This paper appeared at CLEEN 2020, workshop of ICC 2020, Dublin, Ireland, June 7 2020

A co-simulation framework to evaluate edge deployment options and performance

Antonio Virdis[(1)], Giovanni Nardini[(1)], Giovanni Stea[(1)], Yuankun Shi[(2)], Zhaojuan Bian [(2)]

[(1)] Dipartimento di Ingegneria dell'Informazione
University of Pisa, Italy
{name.surname}@unipi.it

[(2)] Intel Architecture, Graphics and Software Group, Intel Corp.
Shanghai, China
yuankun.shi@intel.com bianny.bian@intel.com

***Abstract*—Evaluating edge deployments from a user perspective requires modeling, in a unified framework, the communication part, i.e., the cellular access network, and the computation part, i.e., the Mobile-edge server. This paper presents a framework that enables this, by joining the SimuLTE 4G network simulator and the Intel CoFluent edge-computing simulator. We show how the two tools are integrated and discuss some preliminary validation of the framework.**

Keywords—Multi-access Edge Computing, SimuLTE, CoFluent, simulation, co-simulation

I. INTRODUCTION

Multi-access Edge Computing (MEC) apportions computational capabilities typical of the Cloud near the edge of mobile (4G and beyond) networks. Examples of MEC-based use cases include computational offloading for Internet-of-Things applications, smart transportation and dynamic content optimization [1]. Mobile Edge (ME) servers or MEC hosts, deployed close to the eNBs of a cellular network, interact with the Radio Access Network (RAN) and obtain information on its status and its users. Such information can be exploited to offer context-aware services, in an environment where the latency between the mobile user and the ME is smaller than with a cloud-based service, due to proximity.

A relevant question for operators, manufacturers and service deployers is *where exactly* the "edge" should be located, with respect to the architecture of the mobile network. Possible answers range from *co-located with eNBs*, at one extreme, to *co-located with the cellular network gateway*, at the other. It is clear that the answer to the above question – which may not be unique, and may well be service-dependent – impacts the scalability, the cost, the latency of the service, as well as the amount of bandwidth consumed by MEC-related traffic (e.g., a "far" edge might require a massive transfer of context data to a centralized point, whereas a "near" edge would not). On the other hand, pooling computational resources is more cost effective, and deploying them too near to the user would be inefficient.

From a user perspective, deployment options need be evaluated based on user-perceived metrics, such as round-trip latency. The latter includes the transit in the uplink and downlink of the cellular network, as well as the computation time spent in the MEC host. These times are influenced by different factors: RAN congestion and or poor channel quality influences the first, but not the second; on the other hand, congestion on the computing resources on server influences the second and not the first.

Evaluating the latency of MEC services is not an easy task, because the scientific community lacks tools that incorporate *both* models of computation and communication at the same time. Although several works do exist in the literature (e.g., [2][3]) they rely on numerical evaluations of the system performance. Discrete-event network simulators, such as SimuLTE [4] or Lena [5], incorporate accurate models of the mobile network. However, they do not model MEC computations (SimuLTE incorporates some MEC functionalities [6], but does not have a model of computation, resource contention at the ME, etc.). On the other hand, there are cloud computing simulator – see, e.g., [7][8] – where the computation part is modeled, but the communication part is absent. Therefore, short of building in-house prototypes (which are costly, at the very least), there is currently little that one can do to make a sound end-to-end evaluation of MEC services.

The aim of this paper is to present a framework to assess edge-computing deployments and evaluate their performance. The evaluation is performed via a co-simulation approach involving two tools: SimuLTE [4] and CoFluent. [9] The former is a system-level simulator that models the data plane of the LTE-A RAN. The latter is an Intel product that simulates the EPC and the edge-computing infrastructure. The two simulators are made interoperable to allow the evaluation of end-to-end MEC-based services.

The rest of this paper is organized as follows: Section II describes the background technologies, whereas Section III presents the simulation tools that we used and the way we joined them into an end-to-end tool. Section IV describes the simulation scenarios and the results that we obtained. Section V concludes the paper.

II. BACKGROUND

This section introduces some background on MEC and LTE-advanced.

A. Multi-access Edge Computing

MEC is being standardized by the ETSI [10], following to the architecture shown in Figure 1. With MEC, MEC apps run in a virtualized environment, hence computational resources can be allocated on demand to users requesting a particular service or task. This allows the supervisor of the MEC architecture (e.g., the network operator) to optimize resource utilization, even dynamically, by migrating user applications from a MEC host to another, based on several criteria, including computation and communication requirements.

At the top of the hierarchy, called the MEC System Level, a global vision of the state of all the MEC Hosts in the system exists. More in detail, the MEC system Level receives MEC Application Instantiation requests from user applications, checks their requirements (e.g., maximum communication la-

tency, computational resources, availability of MEC services), and selects – and instructs accordingly – the MEC Host that will host the corresponding ME Application instance. Within the MEC Host, the MEC Platform provides services defined in [11] that can be exploited by MEC Apps, such as: the Smart Relocation Service handles migration of MEC apps to other MEC Hosts; the Radio Network Information Service (RNIS), which is used to gather information from the network elements (e.g. number of users connected to a specific radio base station); the Bandwidth Manager, which defines the priority of data traffic destined to MEC Apps within the MEC Host; the Location Service provides information on the users' position. The Virtualization Infrastructure is the core of the MEC Host, and it runs MEC Apps as instances of virtual machines, allowing them to communicate both within the MEC Host (e.g., with the services of the MEC Platform) and outside it (e.g., with users' local applications).

B. *LTE-Advanced*

MEC interacts with the underlying access networks. Among these, a key role will be played by cellular networks, i.e., LTE-Advanced (LTE-A) and its 5G evolution. The architecture of LTE-A is composed of a Radio Access Network (RAN) part and an Evolved Packet Core (EPC) part. The main components of the LTE RAN are the radio base stations, called eNodeBs (eNBs), and the cellular users, called User Equipments (UEs), as shown in Figure 1. The eNB handles cell-wise operations, such as radio resource allocation to UEs. In particular, every Transmission Time Interval (TTI - 1ms in LTE, possibly shorter in 5G new radio), the eNB schedules a subset of Resource Blocks (RBs) to allow UEs to send/receive data to/from the eNB uplink/downlink (UL/DL) directions. The EPC is a flat IP-based network, where the Packet Data Network Gateway (PGW) represents the entry/exit point of the LTE-A network.

In such context, MEC Hosts can be deployed anywhere in the EPC, possibly close to the eNBs as depicted in Figure 1 so that a single MEC Host is responsible to handle the UEs connected to a small number of adjacent eNBs.

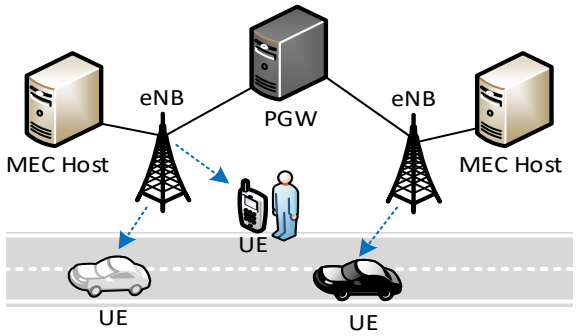

Figure 1 – Example of a MEC-enabled LTE-A system.

III. End-to-end Simulations with 4G and MEC

This section describes the tools that we used to realize the end-to-end co-simulation framework described in the Introduction.

A. *SimuLTE*

A detailed description of SimuLTE is provided in [4]. In this section we highlight the information most relevant to our envisioned framework. SimuLTE is based on the OMNeT++ framework [13], and it simulates the data plane of the LTE/LTE-A RAN and EPC.

UEs and eNBs are implemented as compound modules. These can be connected with each other and with other nodes (e.g. routers, applications, etc.) in order to compose networks. SimuLTE allows simulation of LTE/LTE-A in Frequency Division Duplexing (FDD) mode, with heterogeneous eNBs (macro, micro, pico etc.), using omnidirectional and anisotropic antennas. Its main component nodes are shown in Figure 3. The Binder module is visible by other nodes in the system and stores information about them, such as identifying the interfering eNBs in order to compute the inter-cell interference perceived by a UE in its serving cell. Using the Binder as an information repository allows one to implement control functions as queries to the latter, without the need of modeling or implementing control-plane protocols. The latter can be added to SimuLTE, of course, should one wish to evaluate them specifically.

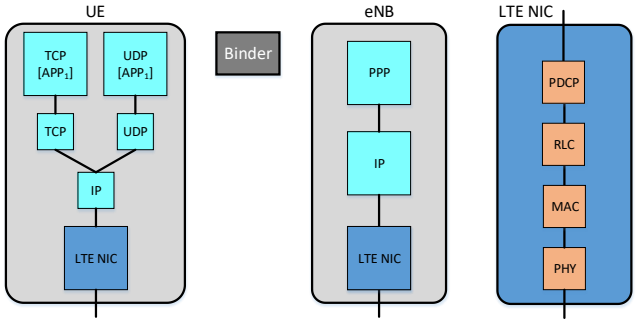

Figure 3 SimuLTE UE and eNB modules and the LTE stack

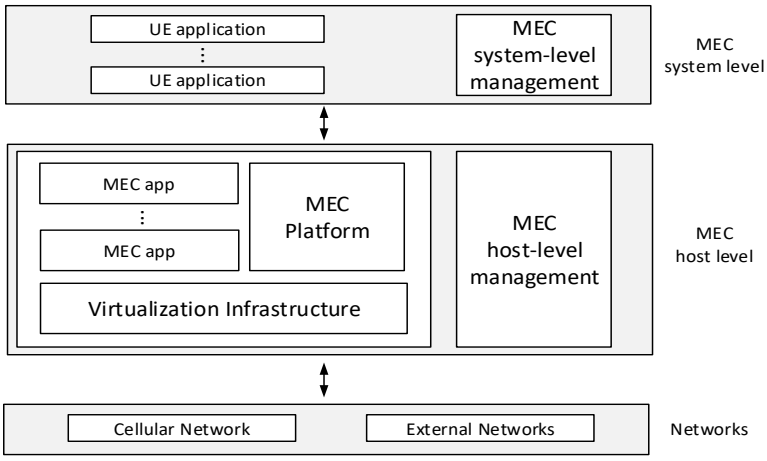

Figure 2 – Overview of the Multi-access Edge Computing Framework

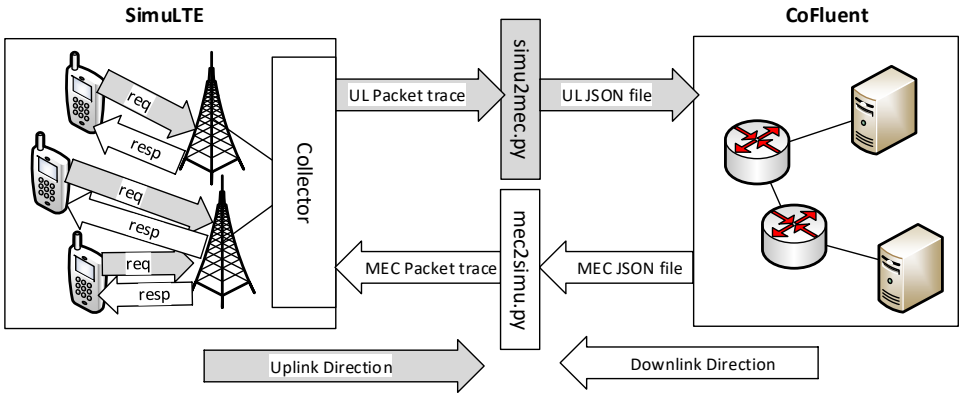

Figure 4 - Implementation of the Uplink leg of the co-simulation

The Network Interface Card (NIC) module, whose structure is shown in Figure 3, implements the LTE stack.

In [4], a detailed description of each component is presented. Detailed information on the input/output parameters of SimuLTE can be found in [12]. SimuLTE is a *system-level* simulator, i.e. it aims at evaluating the performance of a large number of interconnected nodes, focusing more on accurately modeling their interactions and on capturing the overall effects system-wise. This is in contrast with *link-level* simulators, e.g., [14][15], whose focus is on the accurate modeling of the physical layer, which normally do not allow a user to setup end-to-end scenarios with applications and upper layers involved.

As far as the MAC layer is concerned, at the beginning of each TTI the eNB prepares a schedule list in both the UL and the DL directions. In the latter, the available resources are shared among the active connections according to a given scheduling policy. SimuLTE includes the following schedulers: *Maximum Carrier- over-Interference (Max C/I), Proportional Fair (PF), Deficit Round Robin (DRR).* Scheduling processes in the two directions are independent, and they may follow different policies. In SimuLTE, eNBs can also be configured as *external cells*. An external cell generates interference on the UL/DL spectrum according to a user-specified pattern (e.g., a percentage of occupied RBs), without the need of simulating its users, its scheduling process, the LTE protocol stack or actual traffic. External cells are useful in multi-cell scenarios. A user can instantiate a relatively small number of UEs in one or few *target cells*, and simulate accurately what happens in these, while at the same time keeping a correct accounting of inter-cell interference by placing an outer tier of external cells. This considerably reduces the duration of simulations, with no appreciable degradation of the accuracy of its results.

Note that SimuLTE does include models of MEC entities, which are described in [19]. Therefore, a SimuLTE user can generate scenarios involving MEC entities and 4G traversal. However, only the *functional* aspects of MEC entities are modeled therein, whereas their *performance* implications are not. In other words, the computations occurring on a simulated MEC Host in SimuLTE occupy a null *simulated* time. Therefore, modeling the response time at a MEC Host - due to computational resource contention, virtualization, disk swaps, etc., - is not possible, unless a suitable model of all the above aspects is added to SimuLTE. The effort of modeling all the above is considerable, and therefore we prefer to capitalize on the expertise which has already been embodied in CoFluent, which we describe in the next subsection.

B. CoFluent

In this section, we focus on how we use Intel CoFluent Technology in this E2E co-simulation framework. Intel® CoFluent™ studio is a modeling and simulation tool for optimizing, analyzing and predicting the performance of complex systems. A detailed description of CoFluent is available in [9]. CoFluent speeds up the deployment of systems by modeling and simulating both hardware and software interactions. Within this work, we use CoFluent to simulate MEC servers at different deployment locations. The MEC servers can be deployed at different levels in the CoFluent behavioral model, which can be configured through JSON files. Typical examples for MEC deployment options include: universal Customer Premises Equipment (uCPE), RAN-edge, Smart Central Office (CO) and Edge Data Centre (DC).

CoFluent can be used to evaluate the latency and throughput of data flows coming to and leaving from the MEC system, as it simulates computing and communication cost with high accuracy and high speed.

C. Combining SimuLTE and CoFluent in a single co-simulation framework

We now describe the architecture of the co-simulation framework for end-to-end simulation of MEC-enabled cellular networks. The two simulation tools are made interoperable to create a co-simulation environment, i.e. a simulation system wherein SimuLTE and CoFluent are treated as independent simulators, having their own models and configuration files. They are also executed as independent pieces of software, and coordination is achieved through *information sharing*.

Each tool will aim at assessing the impact of the two involved network portions on the E2E service. More in detail, SimuLTE will consider the impact of the RAN, including channel quality, mobility, radio resource scheduling, etc. CoFluent instead, will assess the impact of the EPC and MEC environment, including core-network delay, processing delay at the MEC servers, etc. The logical entities involved and the information exchanged in the co-simulation process are shown in Figure 5.

SimuLTE will be used to simulate a RAN deployment composed of a configurable number of UEs and eNBs. Each UE will execute an application client, modeling the behavior of

the selected use case, and will request services to the MEC infrastructure. Application clients can be implemented as TCP- or UDP-based OMNeT++ applications, defining the logic for the generation of requests and handling of responses. The requests will travel as data packets through the LTE network, thus using UL resources managed by the UE's serving eNB. MEC responses, in turn, will be generated by the MEC server and sent back to the UE through the eNB, using DL resources.

CoFluent will instead be used to simulate the EPC and the MEC environment. Service requests coming from SimuLTE will travel through the EPC and will be processed by the MEC host. The latter will implement a model of the MEC app for the specific use-case, and will generate service responses. Note that SimuLTE includes also a model of the EPC, which is not instantiated to avoid duplication.

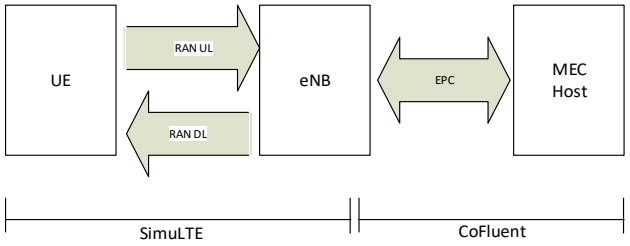

Figure 5 - High-level view of the entities involved in the E2E communication

The two tools interact via file exchange as follows:

Step 1: SimuLTE is run with the required UE drop and mobility model. The simulated UEs generate traffic requests, which are then transmitted as application-level messages via the UL. Traffic requests incur delays as they travel through the RAN, due to several factors, such as channel quality, resource occupancy, etc. When UE requests reach the EPC entry point, they leave SimuLTE. The size and timings of the traffic requests are written to *Logfile#1* file. Each line in the above file will include the simulated time at which the request was *generated at the UE*, as well as the simulated time at which it reached the EPC entry point. Both times are expressed in SimuLTE's timeframe.

Step 2: CoFluent is run, feeding it with *Logfile#1* including the UE requests. The models of the MEC elements will process the requests, and the MEC server will generate the data payload according to the traffic model in use. Payload will be sent back *towards* the access network. Packet size and timings of the data payload are written to *Logfile#2* file.

Step 3: SimuLTE is run again in the same scenario used for Step 1, i.e. with the same initial positions and mobility of UEs. *Logfile#2* is used as an input to generate DL traffic from eNBs to be sent towards the intended UEs. Traffic requests are delayed as they travel through the DL of the RAN, until they reach the application layer at the intended UE.

The above architecture has been implemented by extending both SimuLTE and CoFluent, and setting up tools for information exchange among the two. With reference to Figure 4, in the UL direction the following elements have been implemented:

1. A C++ application that generates periodic service requests, which is executed on every UE;
2. A C++ application that collects all the requests from the eNBs, and prints the UL packet trace to *Logfile#1*;
3. a python program, *simu2mec.py*, which parses the UL packet trace and generates a structured JSON file to be given as input to CoFluent.

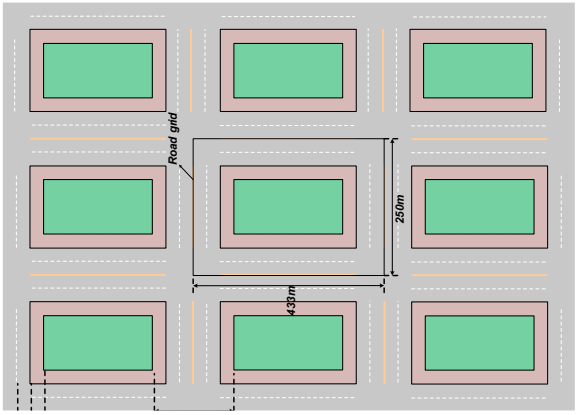

Figure 6 - The eight roads in the urban-grid scenarios

In the DL direction the following elements have been implemented:

1. A program written in python, referred to as mec2simu.py, which parses the DL packet trace produced by CoFluent, and generates a structured JSON file, containing the ID of the packets, the ID of the generating UE, and the timestamp.
2. An application written in C++ that reads the JSON file produced by the MEC environment, generates service responses accordingly and sends them towards the interested UEs, going through their serving eNBs
3. An application written in C++, running on each UE, which receives service responses and computes statistics according to the considered use case;

IV. Performance Evaluation

We have evaluated the above framework using an infotainment data traffic model, on a deployment/mobility model based on an urban grid scenario.

Our infotainment application is a bidirectional communication between a UE and a MEC server. The UE periodically generates and transmits to the MEC server a V*ideo Request*, which translates to a constant-size application message. The MEC server responds with a *Video Stream*, composed of a given number of frames. Frames have constant size and are transmitted at a rate of 25 per seconds. A selection of frame sizes for various video bitrates is given in Table 1. The number of frames within a video stream is proportional to the duration of the latter.

Table 1 - Exemplary choice of frame sizes for various video bitrates.

Video Format	Bitrate	Frame Size
240p	0.64 Mbps	3200 Bytes
360p	0.96 Mbps	4800 Bytes
480p	1.28 Mbps	6400 Bytes
720p	2.56 Mbps	12800 Bytes

We measure the following KPIs:

- *Packet Delay (PD)*: the time between the transmission of the frame from the MEC node and the reception at the UE. This gives a measure of the freshness of video

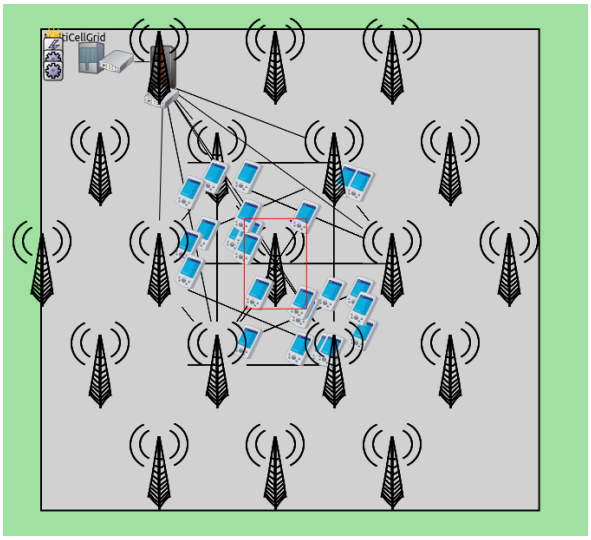

Figure 7 - Representation of the urban-grid deployment taken from the SimuLTE environment.

Table 2 - Simulation parameters

Parameter Name	Value
SimuLTE	
#eNBs	57
#UEs	24 (3 per road)
Fading + shadowing	Enabled
Bandwidth	20 MHz (100 PRBs)
eNB Tx Power	46 dBm
UE Tx Power	23 dBm
External Cells Tx Power	41 dBm
External Cells load level	100%
UE speed	14 m/s
CoFluent	
Mp3	4.77 us/m (5000m)
Mp1	5 us
N3	5.4 us
N6	5.4 us
FrontHaul	4.77 us/m (500m)
F	2x4GHz
B	100 Hz/bit

frames, e.g. to evaluate the performance of a video stream.

- *Inter-frame time (IFT):* the time between the reception of two consecutive frames of the same video stream.

The first scenario that we take into account considers a grid of 8 roads in an urban scenario, as we show in Figure 6.
The grid is placed at the center of a 57-cell floorplan and is populated with 24 UEs (three per road) moving across the roads. In Figure 7, we give a graphical representation of the scenario, wherein each represented eNB is a trisectorial site.
The eNBs belonging to the innermost two tiers (21 eNBs) are simulated with a full protocol stack and they transmit/receive actual traffic. The 36 eNBs in the external tier, instead, are configured as *external cells* in SimuLTE to reduce the duration of the simulations. Table 2 reports the main simulation parameters for SimuLTE.
The MEC protocol and Workload processing latency are simulated in CoFluent model. They can be divided into 3 parts:

1) UL latency of MEC service,
2) DL latency of MEC service,
3) processing latency of MEC service.

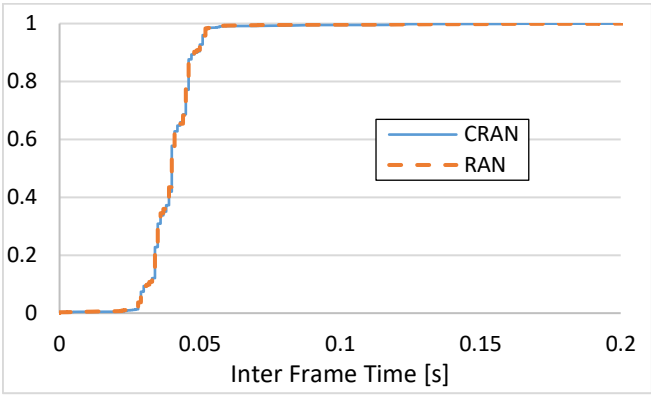

Figure 8 - Distribution of the Inter Frame Times for two MEC deployments.

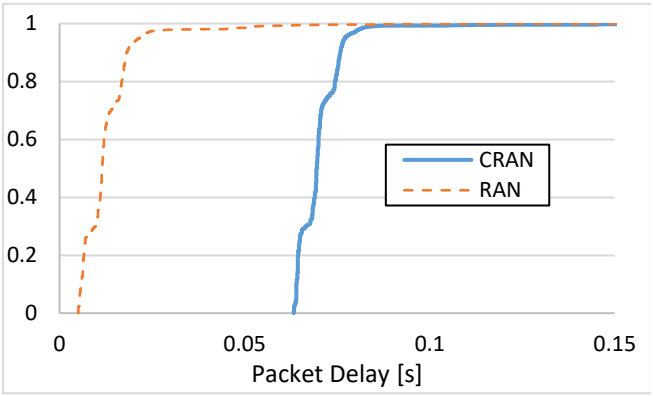

Figure 9 - Distribution of the packet delays for two MEC deployments

The simulation parameters of MEC service are reported in the second part of Table 2. Two deployment options are supported: RAN and CRAN. In the former, the MEC nodes are placed within the RAN, closer to the eNB. In the latter instead, MEC nodes are deployed farther from the RAN.
The processing latency of each response is calculated as

$$MEC_{lat} = \frac{N \cdot l \cdot B}{F}$$

N is the number of request number, l is the size of each response, B is the CPU frequency required for each bit l, F is the processing capacity of the CPU (in frequency). In Figure 8 we show the distribution of the interpacket times for the two considered MEC deployments. The two curves have a similar shape – as one would indeed expect – as the deployment distance has a negligible impact on the relative delay variation. The effect of using a CRAN deployment are instead visible in Figure 9, where we show the distributions of packet delays for the two scenarios.

V. Conclusions

This paper has presented a simulation framework to evaluate MEC services from a user perspective. The framework combines network simulation, using SimuLTE, and computing system simulation, using Intel CoFluent. The two simulators are run independently and are connected via file sharing. This allows one to factor in the service impairments (notably, the delay) due to both computation and network traversal.
Future work on the topic will include running the two simulators in parallel, using a shared simulated time reference. Moreover, we are currently finalizing the development of

Simu5G, the evolution of SimuLTE that incorporates NewRadio technology. As the latter inherits the same architecture of its predecessor, all the work described in this paper can be seamlessly reused with a 5G NewRadio access network.

ACKNOWLEDGMENTS

Work partially supported by the Italian Ministry of Education and Research (MIUR) in the framework of the CrossLab project (Departments of Excellence).

REFERENCES

[1] Y. Ch. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, “Mobile Edge Computing A key technology towards 5G-First edition”, 2015

[2] M. Emara, M. C. Filippou and D. Sabella, "MEC-Assisted End-to-End Latency Evaluations for C-V2X Communications," *2018 European Conference on Networks and Communications (EuCNC)*, Ljubljana, Slovenia, 2018, pp. 1-9.

[3] E. Pencheva, I. Atanasov, D. Velkova and V. Trifonov, "Evaluation of Multi-access Edge Computing Deployment Scenarios," *2019 9th Annual Information Technology, Electromechanical Engineering and Microelectronics Conference (IEMECON)*, Jaipur, India, 2019, pp. 155-160.

[4] A. Virdis, G. Stea, G. Nardini, "Simulating LTE/LTE-Advanced Networks with SimuLTE", DOI 10.1007/978-3-319-26470-7_5, in: M.S. Obaidat, J. Kacprzyk, T. Ören, J. Filipe, (eds.) "Simulation and Modeling Methodologies, Technologies and Applications", Volume 402 of the series Advances in Intelligent Systems and Computing, pp. 83-105, Springer, ISBN 978-3-319-26469-1, 15 January 2016.

[5] N. Baldo, M. Miozzo, M. Requena-Esteso, J. Nin-Guerrero, “An Open Source Product-Oriented LTE Network Simulator based on ns-3”, in Proc. of ACM MSWiM’11, Miami, US, Nov. 2011.

[6] G. Nardini, A. Virdis, G. Stea, A. Buono, "SimuLTE-MEC: extending SimuLTE for Multi-access Edge Computing", OMNeT++ summit 2018, Pisa, Italy, September 2018

[7] G. G. Castañé, A. Núñez and J. Carretero, "iCanCloud: A Brief Architecture Overview," 2012 IEEE 10th International Symposium on Parallel and Distributed Processing with Applications, Leganes, 2012, pp. 853-854.

[8] D. Kliazovich, P. Bouvry, S.U. Khan, “GreenCloud: A Packet-Level Simulator of Energy-Aware Cloud Computing Data Centers”. The Journal of Supercomputing, 62. 1-5, 2010.

[9] https://www.intel.com/content/www/us/en/cofluent/overview.html

[10] ETSI GS MEC 003, “Mobile Edge Computing (MEC); Framework and reference architecture”, 2016-03

[11] ETSI GS MEC 002, “Mobile Edge Computing (MEC); Technical requirements”, 2016-03

[12] http://simulte.com/guides.html

[13] A. Varga, R. Hornig, “An overview of the OMNeT++ simulation environment”. In: Proceedings of the SIMUTools ‘08, Marseille, France, March 2008

[14] C. Mehlfuerer, M. Wrulich, J.C. Ikuno, D. Bosanska M. Rupp: “Simulating the long term evolution physical layer”. In: Proceedings of the 17th EUSIPCO, Glasgow, UK (2009).

[15] C. Bouras, G. Diles, V. Kokkinos, K. Kontodimas, A. Papazois: “A Simulation Framework for Evaluating Interference Mitigation Techniques in Heterogeneous Cellurar Environments”. Wireless Personal Communications, Springer (2013)

[16] Markus Rupp, Stefan Schwarz, and Martin Taranetz. 2016. The Vienna Lte-Advanced Simulators: Up and Downlink, Link and System Level Simulation (1st ed.). Springer Publishing Company, Incorporated.

[17] 3rdGeneration Partnership Project (3GPP). Study on3Dchannelmodel forLTE, 3rdGeneration Partnership Project (3GPP), TR 36.873 (2014)

[18] Z. Shen, A. Papasakellariou, J. Montojo, D. Gerstenberger, F. Xu, “Overview of 3GPP LTE-Advanced Carrier Aggregation for 4G Wireless Communications”, IEEE Communications Magazine, February 2012

[19] G. Nardini, A. Virdis, G. Stea, A. Buono, "SimuLTE-MEC: extending SimuLTE for Multi-access Edge Computing", OMNeT++ summit 2018, Pisa, Italy, September 2018