) EPC network, and provides potentially cloud computing capabilities by also maintaining a global view of the system. Network functions can be deployed in both CN and ENs, taking advantage of all the benefits that come from cloudification mechanisms.

In fact, our proposal is to deploy the EN as a virtualized component adapted towards the cloud environment and cloud-native functionalities realized by the MCN cloud support, thus to ease elastic scalability management [4]. Moreover, we exploited MCN support for cloudification to implement EPCaaS and to enable its flexible orchestration from small-scale local deployments to large-scale data center deploy-

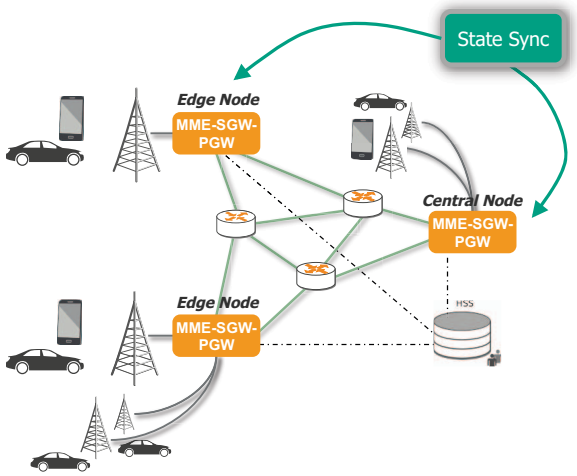

Fig. 3. EPC State Synchronization.

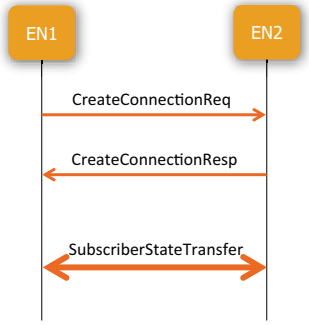

Fig. 4. Synchronization Interface Establishment Procedure.

ments, including the possibility to compose EPCaaS with other corase-grained services such as Radio Access Network as a Service (RANaaS) and IMS as a Service (IMSaaS). Where to execute functionality is completely dynamic and based on advanced policies which depend on the state of the subscriber, the congestion of the network, and the device mobility pattern monitored through the MCN Monitoring as a Service (MaaS) support.

Let us consider now the need for enhanced coordination to enable state transfer among ENs. As motivated above, it is not possible to use existing EPC procedures and some additional mechanisms should be introduced. Among the several possible alternative solutions, we opted for *state sharing* to proactively move state and synchronize state changes across neighbour ENs in a proactive way, whenever a new state appears in the system or is updated.

The advantages of this kind of approach are multiple. First of all, each component has always updated information about all the users and the state is *synchronized and consistent across the system*, allowing components to be transparently added or deleted. In addition, each component logic is unaware that other components of the same type are in the system; that leads to a virtually *stateless control plane*, while each of the components is statefull and can see the complete state information in the system. Focusing on fault tolerance, the possibility to easily replicate state across various redundant components avoids that state is lost in case of component failure and *removes single points of failure*. Finally, state synchronization procedure is *EPC standard compliant*; it is by-design transparent to the EPC procedures and does not require to change standard communication protocols among EPC components, so to grant fast adoption and integration with existing deployments.

Let us note that the introduction of balancing policies and state replication at ENs enable any EN to handle the requests coming from the same subscriber: eNodeB can perform load balancing operations with fine and request-level granularity.

IV. State Sharing for Edge Node Synchronization

In order to be able to transport state information in a controlled manner over the network, there is a need for a specific interface and an afferent communication protocol to be developed. Such an interface has to be defined in the context of 3GPP as well as in a more generic context, enabling any type of state transfer.

In this section we describe the Edge Synchronization Protocol (ESP) as a set of different procedures to coordinate ENs. The first procedure is called *Synchronization Interface Establishment Procedure* and its main goal is to establish an initial common subscriber state between two or more EN entities. The second one, called *Re-synchronization Procedure*, allows the proper re-synchronization of two different ENs when the connection between them is intentionally or unintentionally lost due to different retained states when connectivity comes back. The protocol can be used between multiple ENs in case of standard 3GPP EPC implementations as well as for any subscriber state information which has a similar format to telecom one. To define how a message is represented, Abstract Syntax Notation 1 (ASN.1) is the best solution as standard formalism characterized by efficient data encoding schemes and high compression capacity: a single simple identity and multiple attribute values sums up to less than 1kB (approximately 600 bytes) of information.

A. Synchronization Interface Establishment Procedure

This operation happens when a new EN joins the pool. Assuming EN1 is the joining node, the proposed synchronization procedure starts with EN1 that requests to join the subscriber state synchronization pool where EN2 has already joined. As shown in Fig. 4, the first phase of the procedure starts with the binding between the two entities. EN1 subscribes to state information changes, in terms of new or updated ones, and conversely EN2 makes a subscription to modifications which occur in EN1 (steps 1 and 2). Afterwords, they can exchange to each other subscribers state that they are currently storing. This step (step 3) is also executed whenever a state is updated or a new subscriber attaches, thus after standard Attach, Detach

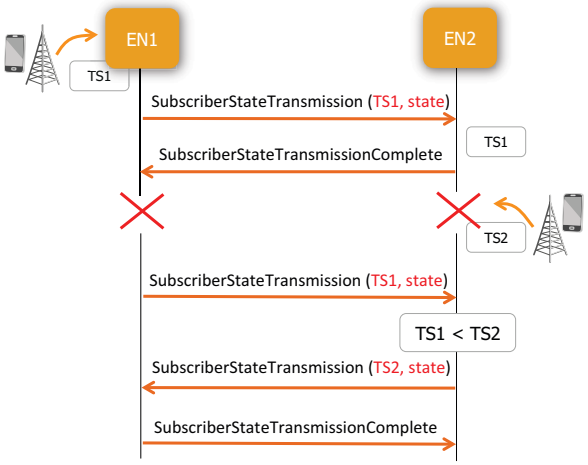

Fig. 5. Re-synchronization Procedure.

and Handover procedures. In addition, to assure consistency of the states in the system, each state is immediately shared upon procedure completion.

B. Re-synchronization Procedure

In case of disconnection between controllers, a synchronization procedure has to be performed. As better explained in the following, the designed solution is based on timestamps, and thus it requires to synchronize the clocks of nearby ENs. We employed the Network Time Protocol (NTP), standard protocol is used to synchronize clocks over packet-switched, variable-latency networks [9]. From the clock accuracy point of view, usual NTP clock skew is in the order of milliseconds and can be tolerated by our solution. In fact, with a rate of 24 updates-per-day a skew less than 5ms will be obtained, that represents the current target clock accuracy. Since the duration of the shortest measured procedure (the UE network Detachment) from the core network perspective - not considering any radio access network delay - is around 7ms, a clock skew of less than 5ms ensures that the events are still synchronized even though the clock values are not, even in this worst case.

Focusing now on the re-synchronization procedure, it consists in a check of the timestamp ENs exchange every time a state is sent, within the "SubscriberStateTransmission" message, so that during the re-connection phase each EN can know which one is owning the newer state. An example is described in Fig. 5, in which during the disconnection of the two ENs, the UE modifies its subscriber state through a procedure executed with EN2 (step 3). When they are getting again connected, EN1 sends the state of the subscriber together with the associated timestamp (TS1) (step 4). As TS1 is older than TS2, EN2 sends back the newer state and EN1 updates its own (steps 5 and 6).

Let us stress that the introduced extensions are transparent to the 3GPP core network; in other words, the described extensions are integrated in EPCaaS by maintaining it still functional wise compatible with the standard, while having a large usage of transparent mechanisms.

V. Experimental Results

This section first reports some more details about the implementation and the employed experimental testbed. Then, it reports obtained experimental results that assess EPCaaS in terms of elastic scalability and dependability.

A. Implementation and Experimental Testbed

The *EN implementation* is based on the Open5GCore toolkit, developed by the Fraunhofer FOKUS Institute in Berlin, while for its cloudification as EPCaaS we employed the SO developed within the context of the MCN project. Moreover, apart the development of the state synchronization protocols, extensively described in the previous section, we introduced several other functional extensions to Open5GCore such as the definition of a load balancer to forward requests received from the radio network to available ENs, the parsing logic to use ASN.1 protocol for expressing messages for sharing the state, and an extension of attachment, detachment, and handover procedures to support graceful state sharing across control components of the same type. All these changes have been carefully designed to be transparent to other exising EPC entities; hence, the current version of the Fraunhofer FOKUS EPC, including this developments, is still functional-wise compliant with the EPC standard.

The *MCN platform* is based on several different technologies. First of all, to grant the widest interoperability it supports OCCI [10]. As regards SO deployments, they are executed within a container managed by OpenShift [11], a widely diffused Platform as a Service (PaaS). MCN provides a Software Development Kit (SDK) written in Python, giving support to developers building their SO who wants to interact with the CC. In this way they can easily discover available supporting services, and request the instantiation of virtual resources required by their Service Instances. Finally, at the Infrastructure as a Service (IaaS) level, MCN leverages the standard-de-facto OpenStack [12], and its resource orchestration support called Heat [13], supported also by the other IaaS employed in MCN, i.e., CloudSigma [14]. Services exposed by MCN-complaint SMs, including EPCaaS, require an interface specification expressed in OCCI.

Finally, for the sake of testing, we employed the *benchmarking tool* part of the Open5GCore platform to emulate the behaviour of a realistic group of users and a set of configurable eNBs. It provides a list of statistics and measurements in order to monitor the performance of the service. An arbitrary number of operations per-second can be set in order to perform UE attachments, detachments, and handovers according to the specified percentage (the operations are run on a round robin base). The benchmarking tool can be configured by dimensioning subscriber and eNodeB pools.

Fig. 6 shows the deployment employed in our experiments. The architecture is composed by two ENs, the HSS, and the benchmarking tool. All the processing components are

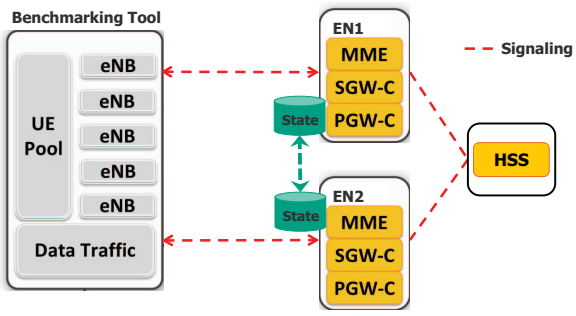

Fig. 6. Testbed Architecture.

TABLE I
TESTBED WORKSTATION CHARACTERISTICS.

Workstation type	Dell Precision T3500
Processor	Intel Xeon E5-1620 2 * @1.6GHz, 6C, 4x4CPU
Memory	4x4 GB RAM
Network Cards	Irrelevant for these evaluations.

virtualized in the form of virtual machines managed by OpenStack, and run on a single compute node. The connectivity between the guest VMs was obtained by virtual LANs, which simulates 1Gbit connections between them. The compute node on which all the testbed measurements were performed has the characteristics shown in Table I. During the evaluation, the system was not loaded to its limit not to produce side effects due to resource saturation. Hence, in all tests, the maximum number of active users was 10.000, the number of attachments reaches 100 attachments/sec, and the number of handovers, more than 500 handovers/sec.

In particular, we realized the following scenarios to prove the efficiency of the proposed state sharing solutions, in terms of high availability and elasticity of the edge network. In all scenarios several setup parameters remained constant. The relevant VMs were EN1, EN2 and HSS that have been evaluated into these scenarios. Each EN component received 4 CPUs and 2GB of memory. The network connectivity of the virtual environment highly surpasses the compute capabilities required, ensuring that no bottleneck will occur due to communication. The benchmarking tool was configured with 5 eNBs, which handle uniformly attachment, detachment and handover operations. In the following, we present obtained results for three test scenarios: scale-out, scale-in, and fault-tolerance.

B. Scale-out Performance Evaluation

This and the following experimental results evaluate EPCaaS elasticity; in this first scale-out experiment, EN1 is the first active controller, while EN2 is elastically started on demand in case of high load situations.

As reported in Table II, the system was pre-loaded with 500 attached users, in order to be able to execute the specific input pattern operations distribution (10% attachments, 10% detachments and 80% handovers); number of operations per second increases at different time intervals from 100ops/sec

TABLE II
SCALE-OUT BENCHMARKING PARAMETERS.

Preparation phase	500 UEs attached to the system
Active procedures	Duration: 60s Input pattern: uniform stairs starting with 100 ops/sec (one operation every 10 ms) ending with 333 ops/sec (one operation every 3 ms) 10% attachments, 10% detachments, 80% handovers
Measured parameters	EN1 and EN2 CPU utilization (%) EN1 and EN2 memory utilization Procedures delay measured at the UE side (only core network delay)

TABLE III
SCALE-IN BENCHMARKING PARAMETERS.

Preparation phase	500 UEs attached to the system
Active Procedures	Duration: 90s Input pattern: uniform stairs starting with 200 ops/sec (one operation every 5 ms) ending with 333 ops/sec (one operation every 3 ms) decreasing to 100 ops/sec (one operation every 10ms) 10% attachments, 10% detachments, 80% handovers
Measured parameters	EN1 and EN2 CPU utilization (%) EN1 and EN2 memory utilization Procedures delay measured at the UE side (only core network delay)

to 333ops/sec. This configuration ensures in this specific deployment that a single EN will not be able by itself to process all incoming requests. Fig. 7 shows obtained results by plotting together: delays for the three operations (numbers on the y-axis on the left to be read as ms); CPU usage percentage (numbers on the y-axis on the left to be read as percentage values); and number of operations-per-second (numbers on the y-axis on the right).

In a more detailed view, at the beginning of the experiment EN1 is active and receives all the requests, while EN2 is a hot standby component (it only synchronizes its internal state). When EN1 CPU load exceeds 80% (with delay to smooth possible ping-pong effects), EN2 is actived (at second 29) and the load balancer splits incoming load between EN1 and EN2. It was observed that the CPU usage of EN1 is not decreasing to half as expected into such situations. It remains with 2% higher due to the bidirectional state protocol synchronization with EN2. In addition, EN2 will take half of the load (and thus have the same CPU usage as EN1) very fast, in less than 1 second.

C. Scale-in Performance Evaluation

This second experiment focuses on the scale-in procedure; initially, both EN1 and EN2 are active, but when incoming load is decreasing EN1 is removed from the system and load transferred from the removed component to EN2.

As in the previous scenario, the system was pre-loaded with 500 attached users. To warm up the system, the input pattern increases from 200ops/sec to 333ops/sec, then the experiment begins and we decrease the input pattern to 100ops/sec,

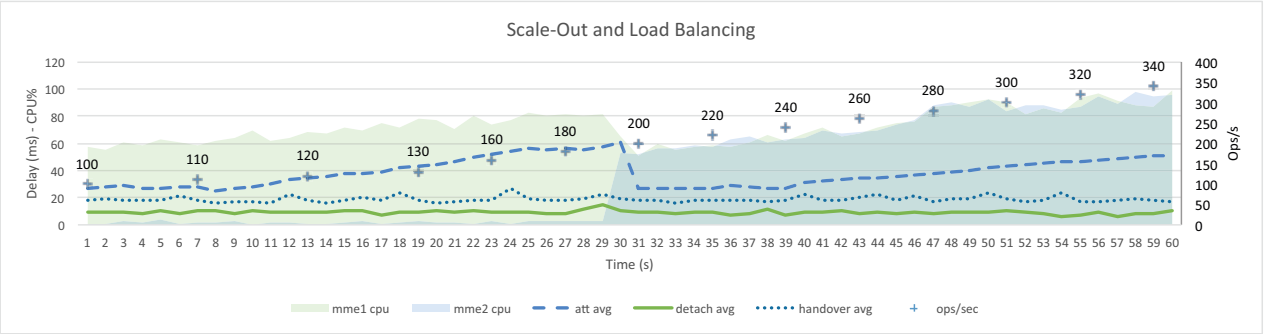

Fig. 7. EN Scale-out Evaluation.

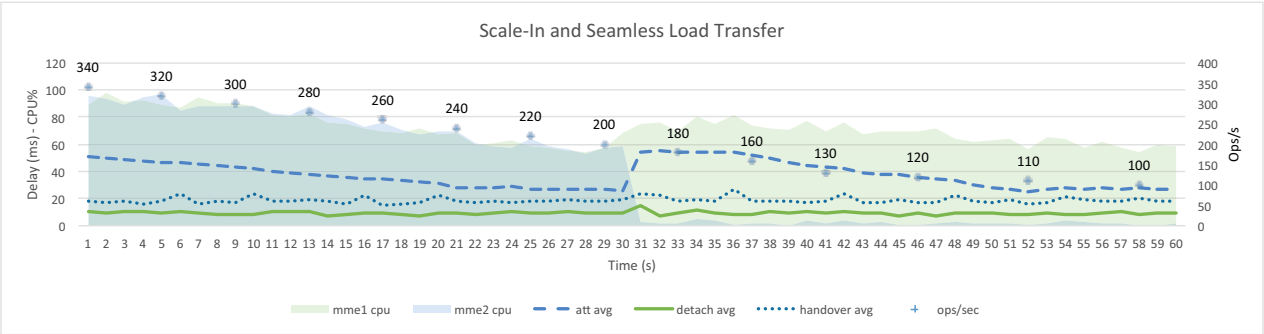

Fig. 8. EN Scale-in Evaluation.

as reported in the configuration settings in Table III; the conventions used in Fig. 8 are the same used in Fig. 7.

The experiment starts when both CPUs are close to 100% CPU utilization; then, due to the decreasing input pattern, the CPU usage of both EN1 and EN2 lowers to 60%. When reaching 200 ops/sec, EN1 is removed from the system and EN2 can handle the remaining requests with less than 80% of its own CPU, while operations delay is momentary increasing. Let us also note that with the removal of EN1, EN2 does not have to synchronize with the state received from EN1 anymore as EN1 is not modifying any subscriber state and so there is no CPU overhead due to synchronization. Finally, it was observed that there is no transaction lost, thus no failure is seen by the UEs.

D. Fault-Tolerance Evaluation

In our last experiment, we evaluate the ability of our solution to overcome possible failures transparently to the UEs. Table IV reports experimental configuration and employed settings.

The active experimentation part includes three phases. In the first phase, EN1 is active and receives all the requests processing them locally and replying to UEs through the eNBs in the benchmarking tool. As part of the operations, EN1 uses ESP to update the internal state of EN2 that acts as a hot standby component. In the second phase, a failure of EN1 is emulated by removing it from the active system. For testbed consistency reasons the VM of EN1 is not stopped, however all its communications are blocked. The benchmarking tool is automatically instructed to forward the signalling to EN2. This is equivalent to the result of an immediate failure mitigation, abstracting the failure processing in the cloud infrastructure (as we wanted to measure only the EPCaaS functionality and not the overall system): this step was executed directly in the benchmarking tool at second 30 of the experiment. In the third and final phase, EN2 becomes the active component that receives and processes all the requests.

TABLE IV
FAILOVER BENCHMARKING PARAMETERS.

Preparation phase	500 UEs attached to the system
Active Procedures	Duration: 60s Input pattern: 100 ops/sec 10% attachments, 10% detachments, 80% handovers
Measured parameters	EN1 and EN2 CPU utilization (%) EN1 and EN2 memory utilization Procedures delay measured at the UE side (only core network delay)

Fig. 9 shows obtained results; it reports on the left-side of the y-axis the operation delay while on the right-side of the y-axis the CPU usage percentage. After second 30 EN2 takes over and the measurements point out that there

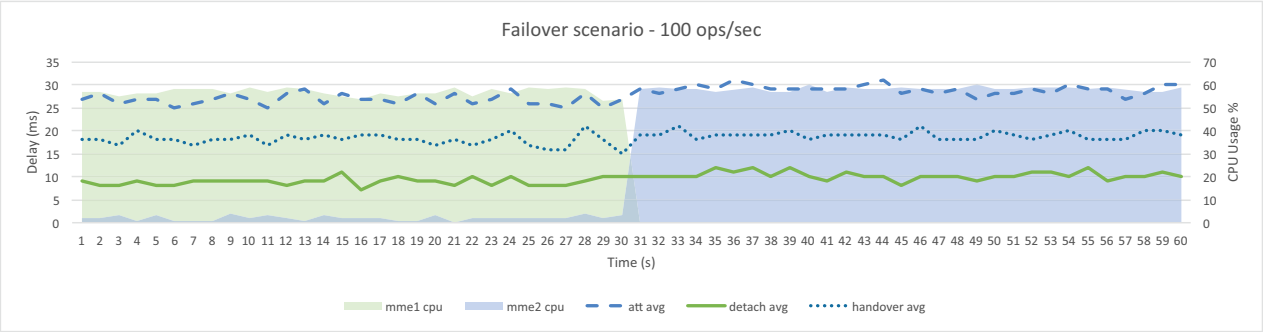

Fig. 9. EN Failover Evaluation.

is no significant increase in delay of any of the procedures from the UE perspective, having under 30ms for attachment, 20 ms for handover, and 12ms for detachment procedures almost constant for the whole duration of the experiment. Additionally, no session has been lost during the emulated failure. At the same time, let us note that having the delay of the radio into the procedures could supposedly lead to a loss of some transactions.

Let us conclude this section with some lessons learnt that apply to all experiments. First of all, from a functional perspective obtained delays are always below 60ms, even in worst case scenarios; hence, notwithstanding additional delays due to our ESP procedures, in all tests our solution is able to comply with the strict requirements of telephony and telecom services [3]. From the perspective of the compute resources used, the CPUs of EN1 and EN2 were utilized in proportion of around 60% when active (4 virtual CPUs allocated) while the memory usage was very low - less than 200MB per EN component in all experiments. Furthermore, it was observed that the synchronization procedure consumed around 1% CPU on the hot standby component. From other additional measurements, we have learnt that passive synchronization of the subscriber state information sums up to 1% CPU for each of the subscribers of each controlled entity. Thus, in case a single hot standby component is deployed for 5 control entities, the hot standby component will have its CPU around 5% only due to the state synchronization operations. Hence, an efficient deployment would activate a single hot standby for multiple control entities and not one hot standby dedicated to each active component, since there would be not actual benefits.

VI. Related Work

Several research activities have recently shown their interest in dynamically scaling services in cloud environments. Due to space limitations we will only consider a few of them by focusing on three main research directions: i) theoretical approaches to scale generic applications in the cloud; ii) more practical research efforts that concentrate on cloudbursting of telecom services via proper oorchestration of services and management of cloud-based (network) resources.

Along the first research direction, Shifrin et al. [15] propose an analytical approach, based on Markov Decision Process, to schedule and cloudburst application tasks in a hybrid cloud environment. [16] proposes a solution that monitors system performance and distributes load across a set of resources provided by both private infrastructures and public clouds. This approach is similar to our solution, but they do not consider at all user preferences and quality requirements when dynamically selecting the targeted cloud platform. [17] provides a study of the state-of-the-art application scalability in cloud environments by illustrating several solutions that focus on different levels of scalability, i.e., horizontal/vertical infrastructure scalability, network scalability and platform scalability. The proposed algorithms are orthogonal for the proposed solution and can be deployed on top of our solution, in a final product. The main drawbacks of these efforts are that they are all based on analytical approaches, provide only numerical evaluation of their solutions without evaluating them in a real cloud testbed scenario and do not take into account QoS requirements.

Along the second direction, only few seminal works have started to analyze the deployment of IMS services over a cloud-based infrastructure. [18] discusses the possibility of building up a Service Delivery Platform [19] in the cloud stating that this approach can limit both capital and operational expenses. Differently from our original proposal, the authors do not provide a solution on how to exploit cloud elasticity and do not consider public clouds. [20] proposes and evaluates a call load forecasting algorithm to decide whether to in-/out-scale telco services together with two protocols to safely coordinate session migration between SIP servers. Differently from our solution, they neither provide quality audit and monitoring mechanisms nor algorithms for choosing where to migrate resources. An improved HSS architecture, based on cloud computing, is described in [21]: the traditional HSS architecture is enriched here with i) a resource pool layer responsible for balancing resource utilization and ii) a management layer responsible for managing the distributed HSS resources. However, [21] only provides simulation results without reporting performance behaviors over real-world sce-

narios of industrial interest. In [22] several IMS deployment models are proposed which could be integrated in our solution. [23] proposes SipCloud, a highly scalable 3-tier SIP proxy architecture based on cloud computing platforms. The proposed approach is, however, limited by the fact that it considers only the monitoring of call load and not other resource-/quality-related indicators such as CPU utilization, RAM consumption and network latencies. In [24] some concepts of cloud brokering are presented, but they are not integrated with the ETSI NFV Management and Orchestration (MANO) architecture.

VII. Conclusion

5G aims to provide ubiquitous connectivity and unlimited access to information, with first commercial systems expected in 2020.

To deal with this ever-growing demand and required levels of elasticity, the solution proposed in this paper leverages the new virtualized network functions provided by SDN and NFV and the new concept of MEC, and proposes to evolve the standard EPC network by moving network control functions (i.e., MME, SGW, and PGW) close to eNB and co-locating them in the new EN component. ENs are flexibly managed exploiting our MCN cloud orchestrator. We evaluated the proposed solution in terms of its ability to provide scalability and scale-in/-out support, as well as enhanced reliability.

Fueled by the significant results, we are working on two main ingoing research directions. On the one hand, we are deploying the realized solution, already widely tested in the geographically-distributed MCN cloud testbed, in a federated cloud environment that introduces new challenges due to the need to deploy and manage EN execution across different domains. On the other hand, we are running extensive experiments to thoroughly assess the impact of EN virtualization on the performances of multimedia services provided Over-The-Top (OTT) of our evolved EPC.

Acknowledgment

We want to thank the European Commission for co-founding the FP7 Large-scale Integrating Project (IP) Mobile Cloud Networking (MCN) project under the 7th Framework Programme, grant agreement no. 318109.

References

[1] O. Makinen, "Streaming at the edge: Local service concepts utilizing mobile edge computing," in *Proc. of IEEE Int. Conf. on Next Generation Mobile Applications, Services and Technologies (NGMAST'15)*, Sept 2015, pp. 1–6.

[2] J. Fajardo, I. Taboada, and F. Liberal, "Improving content delivery efficiency through multi-layer mobile edge adaptation," *IEEE Network*, vol. 29, no. 6, pp. 40–46, Nov 2015.

[3] "Recommendation g.114: One-way transmission time," in *ITU-T Series G: Transmission systems and media, digital systems and networks*, May 2003, pp. 1–20.

[4] T. Taleb, M. Corici, C. Parada, A. Jamakovic, S. Ruffino, G. Karagiannis, and T. Magedanz, "Ease: Epc as a service to ease mobile core network deployment over cloud," *IEEE Network*, vol. 29, no. 2, pp. 78–88, March 2015.

[5] T. M. Bohnert and A. Edmonds, "MCN D2.2: Overall Architecture Definition, Release 1," mobile-cloud-networking.eu/site/index.php?process=download&id=124 &code=93b79f8f5b99f67a6cdc28369c05b65f624cfee7, Oct 2013.

[6] A. Edmonds and T. M. Bohnert, "MCN D2.5: Final Overall Architecture Definition, Release 2," mobile-cloud-networking.eu/site/index.php?process=download&id=263 &code=aa37ba15e0479f1ecb7a696876ab498d7f3ff0ef, April 2015.

[7] "Hurtle home page," http://hurtle.it.

[8] P. Lescuyer and T. Lucidarme, *Evolved Packet System (EPS) - The LTE and SAE Evolution of 3G UMTS*, J. W. . Sons, Ed., 2008.

[9] D. Mills, U. Delaware, J. Martin, J. Burbank, and W. Kasch, "Network time protocol version 4: Protocol and algorithms specification," in *Internet Engineering Task Force (IETF) Request For Comments (RFC) 5905*, June 2010, pp. 1–109.

[10] A. Edmonds, T. Metsch, A. Papaspyrou, and A. Richardson, "Toward an open cloud standard," *IEEE Internet Computing*, vol. 16, no. 4, pp. 15–25, July 2012.

[11] "OpenShift home page," https://www.openshift.com/.

[12] "OpenStack home page," https://www.openstack.org/.

[13] "OpenStack Heat home page," https://wiki.openstack.org/wiki/Heat.

[14] "CloudSigma home page," https://www.cloudsigma.com/.

[15] M. Shifrin, R. Atar, and I. Cidon, "Optimal scheduling in the hybrid-cloud," in *Proc. of IFIP/IEEE Int. Symp. on Integrated Network Management (IM'13)*, May 2013, pp. 51–59.

[16] Y. Seung, T. Lam, L. Li, and T. Woo, "Cloudflex: Seamless scaling of enterprise applications into the cloud," in *Proc. of IEEE INFOCOM*, April 2011, pp. 211–215.

[17] L. M. Vaquero, L. Rodero-Merino, and R. Buyya, "Dynamically scaling applications in the cloud," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 1, pp. 45–52, Jan. 2011.

[18] L. Sanchez and J. Rodriguez, "Sdp as a service (sdpaas): A new revenue stream for operators," in *Proc. of IEEE Int. Symp. on Parallel and Distributed Processing with Applications (ISPA'12)*, July 2012, pp. 442–447.

[19] H. Ohnishi, Y. Yamato, M. Kaneko, T. Moriya, M. Hirano, and H. Sunaga, "Service delivery platform for telecom-enterprise-internet combined services," in *Proc. of IEEE Global Telecommunications Conference (GLOBECOM'07)*, Nov 2007, pp. 108–112.

[20] N. Janssens, X. An, K. Daenen, and C. Forlivesi, "Dynamic scaling of call-stateful sip services in the cloud," in *Proc. of Int. IFIP TC6 Conference on Networking (IFIP'12)*. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 175–189.

[21] T. Yang, X. Wen, Y. Sun, Z. Zhao, and Y. Wang, "A new architecture of hss based on cloud computing," in *Proc. of int. Conf. on Communication Technology (ICCT'11)*, Sept 2011, pp. 526–530.

[22] G. Carella, M. Corici, P. Crosta, P. Comi, T. Bohnert, A. Corici, D. Vingarzan, and T. Magedanz, "Cloudified ip multimedia subsystem (ims) for network function virtualization (nfv)-based architectures," in *Proc. of IEEE Int. Symp. on Computers and Communication (ISCC'14)*, vol. Workshops, June 2014, pp. 1–6.

[23] J. Y. Kim and H. Schulzrinne, "Sipcloud: Dynamically scalable sip proxies in the cloud," in *Proc. of Int. Conf. on Principles, Systems and Applications of IP Telecommunications (IPTcomm'11)*. New York, NY, USA: ACM, 2011, pp. 11:1–11:6.

[24] T. Magedanz and F. Schreiner, "Qos-aware multi-cloud brokering for ngn services: Tangible benefits of elastic resource allocation mechanisms," in *Proc. of IEEE Int. Conf. on Communications and Electronics (ICCE'14)*, July 2014, pp. 168–173.