Ns3Sionna: Realistic Wireless Network Simulation with Ray Tracing in ns-3

Anatolij Zubow, Sascha Rösler, Falko Dressler
School of Electrical Engineering and Computer Science, TU Berlin, Germany
{zubow,roesler,dressler}@tkn.tu-berlin.de

***Abstract*—Network simulators are essential tools for advancing wireless communication technologies, providing cost-effective, reproducible, and scalable environments for system evaluation. However, conventional simulators such as ns-3 rely on simplified statistical or stochastic channel models that inadequately represent physical propagation phenomena such as multipath fading, diffraction, and shadowing. We present `Ns3Sionna`, a framework that integrates a ray-tracing-based channel model implemented using the Sionna RT engine into the ns-3 network simulator. This integration enables environment-specific, physically accurate channel realizations for arbitrary 3D scenes and device configurations. `Ns3Sionna` also introduces a ray-tracing-based mobility model that ensures realistic node movement within complex indoor and outdoor environments. Compared with existing ns-3 models, `Ns3Sionna` produces more realistic path loss, fading and delay characteristics, exhibiting spatial and temporal correlations consistent with measured wireless channels. Fine-grained channel state information generated by the framework can further support sensing and localization research. To address the high computational complexity of ray tracing, `Ns3Sionna` leverages GPU and multi-core CPU parallelization together with intelligent pre-caching mechanisms based on channel reciprocity and coherence time. This approach enables practical, high-fidelity simulations for small- to medium-scale mobile wireless networks.**

***Index Terms*—Ns-3, simulation, channel modeling, ray tracing**

I. Introduction

Wireless communication has become a cornerstone of modern society, underpinning countless applications from home connectivity to industrial automation. Technologies such as Wi-Fi or LTE/5G are pervasive, forming the backbone of our modern digital infrastructure. The continuous evolution of wireless systems depends on rigorous testing and validation of new communication protocols and architectures. However, real-world experimentation is often costly, time-consuming, and constrained by hardware availability and environmental control. As a result, network simulators play a crucial role by providing a scalable, repeatable, and cost-effective environment for evaluating wireless technologies.

Among the available network simulators, *ns-3* [1] has emerged as a leading open-source platform due to its modular design, performance [2], and extensive support for communication protocols such as Wi-Fi and 4G/5G. It is widely adopted in both academia and industry for system-level studies of wireless networks. Despite these advantages, *ns-3* and similar simulators are inherently limited by their simplified statistical and stochastic channel models, which only approximate signal propagation phenomena such as multipath propagation and shadowing. While sufficient for coarse-grained evaluations, these models fail to capture spatial and temporal channel correlations observed in realistic indoor and outdoor environments. Consequently, simulation results often diverge from real-world performance, reducing their value for high-fidelity research in next-generation networks.

A promising alternative is ray tracing, which deterministically models electromagnetic (EM) wave propagation through complex 3D environments [3]. By accounting for reflections, diffractions, and scattering from physical objects, ray tracing enables the computation of spatially consistent Channel Impulse Responses (CIRs) which are a critical feature for studying advanced wireless techniques such as beamforming and sensing. *Sionna* is a framework that implements differentiable ray tracing for radio propagation within user-defined 3D scenes [4]. It supports detailed material modeling and GPU acceleration, providing accurate channel realizations for both indoor and outdoor. However, *Sionna* operates primarily at the link level, lacking integration with higher-layer network functionality which are needed for system-level analysis.

In this work, we introduce `Ns3Sionna`, a framework that bridges this gap by integrating *Sionna's* ray tracing-based channel modeling into the *ns-3* network simulator.[1] Our framework enables physically accurate, environment-specific wireless channel simulation within *ns-3*, including realistic mobility in 3D spaces. It leverages intelligent pre-caching mechanisms that exploit channel reciprocity and coherence time as well as parallel computation across GPUs and multi-core CPUs. Our evaluation demonstrates that `Ns3Sionna` provides substantially more realistic propagation characteristics, both spatially and temporally, than existing *ns-3* models, enabling the exploration of advanced wireless technologies and applications under physically accurate conditions.

II. Background

We begin by reviewing key concepts of wireless channels and ray tracing, then briefly describe the *Sionna RT* framework and the *ns-3* simulator.

A. Wireless Channel

Wireless technologies such as Wi-Fi and 5G are built upon the orthogonal frequency division multiplexing (OFDM) waveform where the transmitted signal comprises of multiple

[1] An early version of `Ns3Sionna` was presented in [5].

subcarriers, each experiencing a flat-fading channel. This structure mitigates the issue of frequency selectivity in wideband transmissions [6]. The received signal on the k-th subcarrier in the frequency domain can be modeled as:

$$y[k] = H[k] \cdot s[k] + z[k] \tag{1}$$

where $y[k]$ is the received symbol, $s[k]$ is the transmitted symbol, $z[k]$ represents the additive noise, and $H[k]$ denotes the Channel Frequency Response (CFR) at subcarrier k. The CFR characterizes the radio channel in the frequency domain and is a complex value capturing both amplitude and phase variations introduced by the channel. The CFR is obtained through estimation techniques, and are crucial for: (i) equalization, which aims to compensate the effects of the propagation channel, and (ii) channel knowledge extraction, which is widely used in sensing applications. The CIR is obtained via inverse fast Fourier transformation (IFFT) of the CFR:

$$h[n] = \text{IFFT}\{H[k]\} \tag{2}$$

where, $h[n]$ is the CIR at delay index n. The CIR represents the signal power received over a multipath channel as a function of propagation delay, revealing the arrival times of the individual multipath components [7].

Another important parameter is the coherence time which defines the duration over which the channel can be assumed to remain constant [8]. Channel variations arise from either the relative movement between the transmitter and receiver, or from motion of surrounding objects within the environment. The channel coherence time (T_C) is directly impacted by the Doppler shift (f_D) and is defined as [8]:

$$T_c = \frac{0.423}{f_D} = \frac{0.423 \cdot c}{v \cdot f_c} \tag{3}$$

where c is the speed of light, v is the relative speed, and f_c is the center frequency. As the equation indicates, for a fixed relative velocity, systems operating at higher carrier frequencies experience shorter coherence times, making them more susceptible to time-varying channel effects.

Finally, there is the effect called shadowing or large-scale fading. It represents the random attenuation of signal power due to obstructions between the transmitter and receiver like buildings, vegetation, or terrain, causing received signal strength to fluctuate log-normally over distances of tens to hundreds of meters, far exceeding the wavelength. This slow-fading effect is typically modeled as a zero-mean Gaussian random variable in decibels, added to the deterministic path loss to simulate environmental variability and spatial correlation between links.

B. Ray Tracing

Ray tracing is a method that simulates signal propagation in a specific environment by taking into account effects such as reflection, diffraction and scattering. It has become increasingly important in the planning and prediction of radio networks, as it offers deterministic modeling of propagation channels and can realistically reproduce the time and space dispersion characteristics of the channel that are important for modern wireless systems [9]. The basic idea of EM ray tracing is that electromagnetic waves are treated like rays and that their propagation can be calculated accurately [10]. In a ray tracing simulation, the CIR between a transmitter and a receiver is calculated taking into account the geometric properties of the environment. The simulation is carried out by tracing the rays that are emitted by the transmitter and that are either reflected, diffracted or scattered by the surrounding objects. To do this, ray tracing requires a detailed description of the simulated 3D environment, in particular the geometric and EM properties, i.e. relative permittivity and conductivity, of the objects inside the environment [9]. The ray tracing approach thus enables accurate calculation of path losses and propagation delays and provides a precise prediction of the transmission channel [3].

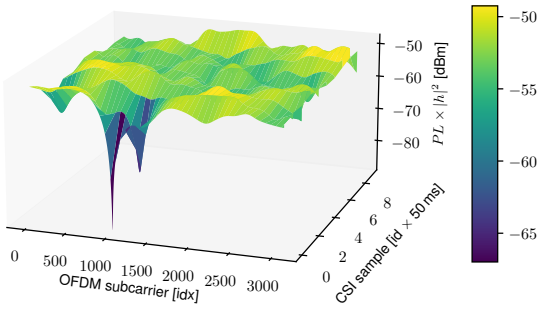


Figure 1. CFR computed with *Sionna RT* ($f_0 = 5\,\text{GHz}$, $B = 80\,\text{MHz}$)

A significant advantage of EM ray tracing over the statistical and stochastic models from network simulators is its ability to generate spatially consistent CIRs [11]. This ability is particularly needed for research topics in the field of wireless communication and sensing like 6G/Wi-Fi [12]. Although EM ray tracing provides precise results, the calculation of signal propagation requires a lot of computing power. However, advancements in the computing power of modern computers, e.g., graphics processing unit (GPU), allow ray tracing to be carried out more efficiently today [9].

C. Sionna RT Simulator

Sionna is an open-source Python library based on tensorflow for link-level simulations [4]. It has a ray-tracing extension called *Sionna RT*, which builds on a differentiable ray tracing model for accurate radio wave propagation. *Sionna RT* enables the simulation of specific radio environments which are described by a 3D model in which objects can be assigned with different materials having specific EM properties. These materials are characterized by parameters such as relative permittivity and conductivity, which allows the simulation of realistic interactions of radio waves with the surfaces [12]. *Sionna RT* calculates the paths of radio waves through the environment and tracks effects such as reflection, diffraction and scattering, enabling realistic modeling of the signal paths from which the CIR can be computed. It is suitable for both indoor and outdoor radio wave simulation and provides a realistic and accurate modeling thanks to the detailed consideration of material properties and environmental structures - see example CFR in Fig. 1. In it latest version *Sionna RT* integrates the

additional phenomena of refraction for channel computation. It is currently limited to the transmission through a single-layer slab. In summary, *Sionna RT* provides a tool for the simulation of radio wave propagation that enables the precise calculation of spatially consistent CIRs. Moreover, the computation is GPU-accelerated enabling orders-of-magnitude faster wireless propagation simulations compared to CPU-based methods.

D. Ns-3 Network Simulator

The network simulator *ns-3* is an event-driven, packet-based simulator developed primarily for research and academia in the field of wired and wireless communication networks. It is an open-source software developed in C++ using object-oriented programming model [13]. The broad research community continuously contributes to the expansion of the network simulator's functionality, which has led to a wide range of technologies (e.g., Ethernet, WiFi, LTE) and statistical and stochastic models for the simulation of channel propagation, mobility and traffic generation [14]. *ns-3* tries to reflect the reality as close as possible, therefore it uses several core concepts and abstractions that map well to how computer networks are built, i.e., a `Node` is a fundamental entity connected to the network. It is a container for `Applications`, `Protocols` and `Network Devices`. An application is a user program that generates packet flows. A protocol represents a logic of network and transport level protocols. A `Network Device` is an entity connected to a `Channel` which represent the transmission medium between the devices.

The simulation of channel propagation is as follows. For LTE *ns-3* employs the `MultiModelSpectrumChannel`, implementing the `SpectrumChannel` interface, to model wideband propagation across multiple frequencies and bandwidths, supporting pathloss, small-scale fading, and dynamic spectrum access for cellular scenarios with overlapping carriers. Conversely, WiFi utilizes the `YansWifiChannel`, subclassing the generic `Channel` class, for narrowband signal propagation incorporating interference, delay, and loss models such as `Friis` or `LogDistance`. However, for advanced simulation of interference the optional `Spectrum` module integration is possible via `WifiSpectrumPhy`. It also enables studies to be conducted on hybrid network, e.g. LTE and Wi-Fi operating in shared spectrum. Note, that `SpectrumChannel` and `YansWifiChannel` differ in propagation modeling paradigms: frequency-selective versus aggregate signal processing. `SpectrumChannel` employs `SpectrumPropagationLossModel` to impose subband-specific attenuations on transmitted power spectral densities (PSDs), thereby accounting for wideband phenomena such as frequency-selective fading, carrier-specific pathloss, and beamforming. It integrates `PropagationDelayModel` for distance-based temporal delays and operates on PSDs, facilitating accurate inter-technology interference. Conversely, `YansWifiChannel` utilizes `PropagationLossModel`, consisting of chainable scalar variants like `Friis` or `LogDistance`, to apply uniform attenuation across entire packets while foregoing frequency selectivity for computational efficiency. This is a useful simplification as long as the channel is narrowband, which is the case at least for the original versions of WiFi. For modern Wi-Fi standards such as 802.11be with bandwidths up to 320 MHz, integration with the `Spectrum` models through `WifiSpectrumPhy` is advised to achieve more realistic, subband-resolved multi-frequency simulations.

In *ns-3*, shadowing is modeled as zero-mean Gaussian log-normal fading with scenario-specific standard deviations. In case of LTE it follows the modeling in [15] and is integrated into the `ThreeGppPropagationLossModel` base class and derivatives. It includes spatial correlation across BS-UE links, computed alongside path loss using a `ChannelConditionModel` for LOS/N-LOS determination. In contrast, WiFi simulations via `YansWifiChannel` lack native shadowing in default models, requiring custom implementation (e.g., via model chaining or `RandomPropagationLossModel`) to approximate it, prioritizing simplicity (i.e., no spatial correlation) and efficient computation over realistic modeling.

Finally, in *ns-3*, mobility is modeled through the `Mobility` module which dynamically tracks node positions and velocities. It incorporates diverse models ranging from `ConstantPositionMobilityModel` for static configurations to `RandomWalk2dMobilityModel` for diffusive motion and `HierarchicalMobilityModel` for simulation of collective dynamics.

III. Motivation

The *ns-3* network simulator provides a comprehensive suite of statistical and stochastic models for simulating wireless propagation and mobility. While these models are computationally efficient and suitable for large-scale evaluations, they are fundamentally limited in realism. Existing propagation loss models such as *LogDistance* or *Nakagami* approximate path loss and fading using distance-based attenuation or random variables, often considering only single reflections or probabilistic line-of-sight assumptions. Such simplifications ignore complex multipath effects, material-dependent attenuation, and spatial correlation, all of which significantly affect link quality and delay in real-world deployments. These limitations become critical in indoor and dense urban scenarios, where wireless signals interact with walls, furniture, and other obstacles. In such environments, path loss, fading and delay cannot be accurately represented by purely statistical distributions. For example, obstruction and reflection can lead to power variations of tens of decibels even at equal distances. Likewise, the default mobility and building models in *ns-3* restrict movement to simple geometric spaces (e.g., boxes), preventing the realistic modeling of device trajectories in complex 3D environments.

To overcome these shortcomings, deterministic ray-tracing techniques have emerged as a powerful method for accurate modeling of radio propagation. By tracing EM rays and accounting for reflection, diffraction, and scattering, ray tracing reproduces spatially consistent CIRs required for wireless research [3]. However, integrating ray tracing into packet-level

Table I
COMPUTATION TIME FOR A SINGLE CFR ON DIFFERENT PLATFORMS: RYZEN9: AMD RYZEN 9 7950X, RYZEN7: AMD RYZEN 7 5800 AND RYZEN7+RTX: AMD RYZEN 7 5800 WITH NVIDIA RTX 3060 GPU

Environment	Average computation time [s] Ryzen9	Ryzen7	Ryzen7+GPU
Free-space	0.03	0.06	0.09
Indoor (simple room)	0.9	1.2	7.7
Outdoor (Munich)	4.54	5.32	18.46

simulators such as *ns-3* has long been regarded as impractical due to its extreme computational cost.

Recent advances in GPU-accelerated computing and differentiable ray-tracing frameworks such as *Sionna RT* have changed this situation. These tools enable efficient, environment-specific propagation modeling in indoor and outdoor while maintaining high physical accuracy. Such developments open the possibility of combining deterministic, physically grounded channel models with system-level simulations.

Building on these insights, we developed `Ns3Sionna`, a framework that integrates *Sionna's* ray-tracing-based propagation modeling with *ns-3*. It provides detailed, deterministic radio wave propagation in arbitrary 3D environments and introduces a mobility model based on ray tracing for realistic device motion. By exploiting channel reciprocity, coherence-time-based caching, and GPU parallelization, `Ns3Sionna` makes high-fidelity, physically consistent wireless simulation practical. This integration enables new types of studies such as indoor localization, wireless sensing, and advanced Wi-Fi/6G protocol evaluation which was previously infeasible.

IV. THE NS3SIONNA FRAMEWORK

The `Ns3Sionna` framework was designed to combine the strengths of the *ns-3* network simulator and the *Sionna RT* ray-tracing engine, enabling physically accurate wireless channel simulation within a packet-level network environment. This section presents the underlying design principles, software architecture, and implementation details of the framework.

A. Design Principles

Integrating a computationally intensive, deterministic ray-tracing model into an event-driven packet simulator presents several challenges. A naive approach, in which every point-to-point (P2P) channel is computed individually for every packet transmission, would lead to prohibitive execution times. As an example Table I shows the average computation time on different platforms for a single P2P channel computation in three different scenarios. The computation time is highest for a complex outdoor scenario resulting in a CFR computation of multiple seconds. Therefore, even a small outdoor scenario with only a few dozen nodes and a simulation duration of just a few minutes can require years of calculations. To overcome this, `Ns3Sionna` adopts the following design principles:

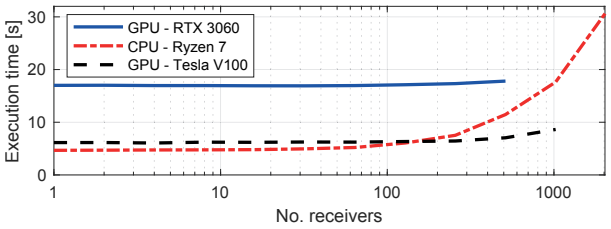


Figure 2. Computation time of a point-to-multipoint channel in *Sionna*

1) **Channel Reciprocity:** The wireless channel between two nodes is reciprocal; the same CSI/CFR can be reused for both transmission directions.
2) **Coherence-Time Caching:** Channels are recomputed only after the coherence time T_C elapses, exploiting temporal stability of the channel [8].
3) **Range Filtering:** Channels toward receivers with negligible signal power are omitted using simple analytical pathloss estimates (e.g., Friis model).
4) **Broadcast Parallelization:** Instead of computing each P2P link independently, `Ns3Sionna` computes a point-to-multipoint (P2MP) channel, offloading all propagation paths from one transmitter to multiple receivers in parallel to the *Sionna* engine.
5) **Predictive Pre-Caching:** Future channel states are precomputed based on predicted node mobility, further amortizing ray-tracing overhead.

These principles leverage the inherent parallelism of modern multi-core CPUs and GPUs to achieve feasible execution times while preserving the physical fidelity of ray tracing. Fig. 2 depicts the computation time using *Sionna RT* of a P2MP channel, i.e. single transmitter and a variable number of receivers R, for the Munich outdoor scenario. The results show that even for large R, the execution time remains nearly constant - up to $\approx R < 100$ on a CPU and $\approx R < 500$ on a Tesla V100 GPU. On GPUs, the available video memory limits the maximum feasible R. Consequently, it is advantageous to avoid computing each P2P channel individually and instead offload the entire P2MP channel from a single transmitter to all potential receivers to *Sionna*. This strategy yields significant performance gains, particularly in dense network scenarios.

Even sparse networks with only a few P2P links can benefit from the same parallelization mechanisms. The key idea is to compute, in one step, not only the current channel but also the channels likely to be required in the near future. Since the employed mobility models (e.g., random walk) are independent of the simulated network traffic, the future positions of mobile nodes together with their prospective channels can be predicted in advance. Fig. 3 illustrates this concept for scenario where both the transmitter and the receiver are mobile: the predicted future receiver positions are represented as virtual nodes with their own positions, allowing the channel computation toward all four nodes to be executed in parallel. Although this approach introduces a minor overhead - because mobility updates are currently processed sequentially per node - it enables efficient large-scale parallelization of ray-tracing computations.

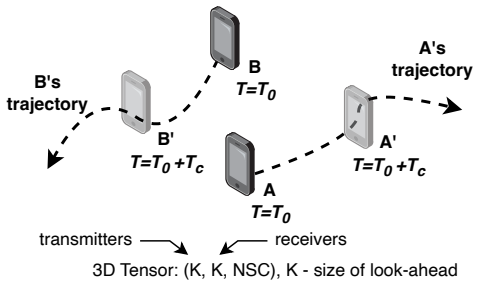


Figure 3. `Ns3Sionna` accelerates execution by treating predicted future node positions as virtual nodes for parallel computation

In summary, by adhering to these design principles, `Ns3Sionna` achieves scalable, high-fidelity wireless simulation suitable for small to medium-sized mobile networks.

B. Architecture

Fig. 4 illustrates the overall architecture of `Ns3Sionna`. The framework consists of two main components: the *ns-3* simulation core and the *Sionna RT* engine. Within *ns-3*, four new models were implemented:

- `SionnaPropagationLossModel`: wraps the *Sionna RT* propagation computation and provides the average path loss to the *ns-3* PHY layer;
- `SionnaSpectrumPropagationLossModel`: wraps the *Sionna RT* propagation computation and provides the frequency-selective fading and CFR to the *ns-3* PHY layer;
- `SionnaPropagationDelayModel`: computes realistic propagation delays based on the shortest propagation paths derived from *Sionna RT*;
- `SionnaMobilityModel`: simulates node movement inside the 3D environment using ray tracing to prevent motion through obstacles.

All received packets are augmented with CFR tags (see `CFRTag`) to make this information available for sensing applications. A `SionnaHelper` class manages the lifecycle, configuration and synchronization of these components, including scene initialization, parameters like frequency and bandwidth, and node position. During runtime, *ns-3* issues propagation requests to *Sionna RT* for each transmission event. Instead of computing a single P2P link, `Ns3Sionna` extends the request to a full P2MP calculation, taking into account the broadcast nature of a wireless transmission and enabling massive parallelization on the GPU. The computed results which include the updated node positions, path loss, delay, and CFR, are returned to *ns-3* and stored in the `SionnaPropagationCache`. The cache handles channel reciprocity and coherence-time validity, ensuring that redundant computations are avoided. In this way, `Ns3Sionna` achieves realistic spatial and temporal correlation characteristics while maintaining tractable runtime.

C. Implementation

To integrate two heterogeneous software stacks - *ns-3* in C++ and *Sionna RT* in Python - we employ a client–server

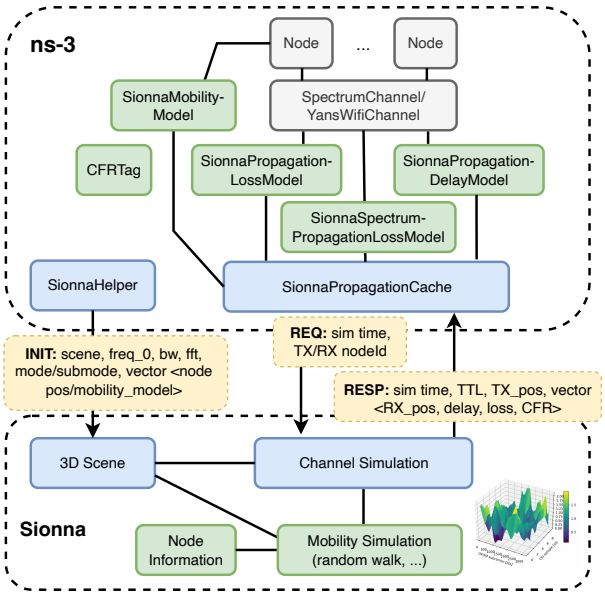


Figure 4. Architecture of the `Ns3Sionna` framework

architecture. The *ns-3* process acts as a client, and *Sionna RT* runs as a server, typically on a GPU-equipped host. Interprocess communication is realized via *ZeroMQ* using a request–response pattern, ensuring synchronous interaction with *ns-3*'s discrete-event scheduler. Message serialization is implemented with *Protocol Buffers* for low-latency data exchange. The mobility model uses the Python ray-tracing library *Mitsuba* to simulate node trajectories that respect object boundaries within the 3D scene. Future positions can be predicted deterministically, enabling speculative pre-computation of corresponding channels. All computed CSI and delay data are stored within the `SionnaPropagationCache` for reuse during the channel's coherence interval. The implementation supports multi-threading on CPUs as well as GPU acceleration in *Sionna*, where multiple transmitter–receiver paths are processed concurrently. The entire framework together with an extensive collection of examples is released as open-source and is available online at: `https://github.com/tkn-tub/ns3sionna`. This enables the research community to reproduce our results and extend the implementation.

V. Demonstration Example

As a demonstration example, we selected an outdoor environment representing the area surrounding the Frauenkirche in Munich (Fig. 5). The simulated wireless system is based on 802.11ac operating in the 5 GHz band with a 20 MHz channel. We consider the downlink scenario, where the access point (AP) is positioned on the rooftop of a building (indicated by the blue dot), and the station (STA) follows a random-walk mobility model with a velocity of $v = 7$ m/s. The resulting STA trajectory is shown in Fig. 6 (a). From the magnitude of the CFR in Fig. 6 (b), we observe a highly frequency-selective channel. As illustrated in Fig. 6 (d), the average received power (P_{rx}) remains relatively constant up to a distance of

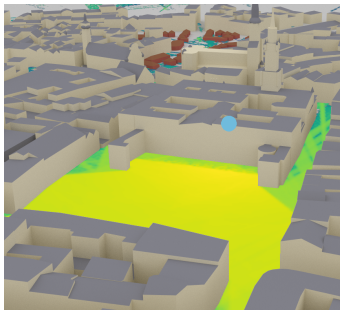
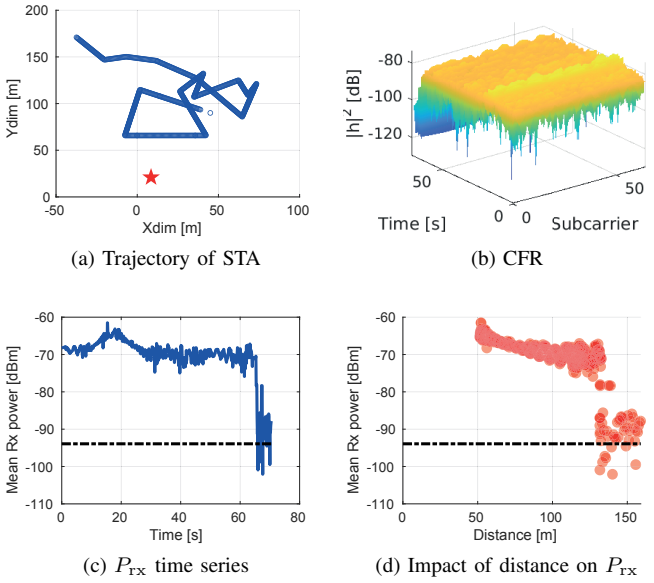


Figure 5. Outdoor example scenario (Frauenkirche in Munich)



(a) Trajectory of STA (b) CFR

(c) P_{rx} time series (d) Impact of distance on P_{rx}

Figure 6. Results for selected outdoor scenario

approximately 125 m, dominated by the line-of-sight (LoS) component. Beyond this range, P_{rx} decreases sharply as the channel transitions to a non-line-of-sight (NLoS) condition.

VI. Benchmarking

In this section, we evaluate the runtime performance of `Ns3Sionna` and compare it with pure *ns-3*, highlighting the factors that most significantly affect execution time. Experiments were conducted on a host equipped with an AMD Ryzen 9 7950X CPU (16 cores) and, where indicated, with GPU acceleration.

A. Simulation Setup

We consider a simple indoor scenario (single room) using IEEE 802.11ac with a 20 MHz channel, a single AP, and a variable number of STAs. Three representative configurations were evaluated, differing in node mobility and traffic load:

- **hT/hM:** high traffic, high mobility ($U = 50$ pkt/s per STA, $v = 7$ m/s, $T_C = 1.6$–3.2 ms);
- **lT/lM:** low traffic, low mobility ($U = 1$ pkt/s per STA, $v = 1$ m/s, $T_C = 11.1$–22.3 ms);
- **hT/zM:** high traffic, no mobility ($U = 50$ pkt/s per STA).

Each simulation lasted 10 s of simulated time.

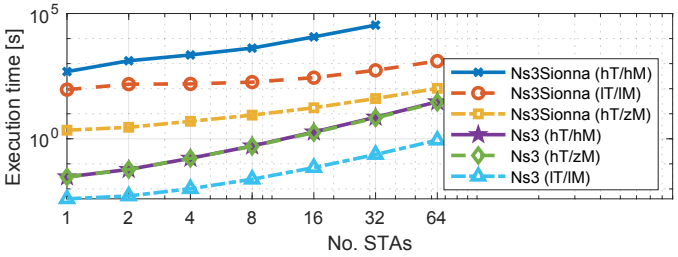


Figure 7. Framework execution time on CPU: ns-3 vs. `Ns3Sionna`

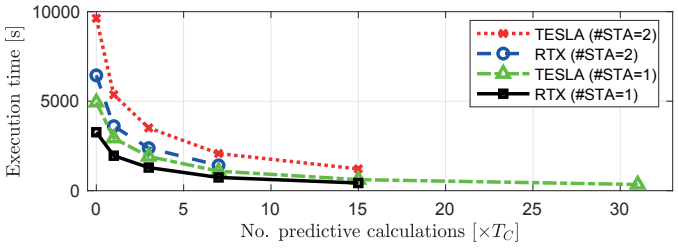


Figure 8. Speedup through predictive calculations on GPU

B. CPU Performance

Fig. 7 shows the execution time of `Ns3Sionna` compared with pure *ns-3*. As expected, the introduction of ray tracing increases computation time, but the magnitude of this overhead depends strongly on network dynamics and traffic intensity. The smallest overhead occurs in the stationary (**hT/zM**) configuration, where each channel must be computed only once. Here, the execution time of `Ns3Sionna` is approximately 3.6–73× that of pure *ns-3*, with the relative overhead decreasing as the network size grows. This reduction is due to efficient P2MP computation and an increasing cache hit rate, both of which amortize ray-tracing cost across multiple links. In mobile configurations, channel recomputation occurs frequently - up to 625 Hz in the **hT/hM** scenario—resulting in significantly longer execution times. Even in these cases, the relative slowdown decreases for larger networks due to parallel computation and caching. For instance, in the **lT/lM** scenario with 64 STAs, execution time is about 1.4×10^3 greater than pure *ns-3*; for **hT/hM** with 16 STAs, the factor increases to roughly 6.2×10^3.

C. GPU Acceleration

Fig. 8 shows the impact of GPU acceleration using NVIDIA Tesla V100 (16 GB) and RTX 3060 (8 GB) GPUs. Predictive channel calculation provides substantial speedups - up to 14.3× for the Tesla V100 and 7.6× for the RTX 3060 in single-STA configurations. As the number of STAs increases, speedup decreases due to GPU memory constraints limiting the number of parallel computations. For example, with two STAs, speedup reduces to approximately 7.9× and 4.6×, respectively. The results clearly indicate that runtime performance scales with the available GPU memory.

VII. Related Work

Related work spans two major areas: (i) the use of ray-tracing methods for wireless channel modeling, and (ii) the integration

of realistic propagation models into network simulators.

Ray-Tracing-Based Channel Modeling: Yun and Iskander [3] provide a comprehensive review of its principles and applications for radio propagation modeling. Degli-Esposti et al. [9] demonstrated that ray-tracing-based modeling can be effectively used to evaluate beamforming strategies at millimeter-wave frequencies. Hoydis et al. [4] introduced *Sionna RT*, a differentiable ray-tracing framework for high-fidelity radio environment simulation, enabling GPU acceleration and material-aware propagation modeling for arbitrary 3D scenes.

Extensions of Network Simulators: Efforts to enhance network simulators with realistic channel modeling date back to Dricot and De Doncker [16], who developed a hybrid ray-tracing model for *ns-2*. Their approach increased physical realism but suffered from computational overheads more than two orders of magnitude greater than conventional models. Wilhelmi et al. [17] emphasized the importance of integrating advanced channel models into simulators to enable machine learning (ML) assisted 5G/6G research. `Ns3Sionna` builds on these ideas by introducing a deterministic, physically grounded propagation model directly into *ns-3*, while employing caching and parallelization to make ray tracing computationally feasible.

Hybrid and Learning-Based Approaches: Complementary research has explored combining ray tracing with data-driven or machine-learning methods to improve modeling accuracy. Seretis and Sarris [18] proposed a hybrid ML-based model that enhances received signal strength prediction by training on both measured and ray-tracing-generated synthetic data, achieving higher accuracy than models based solely on measurements. Such hybrid approaches underscore the value of physically consistent channel data, which frameworks like `Ns3Sionna` can generate efficiently for large-scale training and evaluation.

`Ns3Sionna` uniquely integrates a GPU-accelerated, differentiable ray-tracing engine into a packet-level network simulator. This combination enables system-level studies that jointly capture realistic propagation effects and higher-layer protocol behavior - bridging the gap between deterministic physical modeling and scalable network simulation.

VIII. Conclusion

We have presented `Ns3Sionna`, a software framework that integrates GPU-accelerated ray-tracing-based channel simulation into the widely used *ns-3* network simulator. By coupling physically accurate, environment-specific propagation modeling with system-level simulation, `Ns3Sionna` enables realistic evaluation of wireless networks in both indoor and outdoor 3D scenarios. The framework provides fine-grained channel state information, supports mobility within complex environments, and leverages caching and parallelization to achieve practical execution times.

Future work will focus on further performance optimization through distributed multi-GPU and multi-CPU execution, as well as extending the framework to support multi-antenna systems and reconfigurable intelligent surfaces.

Acknowledgements

We thank Yannik Pilz for implementing the initial version of ns3sionna as presented in [5]. This work was supported by the Federal Ministry of Education and Research (BMBF, Germany) within the 6G Research and Innovation Cluster 6G-RIC under Grant 16KISK020K as well as by the German Research Foundation (DFG) within the project ML4WiFi under grant DR 639/28-1.

References

[1] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena, "Network simulations with the ns-3 simulator," in *ACM SIGCOMM 2008, Demo Session*, Seattle, WA: ACM, Aug. 2008, p. 527.

[2] E. Weingartner, H. Vom Lehn, and K. Wehrle, "A performance comparison of recent network simulators," in *IEEE International Conference on Communications (ICC 2009)*, Dresden, Germany: IEEE, Jun. 2009.

[3] Z. Yun and M. F. Iskander, "Ray Tracing for Radio Propagation Modeling: Principles and Applications," *IEEE Access*, vol. 3, pp. 1089–1100, Jul. 2015.

[4] J. Hoydis, S. Cammerer, F. A. Aoudia, A. Vem, N. Binder, G. Marcus, and A. Keller, "Sionna: An Open-Source Library for Next-Generation Physical Layer Research," arXiv, Information Theory (Cs.IT); Artificial Intelligence (Cs.AI); Machine Learning (Cs.LG) 10.48550/2203.11854, Mar. 2023.

[5] A. Zubow, Y. Pilz, S. Rösler, and F. Dressler, "Ns3 meets Sionna: Using Realistic Channels in Network Simulation," arXiv, cs.NI 2412.20524, Dec. 2024.

[6] J. Heiskala and J. Terry, *OFDM Wireless LANs: A Theoretical and Practical Guide*. Indianapolis, IN: SAMS, 2001.

[7] Y. Xie, Z. Li, and M. Li, "Precise Power Delay Profiling with Commodity Wi-Fi," *IEEE Transactions on Mobile Computing*, vol. 18, no. 6, pp. 1342–1355, Jun. 2019.

[8] T. S. Rappaport, *Wireless Communications: Principles and Practice*, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 2001.

[9] V. Degli-Esposti, F. Fuschini, E. M. Vitucci, M. Barbiroli, M. Zoli, L. Tian, X. Yin, D. A. Dupleich, R. Muller, C. Schneider, and R. S. Thoma, "Ray-Tracing-Based mm-Wave Beamforming Assessment," *IEEE Access*, vol. 2, pp. 1314–1325, Oct. 2014.

[10] P.-H. Tseng, Y.-C. Chan, Y.-J. Lin, D.-B. Lin, N. Wu, and T.-M. Wang, "Ray-Tracing-Assisted Fingerprinting Based on Channel Impulse Response Measurement for Indoor Positioning," *IEEE Transactions on Instrumentation and Measurement*, vol. 66, no. 5, pp. 1032–1045, May 2017.

[11] J. Hoydis, F. A. Aoudia, S. Cammerer, F. Euchner, M. Nimier-David, S. ten Brink, and A. Keller, "Learning Radio Environments by Differentiable Ray Tracing," arXiv, cs.IT 2311.18558, Nov. 2023.

[12] J. Hoydis, F. A. Aoudia, S. Cammerer, M. Nimier-David, N. Binder, G. Marcus, and A. Keller, "Sionna RT: Differentiable Ray Tracing for Radio Propagation Modeling," arXiv, Cs.IT 2303.11103, Mar. 2023.

[13] ns-3 Consortium, *ns-3 Network Simulator*, https://www.nsnam.org/, Version 3.41 or later, 2025.

[14] S. Baidya, Z. Shaikh, and M. Levorato, "FlyNetSim: An Open Source Synchronized UAV Network Simulator based on ns-3 and Ardupilot," in *21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM 2018)*, Montréal, Canada: ACM, Oct. 2018, pp. 37–45.

[15] 3rd Generation Partnership Project (3GPP), TSG RAN WG1, "Study on Channel Model for Frequencies from 0.5 to 100 GHz (Release 16) — 3GPP TR 38.901 version 16.1.0," 3GPP, Tech. Rep. TR 38.901 v16.1.0, Release 16, Nov. 2020. [Online]. Available: https://www.3gpp.org/dynareport/38901.htm.

[16] J.-M. Dricot and P. De Doncker, "High-Accuracy Physical Layer Model for Wireless Network Simulations in NS-2," in *International Workshop on Wireless Ad-Hoc Networks*, Oulu, Finland, Jun. 2004, pp. 249–253.

[17] F. Wilhelmi, M. Carrascosa, C. Cano, A. Jonsson, V. Ram, and B. Bellalta, "Usage of Network Simulators in Machine-Learning-Assisted 5G/6G Networks," *IEEE Wireless Communications*, vol. 28, no. 1, pp. 160–166, Feb. 2021.

[18] A. Seretis and C. D. Sarris, "A Hybrid Machine Learning-Based Model for Indoor Propagation," in *16th European Conference on Antennas and Propagation (EuCAP 2022)*, Madrid, Spain: IEEE, Apr. 2022, pp. 1–5.