ChEmiNet: Multi-UE Channel Emulator for 5G SA Networks

Andrei John Chu, Jason Dagami, Nico Quimsing, Jaybie de Guzman, and Neil Irwin Bernardo
Electrical and Electronics Engineering Institute
University of the Philippines Diliman
Quezon City, Philippines
Email: andrei.john.chu@eee.upd.edu.ph, jason.dagami@eee.upd.edu.ph, nico.quimsing@eee.upd.edu.ph, jaybie.de.guzman@eee.upd.edu.ph, neil.bernardo@eee.upd.edu.ph

***Abstract*—Academic institutions and small-scale network operators face limited access to 5G network testing and development due to the restrictive and proprietary nature of existing cellular systems. Open-source cellular stacks are already available for software-based testing, but these platforms currently lack capabilities in emulating realistic network conditions for multiple-user scenarios. Hence, this project developed a multi-UE emulator integrated with statistical channel models to allow UMa, UMi, RMa, and indoor multi-user scenarios. A fully virtualized 5G SA network is implemented within a single machine using Open5GS, srsgNB, and srsUE from the srsRAN Project, with GNU Radio Companion to allow multi-user connections. This stack is integrated with 3GPP-compliant, python-based 5G channel models Sionna, PySim5G, and PyWiCh, where their characteristics and behavior under different topology scenarios are described using performance metrics bitrate, SNR, error vector magnitude, and latency. This low-cost and accessible alternative to proprietary testing environments holds the potential to empower a wider range of distributors by serving as a testbed to improve 5G network performance.**

***Index Terms*—5G Stand Alone Network, Channel Emulator, Sionna, PySim5G, PyWiCh**

I. Introduction

The fifth generation of wireless technology (5G) is promising faster speeds, lower latency, and expanded connectivity to support emerging technologies in enhanced mobile broadband (eMBB), massive machine-type communication (mMTC), and ultra-reliable low-latency communication (URLLC). To support these use cases, 5G must achieve stringent performance targets, such as peak uplink data rates of up to 10 Gbps, ultra-low latencies as low as 1 ms, and ability to handle high user equipment (UE) densities [1]. However, achieving these targets is challenging due to the complexity of wireless environments with fast-changing urban channel conditions and increasing device mobility. Accurate modeling of these environments is therefore critical for effective design, testing, and deployment of improvements in a 5G network.

Deployment strategies further shape the realization of 5G's full potential. Most networks currently operate in non-standalone (NSA) mode, which integrates 5G radio access networks (RAN) with existing 4G core networks. While faster to deploy, NSA lacks full 5G capabilities. Standalone (SA) deployments offer these benefits but face significant rollout challenges, especially in countries like the Philippines, where deployment is only limited to select urban areas.

Existing open-source cellular stacks have enabled software-based testing [2]–[4], but these fall short in emulating more realistic multi-user network conditions. In this paper, we address this gap by presenting a low-cost, configurable multi-UE channel emulator that integrates standardized 3GPP channel models into a fully-softwarized 5G SA network. Specifically, the main contributions of this paper are as follows:

- Present the design of a fully-softwarized 5G SA network in a single host using Open5GS and srsRAN, and support 10 UE connections via GNU Radio Companion.
- Emulate an uplink channel by integrating open-sourced, Python-based 3GPP channel models Sionna, PySim5G, and PyWiCh into the 5G SA platform.
- Assess and characterize the performance of each channel model using bitrate and SNR at uplink, error vector magnitude (EVM), and latency under the following scenarios: urban macrocell (UMa), urban microcell (UMi), rural macrocell (RMa), and indoor office.

This work is, to our knowledge, the first to incorporate Sionna, PySim5G, and PyWiCh channel models into a virtual 5G SA architecture.

The rest of this paper is organized as follows: Section II reviews related work on network and channel emulation. Section III presents the emulator and simulation setup for data collection. Section IV presents performance metrics, comparison, and discussion of results. Lastly, the conclusion and recommendations of the study are presented in Section V.

II. 5G Emulation and Channel Models

Over the years, several 5G virtual cellular platforms have emerged and been compared for their performance. For the 5G core, one study shows that OpenAirInterface (OAI) 5G CN exhibits a significant increase in round-trip time (RTT) as the number of UEs increases, unlike in Free5GC and Open5GS [5]. Free5GC also demonstrates better data plane performance, while Open5GS outperforms in the control plane [6]. Additionally,

J.A. de Guzman acknowledges the PLDT-Smart Foundation and the UP Engineering Research and Development Foundation, Inc. for their support through the PLDT-Smart Professorial Chair Award.

N.I. Bernardo acknowledges the Office of the Chancellor of the University of the Philippines Diliman, through the Office of the Vice Chancellor for Research and Development, for funding support through the PhD Incentive Award Grant 252509 YEAR 1.

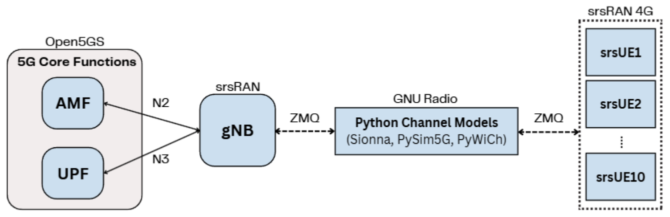


Fig. 1: Block Diagram of the Channel Emulator

OAI 5G CN is reported to have the highest processing load, while Open5GS has the lowest [5].

For the RAN, both srsRAN and OAI provide software radio implementations of full 5G cellular stacks that comply with the 3GPP standards; in contrast, UERANSIM offers only 5G SA gNB and UE functions [3]. srsRAN is widely adopted due to its open-source nature, ease of deployment, and modular implementation on standard desktops, which contrasts with the complexity of OAI [2], [4]. Studies have shown how srsRAN can be implemented in a 5G SA testbed for SDRs [7] and commercial mobile devices [8]. Based on these studies, Open5GS and srsRAN are good software choices for the emulator. These softwares are already tested to work together in multiple testbeds and setups.

Recent studies have tried expanding their testbeds for multi-user connections. In [9], three COTS UE are connected to a 5G SA indoor testbed and showed valuable insights on the interoperability and performance of a 5G SA network. However, this setup lacks scalability to replicate the variability of larger real-world networks. Another study demonstrated a fully-softwarized setup using srsUE with a simulated radio channel based on a constant path loss only [10]. This setup fails to consider key factors such as user mobility, interference, and environmental obstructions, which are critical for emulating 5G networks.

Some emulators have integrated channel effects into 5G cellular stacks. For example. OAI-based studies have incorporated basic path loss model with AWGN only [11]. Simu5G has also integrated channel modelling limited to UMa scenarios and a Jakes fading model [12]. While other efforts focus on channel emulation using dedicated hardware using FPGA [13]. Such designs however can only support a limited number of specifications. In contrast, a fully softwarized setup enables rapid development of new 5G features, such as scheduling algorithms or resource allocation mechanisms, without costly hardware and time-consuming over-the-air testbed prototypes.

While these approaches provide a baseline for channel emulation, more sophisticated models are needed for high-fidelity 5G research. Sionna enables link-level simulations using ray-tracing [14]. PySim5G offers techno-economic modeling of 5G networks by integrating engineering and cost factors with geospatial data [15]. PyWiCh simulates wireless channels with spatial consistency and scattering effects [16]. All three are Python-based and 3GPP-compliant, though each implements 3GPP guidelines differently, potentially yielding varied results. UMa and UMi scenarios are common across all models, enabling reliable comparisons. Additionally, Sionna and PySim5G support RMa scenarios, while PyWiCh includes indoor office environments. Therefore, these three models were selected for integration into the emulator.

III. Methodology

A. Machine Setup

Figure 1 illustrates the overall block diagram of the developed multi-UE channel emulator. This was installed in a workstation equipped with 32GB of DDR5 RAM, 4TB SSD, 8-core/16-thread processor, and NVIDIA RTX 4080 Super GPU, operating in an Ubuntu 22.04.5 LTS. Its architecture consists of a core, RAN, and UE using a dockerized Open5GS, srsgNB from srsRAN Project and srsUE from srsRAN 4G, respectively. This UE is only a prototype 5G UE module which limits the subcarrier spacing to 15kHz and bandwidths of 5, 10, 15, or 20 MHz only. Among these, the study used the 10 MHz bandwidth of the n3 band, having a carrier frequency of 1.8425 GHz uplink and 1.7475 GHz downlink. These UEs were connected via ZMQ sockets and separated via namespaces within the workstation. To allow multiple UE connections, a broker such as GNU Radio was utilized, with a sampling rate of 11.52 MHz. A summary of the configurations used is shown in Table I.A.

B. Channel Model Integration

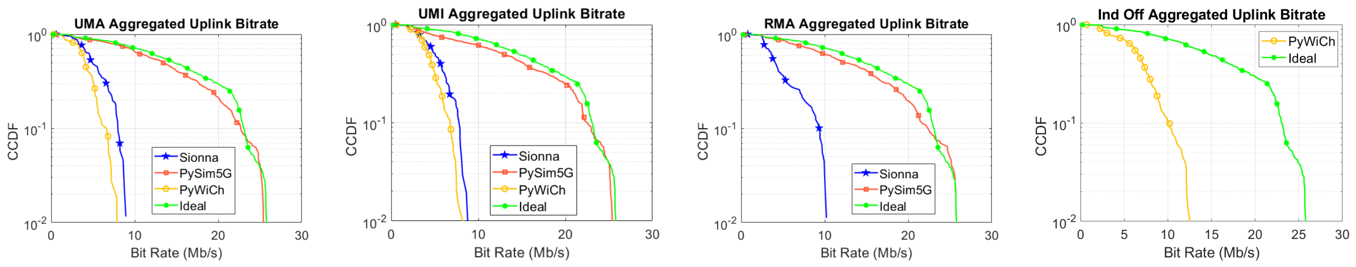
To emulate realistic channel conditions, Python-based channel models, Sionna, PySim5G, and PyWiCh were integrated into GNU Radio using embedded Python blocks. These models implement 3GPP-based scenarios which include UMa, UMi, RMa, and Indoor Office. Due to the limitations of srsUE, only the uplink channel is modeled, and the downlink channel is ideal. Specifically, srsUE is currently restricted to a three-tap filter downlink channel in a single-UE scenario and disconnects in a multi-UE scenario. This constraint is further discussed in Section IV-D. For each channel model, the embedded Python block applies the appropriate path-loss model and, for Sionna and PyWiCh, convolves the generated channel coefficients to the radio samples generated by the UE and gNB. For Sionna, these channel coefficients are time-varying, with a coherence time of 200ms. A

TABLE I: gNB Configuration, and Topology Specifications

A. gNB Configuration	
Band	n3
UL Frequency	1.7475 GHz
DL Frequency	1.8425 GHz
Subcarrier Spacing	15 kHz
Bandwidth	10 MHz
Equalizer	Zero Forcing
TX Gain	70
RX Gain	60
MCS	Adaptive

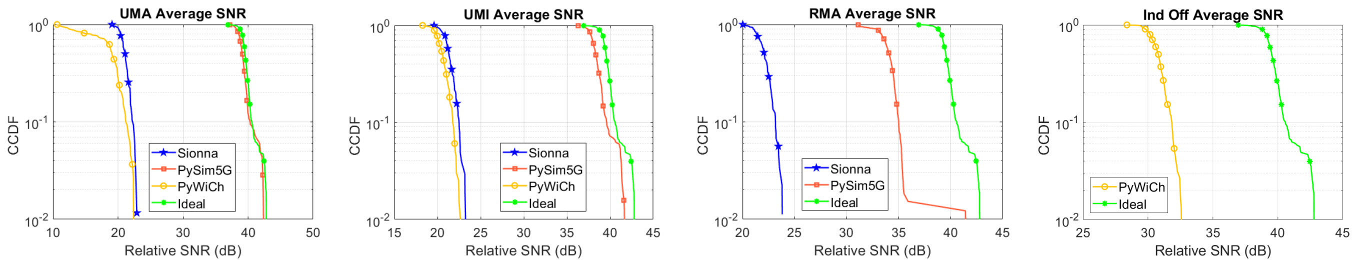
B. Common Topology Specifications		
	Topology 1	Topology 2
Scenario	UMa	UMi - Street Canyon
Grid Size	500 m x 500 m	200 m x 200 m
BS Height	25 m	10 m
UE Height	1.5 m	1.5 m
UE State	outdoors	outdoors
UE Mobility	static	static
LOS/NLOS	NLOS	NLOS

C. Specific Topology Specifications			
	Topology 3		Topology 4
Channel Model	Sionna	PySim5G	PyWiCh
Scenario	RMa		Indoor-Office
Grid Size	5 km x 5 km		120 m x 50 m
BS Height	35 m		3 m
UE Height	1.5 m		1 m
UE State	outdoors		indoors
UE Mobility	static		static
LOS/NLOS	NLOS		NLOS



(a) UMa (b) UMi (c) RMa (d) Indoor Office

Fig. 2: Aggregated Uplink Bitrate of 10 UEs for each Channel Model and Topology



(a) UMa (b) UMi (c) RMa (d) Indoor Office

Fig. 3: Average SNR across 10 UEs for each Channel Model and Topology

summary of shared topology configurations (e.g., grid size, BS height, UE height) can be found in Table I.B, while topologies specific to certain channel models can be found in Table I.C.

C. Experiment Setup

To test these channel models, the study simulated 10 UEs, uniformly distributed across a rectangular grid with dimensions determined by the scenario. While the gNB is fixed at the origin, the 2D coordinates of each UE were generated using MATLAB. Moreover, these positions remain constant for the same scenario across different channel models. During testing, logs from the gNB were processed into a CSV file to generate performance metrics. These metrics include complementary cumulative distribution function (CCDF) of bitrate and uplink SNR, error vector magnitude (EVM), and user plane latency. The CCDF and EVM metrics were derived from the gNB logs under a 400-second bidirectional iperf traffic session. Meanwhile, user plane latency was computed using the average RTT generated by 100 pings in the uplink direction. These results were compared to an ideal scenario wherein the gNB is directly connected to the UE, removing any channel impairments in between.

IV. Results and Discussion

A. Bitrate and SNR

The individual and aggregated uplink bitrate and SNR results for UMa, UMi, RMa, and indoor office scenarios are shown in CCDF curves in Figures 2 and 3. The CCDF of the uplink bitrate and SNR provides insights regarding their statistical properties such that it presents the probability of a random variable being greater than the specified data point.

Figures 2 and 3 shows PyWiCh obtained the lowest valued trend for the CCDF of average SNR and aggregated uplink bitrate in common topology scenarios. These results are consistent with PyWiCh's application of spatial consistency for a more realistic channel model given a densely populated environment such as UMi and UMa. Similarly, Sionna consistently showed low-valued trends for all applicable scenarios. Sionna's library for channel model capabilities provided a realistic physical layer signal transmission which reflects a similar trend with PyWiCh.

As seen in Figures 2 and 3, PySim5G provides a closer-to-ideal channel model with moderate disruption, showing lower uplink bitrates and average SNR. In the RMA scenario, increased distance leads to greater path loss thus more disruption with SNR and uplink bitrate. The CCDF segment where PySim5G outperforms the ideal in uplink bitrate is due to initially high UE bitrates that drop upon congestion.

Overall, all channel models introduced in the uplink transmission affected the bitrate and SNR metrics of each UE and the network as a whole. PySim5G demonstrated how even a simplified path loss implementation can influence performance. Sionna and PyWiCh both showed significantly degraded channel performance due to the addition of small-scale fading, resulting in a more realistic and distorted channel model. The differences in their performance characteristics highlight the trade-offs between model complexity and accuracy in simulating real-world wireless environments.

B. Error Vector Magnitude

Table II presents the average uplink EVM root-mean-square (RMS) across all UEs for each channel model under different scenarios. It can be observed that the EVM values for PySim5G

TABLE II: Average EVM of 10 UEs across different Channel Models and Topologies

	Channel Model		
Scenario	Sionna	PySim5G	PyWiCh
UMa	0.2419	0.036	0.2511
UMi	0.2194	0.036	0.1876
RMa	0.1670	0.034	-
Indoor Office	-	-	0.0677

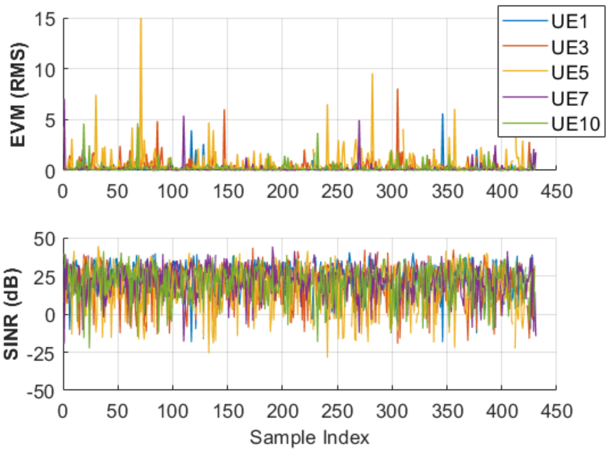
are consistent at approximately 0.036, regardless of the scenario. This arises from PySim5G's path loss only model which does not inherently increase noise power. Furthermore, the pathloss it applies does not reduce the received signal below the noise floor, meaning, it does not significantly affect the bit error rate and EVM [17].

In contrast, both Sionna and PyWiCh include multipath channel impulse responses in their channel models, which introduce inter-symbol interference (ISI) and frequency selective fading, resulting in higher EVM values. Additionally, srsgNB implements a Zero Forcing (ZF) equalizer, which inverts the channel to nullify its effects. However, this inversion can lead to noise amplification, especially in the presence of spectral nulls or deep fades, thus increasing the EVM [18].

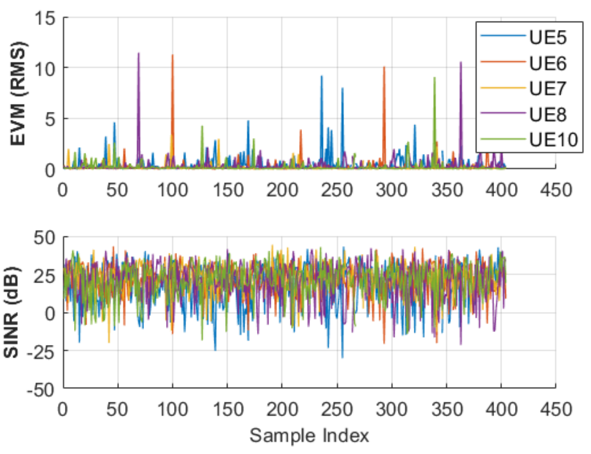
Figures 4 and 5 show the EVM and Signal-to-Interference-plus-Noise-Ratio (SINR) values for selected UEs in UMa and UMi. For clarity, only 5 UEs exhibiting the highest EVM values are shown. For both Sionna and PyWiCh, it can be seen that certain UEs experience SINR values below the 0 dB line, indicating the presence of interference caused by multipath fading. Furthermore, EVM spikes can be noticed, which suggests that the channel contains spectral nulls and deep fades. As shown in Figures 4a and 4b, these spikes occur more frequently in Sionna, which can be attributed to its time-varying nature, in contrast to the time-invariant nature of PyWiCh for stationary UEs.

Although RMa has the highest path loss, it is insufficient to reduce the received signal power below the noise floor. As such, the dominant factor affecting the EVM is the (RMS) delay spread of the channel. Since RMa models a less dense environment compared to UMi and UMa, it has a lower delay spread [19]. This leads to a more frequency-flat channel per subcarrier, reducing the likelihood of spectral nulls, therefore decreasing the chance of noise amplification due to the ZF equalizer. This is also consistent with Indoor Office having an average EVM of only 0.0677. This is expected as the UEs are very close to the gNB and the channel is modeled as having the lowest (RMS) delay spread.

From the results gathered, it suggests that the main factor affecting EVM across scenarios is the channel impulse response (CIR) generated by the channel model. As discussed previously, this is due to the ZF equalizer, which increases the noise floor and introduces deviations from ideal symbol constellations. Furthermore, the CIR introduces multi-path fading which results in ISI, effectively lowering the SINR. Therefore, this suggests that higher EVM values occur at denser scenarios such as UMa and UMi due to higher delay spreads and severe interferences, leading to a more distorted received signal.



(a) UMa



(b) UMi

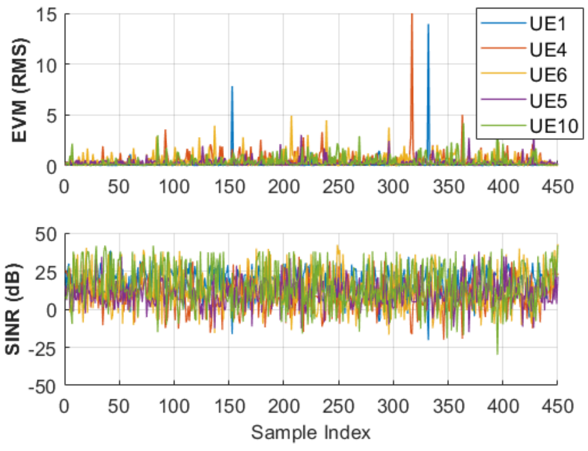
Fig. 4: EVM and SINR Values for Selected UEs in Sionna

C. Latency

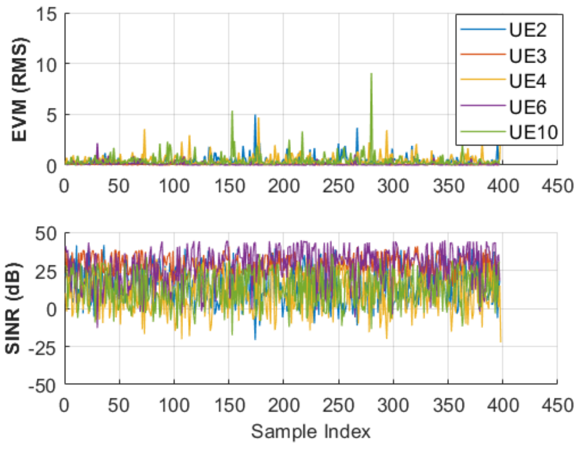
In the simulated topologies, the average distances of the 10 UEs are approximately 39.63 m (indoor), 64.98 m (UMi), 181.77 m (UMa), and 585.38 m (RMa). As shown in Figure 6 for an UMa scenario, the average RTTs per UE do not correlate with distance. The same behavior is observed for all topologies. Although greater separation suggests increased latency due to longer propagation paths, the actual delay introduced is minimal. A 100 m distance adds 333.6 ns only, which is negligible compared to RTTs that span several hundred milliseconds.

But when aggregated, there appears some visible trends. Figure 7 shows that the average RTT in a RMa scenario exhibits around 80 ms more latency than UMa and UMi using Sionna. A similar latency gap is also observed in PyWiCh when comparing UMa/UMi to the indoor scenario. However, differences between UMa and UMi are insignificant, reinforcing that propagation delay alone does not account for the observed latency patterns.

In Pysim5G, latency remains largely consistent across topologies. Its RTT values are also near that of an ideal setup.



(a) UMa



(b) UMi

Fig. 5: EVM and SINR Values for Selected UEs in PyWiCh

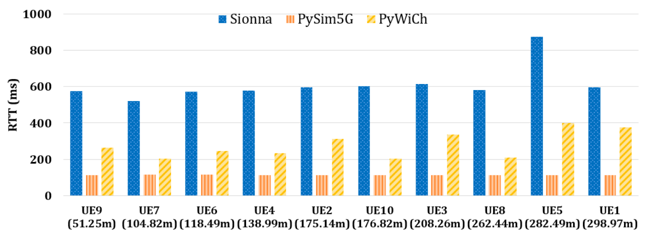


Fig. 6: Average Latency per UE in an UMa Scenario

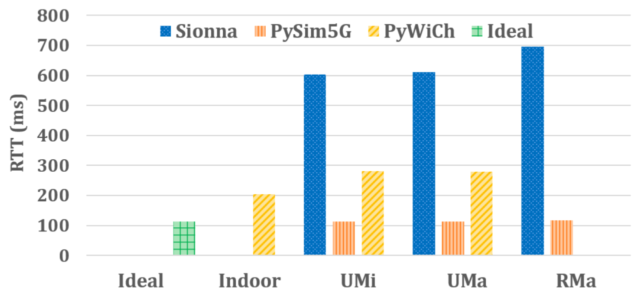


Fig. 7: Latency Averaged across 10 UEs for each Channel Model and Topology

This is expected, as the model applies only path loss without simulating multipath effects or time-varying channels, resulting in minimal processing overhead.

Sionna consistently exhibits the highest latency primarily due to its time-varying behavior, updating filter taps every 200 ms. This time variation introduces additional computational overhead at receiver-side channel estimation. Moreover, repeated filter tap generation and convolution-based signal processing contribute further to the delay. These effects are observed from the noticeably laggy updates in the receiver's frequency spectrum, in contrast to the smoother and more responsive performance in PyWiCh and Pysim5G. In PyWiCh, filter taps are generated only once at initialization, significantly reducing runtime processing complexity and latency.

According to [20], RTT consists of several components, including transmission processing latency, scheduling latency, transmission delay, propagation latency, and reception processing latency. With propagation delays being on the order of hundred of nanoseconds to microsecond, even for UE distances up to 883.18 m, their effect on total latency is minimal as observed in RMa. Thus, the dominant factors influencing latency are related to processing delays, both in signal processing within GNU Radio and in receiver-side computations influenced by the complexity of channel models.

D. Discussion

With the current setup, only the uplink channel is modeled due to limitations of srsUE. As we've observed in single-UE connections, the UE can handle only up to three filter taps without releasing its RRC connection. While for multi-UE setups, assertion failure in the RLC layer appears due to out-of-order segment rearrangement. Looking at the source codes, it seems that channel estimation capabilities of srsUE is only limited to three pilot taps per resource block, compared to the gNB which is more adaptive.

Given the 3GPP compliance of all channel models, their behaviors under different network conditions are varying. This is due to the probabilistic calculation for small-scale parameters in Sionna and PyWiCh. However, each channel model implemented the same path loss computations provided by the 3GPP TR 38.901. Using this commonality, we compared the models under UMa and UMi conditions to verify consistency at least in the large scale parameters. The results showed that the path loss values were largely consistent across all models, with slight variations due to rounding values and the addition of a log-normal distrubtion of PySim5G.

Based on the performance characteristics of each channel model, users can configure the appropriate channel specifications for network testing. PySim5G is suitable for implementing distance and topology-dependent path loss models. When needing the effects of small-scale fading, users can use PyWich for non–time-varying channels and Sionna for time-varying channels. Additionally, the scripts were developed for automated data collection enabling users to characterize the channel conditions. Hence, the characterization can be utilized for assessment when the emulator is tested with third-party or additional algorithms.

V. Conclusion and Recommendations

In this study, a fully softwarized multi-UE channel emulator in a 5G SA network was developed using Open5GS and srsRAN without any proprietary equipment. This study also integrated three open-source implementations of 3GPP based channel models with multiple topology scenarios, which can be used as a tool for the actual deployment of 5G networks. The validation of the channel model emulator can be concluded from the network reliability assessments, such that the channel models provide a source of realistic interference and distortion within the 5G network. Moving on, the authors suggest exploring the use of other UE simulators. Specifically, the developers of srsRAN recommend using the Amarisoft UE. Although not free, this software can be used for experiments that involve a larger number of UE connections, non-ideal downlink channels, non-terrestrial channel models, and more complex traffic patterns.

Finally, with these integrated channel models and scripts for metric generation, the emulator presented could serve as a testbed for 5G network optimization, open RAN deployments, and testing of resource allocation algorithms for RAN intelligent controllers (RIC). This emulator can also function as part of a backend component for a larger network emulator or digital twins similar to Colosseum. The provided characterizations and descriptions for each channel model can be used by future users in selecting and configuring the model that best fits their specific use case.

References

[1] "Report ITU-R M.2410-0; Minimum requirements related to technical performance for IMT-2020 radio interface(s)," ITU. (Nov. 2017).

[2] E. Chirivella-Perez, J. M. Alcaraz Calero, Q. Wang, and J. Gutiérrez-Aguado, "Towards a realistic 5g infrastructure emulator for experimental service deployment and performance evaluation," in *2018 IEEE/ACM 22nd International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, 2018, pp. 1–7.

[3] J. E. Hakegard, H. Lundkvist, A. Rauniyar, and P. Morris, "Performance evaluation of an open source implementation of a 5g standalone platform," *IEEE Access*, vol. 12, pp. 25 809–25 819, 2024. DOI: 10.1109/ACCESS.2024.3367120.

[4] srsRAN, *Srsran gnb with srsue*, , n.d.

[5] R. Reddy, M. Gundall, C. Lipps, and H. D. Schotten, "Open source 5g core network implementations: A qualitative and quantitative analysis," in *2023 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, 2023, pp. 253–258.

[6] G. Lando, L. A. F. Schierholt, M. P. Milesi, and J. A. Wickboldt, "Evaluating the performance of open source software implementations of the 5g network core," in *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*, 2023, pp. 1–7. DOI: 10.1109/NOMS56928.2023.10154399.

[7] M. Chepkoech, N. Mombeshora, B. Malila, and J. Mwangama, "Evaluation of open-source mobile network software stacks: A guide to low-cost deployment of 5g testbeds," in *2023 18th Wireless On-Demand Network Systems and Services Conference (WONS)*, 2023, pp. 56–63.

[8] L. Mamushiane, A. Lysko, H. Kobo, and J. Mwangama, "Deploying a stable 5g sa testbed using srsran and open5gs: Ue integration and troubleshooting towards network slicing," in *2023 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD)*, 2023, pp. 1–10.

[9] M. Amini and C. Rosenberg, *Performance analysis and comparison of full-fledged 5g standalone experimental tdd testbeds in single multi-ue scenarios*, Jul. 2024.

[10] H. Gao, Z. Wang, X. Zhang, *et al.*, "Over-the-air performance testing of 5g new radio user equipment: Standardization and challenges," *IEEE Communications Standards Magazine*, vol. 6, no. 2, pp. 71–78, 2022.

[11] K. Takeda, K. Mizutani, and H. Harada, "Open-source software-based beyond 5g emulator for evaluating dynamic full-duplex cellular system," in *2024 IEEE 100th Vehicular Technology Conference (VTC2024-Fall)*, 2024, pp. 1–5.

[12] G. Nardini, G. Stea, and A. Virdis, "Scalable real-time emulation of 5g networks with simu5g," *IEEE Access*, vol. 9, pp. 148 504–148 520, 2021.

[13] N. A. Shah, M. T. Lazarescu, R. Quasso, S. Scarpina, and L. Lavagno, "Fpga acceleration of 3gpp channel model emulator for 5g new radio," *IEEE Access*, vol. 10, pp. 119 386–119 401, 2022. DOI: 10.1109/ACCESS.2022.3221124.

[14] J. Hoydis, F. A. Aoudia, S. Cammerer, *et al.*, "Sionna rt: Differentiable ray tracing for radio propagation modeling," in *2023 IEEE Globecom Workshops (GC Wkshps)*, 2023, pp. 317–321.

[15] E. J. Oughton, K. Katsaros, F. Entezami, D. Kaleshi, and J. Crowcroft, "An open-source techno-economic assessment framework for 5g deployment," *IEEE Access*, vol. 7, pp. 155 930–155 940, 2019.

[16] M. Kurras, S. Dai, S. Jaeckel, and L. Thiele, "Evaluation of the spatial consistency feature in the 3gpp geometry-based stochastic channel model," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, 2019, pp. 1–6.

[17] N. Kumar and K. Kumar, "Performance analysis and comparison of m x n zero forcing and mmse equalizer based receiver for mimo wireless channel," *Songklanakarin Journal of Science and Technology*, vol. 33, Jun. 2011.

[18] A. Rahmati, K. Raahemifar, T. A. Tsiftsis, A. Anpalagan, and P. Azmi, "Ofdm signal recovery in deep faded erasure channel," *IEEE Access*, vol. 7, pp. 38 798–38 812, 2019.

[19] "Study on channel model for frequencies from 0.5 to 100 GHz (Release 18)," 3GPP. (Mar. 2024).

[20] Y. Zhao and W. Xie, "Physical layer round trip latency analysis and estimation for 5g nr," in *2023 International Wireless Communications and Mobile Computing (IWCMC)*, 2023, pp. 971–976.