openLEON: An End-to-End Emulation Platform from the Edge Data Center to the Mobile User

Claudio Fiandrino[a,*], Alejandro Blanco Pizarro[a], Pablo Jiménez Mateo[a], Carlos Andrés Ramiro[a], Norbert Ludant[b], Joerg Widmer[a]

[a] *IMDEA Networks Institute, Madrid, Spain*
[b] *Khoury College of Computer Sciences, Northeastern University, Boston, MA, USA*

Abstract

To support next generation services, 5G mobile network architectures are increasingly adopting emerging technlogies like software-defined networking (SDN) and network function virtualization (NFV). Core and radio access functionalities are virtualized and executed in edge data centers, in accordance with the Multi-Access Edge Computing (MEC) principle. While testbeds are an essential research tool for experimental evaluation in such environments, the landscape of data center and mobile network testbeds is fragmented. In this work, we aim at filling this gap by presenting openLEON, an open source muLti-access Edge cOmputiNg end-to-end emulator that operates from the edge data center to the mobile users. openLEON bridges the functionalities of existing emulators for data centers and mobile networks, i.e., Containernet and srsLTE, and makes it possible to evaluate and validate research ideas on all the components of an end-to-end mobile edge architecture.

Keywords: Edge Computing, data center networks, mobile networks, emulation.

1. Introduction

The concept of Multi-Access Edge Computing (MEC), formerly known as Mobile Edge Computing, was standardized by the European Telecommunications Standards Institute (ETSI) and is one of the key enablers for fifth-generation (5G) mobile networks [1, 2]. The MEC paradigm aims at providing computing service closer to the end user by bringing applications and services at close distance to the end-user. MEC is applicable in scenarios where locality and low-latency requirements are essential [3]. As its definition suggests, MEC is not tied to a single radio technology, but embraces both cellular and other radio access technologies such as WiFi. It is also agnostic to the evolution of the mobile network itself and can be deployed in LTE, 4G or 5G networks. For these reason, it is crucial for mobile network

*Corresponding author. Tel.: (+34) 91 481 6932
Email addresses: claudio.fiandrino@imdea.org (Claudio Fiandrino), alejandro.blanco@imdea.org (Alejandro Blanco Pizarro), pablo.jimenezmateo@imdea.org (Pablo Jiménez Mateo), carlos.andres@imdea.org (Carlos Andrés Ramiro), ludant.n@northeastern.edu (Norbert Ludant), joerg.widmer@imdea.org (Joerg Widmer)

operators to understand the impact of MEC on overall mobile system performance in existing networks and to plan network upgrades.

The "edge" is a data center or nano data center deployed close to the base stations inside an operator-owned infrastructure, typically called MEC host, that provides computing functionalities and can aggregate virtualized core and radio network functions of the mobile network. Hence, both the Evolved Packet Core (EPC) and Radio Access Network (RAN), in the form of a Cloud-RAN, can run in the same data center in a virtualized manner. MEC exploits emerging technologies such as software-defined networking (SDN) [4] and network function virtualization (NFV) [5]. While testbeds are essential for research, experimental evaluation and prototype development, the existing landscape of emulators and testbeds does not offer much in the context of MEC as the available ones either target mobile or data center networks separately.

In this work, we aim at filling this gap by presenting openLEON, a muLti-access Edge cOmputiNg end-to-end emulator which spans from the edge data center to the mobile users. openLEON bridges the functionalities of existing emulators, namely srsLTE [6] for the mobile network and Containernet [7], an extension of the popular Mininet [8], to emulate a SDN-based data center network. We make our platform available to the community [9] and share configuration examples and application use cases which allows other researchers to build on them. The objective of openLEON is to enable research experiments in the MEC domain, providing emulation of both data center and mobile network components. While some prior work in this area exists (e.g., [10]), it onlys supports mobile network emulation and does not include the data center environment. With Containernet, it is possible to connect virtual data center hosts and switches through virtual Ethernet links, while packets are processed using the real Linux protocol stack. Furthermore, hosts have access to all kernel functions of the host. To validate openLEON, we implement and assess several use cases, such as the caching proof-of-concept of [10], the impact of Radio Link Control (RLC) buffer size [11] on latency and video streaming with Dynamic Adaptive Streaming over HTTP (DASH) [12]. We also profile the computational efficiency of openLEON for the above use cases. Note that we intentionally leave out from the openLEON implementation some typical cloud computing aspects such as the tuning of virtual machine allocation policies or measuring energy consumption that are provided by simulators like CloudSimSDN [13].

The rest of the paper is organized as follows. Section 2 illustrates the rationale for the choice of srsLTE and Containernet as basis for openLEON and provides an overview of existing emulators in the two domains. Section 3 illustrates the openLEON platform and Section 4 provides an evaluation of the use cases and analyzes the computational efficiency of the platform. Section 5 outlines further open research challenges where openLEON can be applied and Section 6 overviews related works in the area. Finally, Section 7 concludes the work and provides final remarks.

2. Background and Motivation

The objective of this section is to answer the question of whether the combination of srsLTE and Containernet is the right set of tools to develop such a end-to-end emulator.

2.1. Emulation of Mobile Networks/LTE

Emulators duplicate both hardware and software aspects of a real-world testbed, hence providing a significant advantage over simulators that abstract from the hardware behavior and are bounded by the correctness of the simulation model. In the context of LTE networks, emulators can capture real-world channel conditions through parameters like path loss, multipath fading, delay spread, Doppler spread, and other spatial parameters.

srsLTE [6] and OpenAirInterface (OAI) [14] are fully operational open source software implementations of LTE cellular systems. Both platforms enable the emulation on standard Linux-equipment of the main components of an LTE network, such as the user equipment (UE) and base station (eNodeB/eNB) in the Radio Access Network (RAN), and the Mobility Management Entity (MME), Home Subscriber Server (HSS), Serving and Packet Gateways (SGw and PGw) in the Evolved Packet Core (EPC). The main difference among the two platforms is the supported specifications. OAI implements the 3rd Generation Partnership Project (3GPP) Releases 9 and 10, whereas srsLTE is more limited and only implements 3GPP Release 8.

Previous research has compared the relative performance between srsLTE and OAI, highlighting the higher computational efficiency of the latter [15]. At the same time, the modularity of the former makes it easier for developers to customize and extend the code. In addition, other metrics should be considered, such as the stability of the connection between core, eNB and UE. We have tested both platforms extensively to assess such metric and the achievable data rates for different channel bandwidth options. A desktop computer with an Intel i7-6700 processor at 3.4 GHz and 16 GB RAM was used to run the eNB implementation with a NI USRP-2942R as LTE base station. In addition, the EPC application is executed in a virtual machine hosted in the same desktop computer. For the user equipment, we use a Motorola Moto G5 Plus with a custom SIM card (the SIM card features factory-default unique IMSI and a card-individual random subscriber authentication key (K) and operator code (OPC) manually registered in the HSS) to have full access to the emulated LTE network environment.

Table 1 compares the results. While srsLTE achieves a connection stability of more than 1 hour for 5 and 10 MHz channel bandwidth, OAI stability is lower, even dropping to less than 5 minutes of connectivity at 20 MHz. For this reason, we opted for srsLTE as mobile network emulator in openLEON.

Table 1: Performance Analysis of srsLTE [6] and OAI [14]

Frequency [MHz]	Speed Test [Mbps]		Connection Stability [min]	
	srsLTE	OAI	srsLTE	OAI
5	DL:15 - UL:9	DL:10 - UL:8	> 60	> 60
10	DL:22 - UL:13	DL:25 - UL:18	> 60	≤ 60
20	DL:23 - UL:22	DL:45 - UL:35	≤ 60	≤ 5

2.2. Emulating SDN Data Center Networks

To emulate data center networks, the most widely adopted solution is Mininet [8]. Mininet is a flexible experimental platform that allows to easily configure SDN-based network topologies comprising virtual hosts, switches and interconnecting links. Mininet creates separated network contexts (the so-called network namespace mechanism) for the processes running together on a single OS kernel with process-based virtualization. The key features of Mininet are the capability of virtual hosts to run Unix/Linux-based applications and the use of the actual Linux network stack to process packets. For example, it allows to perform live migration of a virtual machine [16]. The main, well-known disadvantage of Mininet is its poor scalability on standard servers. When emulating large testbeds with thousands of virtual switches and high traffic load, the computational complexity of the experiment is overwhelming. Since virtual hosts, switches and applications share the CPU cycles, such scenarios prevent the CPU scheduler of the Linux kernel to precisely control the order of operations. Several solutions help to overcome such shortcomings. MaxiNet [17] spans the computational effort over multiple physical machines, achieving precise emulation of large-scale topologies with thousands of nodes. Virtual-Time Mininet [18] resorts on the *time-dilatation* technique, i.e., the emulated time slows down with respect to real time by a given factor (time-dilatation factor). For example, with a time-dilatation factor equal to 10, every 10 s of real-time corresponds to 1 s of emulated time.

A common disadvantage of all the above platforms is the lack in providing isolation to the applications. While Mininet features network isolation, the processes and the file system of the physical machine remain shared among all the Mininet hosts. Conversely, Docker containers provide full isolation of the applications and can replace traditional Mininet hosts as both support virtual Ethernet (veth) devices technology. In Mininet, hosts and switches are connected through a veth pair, which allows Docker containers to seamlessly substitute Mininet hosts. This insight is the base for MEDICINE [7] and Containernet [19]. The former is a NFV prototype platform that allows researchers and practitioners to develop services controlled by standard Management and Orchestration (MANO) systems as in real-world deployments. MEDICINE builds on Containernet, which extends Mininet in numerous ways. First and foremost, it supports Docker containers in addition to regular hosts. Second, it allows to

configure resource limitations such as the memory limit and the CPU share of each container as well as to bind it to a specific CPU core. Third, it allows for changes of such configuration at runtime.

For the prototype implementation of openLEON, we rely on Containernet. Given that edge data centers are much smaller than conventional clouds, the computational effort is not excessive to require to be split over multiple physical machines. Thus, Containernet that extends Mininet is sufficient and general enough to emulate both networking and computing aspects of the edge data center.

3. The openLEON platform

This section presents the design of openLEON and the methodology to interconnect srsLTE and Containernet.

3.1. Requirements

The design of openLEON satisfies the following requirements:

- Maintain high fidelity in the respective environments, i.e., to incorporate network topologies that faithfully reproduce in smaller scale those of data center networks and the current LTE stack to guarantee correct handling of the mobile traffic.
- At the same time, the programming environment should not restrict the users to explore and research alternative solutions, i.e., the platform should be flexible enough to allow to easily emulate other data center network topologies and to extend the functionalities of the mobile network.

In order to meet these design requirements, openLEON has to overcome a number of challenges that are explained in the next section.

3.2. The Methodology for Interconnection

In conventional EPC, specific interfaces interconnect the components, namely S1: S/PGw $\leftrightarrow$ eNB, S11: SGw $\leftrightarrow$ MME, and S6: HSS $\leftrightarrow$ MME. To build openLEON, we seek a solution that allows to execute the functionalities of such components *within* a nano-data center topology. This translates into executing the corresponding script `srsEPC` from regular Mininet hosts, which is possible given the capability of the hosts to run applications through `xterm` terminals. For the sake of simplicity, we resort to execute all the scripts in one host (see Fig. 1), thus not making explicit the aforementioned S1, S6 and S11 interfaces.

The overall architecture of openLEON is shown in Fig. 2. The eNB application (srsENB) is executed on a physical machine (see left part of 1). Another machine hosts Containernet and executes the EPC functionalities (srsEPC application). We set a `10.0.0.0/12` network for Containernet and a

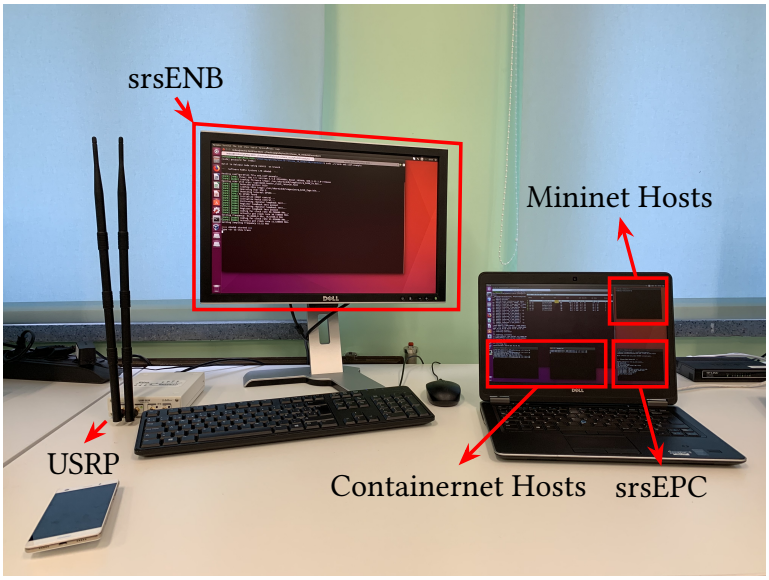


Figure 1: The openLEON testbed

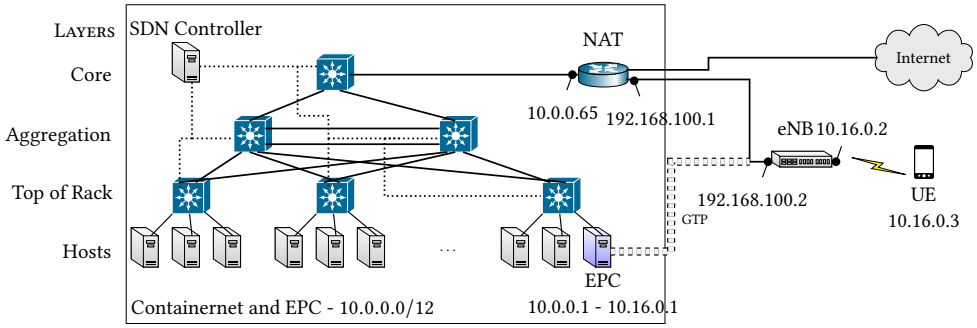


Figure 2: The architectural components - block diagram

`192.168.100.0/24` network to interconnect the physical machines. A PCI Express link interconnects the machine performing eNB processing with the USRP. For the development of openLEON, we tested both Ettus B210 and NI USRP-2942R (with updated PCI Express driver). Such a solution offers the same sample rate (200 MS/s with 16-bit I/Q) as a 10 Gigabit Ethernet connection, with the advantage that no manual IP and MTU size configuration is required from user side. A regular Mininet host (not Docker container) performs NAT functions, bridging the virtual environment interfaces of the edge data center network with the physical machine. We use a `10.16.0.0/12` network for the GPRS Tunneling Protocol (GTP) that provides user-plane connectivity between the S/PGw and the eNB. Given the multi-level virtualized environment and the presence of the GTP tunnel, it is important to properly set up the routing.

openLEON makes the following changes to the routing tables to enable end-to-end connectivity:

- Configuration of the `gtp_bind_addr` of the eNB setting (`enb.conf`) to be in the `192.168.100.0/24` network, e.g., `192.168.100.1`.

- In the physical machine, two entries are included to route traffic from the data center and from the mobile users;
- The traffic generated by the UE and destined to the edge data center is routed via the eNB;
- Traffic generated by Containernet hosts and destined to the mobile users is routed through the EPC host.

We tested the routing configuration using pings, for both smartphones equipped with custom SIM cards and laptops with Nuand BladeRF x40 Software Defined Radios (SDRs) and Ettus B210 as the end devices. Both BladeRFs and Ettus B210 are inexpensive SDRs enabling the setup of a LTE compliant pico cell or UE, offering respectively up to 30 and 56 MHz I/Q sampling rate which is sufficient to decode the 20 MHz LTE channel. For the case of BladeRF and Ettus B210 UEs, bidirectional connectivity is established directly, while for smartphones root privileges are necessary, since commercially available smartphones do not reply to pings.

Finally, we tested the connection stability. Whenever the USRP UHD driver faces problems in sending or receiving samples, it issues late packet messages to inform the user about the internal clock desynchronization that disrupts the connectivity and brings down the network interfaces. As a consequence, the route from the eNB to the EPC host is lost. To overcome this problem, we periodically check the configuration with `traceroute` and re-establish connectivity if necessary.

3.3. The Data Center Topology

Currently, the vast majority of data centers implement a 3-Tier architecture, which is based on a classical 5-stage Clos network and consists of three levels of switches, Top of Rack (ToR), aggregation and core [20]. As default data center topology, openLEON implements in Containernet a 3-Tier architecture with 2 core switches, 2 aggregation switch per-core switch and a total of 64 hosts arranged into 8 racks (where a rack comprises the servers and ToR switches within one physical cabinet). In addition, we create an additional host with NAT functionalities interconnected with the core switches.

In Containernet/Mininet, the reference SDN controller (*ovs-controller* from Mininet 2.0.0) does not implement the spanning-tree protocol by default. Since data center topologies have inherent loops, we instead use a RYU controller. RYU is a Python module that supports OpenFlow protocol and provides an API that facilitates network management and control.

4. Use Case Evaluation

This section validates the performance of the platform by evaluating a series of use cases (Section 4.2) and analyzing its computational efficiency (Section 4.3).

4.1. Testbed Configuration

For the experiments, we configured openLEON as follows. The laptop running the `srsEPC` application from srsLTE version 18.3.1 and Containernet is equipped with an Intel i7-4600U processor at 2.10 GHz, 8 GB RAM and Linux Ubuntu 16.04 LTS. The `srsENB` application from srsLTE version 18.3.1 runs over a desktop computer equipped with an Intel i7-6700 processor running at 3.4 GHz, 16 GB RAM and Linux Ubuntu 16.04 LTS. The LTE station consists of an Ettus B210 connected with USB 3.0 connectivity with the desktop computer. By default, as UEs, we use preferentially a laptop with an Intel i7-4600U processor up to 2.1 GHz and 8 GB of RAM and as second UE a laptop with an Intel i5-4200U processor up to 2.6 GHz and 16 GB of RAM. Those are connected to Nuand BladeRF x40 SDRs. For the experiment with distributed edge data centers and video streaming, the UE is a desktop computer equipped with an Intel i7-6700 processor running at 3.4 GHz, 16 GB RAM and Linux Ubuntu 16.04 LTS wireless connected to the base station with an Ettus B210. Unless otherwise stated, the data center network is the one described in § 3.3.

4.2. Experiments

Caching. ETSI has defined six relevant use cases for MEC: video analytics, location services, Internet-of-Things (IoT), augmented reality, optimized local content distribution, and data caching [1]. The latter use case enables faster response and alleviates congestion in the core. Previous work with OAI platform highlighted the benefit of local caching at the eNB or at the MEC host compared to non-local caching [10]. However, the case with packet caching at MEC hosts does not take into account the characteristics of a data center topology. In this experiment, we fill this gap and analyze the RTT achieved with different options for content placement, namely local caching at the edge data center in a randomly selected host in the same and different rack of the host with EPC functionalities. We conduct experiments by generating traffic with the ping utility, and compare the performance of local caching versus non-local caching, where the UE connects to the Global Amazon AWS service based in Seattle. For the experiment, a total of 120 echo requests over 120 seconds are transmitted, with the packet size set to 64, 768, 2048, 4096, and 8192 bytes. The UE is placed at 2 m in line of sight of the USRP to ensure a stable channel quality. The hosts in the data center generate UDP background traffic with 200 Mbps of target bandwidth with the iperf tool according to a pre-configured permutation matrix (each host generates iperf traffic towards another host randomly chosen so that each host is at the same time transmitting and receiving traffic).

Fig. 3 shows the results in form of a box-plot (left axis) and depicts the fraction of timed-out packets (right axis), i.e., the packets for which the ping utility has waited a reply for a certain amount of time (twice the length of the maximum RTT by default). First, we observe that as expected the higher number of outliers (depicted with crosses) and timed-out packets occur for non-local caching tests.

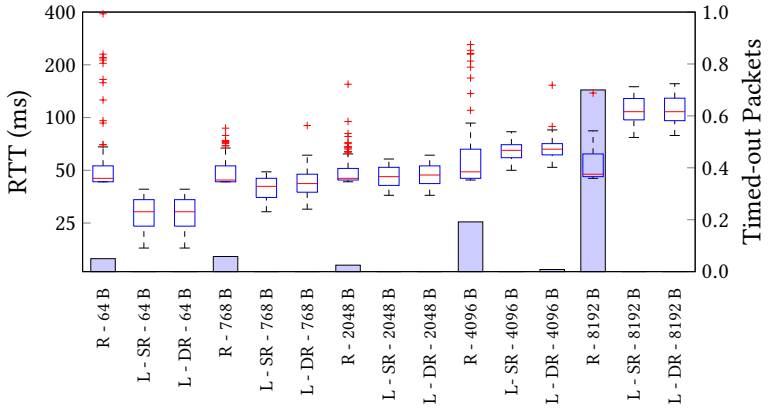


Figure 3: Measured RTT from UE to a remote data center (R) and local host (L) in same rack (SR) and different rack (DR)

Although latency-sensitive applications can considerably benefit from RTTs with low-variability, the ping utility can not measure jitter accurately as the echo replies depend on the OS scheduler of the end host, which introduces jitter itself. Second, for a small packet size (64-768 bytes), the advantage of local caching is significant. For intermediate packet sizes (2048-4096 bytes), the statistical difference in RTTs is negligible while for large packet size (8192 bytes) local caching does not bring any benefit apparently. However, this is not true, as the fraction of timed-out packets for non-local caching exceeds 70% while it is null for both local caching cases. Interestingly, we note that the placement in local or different rack does not affect significantly the RTTs. The next experiment, however, highlights that this is not always true.

RLC Buffer Size. End-to-end latency is one of the main goals of 5G networks to enable Ultra-Reliable and Low Latency Communications (URLLC). For URLLC, 3GPP has defined a target user plane latency of 0.5 ms for both uplink and downlink transmissions, and a reliability requirement of 5-nines $(1-10^{-5})$ for the transmission of a 32 byte packet[1]. Queueing delay is one of the main factors preventing a low end-to-end latency. The total per-user buffer space allocated for RLC in acknowledge mode is given in number of SDU, typically 1024 for a UE cat 3, i.e., 1.4 MB to accommodate 1500 bytes long RLC SDUs. When large TCP flows and traffic from interactive applications are simultaneously destined to the same user, they share the same RLC buffer. As TCP aims at filling the buffer, the queuing delay grows significantly which is detrimental to the performance of the interactive applications [11]. In turn, this also reduces the precision of the TCP retransmission timeout estimation. Consequently, TCP may experience unnecessary timeouts, causing retransmissions and slow start, and thus leading to poor link utilization.

This experiment analyzes the latency of interactive applications while sharing the RLC buffer with

[1]3GPP, "Tech. Report #: 38.913, Release 14," 2017.

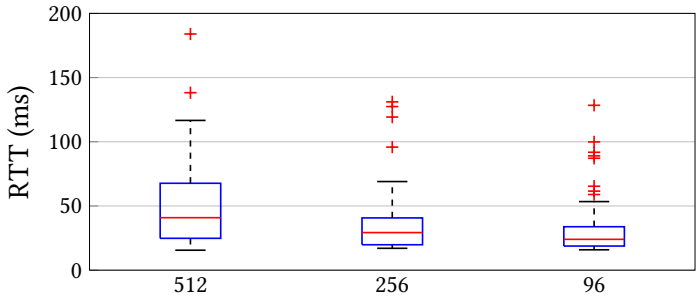
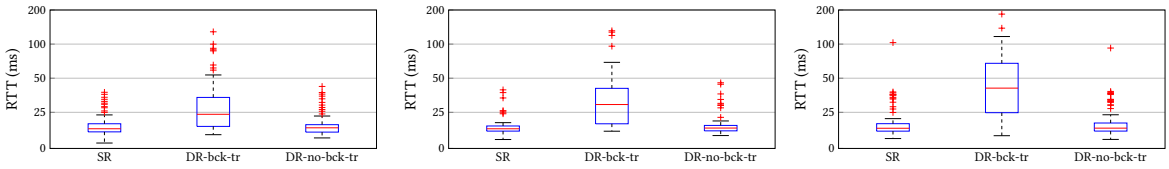


Figure 4: Impact of various RLC buffer sizes on RTT (in number of RLC SDU)



(a) Buffer size 96 (b) Buffer size 256 (c) Buffer size 512

Figure 5: Impact on RTT of various RLC buffer sizes (in number of RLC SDU), application placement in data center network - same rack (SR) or different rack (DR) of the EPC host - and presence of background traffic

a UDP flow. The choice for UDP provides control over the target rate of traffic injection (target bandwidth 4 Mbps for 60 s). Unlike previous research [11], we investigate performance when the sender resides in the same rack of the host performing EPC functionalities or in a different one. Furthermore, we provide an assessment in the presence of background traffic in the edge data center (the hosts exchange UDP traffic with iperf in a permutation matrix with target bandwidth of 60 Mbps for 1000 s). In Fig. 4, we verify the achievable RTTs analyzing `pcap` files captured at UE, eNB and GTP0 interface. As expected, the RTTs increase with the increase of size of the RLC buffer. Next, in Fig. 5 we verify the impact of placement and background traffic on the achievable RTTs. The placement in the same rack (SR) leads to better performance for any buffer size. When the application resides in a different rack with background traffic (DR-bck-tr), the RTTs increase with the increase of the buffer size. To achieve low RTTs while placing the application in a different rack, zero background traffic should flow in the data center (DR-no-bck-tr).

Distributed Edge Data Centers. To achieve low latency, proper configuration of the RLC buffer is of help, but other solutions could be considered as well. For example, to overcome the delays generated by heavy background traffic in the edge data center network, a delay-constrained service could be migrated to a nearby edge data center. Fig. 6 shows the network topology considered: two edge data centers, EDC1 and EDC2 are interconnected through a 2 ms delay link at 1 Gbps. For simplicity, each edge data center consists of 2 regular Mininet hosts and 4 Containers. EDC1 features 2 additional hosts for EPC and NAT functionalities and 2 switches provide interconnection to the hosts with 1 Gbps

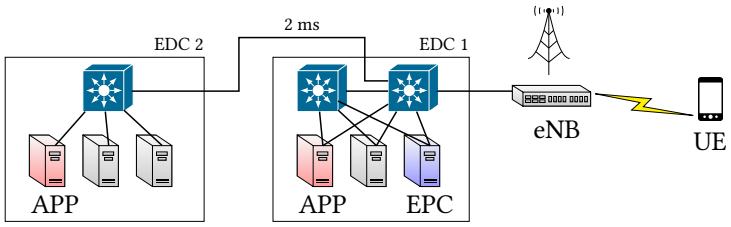


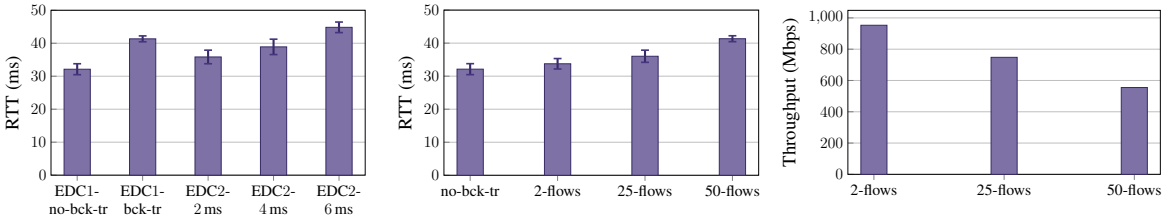
Figure 6: Architecture with distributed data centers

links. Only one switch interconnects the hosts of EDC2 at the same speed. This minimal configuration is realistic enough to verify the benefit of migrating the service from EDC1 to EDC2 when heavy background traffic makes the former congested.

Fig. 7(a) compares the RTT achieved by executing the ping application from the APP node in the edge data centers to the UE. A total of 10 echo requests with packet size set to 64 Bytes are transmitted per run and the results are averaged over 30 runs. An interval of 4 s divides two subsequent runs. Obviously, the lowest RTT is achieved when the service is deployed in EDC1 and no background traffic is present (32.12 ms). By opening 50 parallel iperf TCP sessions among the regular hosts of EDC1 (each host is at the same time server and client, thus accounting for a total of 100 TCP connections contending for network resources) the network becomes congested and the RTT of the ping application spikes to 41.34 ms. Please note that we exploit version 2 of the tool because it is multi-threaded and achieves superior performance with parallel streams than iperf3 which is single-threaded. Under such conditions, migrating the service to EDC2 may be beneficial and we assess how far away EDC2 can be so that service migration is beneficial. When the delay of the link is lower than 6 ms, the achieved RTTs are on average 35.84 and 38.90 ms with 2 and 4 ms link delay respectively. This corresponds to a gain of 15 % and 6 % without having the UE to change any configuration as it remains attached to the same eNB. When the delay of the link interconnecting EDC1 and EDC2 is above 6 ms, there is no advantage in migrating the service as the average RTT is 44.83 ms, much larger than maintaining the service at EDC1. If such a link is optical fiber (3.34 µs propagation delay per kilometer), 100 km length would approximately lead to 1 ms round-trip delay for an uncongested path. Hence, the distance between EDC2 and EDC1 has to be shorter than 400 km approximately in order for the migration to be beneficial.

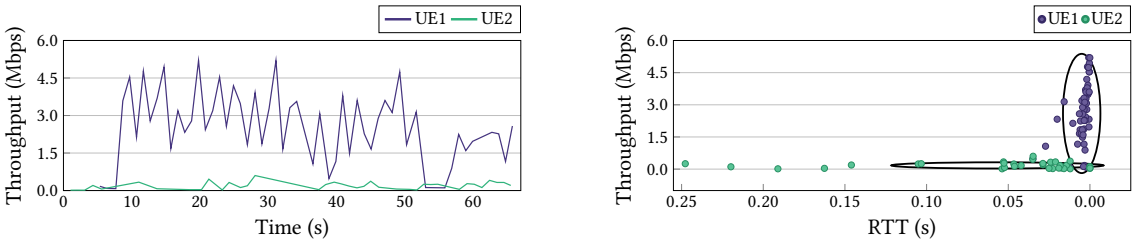
Fig. 7(b) and Fig. 7(c) show respectively the RTT of the interactive application varying the amount of background traffic and the overall network throughput of EDC1. As expected, by opening an increasing number of parallel connections, both RTT and overall throughput worsen.

Mobile Users. This experiment assesses the performance of mobile users with different channel qualities. To this end, one data center host simultaneously opens two TCP connections towards different UEs. UE1 is positioned at 1.4 m from the USRP with line-of-sight (LoS), while UE2 is positioned



(a) Latency comparison (b) Impact of background traffic on latency (c) Throughput of background traffic

Figure 7: Migration of a latency-sensitive service in distributed data centers



(a) Throughput analysis (b) Joint throughput and RTT analysis

Figure 8: Throughput and RTT performance of mobile users

further at 2.2 m with an obstacle obstructing the LoS. The resulting CQI reports are in the range of 13-14 and 7-8 for UE1 and UE2 respectively. The RLC buffer size is set to 512.

Fig. 8(a) shows the achieved throughput of both UEs. As expected, UE1 significantly outperforms UE2 with an average throughput of 2.6 Mbps compared to 187.5 Kbps that UE2 achieves. Fig. 8(b) shows performance of the two users. The graph is a throughput-RTT plot, where for each scenario we take the average results from `pcap` traces captured at the UEs and compute the $2-\sigma$ elliptic contour of the maximum-likelihood 2D Gaussian distribution. The $2-\sigma$ expression defines the level of confidence, i.e., the projection on a one-dimensional sub-space of the elliptic contour is the 95% confidence interval. On the x-axis, lower (better) RTTs are to the right. Hence, the best performing UE, UE1, is on the top-right. Throughput-RTT plots highlight the variability and relative performance of the metrics in the two dimensions. The narrower the ellipses in the axis dimension, the more stable is the protocol in consistently achieving similar throughput or RTT. On the other hand, wider ellipses indicate higher variability. The ellipses' orientations define the relationship between throughput and RTT. UE1 benefits from stable connectivity, achieving low values of RTT and highly variable throughput. On the contrary, the poor channel quality experienced by UE2 leads to the complete opposite performance, with low throughput and highly variable delays due to retransmissions and timeouts.

Exploiting Multiple Paths. With Multipath-TCP (MP-TCP) [21], flows can exploit more than one path simultaneously, by transmitting over multiple interfaces or to different interfaces of a host.

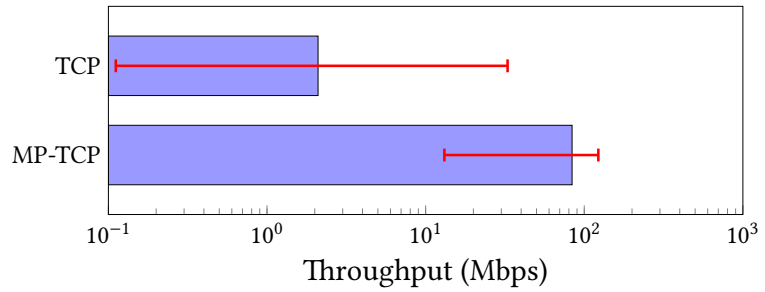


Figure 9: Performance of MP-TCP and TCP

MP-TCP finds applicability in both data centers and wireless networks.

MP-TCP kernel version 0.93[2] does not provide a module to support the GTP interface, hence we downloaded the sources of the kernel version 4.13 which has an available MP-TCP patch and includes the GTP module. Then we patched the kernel with the suitable patch of MP-TCP, enabled all the components of MP-TCP and GTP and built the kernel. This allows to use MP-TCP through GTP tunnels. We enable it by setting `sysctl -w net.mptcp.mptcp_enabled=1`.

For the experiment, we modified the original topology defined in § 3.3, so that each host is interconnected with four ToR switches at a time to enable multiple paths. The hosts generate UDP background traffic with 100 Mbps of set bandwidth according to a permutation matrix. We send traffic from the UE to an end host in the data center and measure the throughput observed by the segment from EPC host to end host. Fig. 9 illustrates the significant advantage that MP-TCP provides with respect to TCP in terms of achieved throughput.

Video Streaming. Nowadays, the majority of video traffic is delivered over the Internet with Dynamic Adaptive Streaming over HTTP (DASH) [12]. With DASH, a video is split into a number of *chunks* (e.g., a 4 s long block) encoded at different bit rates and stored with a descriptor named Media Presentation Description (MPD). The DASH client player estimates the available bandwidth and indicates to the server which chunk representation, i.e., bit rate, to download among those listed by the MDP. This technique is called Adaptive Bit Rate (ABR) selection and allows content providers to optimize video quality. In the literature, numerous ABR techniques have been proposed [22]. Some solely rely on throughput-based estimation for the bit-rate selection, others on buffer occupancy at the client or on estimation of the chunk download time.

With this experiment, we expose how to perform experiments with ABR video streaming over openLEON (see Fig. 10). We choose the popular BigBuckBunny video[3] and encode it with the `ffmpeg` tool at different resolutions:

[2]Available at: `https://www.multipath-tcp.org/`
[3]Available at: `https://peach.blender.org/`

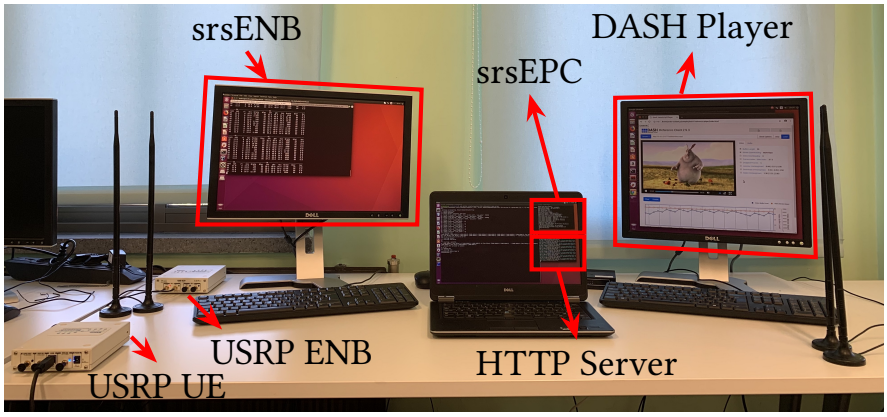


Figure 10: Implementation of DASH application over openLEON

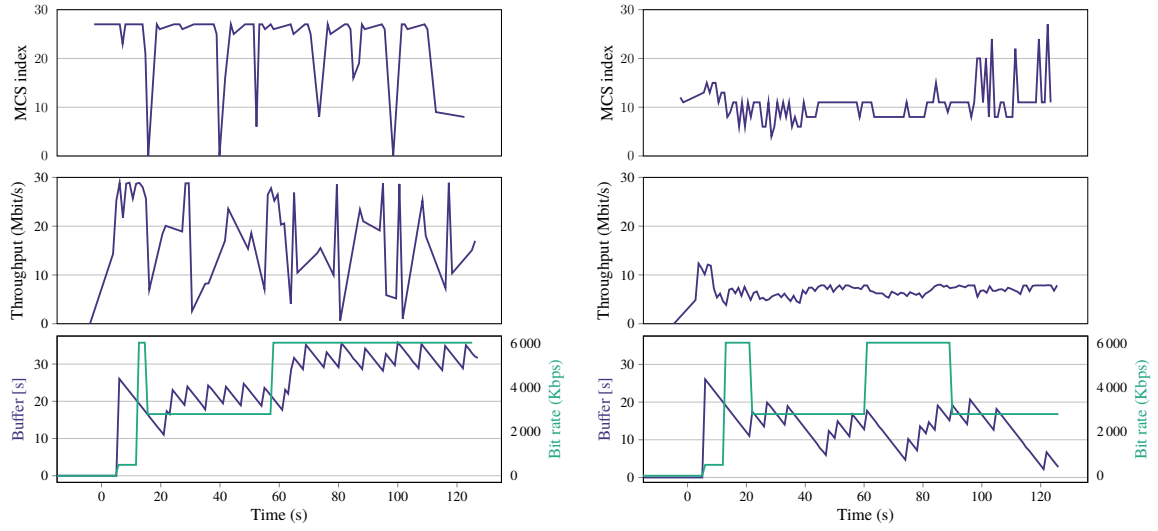
- 256 × 144 at a bit rate of 500 Kbps;
- 858 × 480 at a bit rate of 2800 Kbps;
- 1280 × 720 at a bit rate of 6000 Kbps.

At the server side, we open a Node.js http-server from a regular Mininet node and enable Cross-Origin Resource Sharing (CORS). At the client side, we install the DASH.js reference player[4] and extend it to output on file statistics about video buffer level and video bit rate by default available in the main page (see lower part of DASH player in Fig. 10). By default, the DASH.js player exploits Buffer Occupancy based Lyapunov Algorithm (BOLA) [23], an ABR technique that determines the bit rate selection by solving an optimization problem that only requires knowledge about the amount of buffered data at the player, i.e., the chunks stored prior rendering on the screen.

Fig. 11 analyzes Modulation and Coding Scheme (MCS) used at the eNB, the throughput and video statistics at the UE. We compare two different scenarios. In the first one, the UE experiences good propagation conditions leading to an average throughput of more than 18 Mbps (see Fig. 11(a)). In the second one, the UE is positioned 2.2 m further from the eNB with an obstacle obstructing the LoS, which makes the throughput be consistently below 10 Mbps (see Fig. 11(b)).

At video start-up, in both scenarios BOLA buffers nearly 27 s of video and quickly ramps-up to achieve the highest resolution available, and in turn, the highest bit rate. This drains the buffer quickly and leads to a downgrade in resolution. BOLA's optimization algorithm is driven by two components, the average bit rate of the video (i.e., the encoding quality) and the duration of re-buffering, i.e., the chunks that are download more than once. This makes BOLA robust towards throughput variations that do not significantly affect the bit rate selection as is evident in Fig. 11(a). In such scenario, with

[4]Available at: `https://github.com/Dash-Industry-Forum/dash.js`



(a) High CQI (b) Low CQI

Figure 11: MCS, throughput, bit rate and buffer occupancy of video streaming application

higher available throughput, BOLA successfully recovers the highest encoding quality to provide a richer experience. Conversely, in Fig. 11(b), such an attempt (near the 60 s mark) leads to a quick buffer drain and re-buffering, which makes BOLA downgrade again the encoding quality.

4.3. Computational Efficiency

Fig. 12 shows the CPU utilization measured with the Glances monitoring tool[5] for the majority of the above use cases. We measure both *user* and *system* CPU performance, i.e., the CPU consumption due to userspace and OS kernel processes. The tool records measurements with a 3 ms granularity on the laptop running Containernet and the `srsEPC` application.

Starting Containernet is computationally intensive, with user CPU reaching up to 51.8% and system CPU load reaching up to 19.6%. Subsequent operations such as starting the `srsEPC` application, the connection of the UE and starting the HTTP-server do not impact the CPU load at all. We now analyze three different applications: video streaming, CPU load generator and iperf. The video is streamed for 3 minutes with the set-up described in Subsection 4.2 and streaming has a minor impact on the CPU load. We next instruct two Containernet hosts with LoadGen image that generate CPU load[6] to run for 1 minute on the same core and to load the CPU at 40 %. Within Containernet, the two Containernet hosts were set in the edge data center network topology by using the following command:

```
d1 = net.addDocker('d1', ip='10.0.0.3/24', dimage="furiousgeorge/cpuload",
```

[5]Available at: `https://glances.readthedocs.io/en/stable/`

[6]Available at: `https://github.com/hannah98/docker-cpuload`

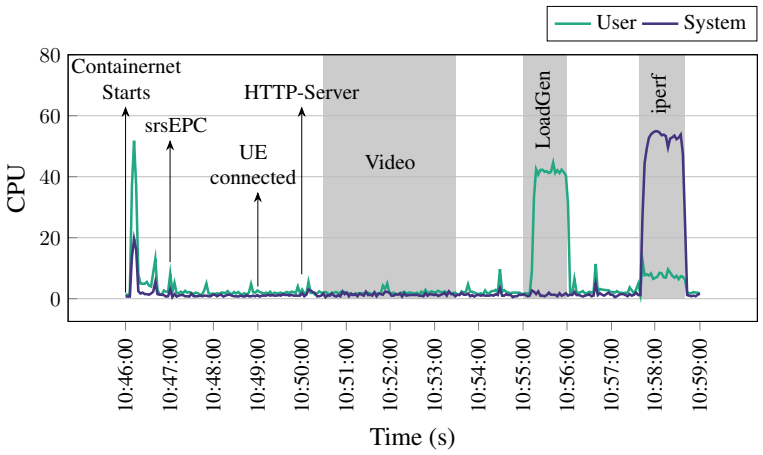


Figure 12: CPU utilization of Containernet for a number of applications

```
cpu_period=50000)
d2 = net.addDocker('d2', ip='10.0.0.4/24', dimage="furiousgeorge/cpuload",
cpu_period=50000)
```

Fig. 12 confirms that the user CPU load remains close to 40 %, with some marginal overhead. The last experiment consists in running from 2 regular Mininet hosts, 50 parallel TCP iperf sessions for 1 minute. Please note that each host is both client and server at the same time, hence a total of 100 connections are active simultaneously. With this application, the system CPU load averages 52 % and the user CPU load remains around 9 %. Unlike the CPU load generator application, packet processing heavily involves the Linux kernel and this explains why this load is accounted as system CPU.

5. Open Research Challenges and Application Scenarios

This section provides a brief overview of possible uses and application areas of openLEON and exposes as well open research challenges.

Transport. In edge data centers the whole transport, from application server to the mobile terminal, is under the control of the mobile network operator. Conventional transport protocols like TCP are known to perform poorly in such a setting. For example, a sudden cell load increase limits user bandwidth availability [24] and the increased delay, due to large queues at base stations, can reduce the precision of the TCP retransmission timeout estimation. Consequently, TCP may experience unnecessary timeouts, causing retransmissions and slow start, and thus leading to poor link utilization. MEC architectures, bringing data center and radio networks together, should take advantage of resource pooling and information about feedback on channel quality that is available as part of the Radio Network Information Services. openLEON provides the researchers with the capability to evaluate

various transport protocols or cross-layer approaches coupling congestion control with lower layers. Some examples of such protocols are:

- TCP BBR [25], available from kernel version 4.9, responds to actual congestion rather than packet loss like TCP CUBIC, the de-facto protocol implemented in the Linux kernel. Specifically, BBR estimates the delivery rate at the receiver to determine the bottleneck link capacity and send exactly at this rate.

- Quick UDP Internet Connection (QUIC) [26] and Multipath QUIC [27]. Google has developed QUIC by combining functionalities of HTTP/2, TLS, and TCP. QUIC runs over UDP and nowadays it accounts for nearly 10 % of the total Internet traffic. Multipath QUIC extends QUIC to allow a single connection to exploit different paths in analogy to MP-TCP.

Multi-RAT. The srsLTE platform can be employed for performance analysis of multiple radio access technologies (RAT). Previous research provided an assessment of the coexistence of LTE unlicensed and 802.11a/b/g/n [6]. Building on srsLTE, openLEON provides such capability as well. As a follow-up of the experiment on MP-TCP (§ 4.2), a promising direction consists in analyzing the coexistence of LTE and 802.11 family, including 802.11ad that utilizes the mm-wave 60 GHz band. The latter offers much higher data rates, but is susceptible to blockage, leading to interesting considerations, such as how to determine the optimal injection rate over the multiple interfaces to limit out-of-order reception at the receiver side.

The concept of such multi-RAT LTE and WLAN integration (LWA) is not new and was first introduced by 3GPP in Release 13 and successively extended in Release 14 [28]. With LWA, a single TCP connection is split at the Packet Data Convergence Protocol (PDCP) layer of LTE base stations. Thus, the *LWA path scheduler* takes decisions at the last mile and not at the source as MP-TCP does by considering instantaneous radio conditions and congestion situations. A preliminary work in the area exposed the capability of LWA in achieving higher link utilization and fairness than MP-TCP [29].

Mobile Cloud Computing. Outsourcing part of the computing-intensive tasks from the resource-constrained mobile devices to the cloud enables energy-savings and augments the capabilities of mobile devices [30]. The challenge is to precisely estimate the amount of time it takes to offload and execute a task over a remote entity (edge/cloud). This can be done, for example, by means of learning algorithms [31]. The vast majority of experimental works in mobile cloud computing limits the scope of the tests to WiFi. openLEON not only overcomes such limitation, but also offers kernel-level access to applications. For example, for object recognition, hosts in openLEON can execute OpenCV[7] methods for image processing and feature extraction.

[7]Available at: `https://opencv.org/`

6. Related Work

This section surveys prior work on emulators for mobile networks (LTE/5G), SDN data centers, and the few recent proposals in the context of 5G and fog computing.

Natively, Mininet does not support specific features of wireless links such as interference, mobility, or channel selection. To overcome such limitation, Mininet-WiFi [32] emulates the wireless channel by exploiting Linux Traffic Control utility to configure the kernel packet scheduler by setting parameters like channel bandwidth, packet loss and delay. Other attempts aimed at modeling wireless links *within* the Mininet environment use the emulation features of ns-3 for WiFi[8] and the Lena module of ns-3 for the cellular network (OpenNet[9]). While OpenNet is similar to openLEON, it is not an end-to-end emulator and does not model the data center. An emulator for LTE using LENA is available [33].

Closest to our work are [34, 35]. Both are experimental platforms for 5G networks with SDN-control for the EPC. The latter initiative incorporates state-of-the-art components, such as OAI for modeling the EPC and OpenDaylight as SDN controller. POSENS [36] features an end-to-end network slicing protocol stack and implements all the components of the mobile network (UE, RAN, core network) and an orchestration framework. Unlike openLEON, all the above solutions lack the fine modeling of the properties of data center networks.

A number of emulation platforms have been recently proposed in the area of fog computing. EmuFog [37] allows to create fog computing infrastructures enabling developers to incorporate network topologies into MaxiNet. However, MaxiNet does not feature any support for wireless communications. Hence this limits the usability of EmuFog with IoT devices. MockFog [38] is an emulation platform that captures well the computing dynamics and host the entire emulation process of applications over computing, storage and memory resources in the cloud. In other words, MockFog emulates fog architectures consisting of edge devices such as Raspberry Pis, cloudlets and clouds over infrastructures like Amazon EC2. The main shortcoming of such approach is the poor emulation of network dynamics as no notion of protocols nor technology is given. Specifically, MockFog captures simple parameters like incoming/outgoing maximum rates, delay, losses, reordering time, duplicate packets. Unlike MockFog, EmuEdge [39] strengths significantly the precision of network emulation that is based on Linux network namespaces. From the computing size, by means of Xen EmuEdge supports both containers and virtual machines. Unlike openLEON, all the above solutions fail to capture the dynamics of the cellular mobile network.

[8] Available at: `https://github.com/mininet/mininet/wiki/Link-modeling-using-ns-3`
[9] Available at: `https://github.com/dlinknctu/OpenNet`

7. Conclusion

In this work, we presented openLEON, an open-source platform that enables experimentation and prototyping in a MEC context. We described the key components of openLEON, srsLTE and Containernet, and the architectural challenges we solved to combine them. We evaluated its computational efficiency and assessed specific use-cases. The obtained initial results are promising and open up further areas that are interesting for future research, for example in the context of mobile cloud computing. In summary, leveraging the presence of Radio Network Information Services at edge data centers opens the door for cross-layer end-to-end optimizations at transport, network and MAC layers.

Acknowledgment

This work is partially supported by the the Madrid Regional Government through the TAPIR-CM program (S2018/TCS-4496) and the Juan de la Cierva grant from the Spanish Ministry of Science, Innovation and Universities (FJCI-2017-32309).

References

[1] F. Giust, X. Costa-Perez, A. Reznik, Multi-access edge computing: An overview of ETSI MEC ISG, IEEE 5G Tech Focus 1 (4).

[2] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, D. Sabella, On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration, IEEE Communications Surveys Tutorials 19 (3) (2017) 1657–1681. doi:10.1109/COMST.2017.2705720.

[3] F. Giust, G. Verin, K. Antevski, J. Chou, Y. Fang, W. Featherstone, et al., MEC deployments in 4G and evolution towards 5G, ETSI White Paper (Feb 2018).

[4] K. Kirkpatrick, Software-defined networking, Communications of the ACM 56 (9) (2013) 16–19.

[5] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. D. Turck, R. Boutaba, Network function virtualization: State-of-the-art and research challenges, IEEE Communications Surveys Tutorials 18 (1) (2016) 236–262.

[6] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, D. J. Leith, srsLTE: An open-source platform for lte evolution and experimentation, in: Proc. of ACM WiNTECH, 2016, pp. 25–32.

[7] M. Peuster, H. Karl, S. van Rossem, MeDICINE: Rapid prototyping of production-ready network services in multi-PoP environments, in: Proc. of IEEE NFV-SDN, 2016, pp. 148–153. doi:10.1109/NFV-SDN.2016.7919490.

[8] B. Lantz, B. Heller, N. McKeown, A network in a laptop: Rapid prototyping for software-defined networks, in: Proc. of ACM Hotnets-IX, 2010, pp. 1–6.

[9] C. Fiandrino, A. B. Pizarro, P. J. Mateo, C. A. Ramiro, N. Ludant, J. Widmer, openLEON: an open source multi-access edge computing emulator, available at `https://openleon.networks.imdea.org/` (May 2019).

[10] C.-Y. Chang, K. Alexandris, N. Nikaein, K. Katsalis, T. Spyropoulos, MEC architectural implications for LTE/LTE-A networks, in: Proc. of ACM MobiArch, 2016, pp. 13–18.

[11] R. Kumar, A. Francini, S. Panwar, S. Sharma, Dynamic control of RLC buffer size for latency minimization in mobile RAN, in: IEEE WCNC, 2018, pp. 1–6.

[12] M. Graf, C. Timmerer, C. Mueller, Towards bandwidth efficient adaptive streaming of omnidirectional video over HTTP: Design, implementation, and evaluation, in: Proc. of ACM MMSys, 2017, pp. 261–271. doi:10.1145/3083187.3084016.

[13] J. Son, A. V. Dastjerdi, R. N. Calheiros, X. Ji, Y. Yoon, R. Buyya, CloudSimSDN: Modeling and simulation of software-defined cloud data centers, in: IEEE/ACM CCGrid, 2015, pp. 475–484.

[14] N. Nikaein, M. K. Marina, S. Manickam, A. Dawson, R. Knopp, C. Bonnet, Openairinterface: A flexible platform for 5G research, ACM SIGCOMM Comput. Commun. Rev. 44 (5) (2014) 33–38.

[15] F. Gringoli, P. Patras, C. Donato, P. Serrano, Y. Grunenberger, Performance assessment of open software platforms for 5G prototyping, IEEE Wireless Communications 25 (5) (2018) 10–15. doi:10.1109/MWC.2018.1800049.

[16] E. Keller, S. Ghorbani, M. Caesar, J. Rexford, Live migration of an entire network (and its hosts), in: Proc. of ACM HotNets-XI, 2012, pp. 109–114.

[17] P. Wette, M. Dräxler, A. Schwabe, MaxiNet: Distributed emulation of software-defined networks, in: IFIP Networking, 2014, pp. 1–9.

[18] J. Yan, D. Jin, VT-Mininet: Virtual-time-enabled mininet for scalable and accurate software-define network emulation, in: Proc. of ACM SOSR, 2015, pp. 1–7.

[19] M. Peuster, J. Kampmeyer, H. Karl, Containernet 2.0: A rapid prototyping platform for hybrid service function chains, in: IEEE Conference on Network Softwarization and Workshops (NetSoft), 2018, pp. 335–337. doi:10.1109/NETSOFT.2018.8459905.

[20] P. Ruiu, C. Fiandrino, P. Giaccone, A. Bianco, D. Kliazovich, P. Bouvry, On the energy-proportionality of data center networks, IEEE Trans. on Sustainable Computing 2 (2) (2017) 197–210.

[21] D. Wischik, C. Raiciu, A. Greenhalgh, M. Handley, Design, implementation and evaluation of congestion control for multipath tcp, in: Proc. of USENIX NSDI, 2011, pp. 99–112.

[22] Y. Sani, A. Mauthe, C. Edwards, Adaptive bitrate selection: A survey, IEEE Communications Surveys Tutorials 19 (4) (2017) 2985–3014. doi:10.1109/COMST.2017.2725241.

[23] K. Spiteri, R. Urgaonkar, R. K. Sitaraman, BOLA: Near-optimal bitrate adaptation for online videos, in: Proc. IEEE INFOCOM, 2016, pp. 1–9. doi:10.1109/INFOCOM.2016.7524428.

[24] B. Nguyen, A. Banerjee, V. Gopalakrishnan, S. Kasera, S. Lee, A. Shaikh, J. Van der Merwe, Towards understanding TCP performance on LTE/EPC mobile networks, in: Proc. of ACM All Things Cellular, 2014, pp. 41–46.

[25] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, V. Jacobson, BBR: Congestion-based congestion control, Commun. ACM 60 (2) (2017) 58–66.

[26] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, W.-T. Chang, Z. Shi, The QUIC transport protocol: Design and internet-scale deployment, in: Proc. of ACM SIGCOMM, 2017, pp. 183–196. doi:10.1145/3098822.3098842.

[27] Q. De Coninck, O. Bonaventure, Multipath QUIC: Design and evaluation, in: Proc. of ACM CoNEXT, 2017, pp. 160–166. doi:10.1145/3143361.3143370.

[28] D. Laselva, D. Lopez-Perez, M. Rinne, T. Henttonen, 3GPP LTE-WLAN aggregation technologies: Functionalities and performance comparison, IEEE Communications Magazine 56 (3) (2018) 195–203. doi:10.1109/MCOM.2018.1700449.

[29] B. Jin, S. Kim, D. Yun, H. Lee, W. Kim, Y. Yi, Aggregating LTE and Wi-Fi: Toward intra-cell fairness and high TCP performance, IEEE Transactions on Wireless Communications 16 (10) (2017) 6295–6308. doi:10.1109/TWC.2017.2721935.

[30] S. Guo, J. Liu, Y. Yang, B. Xiao, Z. Li, Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing, IEEE Transactions on Mobile Computing 18 (2) (2019) 319–333. doi:10.1109/TMC.2018.2831230.

[31] M. Hu, L. Zhuang, D. Wu, Y. Zhou, X. Chen, L. Xiao, Learning driven computation offloading for asymmetrically informed edge computing, IEEE Transactions on Parallel and Distributed Systems (2019) 1–14doi:10.1109/TPDS.2019.2893925.

[32] R. d. R. Fontes, C. E. Rothenberg, Mininet-WiFi: A platform for hybrid physical-virtual software-defined wireless networking research, in: Proc. of ACM SIGCOMM, 2016, pp. 607–608.

[33] V. Mancuso, C. Vitale, R. Gupta, K. Rathi, A. Morelli, A prototyping methodology for SDN-controlled LTE using SDR (Dec 2014).

[34] A. Huang, N. Nikaein, Demo: LL-MEC A SDN-based MEC platform, in: Proc. of ACM MobiCom, 2017, pp. 483–485.

[35] K.Ramantas, E.Kartsakli, M.Irazabal, A. Antonopoulos, C. Verikoukis, Implementation of an SDN-enabled 5G Experimental Platform For Core and Radio Access Network Support, in: Interactive Mobile Communication Technologies and Learning, Springer International Publishing, 2017, pp. 791–796.

[36] G. Garcia-Aviles, M. Gramaglia, P. Serrano, A. Banchs, POSENS: A practical open source solution for end-to-end network slicing, IEEE Wireless Communications 25 (5) (2018) 30–37. doi:10.1109/MWC.2018.1800050.

[37] R. Mayer, L. Graser, H. Gupta, E. Saurez, U. Ramachandran, EmuFog: Extensible and scalable emulation of large-scale fog computing infrastructures, in: Proc. IEEE Fog World Congress, 2017, pp. 1–6. doi:10.1109/FWC.2017.8368525.

[38] J. Hasenburg, M. Grambow, E. Grünewald, S.Huk, D. Bermbach, MockFog: Emulating fog computing infrastructure in the cloud, in: Accepted in Proc. IEEE International Conference on Fog Computing, 2019, pp. 1–6.

[39] Y. Zeng, M. Chao, R. Stoleru, EmuEdge: A hybrid emulator for reproducible and realistic edge computing experiments, in: Accepted in Proc. IEEE International Conference on Fog Computing, 2019, pp. 1–6.

Biographies

Claudio Fiandrino joined as a postdoctoral researcher the IMDEA Networks Institute in December 2016 right after having obtained his Ph.D. degree at the University of Luxembourg. He received the Bachelor Degree in Ingegneria Telematica in 2010 and the Master Degree in Computer and Communication Networks Engineering in 2012 both from Politecnico di Torino. Claudio also holds the 2016 SmartICT Certificate on standardization for business innovation from the joint program of University of Luxembourg and ILNAS, the National Standardization Agency. Claudio has been awarded with the Spanish Juan de la Cierva grant and the Best Paper Awards in IEEE Cloudnet 2016 and in ACM WiNTECH 2018. He is member of IEEE and ACM, served as Publication and Web Chair at IEEE CloudNet 2014, Publicity Chair in ACM/IEEE ANCS 2018, Workshop Co-Chair of MoCS 2019

and TCP Co-Chair of IEEE CAMAD 2019. His primary research interests include multi-access edge computing, ultra-reliable and low latency communications and mobile crowdsensing,.

Alejandro Blanco Pizarro obtained his B.Sc. in Telecommunication Technologies Engineering from Carlos III University of Madrid in October of 2015. In his last year, Alejandro was working in Everis as a Junior Consultant. During the next two years, he continued his studies joining the Double Master's Degree in Telecommunications Engineering and Multimedia and Communications. His final project of M.Sc was focused on developing an scheduling algorithm in a Cloud RAN, where the computational resources are limited. From September of 2017, Alejandro is working as PhD at IMDEA Networks. His current research is focusing on analyzing and measuring cellular network to enhancement its performance.

Pablo Jimenéz Mateo holds two Bachelor degrees, one in Computational Mathematics and another in Computer Engineering, both from Universitat Jaume I at Castellón de la Plana (Spain). He also holds two Masters, one in Intelligent Systems from the same university and a MsC in Telecommunications engineering from Universidad Carlos III at Madrid (Spain) where he is currently a PhD candidate on Telecommunications Engineering. His research focus on mmWave networks, more specifically in transport protocols and medium access. Prior to his incorporation to IMDEA Networks his professional experience has focused on several undergraduate internships: in "Generation of georeferenced alerts based on the study of the interaction of users in social networks" (Big Data), "Design and development of a self-organized system for emergency management traffic accesses on Castellón de la Plana" (Intelligent systems) and "Development of an agent based distributed system for the analysis of real-time traffic" (Intelligent systems).

Carlos Andrés Ramiro is currently a MS student in the Universidad Politecnica de Madrid. He performed his Bachelor thesis as an intern at IMDEA Networks Institute.

Norbert Ludant is a PhD student in the Information Assurance program at Northeastern University's College of Computer and Information Science, advised by Professor Guevara Noubir. He received a BS degree in Communication Systems Engineering in 2015, and MS degrees in Telecommunications Engineering and Multimedia and Communications in 2017, from the University Carlos III de Madrid. During these years he did internships both in industry, in Alcatel-Lucent, and in academia, at the 5G Innovation Centre, University of Surrey, where he was involved in large-scale antenna systems research under an European Erasmus+ grant. He joined IMDEA Networks Institute in 2017, where he has worked on traffic profiling, network optimization, and anticipatory networking, all evaluated with real traffic data. He also joined the Signal Processing Group at University Carlos III de Madrid in 2018, where he worked in novel approaches for channel coding. His research interests are in the area of broadband wireless communications, signal processing, security, privacy, and new emerging wireless technologies, among others.

Joerg Widmer is Research Professor as well as Research Director of IMDEA Networks in Madrid, Spain. His research focuses on wireless networks, ranging from extremely high frequency millimeter-wave communication and MAC layer design to mobile network architectures. From 2005 to 2010, he was manager of the Ubiquitous Networking Research Group at DOCOMO Euro-Labs in Munich, Germany, leading several projects in the area of mobile and cellular networks. Before, he worked as post-doctoral researcher at EPFL, Switzerland on ultra-wide band communication and network coding. He was a visiting researcher at the International Computer Science Institute in Berkeley, USA, University College London, UK, and TU Darmstadt, Germany. Joerg Widmer authored more than 150 conference and journal papers and three IETF RFCs, and holds 13 patents. He serves or served on the editorial board of IEEE Transactions on Mobile Computing, IEEE Transactions on Communications, Elsevier Computer Networks and the program committees of several major conferences. He was awarded an ERC consolidator grant, the Friedrich Wilhelm Bessel Research Award of the Alexander von Humboldt Foundation, a Spanish Ramon y Cajal grant, as well as best paper awards at IEEE ICC, IEEE PIMRC, IEEE WoWMoM, ICST WICON, IEEE MediaWiN, NGC, and the IEEE Communications Society Best Tutorial Paper Award. He is senior member of IEEE and ACM.