

Received July 30, 2020, accepted September 23, 2020, date of publication October 2, 2020, date of current version October 14, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3028550

Simu5G—An OMNeT++ Library for End-to-End Performance Evaluation of 5G Networks

GIOVANNI NARDINI¹, DARIO SABELLA², GIOVANNI STEA¹, (Member, IEEE),
PURVI THAKKAR³, AND ANTONIO VIRDIS¹

¹Dipartimento di Ingegneria dell'Informazione, University of Pisa, 56122 Pisa, Italy

²Intel GmbH, 85579 Neubiberg, Germany

³Intel Corporation, Santa Clara, CA 95054, USA

Corresponding author: Giovanni Stea (giovanni.stea@unipi.it)

This work was supported in part by the Italian Ministry of Education and Research (MIUR) in the framework of the CrossLab Project (Departments of Excellence).

ABSTRACT In this article we introduce Simu5G, a new OMNeT++-based model library to simulate 5G networks. Simu5G allows users to simulate the data plane of 5G New Radio deployments, in an end-to-end perspective and including all protocol layers, making it a valuable tool for researchers and practitioners interested in the performance evaluation of 5G networks and services. We discuss the modelling of the protocol layers, network entities and functions, and validate our abstraction of the physical layer using 3GPP-based scenarios. Moreover, we show how Simu5G can be used to evaluate Multi-access Edge Computing (MEC) and Cellular Vehicle-to-everything (C-V2X) services offered through a 5G network.

INDEX TERMS Computer simulation, object-oriented modeling, computer networks, 5G mobile communication.

I. INTRODUCTION

The two main pillars of the next technology revolution in the field of mobile networks will be the deployment of 5G access and Multi-access Edge Computing (MEC). The former will provide ultra-reliable, high-bandwidth and low-latency connectivity, thus enabling new ICT services, such as smart cities, autonomous vehicles, augmented reality and Industry 4.0. The latter will endow mobile networks with cloud-computing capabilities located at the edge, enabling computation-intensive, context aware services for mobile users, such as those based on artificial intelligence. The next generation of mobile networks will therefore witness a tight integration of computation and communication.

The 5G Radio Access Network (RAN) network, based on the New Radio (NR) 3GPP standard, will be deployed progressively, and will coexist for a relatively long time with the existing 4G (LTE/LTE-Advanced) infrastructure. To favor the above transition, the data plane of the NR technology consists of a stack of layered protocols, which closely resembles that of 4G. NR User Equipments (UEs), e.g. handheld devices, will need to be able to connect to either or both the 5G and the 4G network.

The associate editor coordinating the review of this manuscript and approving it for publication was Tiago Cruz.

5G networks will matter for both the communication performance they offer, and the services that they enable. This calls for credible tools to evaluate the performance of both, in an end-to-end context. As far as communication performance is concerned, most of the intelligence in 5G networks will be realized in software: therefore, being able to assess the Quality of Service offered by the NR RAN to a user (e.g., the latency of a connection), as a function of different network intelligence designs (e.g., admission control, packet scheduling), in a credible way, will help network operators to maximize their network utilization. On the other hand, service providers of next-generation services, such as autonomous driving or factory automation, will need to assess the performance of their services on different 5G network configurations or deployments.

In this article, we present Simu5G, a novel 5G simulation library for the OMNeT++ simulation framework [14]. Simu5G includes a collection of models with well-defined interfaces, which can be instantiated and connected to build arbitrarily complex simulation scenarios, and is fully compatible with the INET library [15], which allows one to simulate end-to-end scenarios involving arbitrarily complex TCP/IP networks including 5G NR layer-2 interfaces. In particular, Simu5G models the data plane of the 5G RAN (rel. 16) and core network. It allows simulation of

5G communications in both Frequency Division Duplexing (FDD) and Time Division Duplexing (TDD) modes, with heterogeneous gNBs (macro, micro, pico etc.), possibly communicating via the X2 interface to support handover and inter-cell interference coordination. Dual connectivity with an eNB (LTE base station) and a gNB (5G NR base station) is also available, making it possible to simulate 4G/5G transitions scenarios. 3GPP-compliant protocol layers are modeled, and physical transmission is modelled via a realistic, customizable channel models. Resource scheduling in both uplink and downlink directions is supported, with support for Carrier Aggregation and multiple numerologies, as specified by the 3GPP standard. Simu5G supports a large variety of models for mobility of UEs, including vehicular mobility.

Simu5G allows one to code and test, for instance, resource allocation and management schemes in 5G networks, e.g. selecting which UEs to target, using which modulation scheme, etc., taking into account inter-cell interference coordination, carrier selection, energy efficiency and so on. As far as services are concerned, it allows a user to instantiate scenarios where a user application, running at the UE, communicates with a MEC application residing at a MEC host, to evaluate (e.g.) the round-trip latency of a new-generation service, inclusive of the computation time at the MEC host. Moreover, it models the Cellular Vehicle-to-Everything (C-V2X) standard, which relies on network-controlled resource allocation for device-to-device (D2D) communications. Simu5G can also run in *real-time emulation* mode [11], enabling interaction with real devices. A user can thus run live networked applications through an emulated 5G network, using the *same* codebase for both simulations and live prototyping, which abates the developing time and makes results more reliable and easier to demonstrate.

In this article, we present the modeling and the capabilities of Simu5G, with the aim of helping researchers to understand the level of detail and to get a clear idea of its functionalities. We also discuss the provisions that were made to make Simu5G scalable (e.g., able to simulate tens of cells as required by 3GPP scenarios), and how it was validated. To show the type of studies enabled by Simu5G, we report two representative case studies involving the evaluation of next-generation services running on a 5G network: the first one compares MEC service migration policies, and the second one compares the impact of communication modes on platooning services in a C-V2X scenario.

To the best of our knowledge, few tools that simulate the 5G NR data plane are available to the research community. Leaving aside *physical-level* 5G simulators, such as the Vienna 5G SL simulator [4], which model the physical layer (and possibly the MAC), but nothing above that, hence are not suitable for the kind of evaluations that we discussed earlier, the only other *end-to-end* 5G simulators that we know of are 5G-LENA [5] and the 5G-air-simulator [6]. However, both lack several functionalities included in Simu5G, e.g., 5G-LENA lacks FDD, and both lack network-controlled

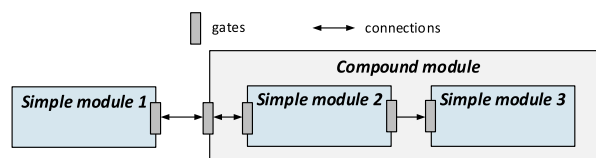


FIGURE 1. OMNeT++ module connection.

D2D communications and do not model dual-connectivity scenarios.

The rest of the paper is organized as follows: Section II reports background on OMNeT++. Section III discusses the related work on simulation of 5G networks in detail. In Section IV we present the architecture of the 5G network, the way Simu5G models it, and discuss its validation. Section V presents topical use cases, and Section VI concludes the paper.

II. THE OMNeT++ SIMULATION FRAMEWORK

OMNeT++¹ is a well-known discrete-event simulation framework that can be used to model virtually any kind of networks, such as wired, wireless, on-chip, sensors, photonic, etc. Its main building blocks are *modules*, which can be either *simple* or *compound*. Modules exchange *messages* through *connections* linking their *gates*, which act as interfaces. A *network* is a special compound module, with no gates to the outside world, which sits at the top of the hierarchy. Connections must respect module hierarchy: with reference to Figure 1, simple module 3 cannot connect to 2 directly, but must instead pass through the compound module gate. Simple modules implement model behavior via *event handlers*, called by the simulation kernel on receipt of *messages*. For instance, a node can schedule a timer by sending a message to itself. Simple modules have an *initialization* and *finalization* function, that can be called in user-defined order at the start and the end of a simulation. OMNeT++ offers support for basic simulation functionalities (e.g., event queueing, random number generation, etc.), allowing users to concentrate on writing their own simulation models.

OMNeT++ separates a model's *behavior*, *description* and *parameter values*. The *behavior* is coded in object-oriented C++. The description (i.e., gates, connections and parameter definition) is expressed in separate files written in Network Description (NED) language. Parameter values are written in initialization (INI) files. NED is a declarative language, which exploits inheritance and interfaces, and it is fully convertible into XML. NED allows one to write *parametric* simulation scenarios, e.g. rings or trees of variable size, via both a GUI (for basic/novice editing) and textual editing (for advanced/expert editing). INI files contain the parameter values that will be used to initialize the model. Multiple values or intervals can be specified for a parameter.

¹<http://omnetpp.org>, last accessed May 2020

The OMNeT++ software offers support to *simulation workflow automation* [9], facilitating the steps of a simulation study that are most time consuming and error prone for a user. Its Eclipse-based Integrated Development Environment (IDE) facilitates debugging by allowing a user to inspect modules, turn on/off textual output during execution, visualizing the message flow in an animation, and displaying events on a time chart. OMNeT++ *studies* are generated automatically from INI files, computing the Cartesian product of all the parameter values and generating independent replicas with different seeds for the random number generators. Multiple runs can be executed in parallel on a multicore machine. Rule-based data analysis allows a user to construct *recipes* to filter or aggregate data, which can then be applied to selected data files or folders.

INET [15] is a model library for OMNeT++. It implements models of many components of a communication network, such as communication protocols, network nodes, connections, etc. INET contains models for the Internet stack (TCP, UDP, IPv4, IPv6, OSPF, BGP, etc.), wired and wireless link layer protocols (Ethernet, PPP, IEEE 802.11, etc.), and provides support for developing custom mobility models, QoS architectures, etc. Thanks to OMNeT++ modular structure, by incorporating the INET library a user can instantiate and connect protocol layers (e.g., an entire TCP/IP stack at a host, from the application to the MAC), and quickly setup composite models, e.g., an IP router with an Ethernet card and a PPP WAN connection.

An interesting feature of OMNeT++ is that it can *slow down* the flow of simulation time to the pace of real time. This is only possible if simulated time flows faster than real time, which depends on the density of events, their processing, and the hardware running the simulation. Since the INET library allows one to attach a host's network interface to a simulation module, to inject *real* TCP/IP packets into a simulator, and collect them from it, it is possible to run simulators based on OMNeT++ and INET, such as Simu5G, as *real-time network emulators*, carrying packets between real applications.

III. RELATED WORK

In this section we compare Simu5G with the related work. In order to give the reader a fair and thorough comparison, we first need to lay down some terminology.

As far as wireless network simulation is concerned, a meaningful distinction is between *physical-level* and *end-to-end* simulators. With the former, one is interested in measuring physical-layer quantities, such as the Signal to Interference to Noise Ratio (SINR) or spectral efficiency, as a function of physical-layer designs, such as antenna layout, transmission schemes, etc. With the latter, one is interested in the performance of application-level quantities (e.g., the end-to-end delay, the throughput of user transactions, etc.), as a function of higher-layer designs (e.g., an admission control scheme, or an airtime scheduling algorithm). Link-level simulators model physical links to a high level of fidelity, but

typically do not model layers above the MAC. What is above the MAC is normally abstracted as a “traffic generator”, its only purpose being generating backlog for the physical layer. Quite often, link-level simulators follow a Monte-Carlo approach rather than a discrete-event one. On the other hand, end-to-end simulators normally include models of application logic, layer-4, layer-3 and layer-2 protocols, as well as network equipment running those protocols, and they always are discrete-event (typically, events include packet arrivals and departures at some interface).

A recurring term in simulation parlance, which can however be misleading with respect to the above distinction, is *system-level* simulator: the latter defines a simulator where the *interaction* among composing models is more important than the level of detail of each single model. End-to-end simulators are certainly system-level ones (think, for instance, of the interaction between protocol models), but physical-level simulators can be too (e.g., a cellular network simulator that account for the interaction among different cells in order to compute the SINR of each UE).

Orthogonally, we distinguish *standalone simulators* from *model libraries for simulation frameworks*. A standalone simulator is a software designed to serve a specific purpose. A *simulation framework* (e.g., OMNeT++ or ns3) is a software that allows developers to write their own *model libraries*. Model libraries, if correctly coded, are interoperable, which means that they get reused, hence validated, by the users of the framework. The same users can thus setup complex scenarios quickly and reliably, just collating existing validated models. On the contrary, adding a model to a standalone simulator takes a considerable amount of time and effort. Simulation frameworks often come with prebuilt functionalities, such as event-handling routines, random number generation, statistics collection, thus freeing developers from the burden of writing them. Moreover, they often offer support to *simulation workflow automation*. This includes the ability to define parametric scenarios, to manage multiple repetitions with independent initial conditions, to efficiently store, retrieve, parse, analyze and plot simulation results, etc. The lack of workflow automation support forces users to develop home-brewed, error-prone or however unstructured solutions (e.g., one-shot scripts), which is a known cause of delay and errors in simulation studies, especially when considering large-scale ones, as shown in [9] and [10].

According to the above classifications, Simu5G is a model library, written for the OMNeT++/INET framework, for end-to-end simulation studies. A previous conference paper of ours [1] first introduced it. With respect to [1], this article reports a more comprehensive description of functionalities, including device-to-device transmissions, and topical examples of case studies, namely modeling of C-V2X scenarios in a 5G New Radio network. Paper [11] discusses the real-time emulation capabilities of Simu5G, showing that a desktop computer can run an emulation of a multicell 5G network carrying application traffic up to several Mbps.

To the best of our knowledge, few other tools to simulate 5G networks are available to the research community. Hereafter, we review them according to the above classifications.

A well-known physical-level simulator is the Vienna 5G SL simulator [4]. It is a MATLAB-based simulator that allows one to evaluate average PHY-layer performance by means of Monte Carlo simulations. A system-level version of it, called Vienna 5G System Level Simulator, allows one to trade accuracy for scale, thus enabling the evaluation of larger-scale networks in terms of average performance. This simulator is well tailored for the evaluation of lower-layer procedures, including signal-processing techniques. However, it cannot be used to evaluate multi-layer, end-to-end scenarios.

The 5G K-Simulator [3] is a standalone simulator *suite* for analyzing various aspects of 5G networks. The suite is composed of three tools: K-SimLink simulates the PHY layer of a single UE-gNB pair; K-SimSys allows the evaluation of MAC-layer interactions of multiple UEs and gNBs; both are physical-level simulators in the above classification. The third tool, K-SimNet evaluates the RLC, RRC, PDCP layers. The three tools are interoperable. However, they only interact by passing *average values* from one to the other, and it is not possible to perform end-to-end simulations. For instance, one cannot observe a *single packet* traveling from a source application to its destination and traversing all protocol layers. Application models are limited only to full buffer and non-full buffer, no explicit mention is made of mobility models. Finally, the MAC layer is based on the ns3 mm-wave module [8], which was developed at the earliest stages of the 5G standardization process and is known to be non-fully standard compliant [6]. Other physical-level simulators, with fewer functionalities than the above, are [20]–[23].

Among end-to-end simulators of 5G networks, we find 5G-LENA [5] and the 5G-air-simulator [6], both recently released. The 5G-LENA model library for the ns3 framework evolves from the LENA 4G library [7] and includes an upgraded, standard-compliant version of the ns3 mm-wave module [8]. It is focused on the simulation of MAC and PHY layer of NR and provides tools for the evaluation of Bandwidth Parts management, which Simu5G lacks. However, it lacks support for FDD mode [6], it does not model dual-connectivity, and we have no indications that it supports network-controlled D2D, MEC or C-V2X scenarios, hence it would not be possible to perform the analysis reported in Section 5 with it.

The 5G-air-simulator is an end-to-end standalone simulator, developed as an extension of the preexisting LTE-Sim, that simulates LTE and LTE-Advanced networks. As such, it inherits most of LTE-Sim pros and cons. On the pros side, it models a wide variety of standard 5G functionalities, some of which are – to the best of our knowledge – exclusive to it, such as Massive MIMO and broadcasting. On the cons side, it lacks others, notably D2D and 4G/5G dual connectivity. Moreover – and more importantly – like its predecessor, it lacks tools for simulation workflow automation, which

reduce its usability for large-scale simulations. For example, scenario definitions are written as static C++ functions, and are compiled together with the simulator. Mixing models, scenarios and definition of experiments is a clear *don't* in simulation practice [9]. Moreover, despite being designed to be an application-level tool, it does not model layer-4 protocols such as TCP and UDP (they are listed, but their behavior is not defined), which makes it impossible to perform a full-stack end-to-end analysis of (e.g.) TCP-based services, such as the ones run in [11].

IV. 5G NETWORKS AND THE Simu5G LIBRARY

This section describes the main elements of the data plane of a 5G cellular network, and the way Simu5G models these elements and functionalities. Moreover, we describe how Simu5G was validated.

A 5G cellular network consists of a Radio Access Network (RAN) and a Core Network (CN), as shown in Figure 2. The RAN is composed of *cells*, under the control of a single base station (BS). 5G base stations are called gNodeBs (gNBs), and they represent an evolution of 4G base stations, which are called eNodeBs (eNBs). UEs are attached to a BS and can change the serving BS through a handover procedure. BSs communicate with each other via the X2 interface, a logical connection which normally runs on a wired network. The data plane of the CN consists of one or more User Plane Functions (UPFs) that provide the interconnection between the RAN and the data network. Forwarding in the CN is carried out using the GPRS tunneling protocol (GTP).

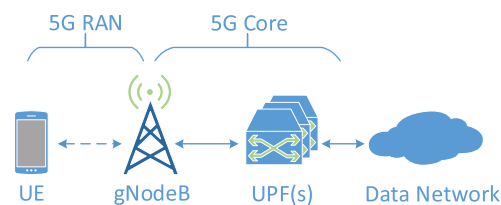


FIGURE 2. Architecture of the data plane of a 5G cellular network.

In the RAN, communications between the BS and the UE occur at layer 2 of the OSI reference model. Layer 1 and 2 are implemented using a stack of four protocols, on both the BS and the UE. From the top down, we first find the Packet Data Convergence Protocol (PDCP), which receives IP datagrams, performs cyphering and numbering, and sends them to the Radio Link Control (RLC) layer. RLC Service Data Units are stored in the RLC buffer, and they are fetched by the underlying MAC (Media Access Control) layer when the latter needs to compose a transmission. The MAC assembles the RLC Protocol Data Units (PDUs) into Transport Blocks, adds a MAC header, and sends everything through the physical (PHY) layer for transmission.

Resource scheduling is done by the BS periodically, every Transmission Time Interval (TTI). On each TTI the BS allocates a vector of Resource Blocks (RBs) to backlogged UEs, according to its scheduling policy. A Transport Block

occupies a variable number of RBs, based on the Modulation and Coding Scheme (MCS) chosen for transmission. The MCS defines the number of bits that an RB can carry, and is selected by the BS, based on the Channel Quality Indicator (CQI) reported by the UE. The latter mirrors the SINR perceived by the UE, quantized over a range going from 0 (i.e., very poor) to 15 (i.e., optimal).

In the downlink (DL), the BS transmits the TB to the scheduled UEs on the allocated RBs. In the uplink (UL), the BS sends *transmission grants* to UEs, specifying which RBs and MCS to use. UEs signal to the BS that they have UL backlog by sending Buffer Status Reports (BSRs) after a scheduled transmission, or by starting a random access procedure in order to obtain a scheduling grant by the BS, if they are not scheduled. Scheduling and transmissions in the UL and DL directions are independent. The partitioning between UL and DL resources can be in frequency or time, leading to FDD and TDD. In FDD, each direction uses a separate spectrum. In TDD, instead, the DL and UL legs share the same spectrum, and the two transmission directions alternate over time.

MAC transmissions are protected by a Hybrid-Automatic Repeat reQuests (H-ARQ). After a configurable number of TTIs, the receiver sends an ACK/NACK to the sender, which can then re-schedule a failed transmission.

A. THE Simu5G LIBRARY

Simu5G simulates the data plane of the 5G New Radio RAN and CN. Signalling and management protocols are not implemented in the current version (but they can easily be added by the interested user). The main elements of the Simu5G library are the *NrUe* and *gNodeB* compound modules, which model a UE and a gNB with NR capabilities. Their internal architecture is shown in Figure 3. All nodes are geolocated on a three-dimensional cartesian plane, which allows one to measure inter-node distances (e.g., to compute path loss). A UE model includes all protocol layers, from the physical to the application layer. Notably, it also includes the IP and TCP/UDP protocols, as well as vectors of TCP/UDP applications. The UE’s NR functionalities are implemented in its Network Interface Card (NIC), called *NrNicUe*. On the

other hand, a gNB compound module includes protocols up to layer 3 (IP) and has two network interfaces: a NR one, implemented in the *NrNicGnb* module, and one running the Point-to-Point Protocol, for wired connectivity towards the CN. The internal structure of both NICs is shown in Figure 4, and will be described in more detail later. For the time being, we just need to observe that the *NrNicUe* module has a dual stack of protocols (identified by the LTE and NR prefixes) to allow the dual connectivity with both LTE and NR, as foreseen by the deployment roadmap.

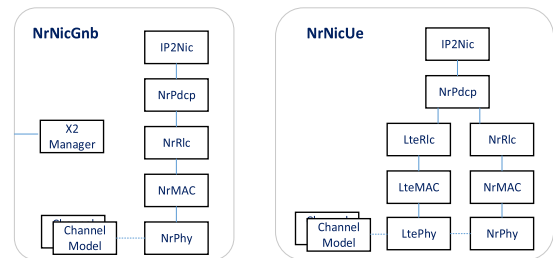


FIGURE 4. Structure of the NR NIC modules.

An arbitrary number of *NrUe* and *gNodeB* modules can be instantiated in a simulation scenario. Conversely, there are two modules, namely the *binder* and the *carrierAggregation* modules, that exist in a single copy. Both maintain global information and can be queried by the other modules. The *binder* maintains data structures containing network-wide information and can be accessed via method calls by every node (both UEs and gNBs). Examples of information stored in the binder are: membership of UEs to multicast groups; which gNB used which frequency resources in the last TTI, etc. This modeling choice has two advantages: first, maintaining a centralized repository of relevant information simplifies the handling of distributed tasks. Second, it allows users to abstract control-plane functions and elements (e.g., servers or signaling protocols), substituting them with queries to the *binder* for the relevant information. This last aspect considerably speeds up the setup of a simulation scenario and the implementation of new functionalities, making Simu5G easier to use and evolvable. The alert reader will notice that the architecture of Simu5G draws heavily from the one of our previous SimuLTE library [12]. This is a design choice, which allows us to incorporate into Simu5G all the models and functionalities of SimuLTE at no cost. These functionalities include, among others, UE handover and network-controlled D2D communications, both one-to-one and one-to-many [16].

The *carrierAggregation* module stores all the information related to the *carrier components* (CC) deployed in the network. CCs are disjoint portions of frequency, characterized by a carrier frequency and a number of available RBs. NR communication can occur on multiple CCs simultaneously, in the so-called *Carrier Aggregation* (CA) mechanism. The *carrierAggregation* module includes a vector of *N componentCarrier* submodules, whose parameters can be configured via

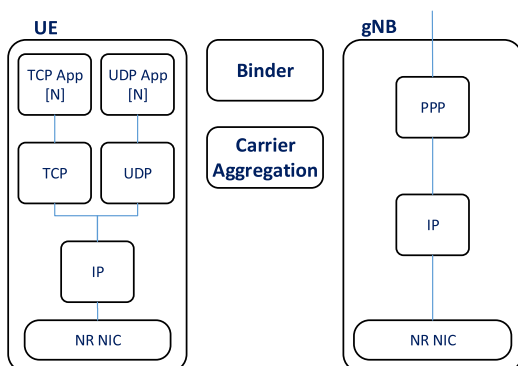


FIGURE 3. Main modules of the Simu5G model library.

NED/INI. The parameters associated to a CC are its carrier frequency, the number of RBs, the *numerology* employed, the selection of either FDD or TDD mode and the *slot format* in case TDD is employed. Both the numerology and the TDD slot format will be described later in more detail. A gNB or a UE might not support all the CCs, due e.g. to deployment choices or equipment limitations. A UE can only attach to a gNB that supports at least one of the CCs supported by the UE itself, and communication can only occur on CCs supported by both.

As far as the CN is concerned, Simu5G provides an implementation of the UPF element of the 5G Core. The UPF module implements the GTP for routing IP datagrams between the gNBs and the data network. A gNB can be connected to the data network through the CN, as shown in Figure 5 (left). This architecture is called StandAlone (SA) deployment, and is expected to occur in new, 5G-only deployments. However, in the next few years, it is expected that 5G will be deployed alongside 4G and coexist with the latter, allowing for a smoother transition. To favor this, 3GPP defines the E-UTRA/NR Dual Connectivity (ENDC) deployment [27], shown in Figure 5 (right). In the latter, there are both an eNB, working as a Master Node (MN) and a gNB, working as a Secondary Node (SN). The eNB is connected to the CN, which is composed by a model of the Packet Data Network Gateway (PGW) of the LTE Evolved Packet Core in this case. LTE traffic flows through the eNB only, while NR traffic goes through the eNB first, and then reaches the gNB via the X2 connection between the two nodes, as shown in Figure 6. Three types of bearer are defined by 3GPP for an ENDC deployment:

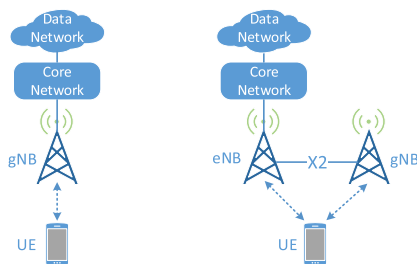


FIGURE 5. SA (left) and ENDC (right) deployment.

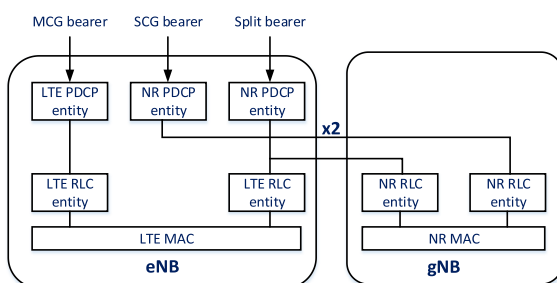


FIGURE 6. Interactions between eNB and gNB in an ENDC deployment.

- Master Cell Group bearer: the UE is served by the eNB, and traffic traverses the LTE protocol stack at both;
- Secondary Cell Group bearer: the UE is served by the gNB. Data destined to the UE gets into the NR PDCP entity at the eNB and is transferred to its peering NR RLC entity in the gNB via the X2 interface.
- Split Bearers: the UE is served by both the eNB and the gNB. Data belonging to the same connection traverses either the eNB or the gNB. The PDCP layer at the UE side will then reorder PDUs coming from LTE/NR RLC layers before presenting traffic to the upper layers.

Simu5G allows one to simulate both SA and ENDC scenarios. The internal structure of the *NRNicGnb* module is shown in Figure 4 (left). It is composed of one submodule for each layer of the protocol stack, plus one *Ip2Nic* module that acts as a bridge with the IP layer. Data packets can be received from the UPF (through the *Ip2Nic* module) in SA scenarios, or from the MN (through the *X2Manager* module) in ENDC scenarios. Figure 4 (right) shows the *NrNicUe* module, which is equipped with two sets of PHY, MAC and RLC submodules to enable dual connectivity. The NR versions of the layers are used for processing data coming from/going to the gNB, whereas the LTE ones are used for processing data coming from/going to the eNB, if any. As shown in the figure, the PDCP layer is unique. This way, packets belonging to a Split Bearer are handled by the same PDCP entity, ensuring in-sequence delivery to upper layers.

B. MODELING OF NR PROTOCOL STACK

As Figure 4 shows, in Simu5G NICs are compound modules which include one simple module per protocol of the NR stack. We describe the main modeling choices and functionalities of each sublayer.

1) PDCP LAYER

The *NrPdcP* module implements the PDCP protocol. Assuming a SA deployment, on the transmission path it performs Robust Header Compression and assigns/creates a Connection Identifier (CID) to packets. The pair CID, UE ID is unique in the whole network. A Logical Connection Identifier (LCID) is kept for each 4-tuple in the form $\langle \text{sourceAddr}, \text{destAddr}, \text{sourcePort}, \text{destPort} \rangle$. When an IP packet arrives at the *NrPdcP* module, its LCID is attached to it (or created, if it does not exist). A PDCP Protocol Data Unit (PDU) is then created and sent to the RLC layer below. On the reception path, a PDCP PDU coming from the RLC is decapsulated, its header is decompressed and the resulting PDCP PDU is sent to the upper layer.

Some extra functionalities are required to handle ENDC settings. In this case, in fact, the *NrPdcP* module is also instantiated in the NIC of an eNB acting as a MN, where packets arriving from the upper layers need to be forwarded to either the eNB's RLC, or to the gNB acting as SN. This is accomplished by marking each packet in the *Ip2Nic* module

with the intended destination. The *NrPdcP* entity then redirects packets towards the RLC layer of either the eNB or the gNB via the X2 interface accordingly. The marking policy within the *Ip2Nic* module works at the *packet* level (rather than at the connection level). This allows finer granularity and dynamic management of SB functionalities, and allows users to design and evaluate, e.g., eNB/gNB load-balancing policies. On the reception path, the PDCP ensures in-sequence delivery to the IP layer in all cases.

2) RLC LAYER

The NR RLC can be configured in *Transparent Mode (TM)*, *Unacknowledged Mode (UM)* and *Acknowledged Mode (AM)*. The TM has no buffering, and it forwards packets transparently to the MAC layer. On the other hand, AM and UM have their own set of transmission/reception buffers. On the transmission path, RLC PDUs are buffered in the transmission buffers, and are fetched by the underlying MAC module on each TTI, as a result of the MAC scheduling process. On the reception path, RLC PDUs are stored in the reception buffer until reassembly of a PDCP PDU is completed, and the latter is sent to the PDCP.

3) MAC LAYER

A NR radio frame is 10 ms long and consists of 10 subframes, each having a duration of 1 ms. NR subframes can be further divided into a variable number of time *slots*, which are called TTIs. A *numerology index* μ defines the slot duration, as shown in Table 1, and – accordingly – the number of TTIs per subframe. UEs are scheduled in slots. In our model, a different μ can be associated to each CC, and configured via NED/INI. We allow for gNBs and UEs to support only a subset of the available numerologies. The example in Figure 7 shows a gNB supporting three CCs that employ different carrier frequencies and numerologies. The gNB serves UE 1 and UE 2, which have different capabilities in terms of supported frequencies and numerologies, as shown in the figure. For instance, UE 1 only supports $\mu < 3$, whereas UE 2 supports frequencies below 6GHz. According to that configuration, UE1 can be served by the gNB using CC 0 and CC 1, whereas UE 2 can be served by the gNB using CC 0 and CC 2.

TABLE 1. NR numerologies.

μ	0	1	2	3	4
TTI duration (ms)	1	0.5	0.25	0.125	0.0625
no. of TTI per subframe	1	2	4	8	16

Simu5G supports both FDD and TDD. In FDD, each CC has separate portions of spectra for the UL and the DL. In TDD, the same spectrum is used for both the DL and the UL, which are instead separated in time. NR TDD allows one to choose among 62 possible slot formats [28], where individual symbols in a slot can be DL, UL or *flexible*. Examples of slot formats are shown in Figure 8.

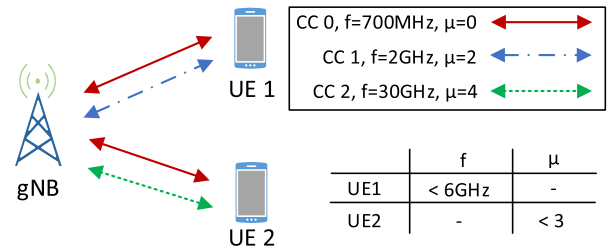


FIGURE 7. Example of UEs' capabilities.

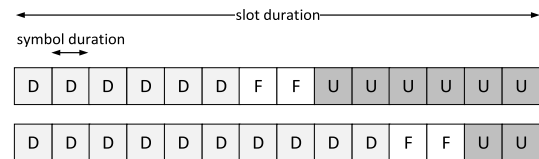


FIGURE 8. Examples of TDD slot formats.

Flexible symbols can be assigned dynamically to either DL or UL transmissions, or kept idle as a guard interval to minimize the DL/UL interference. Hence, the number of bytes that can be transmitted to/by a UE in a slot varies with the TDD slot format, as well as with the selected MCS. As we explain later, this affects the computation of the Transport Block Size (TBS) at the MAC level. We model the TDD slot format as a property of the CC: this means that all gNBs using a given CC will use the same slot format on it. Accordingly, we associate the slot format to the *componentCarrier* submodules. This greatly simplifies interference management, since it guarantees that DL and UL transmissions never interfere with each other. Therefore, their exact arrangement within a slot is immaterial, the only relevant information being their total *number*. Accordingly, we model a slot format as a triplet of integers $\langle n_{DL}, n_{UL}, n_F \rangle$, representing the number of DL, UL and flexible symbols, respectively, whose sum is equal to the total number of symbols available within a slot (i.e., 14). Policies to assign flexible symbols to DL or UL dynamically can be easily defined by a user, e.g. by implementing a function to be invoked at every TTI before executing the scheduling operations, which tells the scheduler to consider the flexible symbols as either UL or DL symbols in that TTI. The current default is that flexible symbols are used as guard symbols.

At both the UE and the gNB, the MAC layer runs periodically, on each TTI. Different CCs may employ different numerologies, hence have different TTI durations. This allows a gNB to run an independent scheduler per CC. Figure 9 shows an example of a scheduling procedure, which takes as input the set Q of backlogged UEs, then it allocates one CC at a time, e.g. starting from CC 0. For each CC i , the scheduler considers U_i , i.e. the set of UEs that can use CC i , to obtain subset $Q_i \subseteq Q$, including backlogged UEs schedulable on CC i . Then, the *scheduling* routine sorts Q_i according to a given policy (e.g. MaxC/I or PF, both of which

```

1  Q = set of backlogged UEs
2  for each CC i active in this period
3    U_i = set of UEs allowed to use CC i
4    Q_i = Q & U_i
5    S_i = scheduling(Q_i)
6    update Q
7  end for
8  S = U{S_i}

```

FIGURE 9. Pseudocode for the scheduling procedure.

are already implemented) and scans it to allocate RBs to UEs. The scheduling routine produces a *schedule list* S_i , including the set of UEs allocated on CC i . UEs which clear their backlog are removed from Q , so that subsequent CCs will not consider them. In the above approach, CCs are scanned in sequence. However cross-CC scheduling policies can also be implemented in this framework.

After scheduling each CC, the scheduler obtains the global schedule list $S = \cup_i S_i$. For each element of S , the MAC layer builds a MAC Transport Block (TB) (in the DL) or issues a scheduling grant (in the UL). The TBS depends on both the number of allocated RBs and the MCS. The latter is chosen following the CQI reported by the UE. The *NrAmc* C++ class determines the TBS according to the procedure defined in [29]. According to the formulas therein, the TBS is also a function of the number of DL(UL) symbols in the slot. When TDD is employed, the number of available symbols is defined by the slot format. The *NrAmc* class supports the extended MCS table with higher modulation orders, i.e. up to the 256QAM modulation.

Simu5G supports flexible timing for the NR H-ARQ feedback, which is *asynchronous*: i.e., the timing of ACK/NACKs is not fixed, and can be configured from the NED/INI file. Following the standard [30], we model independent H-ARQ processes for every CC.

4) PHY LAYER

At the physical layer, the effects of signal propagation and interference on a whole MAC TB, rather than on the symbols composing it, are simulated. Each MAC TB is encapsulated within an *AirFrame* OMNeT++ message sent to the destination module. On receipt of the above message, the destination

entity performs the computations summarized in Figure 10, namely:

- Starting from the transmitted power at the sender, it applies a *channel model* to compute the received power. A channel model can be configured to incorporate fading, shadowing, pathloss, etc., and can be made arbitrarily complex. Simu5G comes with a default channel model called *Realistic Channel Model*, which is compliant with the 3D model described in [19].
- On each RB occupied by the TB, it computes the SINR as $SINR = P_{RX} / (\sum_j P_{RX}^j + R)$, where P_{RX} is the received signal power, P_{RX}^j is the power received from the j -th interferer and R is the Gaussian noise. The set of indexes j is computed by querying the Binder to know which other nodes were using the same RB. For each interferer, the received power is computed by applying the channel model, starting from the transmission power.
- Then, it uses Block Error Rate (BLER) curves to compute the reception probability for each RB composing the ongoing transmission. BLER curves can be obtained from a link simulator or from 3GPP documents. This makes it possible to translate a SINR and a transmission format to a probability of correct reception of an RB. More specifically, the `error()` function considers the BLER curve related to the MCS used in transmission, at the abscissa represented by the measured SINR, and it obtains an error probability for that RB, call it P_{err} .
- Finally, a uniform random variable $X \in [0; 1]$ is sampled and the *AirFrame* is assumed to be corrupted if $X < 1 - \prod_i (1 - P_{err}^i)$, and correct otherwise.

Note that a MAC TB is sent on a given CC, hence the corresponding *AirFrame* is subjected to channel effects (e.g. path loss, shadowing etc.) that depend on that CC. To account for this, each gNB/UE is equipped with a vector of *channelModel* modules, as shown in Figure 4, and each of them is associated with one of the CCs available in the *carrierAggregation* module. Figure 11 shows an example of such association, where the *carrierAggregation* module implements two CCs, whose indexes in the *componentCarrier* vector are 0 and 1, respectively. The gNB and UE1 are configured to use both

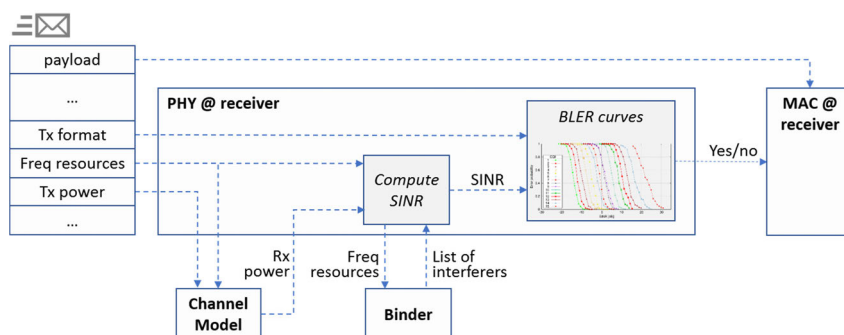


FIGURE 10. Block diagram of the modelling of the physical layer within Simu5G.

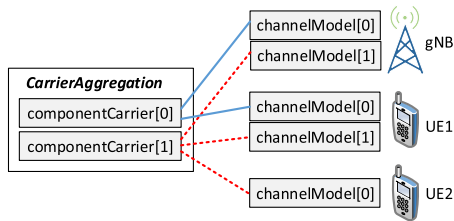


FIGURE 11. Example of CA configuration.

CCs, hence they have two channel models, associated with the two CCs. UE2, instead, is configured with one channel model only associated with CC 1. Each transmitted AirFrame includes a control field specifying the CC it is transmitted onto, thus enabling the receiver to process it via the relevant *channelModel* module.

The PHY layer interacts with the *channelModel* modules via *getSinr()* and *end error()* functions, which compute the SINR and check if the airframe is correctly decoded, respectively. The latter are the functions that need to be redefined when implementing a new channel model.

C. MODELING OF DEVICE-TO-DEVICE TRANSMISSIONS

D2D communication is a feature of LTE-A and NR since release 12 [31] and it allows two UEs to communicate directly when they are in proximity, hence without relaying their traffic through the BS and the CN as in conventional cellular communications. This makes D2D transmissions faster, since they require a single hop (called the *sidelink* – SL) instead of two (i.e., through the UL leg to the serving BS, and the DL one to the receiving UE) and enables proximity services, including those for IoT transmissions and Cellular Vehicle-to-Everything (C-V2X, [36]). In network-controlled D2D, data is sent on the SL, while the BS retains control over resource allocation and takes care of contention and interference management. SL communications can be allocated anywhere, in principle. A common policy (to which we stick in our work) is to use resources in the UL spectrum, which is normally less utilized than the DL one and less prone to interference.

Both point-to-point (P2P) and point-to-multipoint (P2MP) D2D communications are envisaged. With P2P D2D, a UE sends a packet to one (and only one) receiving UE, whereas with P2MP D2D a UE sends a packet to a set of neighboring UEs belonging to a multicast group, identified by a multicast group ID in the MAC Transport Block. P2P transmissions are acknowledged by the receiver, whereas P2MP are not.

Data packets to be sent using D2D (either P2P or P2MP) traverse all the layers of the NR/LTE protocol stack and undergo the same processing as UL/DL packets, with few modifications. This allowed us to reuse the modeling of the protocol layers shown in the previous sections, adding to both the UE and the gNB the required D2D-specific operations, which we will explain in the following.

In addition to the LCID, at the sender UE’s PDCP layer we assign a *flow direction* to each packet coming from the IP layer, as shown in Figure 12. The direction can be UL, P2P D2D or P2MP D2D. This way, packets having the same LCID (i.e., same 4-tuple) but different flow directions will be treated as belonging to different, independent data flows. The flow direction of P2P flows can be either set statically via the NED/INI file or changed dynamically via some *mode switching* policy. In fact, mobile UEs may not always be in communication range, hence the gNB may want to change their communication mode dynamically [32]. In Simu5G, mode-switching policies are user-definable.

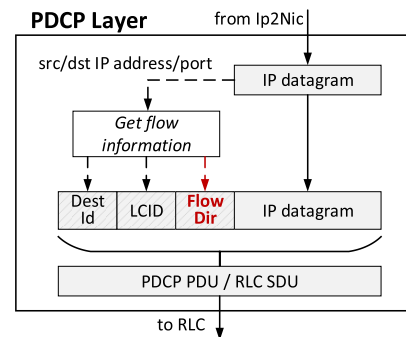


FIGURE 12. Selection of flow direction at PDCP layer.

For P2MP D2D communications, we chose to map multicast IP datagrams (i.e. those with a multicast IP destination address) to the P2MP D2D flow direction. In this case, packets are also assigned a multicast group ID, as specified by the NED/INI file. The information regarding the flow direction will also be included in the BSR sent by the UE to the BS, so that the latter can allocate resources on either the SL or the UL, depending on the UE request.

At the receiver side, the main modifications involved the H-ARQ mechanism of P2P communications. With reference to Figure 13, the H-ARQ (N)ACK is sent directly to the sender without going through the BS. However, the BS still needs to know if a H-ARQ retransmission is due in a future TTI, because it needs to allocate resources for that to occur.

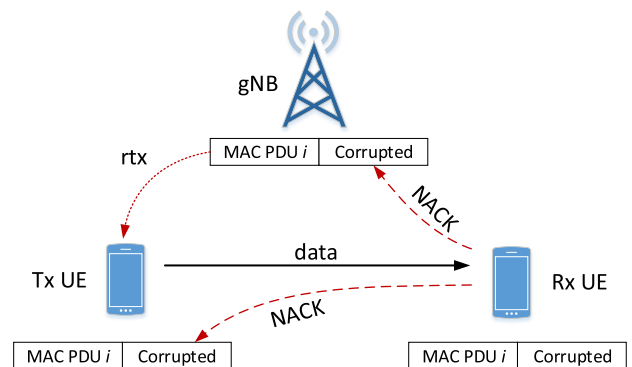


FIGURE 13. H-ARQ mechanism of P2D D2D communications.

In our model, this issue is solved by sending a copy of the H-ARQ (N)ACK to the BS as well, so that it can keep trace of the status of SL transmissions and schedule retransmissions when needed.

As far as resource allocation is concerned, Simu5G models dynamic scheduling of D2D transmissions (which is coherent with C-V2X *Mode 1*, as we discuss in Section V): whenever a UE has data to send using D2D, it sends a random access request and a BSR to its serving BS, which schedules SL resources accordingly and issues a SL grant to the UE, either P2P or P2MP. With this scheme, the BS can schedule D2D transmissions on exclusive resources, or enforce *frequency reuse*. This consists in allocating the same set of RBs to different D2D transmitters at the same TTI, provided that the interference level is kept low (i.e., colliding D2D transmissions happen sufficiently far from one another), and it allows the BS to reduce the overall spectrum occupancy.

D. MODELLING FOR SCALABILITY

A common criticisms of end-to-end network simulators (see, e.g., [4]) is that they are not scalable, because they model the entire protocol stack of every node, instead of, e.g., abstracting everything that lies above the MAC as a traffic generator. Hereafter, we provide evidence of the mechanisms that were employed to make Simu5G scalable.

It is often the case that a user is interested in the performance of a single cell (e.g., its RB occupancy or throughput), or of few neighboring cells (e.g., to account for handover). However, the above performance cannot be correctly assessed unless several other cells are simulated as well, for the correct accounting of inter-cell interference. Simulating more cells implies a higher computation load, which limits the scale of the scenarios that can be simulated.

To solve this issue, Simu5G includes two models of a gNB. Besides the “full” one described in the previous sections, a gNB can be modeled as a *light cell*. A light cell does not run the NR protocol stack. It is configured with a location, a radiation pattern, a transmission power, and a *distribution of occupied RBs*. On each TTI, it samples a value from that distribution and occupies the resulting number of RBs in the DL subframe, so as to produce inter-cell interference. This allows one to produce a configurable level of interference in the cell(s) of interest, without incurring the overhead of simulating many full-stack gNBs and their served UEs. This way, we can run credible simulations with tens of external cells at a tolerable computation cost. This is also useful when running Simu5G in emulation mode, where keeping the computation load low is of paramount importance to enable real-time emulation. Paper [11] shows that real-time emulation of a multicell scenario is possible on a desktop PC, exactly thanks to the usage of light cells.

E. VALIDATION AND CALIBRATION

A simulator should be properly validated, to ensure that the results obtained with it are credible. The techniques used to

do so are several and should be used concurrently. Some of the ones that were used with Simu5G are:

- Extensive parsing of event traces (leveraging the features of the OMNeT++ IDE);
- Continuity, consistency and degeneracy tests [24];
- Backward compliance: for instance, running a 5G scenario with FDD and $\mu = 0$, i.e. TTIs of 1 ms (as LTE's), and forcing all nodes to use MCS schemes which are common to both LTE and NR, yields the same results as SimuLTE does;
- Fingerprinting, to test that results and/or the sequence of function calls does not change following code updates.

In this section we focus on a specific validation technique, i.e. comparison with known analytical models. We simulate the Urban Macro scenario described in Table 4, config. A of [25]. We compare the SINR that we obtain with the one reported as an attachment of that document.

With reference to Figure 14, we simulated 57 cells deployed according to a regular hexagonal tessellation, whose inter-site distance is 500 m. Each site hosts three cells, radiating outwards according to the horizontal and vertical pattern described in [25]. We collect statistics only from the central site (three central cells), whereas the other (light) cells only produce interference occupying the whole spectrum. We randomly deploy 30 UEs in the three central hexagons, which attach to the cell from which they perceive the best SINR. 80% of UEs are assumed to be indoor, 20% outdoor. We assume DL traffic only and each UE receives a 240kbps constant bit rate traffic. The values in the following charts are obtained by averaging statistics from 50 independent repetitions, with 95% confidence intervals. The main simulation parameters are summarized in Table 2. Figure 15 shows that the CDF of the SINR measured by UEs in the scenario described above overlaps the 3GPP reference one.

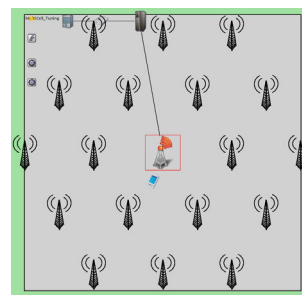


FIGURE 14. 3GPP Urban Macro simulation scenario.

Having validated Simu5G's channel model, we calibrate the sensitivity of a receiver following the guidelines reported in [25]. We employed the BLER curves taken from [6]. Such curves represent the error probability for a transmission occupying one RB. Since 3GPP recommends a target TB error rate of 10%, we carried out simulations with different sensitivities of a 5G receiver, modeled by an offset $\Delta \in [3 \text{ db}, 7 \text{ db}]$ to shift the above BLER curves to the left, in order to determine the sensitivity that allows us to meet

TABLE 2. Main simulation parameters.

Parameter Name	Value
#gNBs	57
Inter-site Distance	500 m
#UEs	30 (uniform distribution)
Carrier frequency	700 MHz
Bandwidth	10 MHz (50 PRBs)
Fading + shadowing	Enabled
gNB Tx Power	46 dBm
gNB antenna gain	8 dBi
gNB noise figure	5 dB
UE antenna gain	0 dBi
UE noise figure	7 dB
Path loss model	(3GPP - TR 36.873)
Load of interfering gNBs	100% (Full buffer)
UE speed	3 km/h (80% indoor, 20% indoor)
Traffic type	Constant Bit Rate (240 kbps)
# of repetitions	50

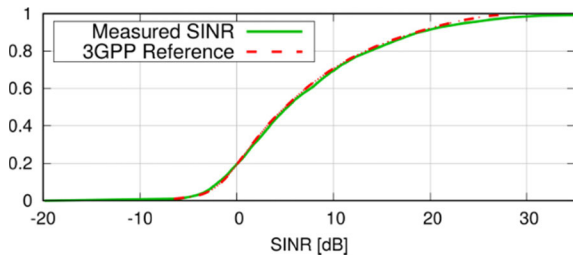


FIGURE 15. Measured SINR with Simu5G.

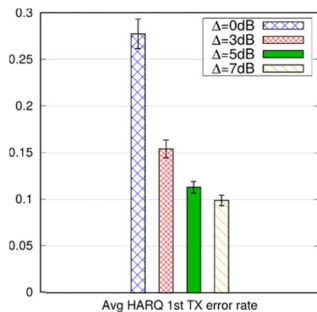


FIGURE 16. Average error rate after 1st TX attempt.

that target. Figure 16 reports the average error rate for the first transmission attempt of Transport Blocks, i.e. the probability that the first transmission of a MAC Transport Blocks is not decoded correctly. The employed shift increases from left to right. We observe that the rate is close to the target 10% error rate when a shift of either $\Delta = 5\text{db}$ or $\Delta = 7\text{db}$ are used. Figure 17 shows the CDF of the error rate for the first H-ARQ transmission attempt, where we observe that the target 10% error rate is around the median value, whereas only a small fraction of UEs (i.e. $\sim 5\%$) gets an error rate larger than 20% when $\Delta = 5\text{db}$ or $\Delta = 7\text{db}$.

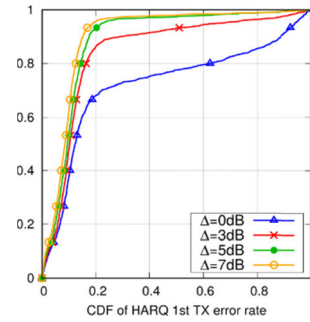


FIGURE 17. CDF of error rate after 1st TX attempt.

V. PERFORMANCE EVALUATION USE-CASES

In this section we show how Simu5G can enable meaningful performance assessment of cutting-edge technologies. We concentrate on two use-cases, namely application offloading with MEC and platooning with C-V2X.

A. APPLICATION OFFLOADING WITH MEC

Although orthogonal to the access technology itself, MEC is expected to unleash its full power in conjunction with cellular access, and specifically 5G. In a MEC environment, computation nodes called *MEC Hosts* are placed close to the RAN and interact tightly with the latter to obtain information on the status of the radio network and its users. MEC applications run on behalf of the user on MEC hosts, in a virtualized environment. Computational resources can be allocated on demand to users requesting a service or task.

According to the ETSI architecture [33], MEC functions are organized in two layers, namely the *MEC System Level* and the *MEC Host Level*. The MEC System Level maintains a global view of the status of all the MEC Hosts in the system. It receives *MEC Application Instantiation* requests from applications running at the user side and routes them to the most suitable MEC Host, e.g. based on requirements such as maximum communication latency, computational resources and availability of MEC services. Within the MEC Host, the *MEC Platform* provides MEC Services [33] that can be exploited by MEC Apps, such as the *Smart Relocation Service*, to handle migration of MEC Apps to other MEC Hosts; the *Radio Network Information Service* (RNIS), 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*, which provides information on the users' position. The *Virtualization Infrastructure* runs MEC Apps as instances of virtual machines and allows them to communicate both within (e.g., with the services within the MEC Platform) and outside the MEC Host (e.g., with users' local application).

Simu5G includes a model of the above infrastructure, as well as functions that allow a UE to dynamically request the initialization and the termination of one or more

applications within a MEC Host and to communicate with such applications so that a specific task is accomplished. The above architecture allows a user to develop its own MEC App and plug it seamlessly within Simu5G. The main building block of our model is the *MecHost*, shown in Figure 18. The latter hosts MEC Apps created *on demand* on reception of a request from UEs. The MEC Host includes a GTP module, so that it can be placed anywhere in the CN of the 5G network, and a Virtualization Infrastructure module that handles the data traffic. The Virtualization Manager submodule manages the life cycle of MEC Apps, handling requests for instantiation and termination of MEC Apps from the UEs and forwarding data packets to the correct MEC App. Moreover, it interacts with the Resource Manager, which keeps track of the computational resources (RAM, storage and CPU) currently in use within the MEC Host. In fact, each MEC Host has a configurable maximum amount of resources, and MEC App creation requests come with an indication of how much of each they are going to use. When a new request reaches the Virtualization Manager, the latter queries the Resource Manager, where admission control is checked. For instance, this allows a user to model the computation delay at the MEC Host based on the amount of occupied resources (e.g., using queueing network models). Finally, the MEC Platform submodule contains MEC services. The number and the type of each MEC Service can be configured. Our implementation comes with a simplified version of the Radio Network Information Service.

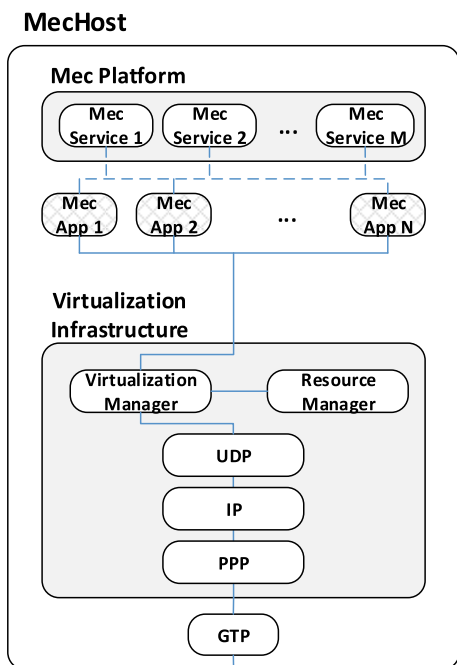


FIGURE 18. Modeling of a MecHost module.

A UE local application exchanges signalling messages with MEC hosts, e.g. to request, respectively, initialization and termination of a MEC App. The initialization message

includes a list of its computational requirements. Once the MEC App is instantiated on the MEC Host, data packets can flow through. If necessary, the MEC App can request the services of the MEC Platform to carry out (part of) its operations. For example, it can contact the Radio Network Information Service to collect information about the radio network.

We consider the scenario of Figure 19, where five servers representing MEC hosts are respectively co-located with five gNBs. The inter-gNB distance is 500m. One UE is linearly moving from the service area of gNB1 to that of gNB2, gNB3 and so on, at a constant speed of 30 km/h. The UE offloads tasks to the MEC Host periodically, with period $T = 1$ s. For each offloaded task, the UE transfers the context to an MEC App running on the MEC Host, which performs computations and sends the context back to the UE. We let $l_i \sim U(8, 12)$ be the context size for task i (hence, the size of the i -th packet to be transmitted), measured in kbits. Moreover, we model the processing time at the MEC Host as $T_{proc} = (l_i \beta_i) / F$, where $\beta_i \sim U(100, 300)$ are the cycles per bit necessary for processing task i and $F = 9$ Gcycles/s is the processing capacity at the MEC [34].

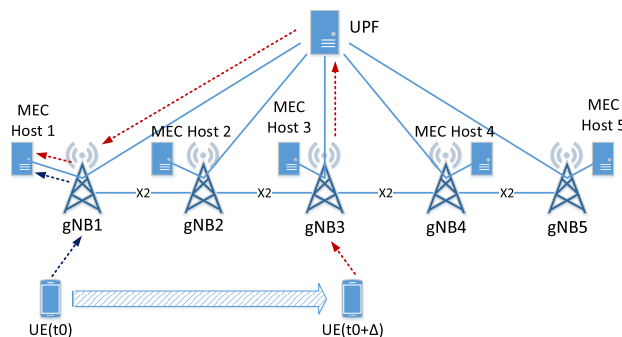


FIGURE 19. Simulation scenario with ME app migration.

We assume that at the beginning of the simulation the UE offloads its tasks to MEC Host 1. When the UE performs the handover to the other gNBs, we consider the two following scenarios: a) the UE keeps offloading its tasks to MEC Host 1, and b) the UE offloads its tasks to the MEC Host co-located with the serving gNB, i.e. the MEC App migrates according to the UE mobility. In the first scenario, data needs to be routed through the UPF, hence the additional latency is uniformly distributed between 15 and 35 ms [34]. In the second scenario, we assume that the migration needs a time in the range (20s, 30s), which is compatible with the results in [35]. More advanced migration algorithms and models can be easily implemented and tested within Simu5G.

Figure 20 shows the latency required for obtaining the result of the task offloading over time. The UE performs the handover at 64s, 124s, 184s and 224s, where the latency increases due to need of rerouting the traffic through the UPF. Without service migration, the latency always stays above 35ms. When the application migrates, the latency goes back to about 20ms after the transient. Figure 21 shows the same

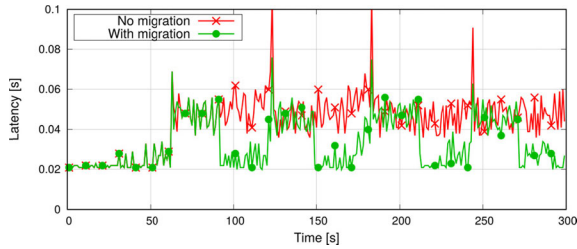


FIGURE 20. Latency of task offloading, $\mu = 0$.

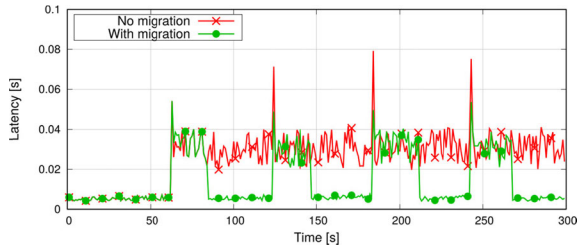


FIGURE 21. Latency of task offloading, $\mu = 3$.

metric when $\mu = 3$ is employed. As expected, the latency has the same evolution, except for scaled-down values due to shorter TTIs.

B. PLATOONING WITH C-V2X

C-V2X has been part of the 3GPP specifications since release 14. With the development of the 5G, the NR standard evolves C-V2X to support ultra-reliable and ultra-low-latency services for vehicular applications like autonomous driving, platooning, infotainment, and so on.

C-V2X communications are realized in two ways: between a vehicle and the network infrastructure, i.e. using the UL/DL path, and between vehicles, i.e. using D2D communications on the SL. In the latter case, we refer to Vehicle-to-vehicle (V2V) communications. While earlier specifications (release 12) for the SL were mostly designed for energy efficiency at the cost of higher latency, the more stringent requirements for V2X services motivated the design of enhanced resource allocation schemes for NR-based C-V2X. In NR, V2V communications can be scheduled according to two resource allocation policies, namely Mode 1 and Mode 2 [36]. With Mode 1, the BS selects the resources to be assigned to vehicles, either dynamically (same as with the UL's) or using Semi-persistent Scheduling (SPS). According to Mode 2, instead, vehicles autonomously select a set of RBs from a V2X resource pool, indicated by the BS. V2V communications can be either P2P or P2MP.

As explained in Section IV.C, a resource allocation method compliant with Mode 1 of C-V2X is already available in Simu5G, as well as P2P and P2MP D2D communications and different numerologies. Moreover, Simu5G can be easily integrated with OMNeT++ vehicular mobility libraries such as Veins [37] and Artery [38].

We now report an example of performance study of a C-V2X application with Simu5G. In particular, we consider

the *platooning* use case, in which vehicles are instructed to travel in a train-like fashion, i.e. keeping the same speed and a small, constant distance to the vehicle ahead. We consider the scenario of Figure 22, where three gNBs are connected to a single MEC host in the CN. Five vehicles move along a straight road and are equipped with a NR NIC, so that they can exploit 5G connectivity and the services offered by the MEC Host to form a platoon. In particular, the first vehicle is the platoon leader (PL) whereas the other ones are the platoon members (PMs). At the beginning of the simulation, vehicles have different speed and distance from their respective predecessor. Each vehicle runs one application that sends the information about the vehicle's position and speed to its corresponding MEC application at the MEC host every $T_{UE} = 50ms$. In the meantime, the MEC Host executes two MEC services, running at period $T_{MEC} = 250ms$: a *platoon formation service* (PFS) that groups vehicles into platoons based on the information gathered from MEC applications, and a *platoon control service* (PCS) that takes as input the platoon(s) formed by the PFS and computes new acceleration values that vehicles must use in order to converge towards a target speed $v_{target} = 13.89m/s = 50km/h$. To do so, the PCS employs the control algorithm in [26], which aims at maintaining safe inter-vehicle distances given the target speed. Then, the output of the PCS, i.e. the acceleration values, are sent to the vehicles which adjust their course accordingly.

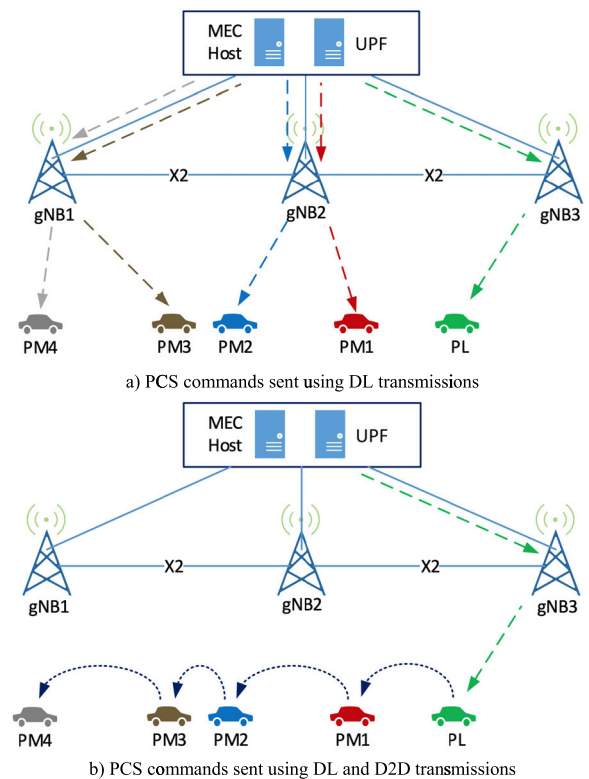


FIGURE 22. Simulation scenario with C-V2X-based platooning.

We run the above service in a RAN whose gNBs operate on a single carrier with 10 RBs, employing numerology index $\mu = 2$. While moving, vehicles perform handover to the gNB they receive the highest SINR from. Since the distance between gNBs is 500m and vehicles' target speed is 50km/h, vehicles remain in the area covered by the gNBs during the 100s of simulated time. We compare two different strategies to deliver PCS commands: in the first mode, each vehicle gets its new acceleration value via one DL transmission from its serving gNB, as shown in Figure 22(a); in the second mode, shown in Figure 22(b), one message including the acceleration of all the vehicles is sent to the PL with a DL transmission, then the PL forwards the message to its follower PM1 using a D2D transmission, which in turn relays it to PM2, and so on. In this last case, it is worth noting that our D2D modeling allows P2P D2D communication between UEs in the same cell only. This means that when the platoon traverses a cell border due to mobility, the P2P D2D connection is temporarily *switched* to the traditional UL-DL path. Figure 23 reports the average end-to-end latency of PCS commands. When only DL transmissions are used, all vehicles obtain the message with minimum delay, whereas with D2D the latency increases with the vehicle's relative position in the platoon, since the message traverses multiple hops through its predecessors. However, since D2D exploits the UL spectrum, using D2D allows one to save 73% of DL frequency resources, which results in less energy consumed by the gNBs. Despite the above differences at the radio level, the two delivery modes produce the same results from the platooning perspective, as shown in Figure 24 and Figure 25. The former shows each vehicle's distance from its predecessor, whereas the latter shows how the speed of vehicles changed over time to obtain the result. In both cases, the platoon reaches stability in less than one minute.

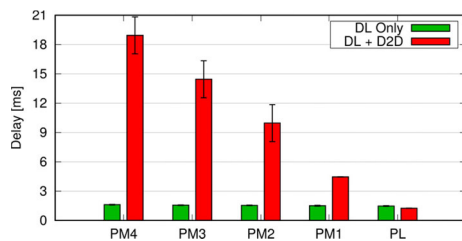


FIGURE 23. Average end-to-end latency of the PCS' commands.

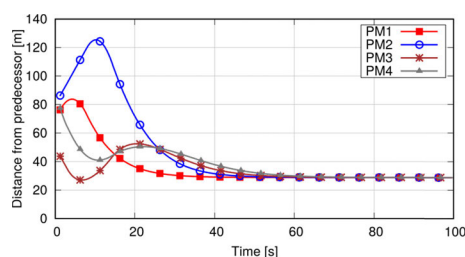


FIGURE 24. Vehicles' distance from their predecessor over time.

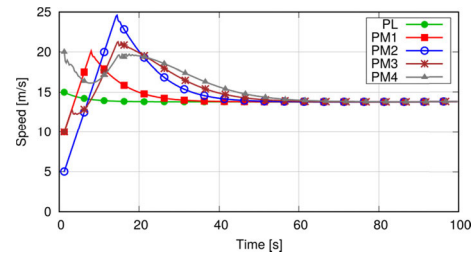


FIGURE 25. Vehicles' speed over time.

VI. CONCLUSION AND FUTURE WORK

This article presented Simu5G, a new model library for 5G NR for the OMNeT++ simulation framework. To the best of our knowledge, this is one of two libraries allowing end-to-end application-level communications in complex, heterogeneous scenario (the other one being 5G-LENA [5]), but the only one modelling ENDC deployments, FDD access, D2D communications, MEC and C-V2X. We have described Simu5G's resource management and protocol layers, to allow prospective users to understand its modeling philosophy, aimed at research on 5G services and 5G resource management. We have presented validation results that show near-perfect compliance with 3GPP requirements. We have shown that Simu5G can be used to evaluate technologies currently at the cutting edge of research, namely MEC and platooning with C-V2X, with little setup effort on the user.

Future work on this topic includes thoroughly investigating Simu5G's capabilities as a real-time emulator. Some preliminary results of this line of research have been presented in [11], and show that Simu5G can be used by application programmers to test their applications and services in a controllable, yet lifelike environment, possibly in conjunction with rapid-prototyping tools, such as the Intel OpenNESS framework for MEC hosting [39], with which Simu5G is perfectly integrated.

ACKNOWLEDGMENT

A preliminary version of this article was presented at the INSTICC SIMULTECH 2020, Paris, France, 8–10 July 2020. The subject matter of this article includes description of results of a joint research project carried out by Intel Corporation and the University of Pisa. Intel Corporation reserves all proprietary rights in any process, procedure, algorithm, article of manufacture, or other results of said project herein described.

REFERENCES

- [1] G. Nardini, G. Stea, A. Virdis, and D. Sabella, "Simu5G: A system-level simulator for 5G networks," in *Proc. 10th Int. Conf. Simulation Modeling Methodologies, Technol. Appl.*, Paris, France, 2020, pp. 68–80, doi: 10.5220/0009826400680080.
- [2] *Simu5G: Simulator for 5G New Radio Networks*. Accessed: Oct. 2020. [Online]. Available: <http://simu5g.org>
- [3] Y. Kim, J. Bae, J. Lim, E. Park, J. Baek, S. I. Han, C. Chu, and Y. Han, "5G K-simulator: 5G system simulator for performance evaluation," in *Proc. IEEE Int. Symp. Dyn. Spectr. Access Netw. (DySPAN)*, Oct. 2018, pp. 1–2, doi: 10.1109/DySPAN.2018.8610404.

- [4] M. K. Müller, F. Ademaj, T. Dittrich, A. Fastenbauer, B. R. Elbal, A. Nabavi, L. Nagel, S. Schwarz, and M. Rupp, "Flexible multi-node simulation of cellular mobile communications: The Vienna 5G system level simulator," *EURASIP J. Wireless Commun. Netw.*, vol. 2018, no. 1, p. 227, Dec. 2018, doi: [10.1186/s13638-018-1238-7](https://doi.org/10.1186/s13638-018-1238-7).
- [5] N. Patriciello, S. Lagen, B. Bojovic, and L. Giupponi, "An E2E simulator for 5G NR networks," *Simul. Model. Pract. Theory*, vol. 96, Nov. 2019, Art. no. 101933, doi: [10.1016/j.simpat.2019.101933](https://doi.org/10.1016/j.simpat.2019.101933).
- [6] S. Martiradonna, A. Grassi, G. Piro, and G. Boggia, "5G-air-simulator: An open-source tool modeling the 5G air interface," *Comput. Netw.*, vol. 173, May 2020, Art. no. 107151.
- [7] N. Baldo, M. Miozzo, M. Requena-Esteso, and J. Nin-Guerrero, "An open source product-oriented LTE network simulator based on ns-3," in *Proc. 14th ACM Int. Conf. Modeling, Anal. Simulation Wireless Mobile Syst. (MSWiM)*, Miami FL, USA, 2011, pp. 293–298.
- [8] M. Mezzavilla, M. Zhang, M. Polese, R. Ford, S. Dutta, S. Rangan, M. Zorzi, "End-to-end simulation of 5G mmWave networks," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2237–2263, 3rd Quart., 2018.
- [9] L. F. Perrone, C. Cicconetti, G. Stea, and B. C. Ward, "On the automation of computer network simulators," in *Proc. 2nd Int. ICST Conf. Simulation Tools Techn.*, Rome, Italy, 2009, pp. 1–10.
- [10] S. Kurkowski, T. Camp, and M. Colagrosso, "MANET simulation studies: The incredibles," *ACM SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 9, no. 4, pp. 50–61, Oct. 2005.
- [11] G. Nardini, G. Stea, A. Virdis, D. Sabella, and P. Thakkar, "Using Simu5G as a real-time network emulator to test MEC Apps in an end-to-end 5G testbed," in *Proc. IEEE PIMRC*, London, U.K., Sep. 2020, pp. 1–3.
- [12] A. Virdis, G. Stea, and G. Nardini, "Simulating LTE/LTE-advanced networks with SimuLTE," in *Simulation and Modeling Methodologies, Technologies and Applications (Advances in Intelligent Systems and Computing)*, vol. 402. Cham, Switzerland: Springer, 2016, pp. 83–105, doi: [10.1007/978-3-319-26470-7_5](https://doi.org/10.1007/978-3-319-26470-7_5).
- [13] Accessed: Jul. 2020. *SimuLTE*. [Online]. Available: <http://simulte.com>
- [14] Accessed: Jul. 2020. *OMNeT++*. [Online]. Available: <https://omnetpp.org>
- [15] Accessed: Jul. 2020. *INET Library*. [Online]. Available: <https://inet.omnetpp.org>
- [16] G. Nardini, A. Virdis, and G. Stea, "Modeling network-controlled device-to-device communications in SimuLTE," *MDPI Sensors*, vol. 18, no. 10, p. 3551, 2018, doi: [10.3390/s18103551](https://doi.org/10.3390/s18103551).
- [17] C. Sommer, R. German, and F. Dressler, "Bidirectionally coupled network and road traffic simulation for improved IVC analysis," *IEEE Trans. Mobile Comput.*, vol. 10, no. 1, pp. 3–15, Jan. 2011.
- [18] B. Sliwa and C. Wietfeld, "LIMoSim: A framework for lightweight simulation of vehicular mobility in intelligent transportation systems," in *Recent Advances in Network Simulation: The OMNeT++ Environment and Its Ecosystem*, A. Virdis and M. Kirsche, Eds. Cham, Switzerland: Springer, May 2019, pp. 183–214.
- [19] *Study on 3D Channel Model for LTE*, document TR 36.873 v12.7.0, 3GPP, Dec. 2017.
- [20] K. Bakowski, M. Rodziewicz, and P. Sroka, "System-level simulations of selected aspects of 5G cellular networks," in *Proc. Int. Symp. Wireless Commun. Syst. (ISWCS)*, Aug. 2015, pp. 711–715, doi: [10.1109/ISWCS.2015.7454442](https://doi.org/10.1109/ISWCS.2015.7454442).
- [21] X. Wang, Y. Chen, and Z. Mai, "A novel design of system level simulator for heterogeneous networks," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2017, pp. 1–6, doi: [10.1109/GLOCOMW.2017.8269059](https://doi.org/10.1109/GLOCOMW.2017.8269059).
- [22] N. Mohsen and K. S. Hassan, "C-RAN simulator: A tool for evaluating 5G cloud-based networks system-level performance," in *Proc. IEEE 11th Int. Conf. Wireless Mobile Comput., Netw. Commun. (WiMob)*, Oct. 2015, pp. 302–309, doi: [10.1109/WIMOB.2015.7347976](https://doi.org/10.1109/WIMOB.2015.7347976).
- [23] M. Liu, P. Ren, Q. Du, W. Ou, X. Xiong, and G. Li, "Design of system-level simulation platform for 5G networks," in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC)*, Jul. 2016, pp. 1–6, doi: [10.1109/ICCCChina.2016.7636796](https://doi.org/10.1109/ICCCChina.2016.7636796).
- [24] A. Law and W. D. Kelton, *Simulation Modeling and Analysis*. New York, NY, USA: McGraw-Hill, 2000.
- [25] *Summary of Calibration Results for IMT-2020 Self Evaluation*, document RP-180524, Huawei, 3GPP TSG RAN Meeting #79, Chennai, India, Mar. 2018.
- [26] B. van Arem, C. J. G. van Driel, and R. Visser, "The impact of cooperative adaptive cruise control on traffic-flow characteristics," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 4, pp. 429–436, Dec. 2006, doi: [10.1109/TITS.2006.884615](https://doi.org/10.1109/TITS.2006.884615).
- [27] *Study on New Radio Access Technology: Radio Access Architecture and Interfaces (Release 14)*, document TR 38.801 v14.0.0, Mar. 2017.
- [28] *NR; NR; Physical Layer Procedures for Control (Release 16)*, document TR 38.213 v16.0.0, 3GPP, Jan. 2020.
- [29] *NR; Physical Layer Procedures for Data (Release 16)*, document TR 38.214 v16.0.0, 3GPP, Jan. 2020.
- [30] *NR; NR and NG-RAN Overall Description; Stage 2 (Release 16)*, document TR 38.300, 3GPP, Dec. 2019.
- [31] *Study on LTE Device to Device Proximity Services: Radio Aspects (Release 12)*, document TS 36.843 v12.0.1, 3GPP, Mar. 2014.
- [32] G. Nardini, G. Stea, and A. Virdis, "A scalable data-plane architecture for one-to-one device-to-device communications in LTE-advanced," *Comput. Netw.*, vol. 131, pp. 77–95, Feb. 2018, doi: [10.1016/j.comnet.2017.12.006](https://doi.org/10.1016/j.comnet.2017.12.006).
- [33] *Multi-access Edge Computing (MEC): Framework and Reference Architecture*, document [33] ETSI GS MEC 003 v2.1.1, Jan. 2019.
- [34] M. Emar, M. C. Filippou, and D. Sabella, "MEC-assisted end-to-end latency evaluations for C-V2X communications," in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, Ljubljana, Slovenia, Jun. 2018, pp. 1–9.
- [35] T. Taleb, A. Ksentini, and P. A. Frangoudis, "Follow-me cloud: When cloud services follow mobile users," *IEEE Trans. Cloud Comput.*, vol. 7, no. 2, pp. 369–382, Apr. 2019.
- [36] G. Naik, B. Choudhury, and J.-M. Park, "IEEE 802.11 bd & 5G NR V2X: Evolution of radio access technologies for V2X communications," *IEEE Access*, vol. 7, pp. 70169–70184, 2019, doi: [10.1109/ACCESS.2019.2919489](https://doi.org/10.1109/ACCESS.2019.2919489).
- [37] C. C. Sommer, D. Eckhoff, A. Brummer, D. S. Buse, F. Hagenauer, S. Joerer, and M. Segata, "Veins: The open source vehicular network simulation framework," in *Recent Advances in Network Simulation: The OMNeT++ Environment and its Ecosystem (EAI/Springer Innovations in Communication and Computing)*, A. Virdis and M. Kirsche, Eds. Cham, Switzerland: Springer, 2019.
- [38] R. Riebl, C. Obermaier, and H. Günther, "Artery: Large scale simulation environment for its applications," in *Recent Advances in Network Simulation: The OMNeT++ Environment and its Ecosystem (EAI/Springer Innovations in Communication and Computing)*, A. Virdis and M. Kirsche, Eds. Cham, Switzerland: Springer, 2019.
- [39] Intel Corporation. *OpenNESS: Enabling Services and Applications on the Network and On-Premise Edge*. Accessed Jul. 2020. [Online]. Available: <https://www.openness.org/templates/openness/images/the-network-and-onpremise-edge-new.pdf>



GIOVANNI NARDINI received the M.Sc. degree in computer systems engineering (*summa cum laude*) and the Ph.D. degree in information engineering from the University of Pisa, in 2013 and 2017, respectively. He is currently an Assistant Professor with the Department of Information Engineering, University of Pisa. His research interests include resource allocation algorithms for Quality of Service in 4G and 5G networks, multi-access edge computing, simulation, and performance evaluation of computer networks. In these fields, he has coauthored six patents and more than 30 peer-reviewed articles. He has been involved in EU-funded and industrial research projects.



DARIO SABELLA is currently working with INTEL as a Senior Manager Standards and Research, driving new technologies and edge cloud innovation for advanced systems, involved in ecosystem engagement and coordinating internal alignment on edge computing across standards and industry groups. In 2019, he was appointed as the Vice-Chairman of ETSI Multi-access Edge Computing (MEC), previously Lead of Industry Groups, and since 2015, he has been the Vice-Chair of IEG WG. Since 2017, he has been delegate of 5G Automotive Association (5GAA). Before 2017, he worked with the Telecom Italia Group (TIM), as responsible in various research, experimental and operational activities on OFDMA technologies (WiMAX, LTE, 5G). He is the author of several publications more than 40 and patents more than 20 in the field of wireless communications, energy efficiency, and edge computing. He has also organized several international workshops and conferences.



GIOVANNI STEA (Member, IEEE) received the Ph.D. degree from the University of Pisa, Italy, in 2003. He is currently an Associate Professor with the Department of Information Engineering, University of Pisa. His current research interests include quality of service and resource allocation in wireline and wireless networks, performance evaluation through simulation, and analytical techniques, traffic engineering. In these fields, he has coauthored more than 100 peer-reviewed articles and 16 patents. He has been involved in national and European research projects, and he has led joint research projects with industrial partners. He has served as a member of the technical and/or organization committees for several international conferences, including SIGCOMM, WoWMoM, and VALUETOOLS, and he is serving on the editorial board of the *Wireless Networks Journal*. He is a Fellow of the European Alliance for Innovation.



ANTONIO VIRDIS received the M.Sc. degree in computer system engineering and the Ph.D. degree in information engineering from the University of Pisa, in 2011 and 2015, respectively. He is currently an Assistant Professor with the University of Pisa. His research interests include quality of service, scheduling and resource allocation in wireless networks, network simulation, and performance evaluation. He has been involved in national, EU-funded and industry-funded research projects. He has coauthored seven patents and more than 40 peer-reviewed articles.

...



PURVI THAKKAR received the Computer Engineering degree. She is currently a Product Manager with Intel and is part of the Edge Computing and Ecosystem Enablement Team. She manages developer outreach activities of OpenNESS—a MEC solution and an enabler of edge services on Intel Architecture (IA). She also leads the partner and customer enablement activities for accelerating the provisioning of edge services on IA with OpenNESS. She is also a management professional trained through executive program in business management. She has a versatile experience across verticals and functions gathered through 20 years of experience.