openLEON: An End-to-End Emulator from the Edge Data Center to the Mobile Users

Carlos Andrés Ramiro
IMDEA Networks Institute
carlos.andres@imdea.org

Claudio Fiandrino
IMDEA Networks Institute
claudio.fiandrino@imdea.org

Alejandro Blanco Pizarro
IMDEA Networks Institute
Universidad Carlos III de Madrid
alejandro.blanco@imdea.org

Pablo Jiménez Mateo
IMDEA Networks Institute
Universidad Carlos III de Madrid
pablo.jimenezmateo@imdea.org

Norbert Ludant
IMDEA Networks Institute
norbert.ludant@imdea.org

Joerg Widmer
IMDEA Networks Institute
joerg.widmer@imdea.org

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., Mininet and srsLTE, and makes it possible to evaluate and validate research ideas on all the components of an end-to-end mobile edge architecture.

CCS CONCEPTS

• **Networks → Network architectures**; *Data center networks*; Mobile networks;

ACM Reference Format:
Carlos Andrés Ramiro, Claudio Fiandrino, Alejandro Blanco Pizarro, Pablo Jiménez Mateo, Norbert Ludant, and Joerg Widmer. 2018. openLEON: An End-to-End Emulator from the Edge Data Center to the Mobile Users. In *12th International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization (WiNTECH '18), November 2, 2018, New Delhi, India.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3267204.3267210

ACM ISBN 978-1-4503-5930-6/18/11…$15.00
https://doi.org/10.1145/3267204.3267210

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 [4]. 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 [5]. 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 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) [11] and network function virtualization (NFV) [17]. 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 Mininet [14] to emulate a SDN-based data center network. 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., [2]), it onlys supports mobile network emulation and does not include the data center environment. With Mininet, 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. To validate openLEON, we implement and assess several use cases, such as the caching proof-of-concept of [2] and impact of Radio Link Control (RLC) buffer size [13]. 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 [21].

The rest of the paper is organized as follows. Section 2 illustrates the rationale for the choice of srsLTE and Mininet 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. 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 Mininet 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) [19] 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 [7]. 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 8 cores at 3.4 GHz and 16 GB RAM was used to run the eNodeB 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.

2.2 Emulating SDN Data Center Networks

To emulate data center networks, the most widely adopted solution is Mininet [14]. 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. 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

Table 1: Performance Analysis of srsLTE [6] and OAI [19]

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

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 [22] spans the computational effort over multiple physical machines, achieving precise emulation of large-scale topologies with thousands of nodes. Virtual-Time Mininet [24] 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.

Given that edge data centers are much smaller than conventional clouds, we use Mininet 2.3.0, an evolution of Mininet High Fidelity (HiFi - 2.1.0), for the prototype implementation of openLEON. Among its advantages is that it allows to perform live migration of a virtual machine [10].

3 THE OPENLEON PLATFORM

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

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.

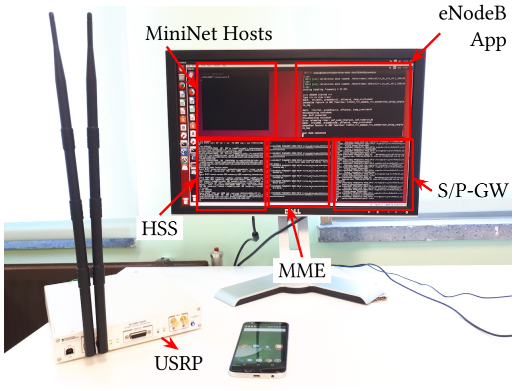


Figure 1: The openLEON testbed

3.2 The Methodology for Interconnection

In conventional EPC, specific interfaces interconnect the components, namely S1: S/PGw ↔ eNB, S11: SGw ↔ MME, and S6: HSS ↔ 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 scripts `run_spgw`, `run_hss` and `run_mme` from 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. Note that no technical restrictions forbid to execute the three functionalities on different hosts (with the caveat of granting the appropriate permissions to access the HSS database, developed with MySQL).

The overall architecture of openLEON is shown in Fig. 2. The eNB application is executed on the physical machine, which also hosts a virtual machine running Mininet and the EPC functionalities. We set a `10.0.0.0/12` network for Mininet and a `172.16.0.0/24` network to interconnect the physical machine with the USRP through PCI Express. For the development of openLEON, we use a 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.

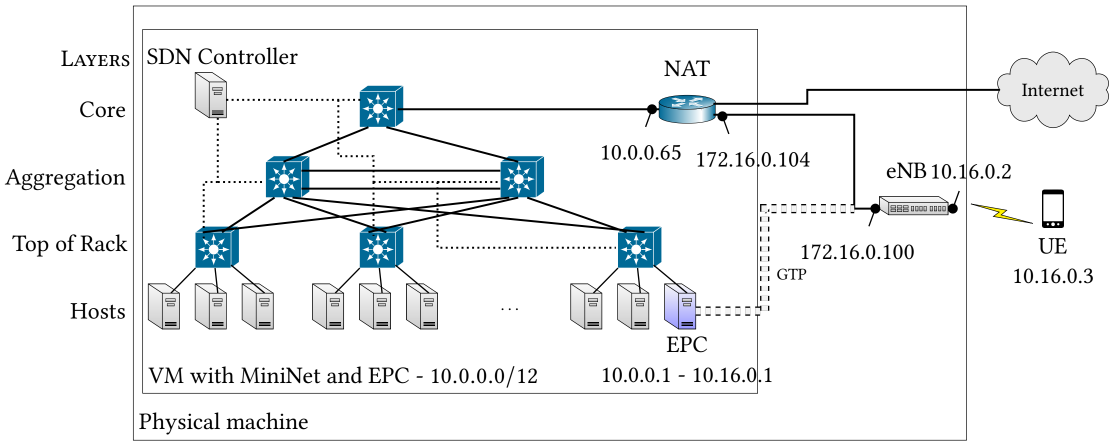


Figure 2: The architectural components - block diagram

One Mininet host performs NAT functions, bridging the virtual machine interfaces 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 `172.16.0.0/24` network, e.g., `172.16.0.104`.
- In the physical machine, two entries are included to route traffic from the data center and from the mobile users through the VM IP;
- The traffic generated by the UE and destined to the edge data center is routed via the eNB;
- Traffic generated by Mininet 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) as the end devices. BladeRFs are inexpensive SDRs enabling the setup of a LTE compliant pico cell or UE, offering up to 30 MHz I/Q sampling rate which is sufficient to decode the 20 MHz LTE channel. For the case of BladeRF 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 if necessary re-establish connectivity.

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 Mininet 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 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

4.1 Testbed Configuration

For the experiments, we configured openLEON as follows. The srseNB application from srsLTE version 18.3.1 runs over a desktop computer equipped with 8 cores running at 3.4 GHz, 16 GB RAM and Linux Ubuntu 14.04.5. The LTE station consists of a NI USRP-2942R connected with PCI Express with the desktop computer. As UEs, we use two laptops, one with 4 cores at 2.6 GHz and 16 GB of RAM, and another with 4 cores at 2.1 GHz and 8 GB of RAM. Those are connected to Nuand BladeRF x40 SDRs. For the virtual machine running the EPC application and Mininet (on Ubuntu with kernel version 4.13.0) we reserved 1 core, 6 GB of RAM, and 50 GiB of space. 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. 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 [2]. 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. 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, highlight that this is not always true.

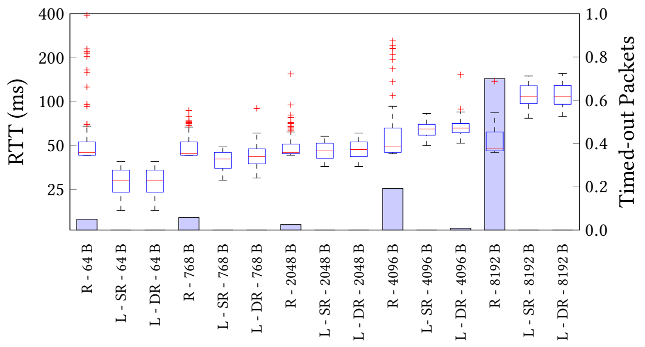


Figure 3: Measured RTT from UE to a remote data center (R) and local host (L) in same rack (SR) and different rack (DR)

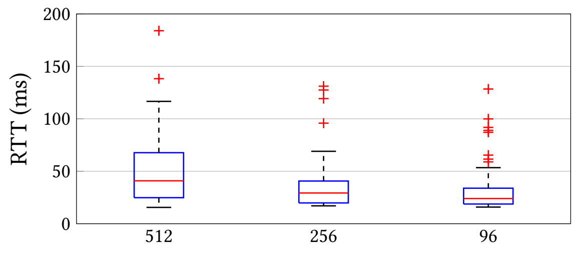
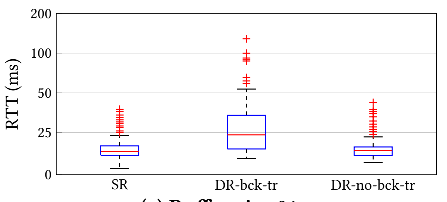


Figure 4: Impact of various RLC buffer sizes on RTT (in number of RLC SDU)

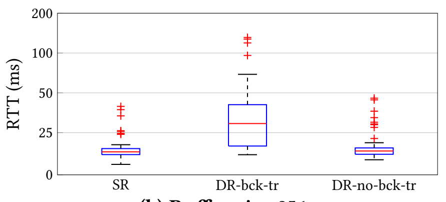
RLC Buffer Size. End-to-end latency is one of the main goals of 5G networks to enable Ultra-reliable and low latency communications (URLLC). For URLLC, 3GPP has defined a target user plane latency of 0.5 ms for both uplink and downlink transmissions, and a reliability requirement of 5-nines ($1-10^{-5}$) 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 [13]. 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 a UDP flow. The choice for UDP provides control over the target rate of traffic injection (target bandwidth 4 Mbps for 60 s). Unlike

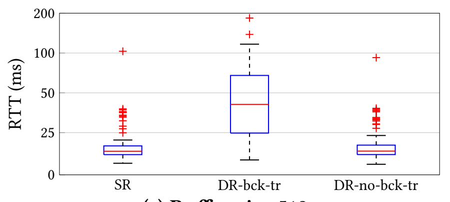
[1]3GPP, "Tech. Report #: 38.913, Release 14," 2017.



(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

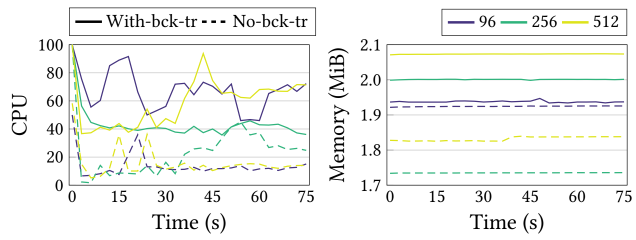


Figure 6: CPU and memory utilization

previous research [13], 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 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).

Fig. 6 shows the CPU and memory utilization measured with the Glances monitoring tool[2]. We assess the case with the sender residing in a different rack of the host performing EPC functionalities. The presence of background traffic increases both CPU and memory utilization. For a maximum RLC buffer size of 96 SDUs, the curves with and without background traffic are close. The reason is that the network topology has been created during the experiment without background traffic.

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

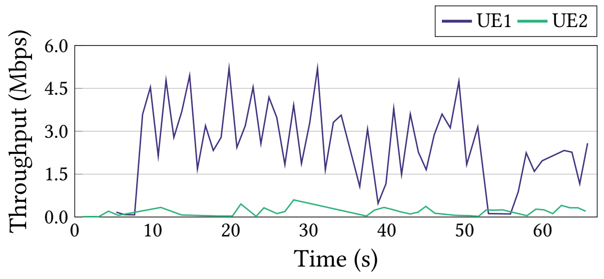
Fig. 7(a) shows the achieved throughput of both UEs. As expected, UE1 significantly outperforms UE2 with an average throughput of 2.6 Mbps and 187.5 Kbps, respectively. Fig. 7(b) shows performance of the two users. The graph is a throughput-RTT plot, where for each scenario we take the average results 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) [23], flows can exploit more than one path simultaneously, by transmitting over multiple interfaces or to different IPs of a host. MP-TCP finds applicability in both data centers and wireless networks.

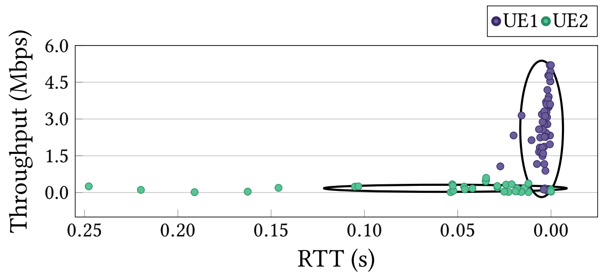
MP-TCP kernel version 0.93[3] does not provide a module to support the GTP0 interface, hence we downloaded the sources of the kernel version 4.13 which has an available MP-TCP patch and includes the GTP module. Then we patched

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

[3]Available at: https://www.multipath-tcp.org/



(a) Throughput analysis



(b) Joint throughput and RTT analysis

Figure 7: Throughput and RTT performance of mobile users

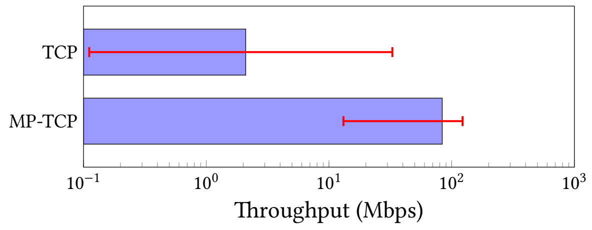


Figure 8: Performance of MP-TCP and TCP

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 target 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. 8 illustrates the significant advantage that MP-TCP provides with respect to TCP in terms of achieved throughput.

5 OPEN RESEARCH CHALLENGES AND APPLICATION SCENARIOS

This section provides a brief overview of possible uses of openLEON.

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 [18] 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 researchers the capability to evaluate new protocols that respond to actual congestion rather than packet loss (e.g., TCP BBR [1] available from kernel version 4.9) or cross-layer approaches coupling congestion control with lower layers.

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.

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 [8]. 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[4] methods for image processing and feature extraction.

[4]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 [3] emulates the wireless channel by exploiting Linux TC 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[5] and the Lena module of ns-3 for the cellular network (OpenNet[6]). 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 [15].

Closest to our work are [9, 12]. 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. Unlike openLEON, both solutions lack the fine modeling of the properties of data center networks. EmuFog [16] 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.

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 Mininet, 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 (RNIS) 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 European Research Council grant ERC CoG 617721, the Ramon y Cajal grant from the Spanish Ministry of Economy and Competitiveness RYC-2012-10788, by the European Union H2020-ICT grant 644399 (MONROE), and the Madrid Regional Government through the TIGRE5-CM program (S2013/ICE-2919).

[5]Available at: https://github.com/mininet/mininet/wiki/Link-modeling-using-ns-3

[6]Available at: https://github.com/dlinknctu/OpenNet

REFERENCES

[1] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2017. BBR: Congestion-based Congestion Control. *Commun. ACM* 60, 2 (2017), 58–66.

[2] Chia-Yu Chang, Konstantinos Alexandris, Navid Nikaein, Kostas Katsalis, and Thrasyvoulos Spyropoulos. 2016. MEC Architectural Implications for LTE/LTE-A Networks. In *Proc. of ACM MobiArch*. 13–18.

[3] Ramon dos Reis Fontes and Christian Esteve Rothenberg. 2016. Mininet-WiFi: A Platform for Hybrid Physical-Virtual Software-Defined Wireless Networking Research. In *Proc. of ACM SIGCOMM*. 607–608.

[4] Fabio Giust, Xavier Costa-Perez, and Alex Reznik. 2017. Multi-Access Edge Computing: An Overview of ETSI MEC ISG. *IEEE 5G Tech Focus* 1, 4 (Dec 2017).

[5] Fabio Giust, Gianluca Verin, Kiril Antevski, Joey Chou, Yonggang Fang, Walter Featherstone, and et al. 2018. MEC Deployments in 4G and Evolution Towards 5G. ETSI White Paper.

[6] Ismael Gomez-Miguelez, Andres Garcia-Saavedra, Paul D. Sutton, Pablo Serrano, Cristina Cano, and Doug J. Leith. 2016. srsLTE: An Open-source Platform for LTE Evolution and Experimentation. In *Proc. of ACM WiNTECH*. 25–32.

[7] F. Gringoli, P. Patras, C. Donato, P. Serrano, and Y. Grunenberger. 2018. Performance Assessment of Open Software Platforms for 5G Prototyping. *To appear in IEEE Wireless Communications* (2018). Available at: http://homepages.inf.ed.ac.uk/ppatras/pub/wcm18.pdf.

[8] S. Guo, J. Liu, Y. Yang, B. Xiao, and Z. Li. 2018. Energy-Efficient Dynamic Computation Offloading and Cooperative Task Scheduling in Mobile Cloud Computing. *IEEE Trans. on Mobile Computing* (Apr 2018), 1–14.

[9] Anta Huang and Navid Nikaein. 2017. Demo: LL-MEC A SDN-based MEC Platform. In *Proc. of ACM MobiCom*. 483–485.

[10] Eric Keller, Soudeh Ghorbani, Matt Caesar, and Jennifer Rexford. 2012. Live Migration of an Entire Network (and Its Hosts). In *Proc. of ACM HotNets-XI*. 109–114.

[11] Keith Kirkpatrick. 2013. Software-defined Networking. *Commun. ACM* 56, 9 (Sep 2013), 16–19.

[12] K.Ramantas, E.Kartsakli, M.Irazabal, A. Antonopoulos, and C. Verikoukis. 2017. 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, 791–796.

[13] R. Kumar, A. Francini, S. Panwar, and S. Sharma. 2018. Dynamic Control of RLC Buffer Size for Latency Minimization in Mobile RAN. In *IEEE WCNC*. 1–6.

[14] Bob Lantz, Brandon Heller, and Nick McKeown. 2010. A Network in a Laptop: Rapid Prototyping for Software-defined Networks. In *Proc. of ACM Hotnets-IX*. Article 19, 6 pages.

[15] Vincenzo Mancuso, Christian Vitale, Rohit Gupta, Karamvir Rathi, and Arianna Morelli. 2014. A prototyping methodology for SDN-controlled LTE using SDR.

[16] Ruben Mayer, Leon Graser, Harshit Gupta, Enrique Saurez, and Umakishore Ramachandran. 2017. EmuFog: Extensible and Scalable Emulation of Large-Scale Fog Computing Infrastructures. *CoRR* abs/1709.07563 (Sep 2017).

[17] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba. 2016. Network Function Virtualization: State-of-the-Art and Research Challenges. *IEEE Communications Surveys Tutorials* 18, 1 (Third Quarter 2016), 236–262.

[18] Binh Nguyen, Arijit Banerjee, Vijay Gopalakrishnan, Sneha Kasera, Seungjoon Lee, Aman Shaikh, and Jacobus Van der Merwe. 2014. Towards Understanding TCP Performance on LTE/EPC Mobile Networks. In *Proc. of ACM All Things Cellular*. 41–46.

[19] Navid Nikaein, Mahesh K. Marina, Saravana Manickam, Alex Dawson, Raymond Knopp, and Christian Bonnet. 2014. OpenAirInterface: A Flexible Platform for 5G Research. *ACM SIGCOMM Comput. Commun. Rev.* 44, 5 (Oct 2014), 33–38.

[20] P. Ruiu, C. Fiandrino, P. Giaccone, A. Bianco, D. Kliazovich, and P. Bouvry. 2017. On the Energy-Proportionality of Data Center Networks. *IEEE Trans. on Sustainable Computing* 2, 2 (Apr 2017), 197–210.

[21] J. Son, A. V. Dastjerdi, R. N. Calheiros, X. Ji, Y. Yoon, and R. Buyya. 2015. CloudSimSDN: Modeling and Simulation of Software-Defined Cloud Data Centers. In *IEEE/ACM CCGrid*. 475–484.

[22] P. Wette, M. Dräxler, and A. Schwabe. 2014. MaxiNet: Distributed emulation of software-defined networks. In *IFIP Networking*. 1–9.

[23] Damon Wischik, Costin Raiciu, Adam Greenhalgh, and Mark Handley. 2011. Design, Implementation and Evaluation of Congestion Control for Multipath TCP. In *Proc. of USENIX NSDI*. 99–112.

[24] Jiaqi Yan and Dong Jin. 2015. VT-Mininet: Virtual-time-enabled Mininet for Scalable and Accurate Software-Define Network Emulation. In *Proc. of ACM SOSR*. 1–7.