Py5cheSim: a 5G Multi-Slice Cell Capacity Simulator

Gabriela Pereyra, Claudina Rattaro and Pablo Belzarena
Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
Email: pereyra.gab@gmail.com; {crattaro,belza}@fing.edu.uy

***Abstract*—The fifth generation of mobile communications (5G) is the new 3GPP technology designed to solve a wide range of requirements. On the one hand, it must be able to support high bit rates and ultra-low latency services, and on the other hand, it should be able to connect a massive amount of devices with loose bandwidth and delay requirements. In this context, as scheduling is always a delicate vendor topic and there are not so many free and complete simulation tools to support all 5G features, in this paper we present Py5cheSim. Py5cheSim is a flexible and open-source simulator based on Python and specially oriented to simulate cell capacity in 3GPP 5G networks and beyond. To the best of our knowledge, Py5cheSim is the first simulator that supports Network Slicing at the Radio Access Network (RAN), one of the main innovations of 5G. The present work describes its design and implementation choices and the principal validation results. Finally, as another contribution, we present an exhaustive analysis of the existing available simulation tools highlighting the novelty of Py5cheSim comparing with the others existing simulation software for 5G.**

***Index Terms*—5G, Network Slicing, simulator.**

I. Introduction

The services supported by the 5th Generation (5G) fall under three categories formulated by ITU-R, i.e., enhanced Mobile BroadBand (eMBB), massive Machine-Type Communications (mMTC) and Ultra-Reliable Low-Latency Communications (URLLC). The different types of services to be supported have vastly heterogeneous traffic characteristics, quality of service requirements and even energy consumption associated [1]. First, eMBB traffic is a direct extension of the 4G broadband service that focuses on a higher data rate (20 and 10 Gbits/s downlink and uplink peak data rates, respectively), with a large payload and prolonged internet connectivity based applications. Second, mMTC focuses on uplink communications of massive low rate devices (connection density about $1,000,000$ devices per km2). Finally, URLLC services target mission-critical communications such as autonomous vehicles, tactile internet or remote surgery. Their main requirements are ultra-high reliability with a PER around 10^{-5} and low latency (1 ms).

3GPP has standardized 5G in two phases. The first phase mainly oriented to eMBB and URLLC services, covered in Release 15, and the second to enhance URLLC services and develop mMTC's ones, covered in the subsequent 3GPP Releases. 5G is designed based on an Orthogonal Frequency Division Multiplexing (OFDM) physical layer, as LTE (3GPP 4G networks). The main difference with LTE is supporting different subcarrier spacing (variable numerology) as a way to have enough flexibility to handle the 5G different services [2], [3]. In order to increase spectral efficiency, 5G adds the use of new parts of the spectrum: the millimeter Waves. The standard provides several bands of frequencies above 6 GHz for NR (New Radio) TDD use. In addition, the standard proposes analog beamforming to improve coverage and introduce high subcarrier space with low TTI (Transmission Time Interval) for lower delays. 5G also adds different features like Carrier Aggregation and massive Multiple Input Multiple Output (MIMO) to increase User Equipment (UE) and cell throughput. One of the new 5G features enabled by Stand Alone operation mode is Network Slicing. The basic idea behind this feature is to allocate network resources to different Network Slices, which act as virtual or logical networks with relative independence between each other. Each Slice is configured according to the service it provides. The Slice is defined end to end, so to support Network Slicing, RAN, Core and Transport Network must be prepared. At transport networks, Software Defined Network (SDN) will be an enabler for Network Slicing, and Network Function Virtualization (NFV) will be used at the Core Network. At the RAN level, the resources allocated to the different slices will be the spectrum, i. e. PRBs (Physical Resource Blocks) in the used band. In this context, different Slices will have different configurations in terms of numerology and features supported as mobility and access random procedures, which will directly impact the slice capacity and performance for the required traffic profile.

A vast bibliography exists on 5G and is composed of research articles [4], vendor white papers and web sites presentations. However, as scheduling is always a delicate vendor topic, few free tools simulate cell capacity in 3GPP networks. Existing tools represent only a selected set of features presented in the 5G standard. Most importantly, none of the existing known simulators have the specific features and flexibility needed to implement and evaluate a complete cell capacity analysis. Even more, no one implements Network Slicing at the RAN level.

Network simulators often implement layer by layer most of the procedures described in the 3GPP standard, so the simulation turns hard to configure and implies high processor

loads. However, most network simulators have a wide range of configurable options giving an excellent reference to compare at the time of validation. On the other hand, System-Level Simulators could be a good option. However, this type of simulators often has a high degree of simplification to cover a wide range of cells with affordable resource use. To tackle these problems, we have developed a new simulator in Python platform, named Py5cheSim (we presented primary ideas of Py5chesim in [5]). The general design goal of the developed simulator is to keep it as simple as possible, trying to be as flexible as possible for scheduler implementation. In addition, the goal of Py5cheSim is to build a specific tool for simulating cell capacity in a 5G network for Frequency Division Duplex (FDD) and Time Division Duplex (TDD) operation, including different types of schedulers for the different slices that 5G Networks can handle. Py5cheSim allows analyzing inter and intra-slice scheduling. Also, as there is no need to implement layer by layer all the procedures defined in the standard, this new simulator is more straightforward, lighter, and quicker than many of the existing free tools. Furthermore, but not less important, as Python offers a vast choice of libraries for Artificial Intelligence (AI) development, Py5cheSim allows to implement easily and test AI-based algorithms.

The rest of the paper is structured as follows. We start in Section II giving a complete overview of the related works in 5G simulation tools. In Section III we briefly describe Py5cheSim characteristics and its architecture. Then, in Section IV we present the validation results and some examples of usage in realistic 5G scenarios considering Network Slicing scheduling. Part of the validation test consists of comparing the performance of Py5cheSim with a reference simulation tool. Finally, Section V discusses our roadmap and plans and concludes the work.

II. Related Work

Different research groups have developed simulators targeting 5G network characteristics in the last years, being 5G-LENA and Vienna the most popular ones. Other examples, considering only software that is openly available for academic purposes, are 5G-K-Sim, Simu5G, SyntheticNET, and OpenAirInterface.

5G-LENA [6] is a GPLv2 simulator designed as a pluggable module to ns-3. It is strongly based on lte-LENA and mmWaves modules [7], [8]. Ns-3 is a network simulator, very rich in terms of technologies supported, and particularly the 5G-LENA module provides several parameter configuration options making it an excellent choice for different scenario simulations. These modules present some disadvantages: the high degree of complexity and processing capacity needed to configure and run a simulation. The first is not a problem if one is a C++ developer and has experience with ns-3, but the second is unavoidable because of the simulator's nature. These modules implement layer by layer most of the procedures described by the 3GPP standard, so a simple 10 minutes simulation with high bandwidth and several users can take hours in a standard PC. Additionally, although 5G-LENA supports many NR features, the current version of this module (NRv1.1 available since March 2021) does not implement mini-slots and network slicing scheduling. It is important to mention that in part of our simulator validation process, we use 5G-LENA as the basis (in particular in everything related to intra-slice scheduler module: MCS (Modulation and Coding Scheme), BER (Bit Error Rate), SINR (Signal-to-Interference-plus-Noise Ratio), TBS (Transport Block Size), and throughput calculation or generation).

Concerning system-level simulators, one piece of software that stands out is the Vienna Simulator [9], [10]. This MATLAB tool, which is available for download under an academic use license, permits link-level and system-level simulations. It is based on its predecessor Vienna LTE simulator. The latest version of Viena available is of 2020. This version does not support key NR features like mini-slot scheduling, network slicing, mmWave propagation models, and 256-QAM modulation. Also, the simulator does not include the possibility to perform an uplink simulation and use non-full buffer traffic model.

Simu5G [11], [12], based on OMNeT++ framework written in C++, is also categorized as a system-level simulator. Simu5G simulates the data plane of the 5G RAN (rel. 16) and core network. It supports many interesting features that are not present in others (e.g. FDD and TDD modes, dual connectivity, carrier aggregation, different numerologies). However, according to its last version 1.1.0, it does not simulate all possible features in relation to resource allocation like network slicing or mini-slot, being essential for URLLC traffic. Unlike Vienna, which is well tailored for the evaluation of lower-layer procedures, including signal-processing techniques, Simu5G as Py5cheSim is a discrete-event, application-level simulator.

5G-K-Sim [13], [14] is a complete C++ tool that includes link-level, system-level, and network-level simulations. Its network version has a SDN/NFV module, which is an essential function for the network slicing technology. However, it does not support end-to-end network slicing capabilities (in particular, RAN-slicing). 5G-K-Sim was developed at the earliest stages of the 5G standardization process and is non-fully standard compliant.

SyntheticNET [15] is a Python simulator that is focused on modeling a realistic handover process, including a realistic urban mobility module. The authors of SyntheticNET said that a free version of SyntheticNET would be available soon for academic purposes. It supports different numerologies and mmWaves, but the authors do not specify what other 5G features it supports.

Finally, it is important to mention OpenAirInterface [16], [17]. It consists of open source software running on general-purpose processors. This development supports many of the NR specifications. Its main limitation the ability to scale simulations up to large networks, but it represents an interesting tool to validate new proposals in real testbeds.

III. System Design

In this section, we present a brief description of our simulator. First, we present the simulator characteristics and features, and at the end of the section, we explain the simulator architecture.

A. Main simulator characteristics

The general design goal for Py5cheSim was to keep it as simple as possible, trying to maintain the biggest freedom degree possible when it comes to scheduler implementation. Python was used to develop the simulator. Python is a powerful and versatile language, provided with tons of packages developed for specific purposes, from discrete event simulation to machine learning tools. The tool used to implement discrete event simulation was SimPy [18].

Py5cheSim implements RAN Slicing as a core feature using a two-level scheduler composed of an Intra Slice Scheduler and an Inter Slice Scheduler. The first one is oriented to solve resource allocation between different UEs of the same Slice, and the second to allocate resources between the different Slices. Each Slice has a set of requirements and a configuration. Configuration is set automatically depending on Slice requirements in terms of delay, band, the number of UEs to serve, traffic profile, UE capabilities, and availability. For each Slice, numerology/SCS (Sub-carrier Spacing)/TTI, duplexing mode, scheduler algorithm to use, signaling load, and allocated PRBs are set at the initializing of the simulator. Slice allocated PRBs can change according to Inter Slice scheduler decision with a TTI granularity.

Py5cheSim supports multiple numerologies, FDD and TDD frame (depending on the cell band set for the simulation), uplink and downlink bearers, and a basic implementation of Carrier Aggregation and Single-User/Multi-User MIMO functionalities. Transport Block Size (TBS) calculation, which depends on both the number of allocated PRBs and the MCS, is based on 3GPP Technical Specifications [19], [20]. At the moment, MCS allocation is based purely on UE's SINR. An SINR-MCS table was generated from 5G-LENA for a wide range of SINRs being an input of Py5cheSim. However, different MCS allocation algorithm could be implemented overwriting the *setMod* method (see IntraSliceSche in Figure 2).

Py5cheSim also supports different traffic profiles configuration by groups of UEs that can emulate the different 5G services (eMBB, URLLC and mMTC). The traffic profile is set in terms of average packet size (in bytes) and inter-arrival times (in ms).

B. Architecture

The Simulator is built on the modules of Figures 1 and 2. UE, Cell, IntraSliceSch and Slice are the simulation core. IntraSliceSch and Cell have implemented the basic schedulers for one and several slices, respectively. The default scheduling algorithm in these classes is Round Robin. Other schedulers must be defined as classes inherited from the Base Scheduler's ones defined in the former modules overwriting the *resAlloc* method. Three possible scheduling algorithms for interslice scheduling are actually supported by Py5cheSim (Round Robin by default, and Proportional Fair and a modified version of Round Robin as examples). In terms of intraslice scheduler, for FDD simulations resource allocation for different UEs here is done in terms of PRB, then the next scheduling algorithms are actually supported: Round Robin and Proportional Fair. For TDD simulations resource allocation is done in a TTI granularity along the entire band. Only Round Robin TDD scheduler is supported at the moment. Other algorithms can be added as new classes inherited from TDD Scheduler class.

Two classes have been developed to support inter Slice Scheduling: InterSliceSch and Slice (the yellow ones in Figure 2). The first one implements the interslice scheduler, as it dynamically allocates band PRBs between the different configured Slices. The second one manages Slices requirements and translates to Slice configuration (each Slice is associated with an instance of intraslice Scheduler). The simulator allows to configure the time granularity for the InterSlice scheduling decision by setting the *granularity* attribute in the interSliceScheduler class.

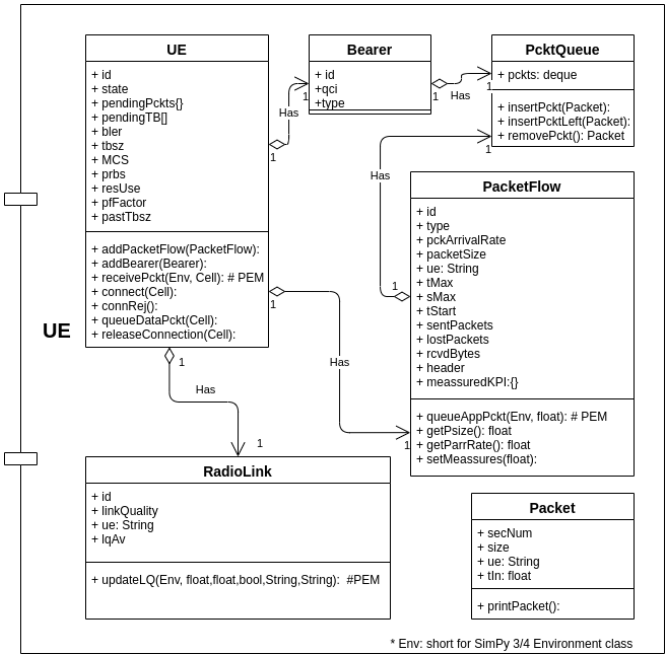

Fig. 1: UE module class diagram. Define UE parameters and traffic. In particular PacketFlow class is responsible for creating different traffic flows.

The simulator basic operation can be seen in the Figure 3. The application generates a packet flow through the *queueAppPckt* method. Each packet is stored in an application queue, the first which appears in Figure 3. Then, when the UE reaches the connected state in the cell, the DRB (Data Radio Bearer) is established and its packets go to the bearer queue through the *receivePckt* method. Then, the scheduler assigns resources for all the active bearers and takes packets from there to make TB (Transport Blocks) with an appropriate MCS according the

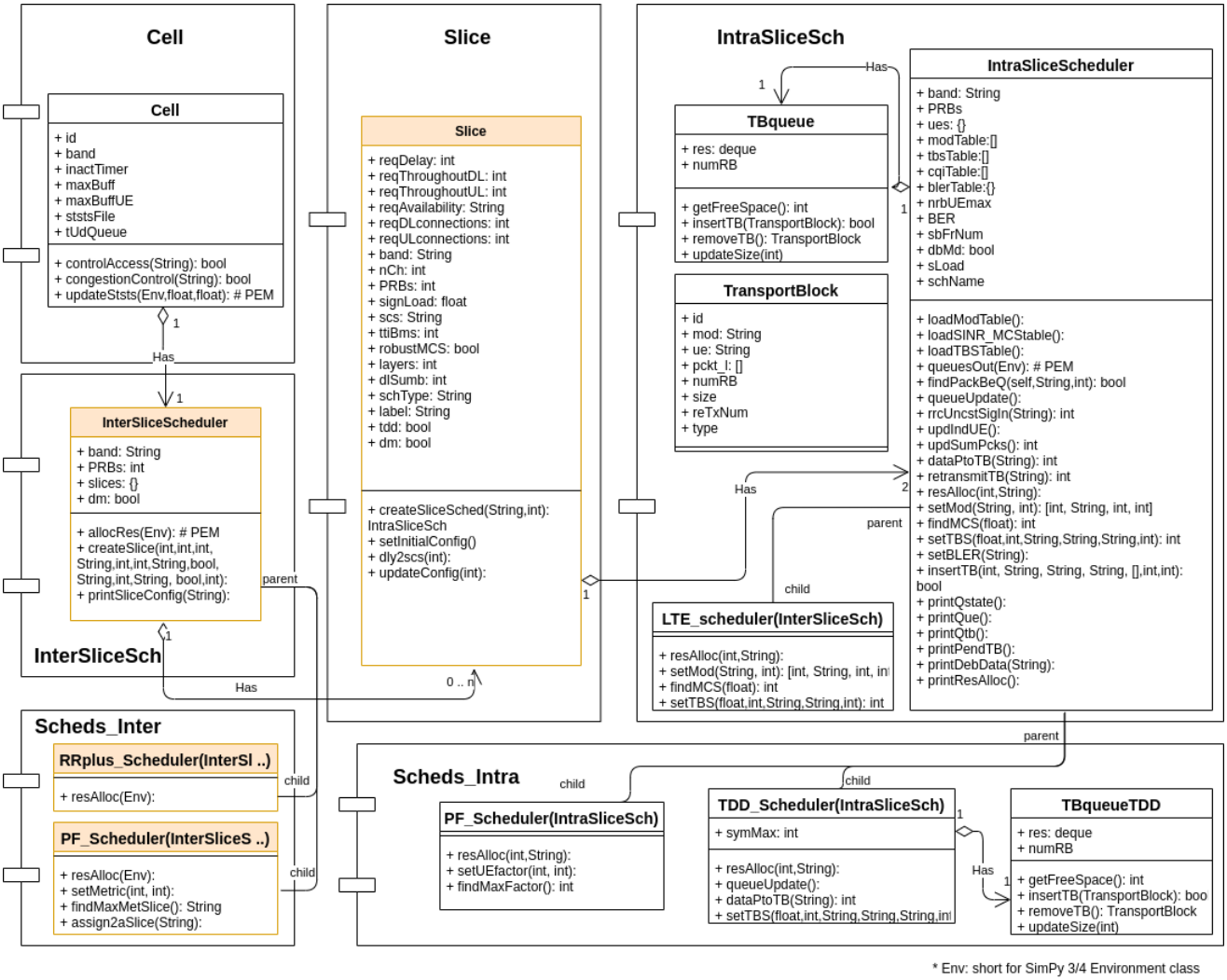

Fig. 2: Cell, Slice, and Schedulers modules class diagrams. Cell module defines cell configuration, statistics management, and base interslice scheduler configuration. On the other hand, IntraSliceSch is the base of intraslice scheduler configuration. Sched_inter and Sched_intra have other inter or intraslice schedulers configuration, respectively.

UE SINR at the moment and put them in the Scheduler queue through the *queueUpdate* method. Finally the scheduler takes the TB from the queue at each TTI and send them through the air interface. The TB are successfully received with a probability of (1-BLER (Block Error Rate)). Please note that BLER can be set overwriting the *setBLER* method.

Naturally, 5G network has been deploying progressively and will coexist for a relatively long time with the existing 4G (LTE/LTE-Advanced) infrastructure. To favor the above transition, we have implemented the LTE_scheduler class, which inherits from IntraSliceSch (see Figure 2). In this simulation environment LTE traffic can be served by a LTE slice in a 5G cell.

Py5cheSim includes a Python script (simulation.py) to configure and run a simulation. It also prints average simulation results in the terminal (see an example in the next Section in Figure 12), and makes charts with the different Key Performance Indicator (KPI) selected to study.

IV. Validation and Results

A partial validation was made through the 5G-LENA module and the throughout calculator web tool from https://5g-tools.com/5g-nr-throughput-calculator/ (the last one represents a quick comparison with known analytical results). We said "partial" because there is no tool to compare ourselves in multi-slice scenarios.

The general idea behind this validation was to test the developed simulator operation and compare performance results with the references in terms of the main KPI considered, using the same configuration scenarios. Py5cheSim AMC (Adaptive Modulation and Coding) and TBS calculation was adjusted to LENA-5G's as much as possible for this purpose. However as Py5cheSim makes a high degree of simplification of NR procedures compared to LENA-5G, an error margin should be expected. The goal is to not exceed a 10% when

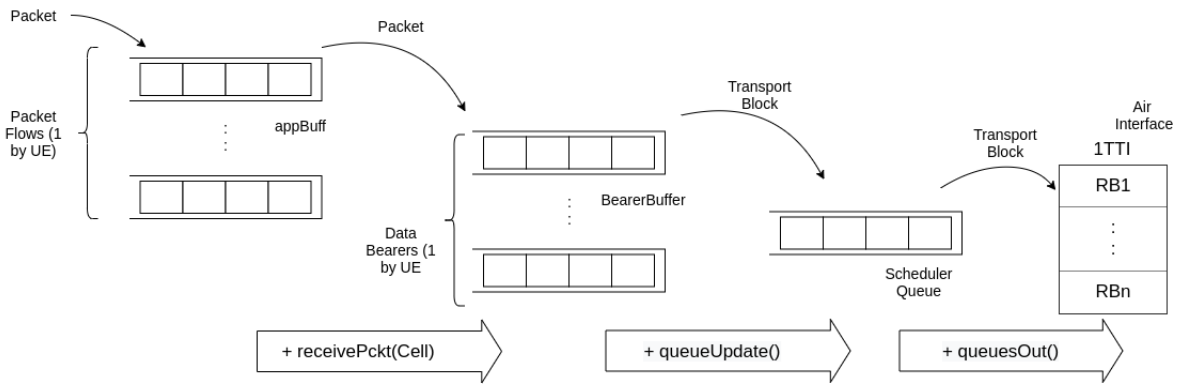

Fig. 3: Intra Slice Scheduler Queues operative.

comparing simulations considering the same configuration parameters. The validation and calibration process has been a comprehensive check verifying in a wide variety of scenarios. Following, we present some representative results.

Firstly, intra-Slice level validation tests are described, and comparison results are shown. We compare MCS allocation for a wide range of SINRs, using the same band and bandwidth, no CA nor MIMO, and only one UE with full buffer DL and UL traffic profile. Figures 4, 5, 6 and 7 show some validation results comparing Py5cheSim and the results of script cttcnr-demo.cc of 5G-LENA.

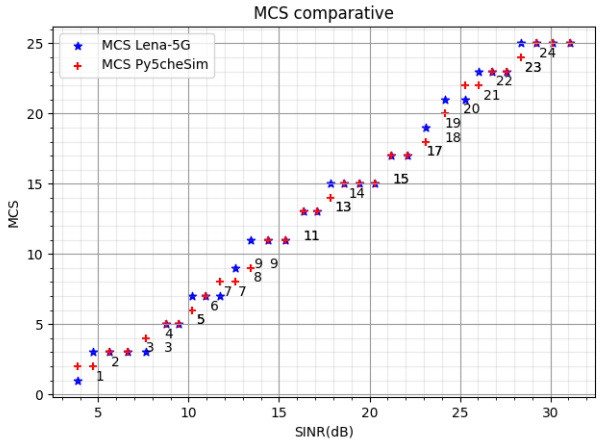

Fig. 4: Py5cheSim vs 5G-LENA MCS (downlink) 10MHz.

Secondly, continuing with intra-Slice level validation, we validate throughput in two ways: comparing the former script results with Py5cheSim's and comparing the latter with the web throughput calculation tool. In Figures 8 and 9 we show some throughput results for a downlink and ulplink scenario (a complementary analysis of the case of Figures 4 and 5). Differences observed respond mainly to different MCS allocated, due to the implementation's high degree of simplification in NR procedures. AMC implementation can be improved rewriting the *setMCS* method. Differences can also be present for the same MCS, due to the differences in the TBS

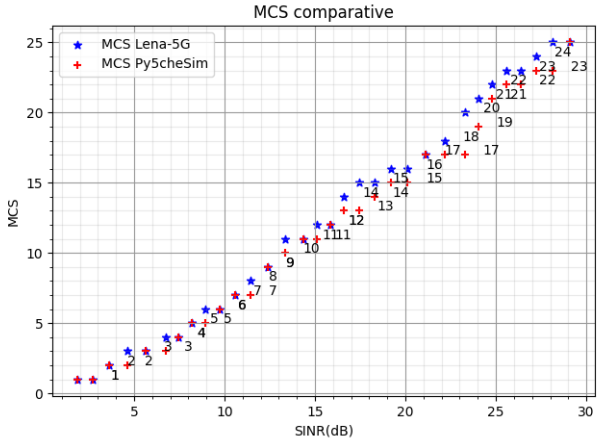

Fig. 5: Py5cheSim vs 5G-LENA MCS (uplink) 10MHz.

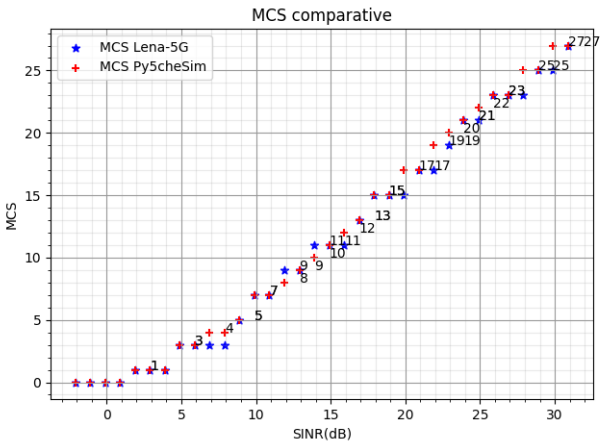

Fig. 6: Py5cheSim vs 5G-LENA MCS (downlink) 100MHz (TDD).

calculation method. However as can be seen in throughput figures, it remains under the error margin considered.

Thirdly, we made scheduling (intra-slice) validation comparing 5G-LENA scripts again results with our simulator. We consider one cell (one slice) and more than one UE with a

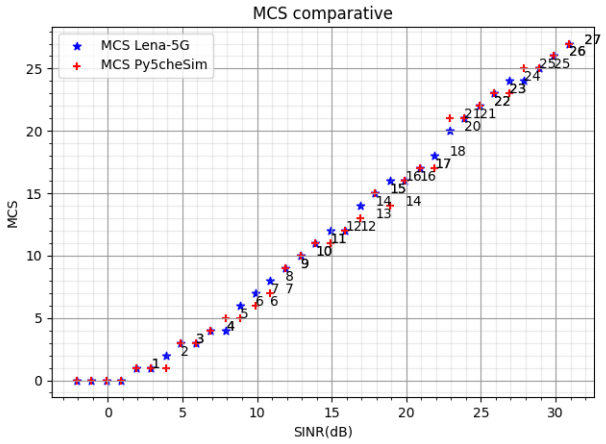

Fig. 7: Py5cheSim vs 5G-LENA MCS (uplink) 100MHz (TDD).

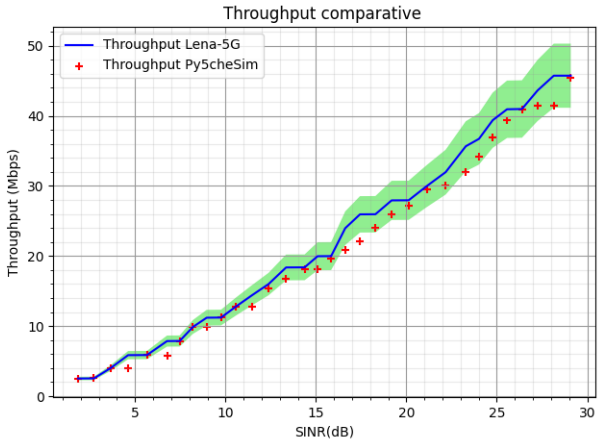

Fig. 9: Throughput of Py5cheSim vs 5G-LENA (uplink) 10MHz

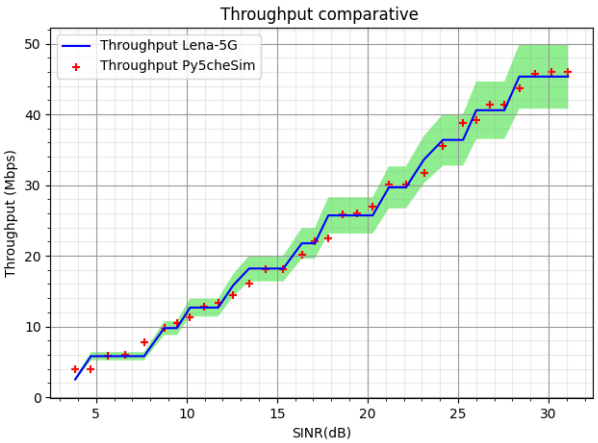

Fig. 8: Throughput of Py5cheSim vs 5G-LENA (downlink) 10MHz

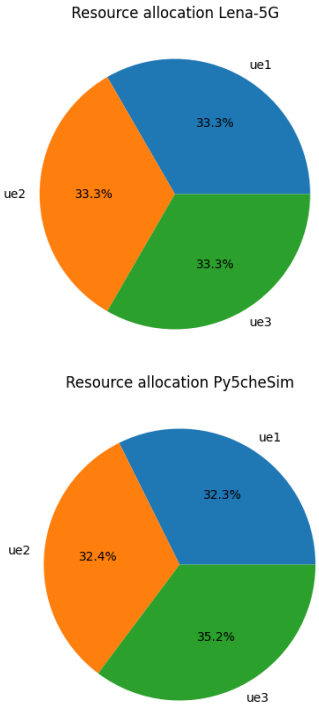

Fig. 10: Py5cheSim vs 5G-LENA FDD PF11 resource use distribution.

full buffer traffic profile. We tested Round Robin (RR) and Proportional Fair (PF) considering different exponent values schedulers. Figure 10 and 11 present representative results; in the first one the three UEs have very different SINR. Above, resource usage for the Proportional Fair of 5G-LENA and below the same but by Py5cheSim using numerator and denominator exponent equal to 1. On the other hand, Figure 11 presents an example using Round Robin scheduler.

Finally, we present a brief description of multi-slice validation. It is important to note that, except of resource allocation, Slice configuration is made at the creation moment, and remains unchanged during the simulation. It is assumed that some service requirements will not change during the simulation. However, as the number of UEs and traffic intensity could change, resource allocation between slices can be updated, with a configurable time granularity.

Table I shows the configured mapping between RAN Delay requirements and SCS configuration for a Slice. Note that the considered thresholds were defined taking into account the delay analysis and results presented in [20] assuming average values for the scheduling timings, and the direct dependence with the slot duration and SCS as a consequence. Different thresholds can be configured modifying the *dly2scs* method in Slice class.

TABLE I: Delay requirement to SCS mapping.

Delay Requirement (ms)	≤ 2.5	≤ 5	≤ 10	> 10
SCS (kHz)	120	60	30	15

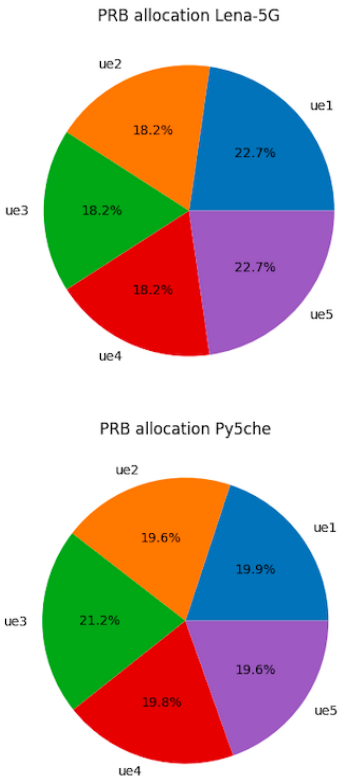

Fig. 11: Py5cheSim vs 5G-LENA RR uplink resource use distribution.

In Figure 12 we show the results printed in a simulation considering two slices in a 5 MHz cell, no MIMO nor CA: one for traffic eMBB and the other for traffic URLLC. Table II shows slice level delay requirements and simulation results. Round Robin scheduler is used for resource allocation between Slices, so for 15 kHz SCS Slice 12 PRB resource allocation is expected from the total of 25. In the URLLC Slice case, as 60 kHz numerology is set according to delay requirements, half band PRB should be 3. In URLLC case, UE throughput respond to traffic profile (set in the simulation script). In eMBB Slice case, as traffic profile is more demanding, so UE throughput matches with the maximum expected for 12 PRB. According to the web throughput calculator tool [21] for a 5 MHz cell on the same conditions (high SINR, no MIMO nor CA, etc.) maximum expected throughput should be 26 Mbps (using 25 PRB). If we estimate the expected througput for 12 PRB under the same condition, the obtenied value here has a difference with the simulation's one lower than 3%.

V. Conclusions and Future Work

This article presented Py5cheSim, a new discrete event Python simulator focused on cell capacity analysis. Py5cheSim constitutes a simple environment to develop and test new 5G scheduler algorithms (inter and intra Slice). We have presented validation results that show near compliance with 3GPP requirements and the reference simulator.

```
------------------------------------------------
                SLICE: eMBB
------------------------------------------------

Accumulated DL indicators by user:
ue1      Sent Packets:59615 Lost Packets:3736
        SINRav: 34 MCSav: 26 PLR: 6.27 % Throughput: 12.15
Average DL Indicators:
Packet Loss Rate av:    6.27 %
Throughput av:  12.15 Mbps
Connections av:         1
Slice Resources:        12 PRBs
Slice Numerology:       15 kHz

------------------------------------------------
                SLICE: URLLC
------------------------------------------------

Accumulated UL indicators by user:
ue1      Sent Packets:1933 Lost Packets:0
        SINRav: 35 MCSav: 27 PLR: 0.0 % Throughput: 0.03
ue2      Sent Packets:2037 Lost Packets:0
        SINRav: 28 MCSav: 23 PLR: 0.0 % Throughput: 0.03
ue3      Sent Packets:1997 Lost Packets:0
        SINRav: 21 MCSav: 17 PLR: 0.0 % Throughput: 0.03
ue4      Sent Packets:2046 Lost Packets:0
        SINRav: 14 MCSav: 11 PLR: 0.0 % Throughput: 0.03
ue5      Sent Packets:2019 Lost Packets:0
        SINRav: 8 MCSav: 5 PLR: 0.0 % Throughput: 0.03
Average UL Indicators:
Packet Loss Rate av:    0.0 %
Throughput av:  0.03 Mbps
Connections av:         5
Slice Resources:        3 PRBs
Slice Numerology:       60 kHz
```

Fig. 12: Example of two slices scenario result.

TABLE II: Two slices main simulation.

Slice Name	eMBB	URLLC
Max Delay (ms)	20	5
Slice PRBs	12	3
Slice Numerology	15kHz	60kHz
Slice Connections	1	5
Avg. UE Throughput (Mbps)	12.15	0.03

In this article, we presented the architecture and the capabilities of Py5cheSim v1.0, with the aim of helping researchers to understand the level of detail and to get a clear idea of its functionalities. This first version of the simulator is available at our web page [22] [1] and we are working on a second version of the simulator improving some features and including others such as: mini-slots support and user mobility possibilities.

VI. Acknowledgements

This work has been supported by the Agencia Nacional de Investigación e Innovación, Uruguay, Project FMV_1_2019_1 _155700, "Artificial Intelligence applied to 5G networks".

References

[1] ITU-R, "Imt vision – framework and overall objectives of the future development of imt for 2020 and beyond," International Telecommunication Union, Geneva, Recommendation M.2083-0, Set 2015.

[2] 3GPP, "NR; NR and NG-RAN Overall description; Stage-2," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.300, 03 2021, version 16.5.0.

[3] ——, "NR; Physical layer; General description," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.201, 01 2020, version 16.0.0.

[1] https://iie.fing.edu.uy/investigacion/grupos/artes/proyectos/inteligencia-artificial-aplicada-a-redes-5g/

[4] M. Shafi, A. F. Molisch, P. J. Smith, T. Haustein, P. Zhu, P. De Silva, F. Tufvesson, A. Benjebbour, and G. Wunder, "5g: A tutorial overview of standards, trials, challenges, deployment, and practice," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 6, pp. 1201–1221, 2017.

[5] G. Pereyra, C. Rattaro, and P. Belzarena, "A 5g multi-slice cell capacity framework," Aug 2021, poster presented at ACM SIGCOMM 2021 Networking Networking Women Professional Development Workshop (N2Women'21).

[6] N. Patriciello, S. Lagen, B. Bojovic, and L. Giupponi, "An e2e simulator for 5g nr networks," *Simulation Modelling Practice and Theory*, vol. 96, p. 101933, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1569190X19300589

[7] "An open source model for the simulation of lte handover scenarios and algorithms in ns-3," p. 289–298, 2013. [Online]. Available: https://doi.org/10.1145/2507924.2507940

[8] M. Mezzavilla, M. Zhang, M. Polese, R. Ford, S. Dutta, S. Rangan, and M. Zorzi, "End-to-end simulation of 5g mmwave networks," *IEEE Communications Surveys Tutorials*, vol. 20, no. 3, pp. 2237–2263, 2018.

[9] 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 Journal on Wireless Communications and Networking*, vol. 2018, no. 1, p. 17, Sep. 2018.

[10] S. Pratschner, B. Tahir, L. Marijanovic, M. Mussbah, K. Kirev, R. Nissel, S. Schwarz, and M. Rupp, "Versatile mobile communications simulation: the Vienna 5G Link Level Simulator," *EURASIP Journal on Wireless Communications and Networking*, vol. 2018, no. 1, p. 226, Sep. 2018.

[11] G. Nardini, G. Stea, A. Virdis, and D. Sabella, "Simu5g: A system-level simulator for 5g networks," 07 2020.

[12] G. Nardini, D. Sabella, G. Stea, P. Thakkar, and A. Virdis, "Simu5g–an omnet++ library for end-to-end performance evaluation of 5g networks," *IEEE Access*, vol. 8, pp. 181 176–181 191, 2020.

[13] 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 *2018 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, 2018, pp. 1–2.

[14] J. Baek, J. Bae, Y. Kim, J. Lim, E. Park, J. Lee, G. Lee, S. I. Han, C. Chu, and Y. Han, "5g k-simulator of flexible, open, modular (fom) structure and web-based 5g k-simplatform," in *2019 16th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 2019, pp. 1–4.

[15] S. M. A. Zaidi, M. Manalastas, H. Farooq, and A. Imran, "Syntheticnet: A 3gpp compliant simulator for ai enabled 5g and beyond," *IEEE Access*, vol. 8, pp. 82 938–82 950, 2020.

[16] F. Kaltenberger, G. d. Souza, R. Knopp, and H. Wang, "The openairinterface 5g new radio implementation: Current status and roadmap," in *WSA 2019; 23rd International ITG Workshop on Smart Antennas*, 2019, pp. 1–5.

[17] F. Kaltenberger, A. P. Silva, A. Gosain, L. Wang, and T.-T. Nguyen, "Openairinterface: Democratizing innovation in the 5g era," *Computer Networks*, vol. 176, p. 107284, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128619314410

[18] T. SimPy, "Documentation for simpy," *https://simpy.readthedocs.io/en/latest/contents.html*, 2020.

[19] 3GPP, "NR; User Equipment (UE) radio access capabilities," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.306, 03 2021, version 16.4.0.

[20] ——, "NR; Physical layer procedures for data," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.214, 03 2021, version 16.5.0.

[21] f. G.-t. Vinogradov Oleg, "5G NR Throughput calculator," *https://5g-tools.com/5g-nr-throughput-calculator/*, 2021.

[22] G. Pereyra, C. Rattaro, and P. Belzarena, "Py5chesim," https://github.com/ClaudinaRattaro/Py5cheSim, 2021.