

OpenNTN: An Open-Source Framework for Non-Terrestrial Network Channel Simulations

Tim Düe, MohammadAmin Vakilifard, Carsten Bockelmann, Dirk Wübben, Armin Dekorsy
Department of Communications Engineering, University of Bremen, Bremen, Germany.
{duee, vakilifard, bockelmann, wuebben, dekorsy}@ant.uni-bremen.de

Abstract—Non-Terrestrial Networks (NTNs) are critical enablers of ubiquitous connectivity in future 6G systems, but they also face challenges such as long propagation delays, significant Doppler effects, and diverse channel conditions. Developing and testing communication technologies for NTNs requires realistic and flexible simulation tools. In this work, OpenNTN is presented as an open-source software framework for simulating NTN channel models based on the 3GPP TR38.811 standard. OpenNTN is designed as an extension to the Python-based Sionna™ framework, providing channel models compatible with existing interfaces, enabling the use of the powerful tools found in Sionna™. The framework offers a flexible user interface, enabling researchers to investigate diverse scenarios. As an open-source implementation, OpenNTN empowers the research community to fully use and adapt the framework, offering a solution for both fast and easy NTN research using realistic channel models, while also providing the opportunity to extend the model for custom research interests beyond the current state of the standards.

Index Terms—Channel Simulations, End-to-End Simulations, Link-Level Simulations, Non-Terrestrial Networks

I. INTRODUCTION

Non-Terrestrial Networks (NTNs) are vital for achieving ubiquitous 6G coverage, but also present challenges, such as varying channel conditions between User Terminals (UTs), large delays, and significant Doppler influences [1]. Accurate simulations supported by comprehensive channel models are essential to address these problems. However, progress is hindered by the limited availability and high cost of datasets from operational NTN setups, and simulation models are often overly complex or oversimplified, limiting their practical value [2].

The open-source framework OpenNTN addresses this by offering implementations of channel models from the 3rd Generation Partnership Project (3GPP) TR38.811 standard [3]. These models are based on extensive measurements from operational setups and support a wide range of adjustable parameters, such as elevation angles, antenna setups, and UT environments.

OpenNTN can be used as an extension of Sionna™, a Python-based framework for communication research [4].

This research was supported in part by the German Federal Ministry of Research, Technology and Space (BMFTR) within the projects Open6GHub under grant number 16KISK016 and 6G-TakeOff under grant number 16KISK068 as well as the European Space Agency (ESA) under contract number 4000139559/22/UK/AL (AICoMS).

This enables link-level simulations using Sionna’s existing 5G-compliant tools, including Orthogonal Frequency-Division Multiplexing (OFDM) setups, channel coding schemes, and utility functions. Created based on TensorFlow and Keras, OpenNTN also supports many Machine Learning (ML) solutions.

As an open-source tool, OpenNTN can be freely expanded or adapted for research beyond current standards. It enables the use of the advanced channel models as a foundation for additional implementations, avoiding the need to implement the full mathematical framework.

While OpenNTN is not the first implementation of 3GPP’s NTN channel models, it differs in key ways. Proprietary implementations exist in MATLAB [5] and Gigayasa Wireless [6], and a C++-based open-source version exists in ns-3 [7]. OpenNTN, in contrast, is open source and Python-based, the preferred language in data processing and especially ML research. Additionally, OpenNTN can be used for simulations on Graphics Processing Units (GPUs), significantly reducing runtime. It is also modular and integrates easily into existing and future Sionna™ workflows.

OpenNTN is available on GitHub [8]. In a previous paper, OpenNTN and its use cases were introduced [9]. After additional experience with OpenNTN and feedback from the community, this paper was written to expand upon the original work and address both the feedback and the new experience gained. Compared to [9], this work introduces the following new contributions and extensions:

- Section III provides more detailed explanations of the mathematical models and their implementations, along with more detailed references to the original sources. This makes it easier to reuse individual parts of the implementation and link individual parts of the implementation to the standards, which is useful if components beyond the standard should be adapted.
- Section II and Section V provide a deeper insight into the capabilities of OpenNTN and the use of OpenNTN. While [10] focused more on the user interface, this new work provides deeper insight on the diverse set of channels that can be simulated. In addition, the examples in Section V are synchronized with the interactive tutorials found on Github, to further ease the introduction to OpenNTN.

- Lastly, the testing procedures in Section IV are explained in greater detail, including the new showcase of the validation of an end-to-end system.

In conclusion, this paper directly addresses the past feedback and experiences since the first paper and provides an even easier introduction into OpenNTN, focusing on the most relevant pieces of information.

II. CAPABILITIES AND APPLICATIONS OF OPENNTN

OpenNTN is an open-source software framework for generating realistic wireless channel simulations in NTN. As it can be used as an extension of SionnaTM, researchers can setup arbitrary end-to-end communications systems with the extensive set of modular components of the SionnaTM framework and simply use OpenNTN in a plug-and-play approach to directly and easily simulate arbitrary setups. Researchers that already use SionnaTM can directly use OpenNTN without needing to adapt any workflows. Researchers already using SionnaTM; can adopt OpenNTN directly without workflow adaptations, while new users are supported by interactive tutorials. The following sections describe OpenNTN's capabilities and available parameters for defining diverse channel scenarios.

OpenNTN enables researchers to simulate the link between a satellite and one or multiple UTs on the ground, following standardized channel models from 3GPP. OpenNTN is implemented in Python using TensorFlow and Keras, and is designed to integrate seamlessly with the SionnaTM framework. This allows users to incorporate complex NTN channel behavior into end-to-end communication system simulations, including those utilizing ML-based components.

The channel models in OpenNTN are compliant with the 3GPP TR38.811 standard [3], which defines various NTN scenarios with different levels of complexity and flexibility [3, Ch. 6]. Out of these, OpenNTN implements the frequency-selective fading models, which use a cluster-based multi-path multi-tap approach [3, Ch. 6.7.2]. A channel is defined as a set of paths between each transmitter and receiver antenna pair, where each path is characterized by a gain and delay. In addition to the fast-fading models, TR38.811 also defines Cluster Delay Line (CDL) and Tapped Delay Line (TDL) models [3, Ch. 6.9]. However, these are currently fully specified only for simulations at a fixed elevation angle of 50° and simulations at other angles require approximation based methods. OpenNTN does not yet implement the CDL and TDL models, but they will later be added, once the complete definition for all elevation angles are provided by the standards.

Next, the notation used to describe the generated channels is explained, following the conventions of the 3GPP standards and the SionnaTM; framework. Between each transmitter and receiver antenna pair, denoted as s and u , multiple clusters n are generated. Each cluster represents one path, with a corresponding time-varying complex gain $a_{u,s,n}(t) \in \mathbb{C}$ and delay $\tau_{u,s,n} \in \mathbb{R}$. Together, these form the Channel Impulse Response (CIR). The set of user-defined parameters that define a simulation setup is called the scenario. As the channel generation process incorporates randomized elements based

on statistical models, each CIR is a sample of the distribution defined by the scenario.

The following is a list of the parameters that are required to define a scenario. There are various additional parameters that can be specified, but if they are not, OpenNTN uses default values for them taken from the standard. This approach provides full control over the scenarios, while also keeping the user interface simple. The necessary parameters are:

- **Environment:** Each UT is located in either a dense urban, urban, or suburban environment. Based on this selection, OpenNTN loads standardized environmental parameters, such as UT distances, building density, UT velocities, and channel model parameters like Line-of-Sight (LOS) probability or delay spread. All parameters can be adjusted via the user interface to define specific scenarios of interest.
- **Link and Frequency:** Each channel consists of one satellite (acting as Base Station (BS)) and one or more UTs. The link direction can be downlink (DL) or uplink (UL). All frequencies within the S-band (1.9–4 GHz) and Ka-band (19–40 GHz) are supported. Although some channel loss models work beyond this range (0.5–350 GHz), the necessary parameters for each environment are currently only defined by the standards for the two listed bands, limiting the possible frequencies for simulations of complete channels.
- **Satellite Altitude:** Orbits from 500 km to 36 000 km are supported.
- **Elevation Angle:** The elevation angle — the angle between the satellite-to-UT line and the horizon — can be selected between 10° and 90°.
- **Antenna Configuration:** All antennas mentioned in 3GPP TR38.811 are implemented in OpenNTN. A list is given in Section III. Both the satellite and UTs can have one or more antennas in various arrangements.
- **Channel Effects:** Doppler shift, path loss, and shadowing effects can be disabled individually. For instance, if the satellite's position and orbit are known, Doppler can often be assumed to be pre-compensated and thus disabled in the simulation [11].

At the moment, additional scenarios of interest, such as multi-satellite channels, yet directly defined by the standard 3GPP TR38.811 and are therefore also not implemented in OpenNTN. However, because OpenNTN is open-source, researchers can easily use OpenNTN as a foundation and integrate new features beyond the standard.

III. NON-TERRESTRIAL CHANNEL MODEL

As described in Section II, the NTN channel models implemented in OpenNTN are the frequency selective fading models mainly defined in 3GPP TR38.811 [3, Ch. 6.7.2]. However, TR38.811 does not fully define the channel model generation process. Instead it extends the process defined for Terrestrial Network (TN) models from 3GPP TR38.901 [10] by adding NTN-specific steps and parameters, with additional required data from sources such as the International Telecommunication

Union (ITU) weather models [12]. The complete definition of the channel generation process is therefore complex and distributed across several documents.

This section provides a simplified summary of the full channel generating process, with references to the original documents for readers who wish to study individual parts in more detail.

On a large scale, the channel generation is divided into three main parts:

- 1) **Topology and Large-Scale Parameters:** A 3D coordinate system is created to define the topology, placing all UTs and the satellite according to scenario parameters. Positions are partly deterministic and partly sampled from scenario-specific distributions. Large-scale parameters such as propagation losses and delay spreads (DSs) are then computed. Each channel is a realistic sample for the scenario and to enable statistical generalization multiple realizations are simulated.
- 2) **Small-Scale Parameters:** Based on the results from step 1, the small-scale parameters are determined. For each transmitter–receiver antenna pair, a number of clusters is generated. Each cluster produces a set of rays, where each ray is characterized by its power, time of arrival, and phase. These rays model the physical propagation through representative samples.
- 3) **Channel Coefficients Generation:** In the final step, the effective channel paths are obtained by grouping all rays with the same time of arrival and summing their contributions to compute the path gain for that channel tap. The resulting set of paths corresponds to the channel coefficients as defined in the standard.

The 16 steps in Fig. 1 illustrate the process in more details. Because the process is based on the channel generation for TNs from 3GPP TR38.901, blue boxes mark steps unique to NTN and red boxes mark those similar to TN models.

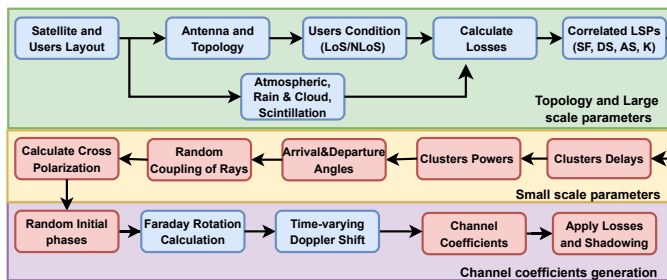


Fig. 1: Channel generation procedure based on 3GPP TR38.811

A. Topology and Large Scale Parameters

1) *Satellite and Users Layout:* A 3D coordinate system is used to define the layout of the satellite and UTs [3, Ch. 6.3]. The origin lies on the Earth’s surface, with the x - y plane representing the horizon and the z -axis pointing upward. The satellite is placed directly above the origin at the desired orbit height, oriented toward the origin. To position the UTs, a point

on the Earth’s surface at the defined elevation angle is selected along one axis. Around this point, UTs are randomly placed using Gaussian-distributed offsets based on the expected inter-UT spacing. Each UT is oriented to point directly at the satellite.

2) *Antenna and Topology:* Both singular antennas and antenna arrays can be simulated [3, Ch. 6.4]. The following radiation patterns are available:

- Antenna with a circular aperture [3, Ch. 6.4.1]. Its gain as a function of the elevation angle θ is given by:

$$G(\theta) = \begin{cases} 1 & \text{for } \theta = 0 \\ 4 \left| \frac{J_1(ka \sin \theta)}{ka \sin \theta} \right|^2 & \text{for } 0 < |\theta| \leq 90^\circ \end{cases} \quad (1)$$

where $J_1(x)$ is the Bessel function of the first kind and first order and a is the radius of the antenna aperture. $k = 2\pi \frac{f}{c}$ is the wave number for f as the carrier frequency, and c is the speed of light,

- Quasi isotropic antennas [10, Ch. 7.7.4.1],
- Dual linear polarized patterns [13],
- Very Small Aperture Terminal (VSAT) antennas modeled as uniform rectangular panel arrays [10, Ch. 7.3].

As defined by the standard [3, Ch. 6.4], all antenna patterns are normalized in OpenNTN, meaning that their maximum antenna gain is 0 dB. For applications requiring more direct link budget analyses, this could easily be adapted by introducing scaling factors for the antenna gains.

3) *Propagation Condition:* Each UT is assigned to either LOS or Non-Line-of-Sight (NLOS) condition by sampling from a Bernoulli distribution [3, Ch. 6.6.1]. Unlike in the simulations of TNs, UTs in the indoor state are not yet considered for NTN [3, Ch. 6.1.2]. However, the necessary structures to add the indoor or other states later, as soon as the standards define them, are already present in OpenNTN.

4) *Atmospheric, Rain & Cloud, Scintillation:* Signal propagation between the satellite and Earth introduces additional losses not present in TNs [12]. These include:

- PL_g : Gas or tropospheric loss, considered only in the Ka-band and set to zero in the S-band.
- PL_s : Scintillation or ionospheric loss, considered only in the S-band and set to zero in the Ka-band.

Additionally, there are explicit approaches to calculate losses for cloud and rain attenuation [3, Ch. 6.6.5]. However, the standard in its base form does the loss calculations as shown in (2), in which they are not explicitly considered. Still, the functions provided in OpenNTN are implemented for the full defined range and additional functions to calculate the rain and cloud losses are provided among the utility functions. These can be used for applications that are interested in more precise link budget analyses.

5) *Calculate Losses:* The total path loss PL_{total} is calculated as the sum of each individual loss [3, Ch. 6.6.2]

$$PL_{\text{total}} = PL_b + PL_g + PL_s + PL_e \quad (2)$$

PL_e is the building entry loss, which is always 0 in OpenNTN, as UTs indoors are not yet considered in satellite

scenarios in the standard [3, Ch. 6.1.2]. However, the necessary structures to add them later have already been implemented, so that OpenNTN can easily be adapted for new versions of the standard or other scenarios of interest beyond the standard. Lastly, PL_b marks the so called basic path loss given by

$$PL_b = FSPL(d, f_c) + SF(\theta, l, f_c) + CL(\theta, l, f_c), \quad (3)$$

with

- FSPL, the free-space path loss depending on the distance d between UT and satellite, and f_c , the carrier frequency,
- SF, the shadow fading, drawn from a Gaussian distribution based on the elevation angle θ and UT LOS state l (LOS or NLOS) ($SF \sim N(0, \sigma_{SF}^2)$, with σ_{SF}^2 read from [3, Ch. 6.6.2]), and
- CL is the clutter loss $CL(\theta, s, f_c)$, which is a deterministic value [3, Ch. 6.6.2]. If l is LOS, CL is 0.

Generally, the possible frequency ranges for models generated by OpenNTN are within the S-band and Ka-band, as the environment parameters are only defined within them. However, most of the losses themselves are defined for significantly larger frequency ranges, in the largest case reaching from 0.5 GHz to 350 GHz. OpenNTN provides implementations for the calculations over the entire possible frequency ranges for the individual losses, even outside the possible frequencies for the environments, so that the functions can be reused for other applications of interest, outside the standard simulations using the possible environments.

6) *Correlated LSPs (SF, DS, AS, K)*: For each UT, the Large Scale Parameters (LSPs)—Shadow Fading (SF), DS, Zenith of Arrival (ZOA), Zenith of Departure (ZOD), Angle of Departure (AOD), Angle of Arrival (AOA), and Rician factor K —are sampled from Gaussian distributions. These distributions are parameterized based on the UT's state (LOS/NLOS), frequency band, and environment, using values from [3, Ch. 6.7.2]. To ensure realistic spatial correlation in multi-user scenarios, the sampled LSPs are then correlated using a distance-dependent cross-correlation matrix based on the method in [14].

B. Small Scale Parameters

As introduced in section II, the channel models use a cluster-based, multipath, multi-tap approach, in which each cluster represents at least one path. During the small scale parameter generation, a set of rays m for each cluster n is generated, which will later be used to calculate the path coefficients. The clusters represent the NLOS component of the channel. For UTs in LOS an additional LOS path is calculated and later added to the NLOS component. The number of clusters and rays per cluster depend on the elevation angle and environment. A visualization is shown in Fig. 2.

1) *Cluster Delays*: Each cluster n has a base delay drawn from an exponential delay distribution based on the sampled DS [3, Ch. 6.7.2]. The delays are sorted and the minimum delay is subtracted, to set the smallest delay to 0 and have the other delays defined as relative to the first cluster. In the

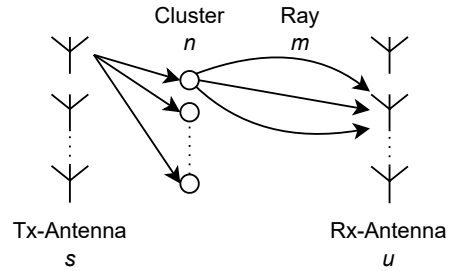


Fig. 2: Visualization of the clusters and rays structure

LOS case, delays are scaled using the Rician factor K [10, Ch. 7.5]. Each ray from the same cluster has the same delay. The resulting delays $\tau_{u,s,n}$ from this step already form part of the final CIR.

2) *Cluster Powers*: Each cluster is assigned a power, based on its respective delay, the DS, a so-called delay distribution proportionality factor, and a shadowing term [10, Ch. 7.5]. The shadowing term is drawn from a Gaussian distribution, parametrized based on the environment, satellite orbit height, and elevation angle θ [3, Ch. 6.7.2]. The powers follow an exponential distribution, decreasing asymptotically when the delay increases. The sum of all cluster powers is normalized to 1. In the LOS case, the individual cluster powers are scaled using the Rician factor K [10, Ch. 7.5].

3) *Arrival & Departure Angles*: For each ray m , the angles of departure and arrival are calculated for both the azimuth and zenith. The angle of arrival depends on the antennas orientations, the sampled angular spreads, the power of the ray's origin cluster, and a scaling factor, which depends on the total number of clusters. All rays receive a deterministic phase offset, which is always constant [10, Ch. 7.5].

4) *Random Coupling of Rays*: Because the phase offsets introduced in the previous step are always the same, they would create an artificial correlation between multiple channel realizations. To avoid this, the order of rays from the same cluster is randomly shuffled in this step. The two strongest clusters will in a later step be separated into three sub-clusters each. For these clusters, the shuffling is executed per sub-cluster, for the other clusters all rays are shuffled.

5) *Calculate Cross Polarization*: For each ray m , the cross polarization power ratio is sampled from a log-Normal distribution [10, Ch. 7.5], based on a polarization factor sampled from a Gaussian distribution, which is parameterized based on the environment and elevation angle θ [3, Ch. 6.7.2].

C. Channel Coefficient Generation

For the generation of the channel coefficients, the rays and parameters from the previous steps are used.

1) *Random Initial Phases*: For each ray, an initial phase is drawn from a uniform distribution within the range $(-\pi, \pi)$.

2) *Faraday Rotation Calculation*: The Earth's magnetic field and ionized medium introduce a phase rotation to the rays known as the Faraday rotation. This is based on the carrier frequency, with higher frequencies causing larger rotations [3, Ch. 6.8.2].

3) *Time-Varying Doppler Shift*: The Doppler shift is determined by the relative velocity between the satellite and the UTs [3, Ch. 6.8.1]. In multiple approaches, the velocity is either considered to be maximal, leading to the maximum possible Doppler, or the Doppler is set to 0. To implement more realistic and diverse Doppler behavior, OpenNTN uses a different approach. However, despite being based on the standards, this approach is not clearly defined in the standard and could therefore be considered an interpretation.

In OpenNTN, the satellite velocity is computed from its orbit height and a orbital orientation. The orientation is sampled uniformly from $[0, 2\pi]$ to reflect realistic Doppler variation for random UT positions. UT velocity is sampled based on the environment [10, Ch. 7.5]. The resulting Doppler phase shift $\Delta\Phi_{\text{Doppler}}(t)$ accumulates over time and is modeled at time t as:

$$\Delta\Phi_{\text{Doppler}}(t) = \exp\left(j2\pi \int_{t_0}^t \frac{r_{\text{UT},n,m}^T(\tilde{t}) \cdot v_{\text{UT}}(\tilde{t})}{\lambda_0} d\tilde{t}\right) \cdot \exp\left(j2\pi \int_{t_0}^t \frac{r_{\text{sat},n,m}^T(\tilde{t}) \cdot v_{\text{sat}}(\tilde{t})}{\lambda_0} d\tilde{t}\right), \quad (4)$$

where $r_{\text{UT},n,m}(\tilde{t})$ and $r_{\text{sat},n,m}(\tilde{t})$ are unit vectors in the direction of wave propagation, v_{UT} and v_{sat} are the velocities of the UT and satellite, λ_0 is the wavelength, and t_0 is the reference time at which the simulation started.

4) *Channel Coefficients*: Using the previous results, the coefficient for each ray $a_{u,s,n,m}(t)$ is calculated [3, Ch. 6.8.2]. Because the full equation is too complex for the scope of this paper, a strongly simplified expression is provided as

$$a_{u,s,n,m}(t) = p \cdot g_{\text{Rx}} \cdot \Phi \cdot \Psi_{\text{Faraday}} \cdot g_{\text{Tx}} \cdot e^{j\omega_{\text{Rx}}} \cdot e^{j\omega_{\text{Tx}}} \cdot e^{j\omega_{\text{Doppler}}}, \quad (5)$$

where

- $a_{u,s,n,m}(t)$ is the coefficient of ray m from cluster n between transmitter s and receiver u ,
- p is the power of this ray,
- g_{Rx} and g_{Tx} are the antenna gains (based on their radiation patterns and the angles of arrival and departure),
- Φ is the phase of the ray, which is scaled by the cross polarization in the NLOS case,
- Ψ_{Faraday} represents the Faraday rotation,
- $e^{j\omega_{\text{Rx}}}$ and $e^{j\omega_{\text{Tx}}}$ account for the phase shifts at the receiver and transmitter due to their 3D position, and
- $e^{j\omega_{\text{Doppler}}}$ captures the Doppler shift.

Next, the two clusters with the largest cluster-powers are separated into three sub-clusters and the rays are divided among them. For each sub-cluster a new cluster delay based on the DS and the original cluster delay is calculated. After this, the new number of clusters is $N_{\text{clusters new}} = 2 \cdot 3 + (N_{\text{clusters old}} -$

2) and the number of rays per cluster is different among some clusters [10, Ch. 7.5].

Each of these final clusters represents a path, so that the number of paths is equal to $N_{\text{clusters new}}$, which is also the number of channel taps. The resulting coefficients for each path are calculated as the sum of all rays M_n for each cluster n

$$a_{u,s,n}(t) = \sum_{m \in M_n} a_{u,s,n,m}(t). \quad (6)$$

In the LOS case, the LOS path is added to (6), because the cluster represent the NLOS components.

5) *Apply Losses and Shadowing*: Finally, the generated channel coefficients are multiplied by PL_{total} to apply the losses and shadowing. The results are the finalized channel coefficients $a_{u,s,n}(t)$ and delays $\tau_{u,s,n}$, which together form the CIR.

IV. CALIBRATION AND TESTING

The implementation was validated in three steps: calibration against the 3GPP standard, unit tests, and performance comparison with other simulations.

First, calibration followed the procedure in 3GPP TR38.821 [15], which defines 30 test scenarios varying in satellite heights, carrier frequencies, and elevation angles and provides benchmark link budget values. For each scenario, our results (available on GitHub [8]) were compared to the references. Table I shows the scenarios (SC) marked as most important by the standard. Only Free-Space Path Loss (FSPL), atmospheric loss, and scintillation loss are listed for brevity. OpenNTN matches the reference values closely, with a maximum Signal to Noise Ratio (SNR) deviation of 0.6 dB for SC1 UL. For comparison, the ns-3 implementation deviates by 0.4 dB in the same SC [16], placing our results well within an acceptable margin. The same holds true for the remaining scenarios.

TABLE I: Calibration results

Scenario	Origin	FSPL [dB]	Atmospheric [dB]	Scintillation [dB]	SNR [dB]
SC1 DL	OpenNTN	210.6	1.3	1.2	11.4
	3GPP	210.6	1.2	1.1	11.6
SC1 UL	OpenNTN	214.2	1.3	1.4	-0.1
	3GPP	214.1	1.1	1.1	0.5
SC6 DL	OpenNTN	179.1	0.5	0.4	8.4
	3GPP	179.1	0.5	0.3	8.5
SC6 UL	OpenNTN	182.6	0.6	0.5	18.1
	3GPP	182.6	0.5	0.3	18.4
SC9 DL	OpenNTN	159.1	0.1	2.2	6.6
	3GPP	159.1	0.1	2.2	6.6
SC9 UL	OpenNTN	159.1	0.1	2.2	2.3
	3GPP	159.1	0.1	2.2	2.8
SC14 DL	OpenNTN	164.5	0.1	2.2	7.3
	3GPP	164.5	0.1	2.2	7.2
SC14 UL	OpenNTN	164.5	0.1	2.2	-3.1
	3GPP	164.5	0.1	2.2	-2.6
SC16 DL	OpenNTN	210.4	0.8	0.7	4.6
	3GPP	210.4	0.8	0.5	4.8
SC16 UL	OpenNTN	213.9	0.8	0.7	-6.8
	3GPP	213.9	0.7	0.5	-6.3

Second, comprehensive unit tests were implemented for all OpenNTN functions, structured to verify each step from Fig. 1. Stochastic steps were tested by generating large sample

sets and checking adherence to the target distributions. All tests produced matching results to independently generated references. Unit tests are available on GitHub [8].

Lastly, OpenNTN was compared to existing NTN simulations from the literature. Where exact setups could not be replicated, reasonable error margins were defined. One example is the link-level investigation from the 6G-NTN project [17] (also shown in Section V), which evaluates multiple variations of a scenario with either pure LOS or NLOS conditions and additional impairments. As expected, OpenNTN's performance falls between the best case (pure LOS) and worst case (pure NLOS) results from [17]. Fig. 5 confirms this, supporting the validity of our implementation.

V. EXAMPLES

In this section, examples generated using OpenNTN are presented to demonstrate its capabilities for evaluating different results across various scenarios. First, scenario parameters are varied to measure received powers, followed by a complete end-to-end link-level simulation.

A. Receive Power for Different Scenarios

The expected receive power is evaluated when varying the elevation angle or satellite height. In a DL suburban scenario, a satellite uses a single circular aperture antenna to transmit to a single UT with a quasi-isotropic antenna at $f_c = 2$ GHz. For elevation variation, the satellite height is fixed at 600 km and 900 km scenarios are simulated from 10° to 90° in 0.1° steps, each with $N_{\text{samples}} = 20000$ CIRs. The expected receive power P is computed as:

$$P = 10 \log_{10} \left(\frac{1}{N_{\text{samples}}} \sum_{i=1}^{N_{\text{samples}}} \left| \sum_{n=1}^{N_{\text{paths}}} a_{i,1,1,n} e^{-j2\pi f_c \tau_{i,1,1,n}} \right|^2 \right). \quad (7)$$

For satellite height variation, 100 scenarios are simulated at each height with a fixed elevation angle of 90° and heights ranging from 600 km to 1600 km in 10 km steps.

The results for the simulations at different elevation angles are shown in Fig. 3. The expected receive power pattern clearly resembles the Bessel function, which results from the satellite's antenna radiation pattern.

The results for the different satellite heights are shown in Fig. 4. Here, the receive power decreases with the height, the values are approximately -170 dB at 600 km and -176 dB at 1200 km. This means that doubling the height decreases the receive power by about 6 dB, which is due to the FSPL. This matches the fundamental assumption that the FSPL decreases with the square of the distance.

Similar simulations have been implemented using ns-3 [16], which show the same results.

B. End-to-End Link Level Simulation

In this subsection, link-level investigations are presented, utilizing the integration of OpenNTN with SionnaTM. To validate the results, an existing scenario from the literature

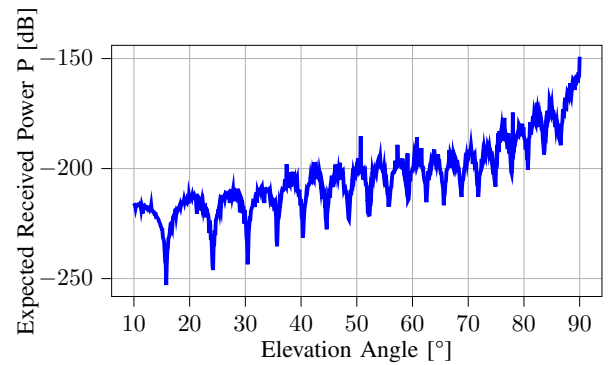


Fig. 3: Expected received power at different elevation angles

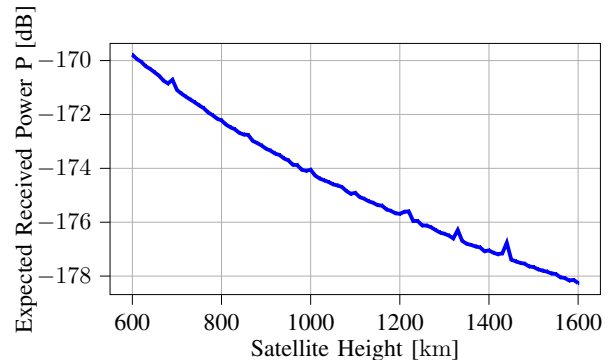


Fig. 4: Expected received power at different satellite heights

is reproduced as a reference [17, Ch. 5.3.1]. The code for this example is available under the examples section on GitHub [8].

A scenario is considered in which a satellite at 600 km altitude communicates with a single UT in DL. The satellite is equipped with a circular aperture antenna, while the UT uses an isotropic antenna. The UT is positioned at an elevation angle of 50° . The carrier frequency is set to 3.5 GHz. The transmission utilizes OFDM with 52 resource blocks and numerology 1. A Low Density Parity Check (LDPC) code with a rate of 526/1024 is applied, and QPSK is used for modulation. At the receiver, a Linear Minimum Mean Squared Error (LMMSE) equalizer is employed under the assumption of perfect Channel State Information (CSI).

Two different receiver configurations are evaluated. In the first, the decoder is omitted and hard decisions are made directly by the demapper. This configuration is used to compute the uncoded Bit Error Rate (BER), which is later compared with the reference setup [17, Ch. 5.3.1]. In the second, the demapper produces log likelihood ratios for a subsequent LDPC decoder. Simulations are performed for dense urban, urban, and suburban environments over an SNR range from 0 dB to 15 dB in 1 dB steps.

The reference [17] utilizes the TDL channel model with the addition of various levels of hardware impairments caused by power amplifiers. As this channel model differs from the

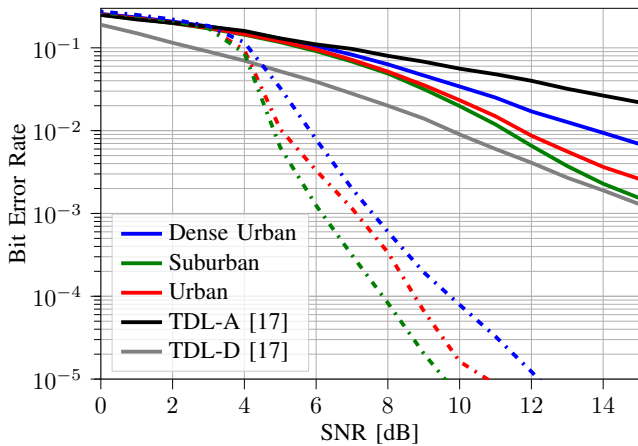


Fig. 5: BER for the coded (---) and uncoded (-) case. Comparing OpenNTN and reference simulations [17, Ch. 5.3.1]

one used in the present work, small differences are expected between the results reported in [17] and the simulations conducted here. To address this, two scenarios from the reference are selected as boundaries. For the lower boundary, the pure NLOS (TDL-A) channel with strong hardware impairments is used, for which the simulations are expected to yield lower error rates. For the upper boundary, the pure LOS (TDL-D) channel with weak hardware impairments is employed, which is expected to produce lower error rates compared to the present simulations. In [17], the BER is provided only prior to decoding. In contrast, the simulations reported here measure both the uncoded BER, for comparison with the reference, and the coded BER, to provide a more realistic assessment of the performance of an OFDM system.

Fig. 5 presents the average BER for the simulated environments alongside the selected references from [17, Ch. 5.3.1]. In the uncoded scenarios, the pure LOS case with weak impairments exhibits the best performance, whereas the NLOS case with strong impairments exhibits the worst. The results obtained lie between these two cases, presumably due to the probabilistic mixture of LOS and NLOS conditions in the simulated scenarios. Furthermore, the dense urban scenario shows the highest BER, while suburban shows the lowest, consistent with their respective LOS probabilities at a 50° elevation angle (53.7% for dense urban, 72.6% for urban, and 93.5% for suburban). The coded BERs are significantly lower than the uncoded BERs, while maintaining the trend across scenarios. As noted in Section V, this outcome matches expectations and is therefore regarded as partial validation of the correctness of OpenNTN.

VI. SUMMARY

In this paper, OpenNTN, an open-source framework for simulating NTN channels between satellites and UTs based on the 3GPP TR38.811 standard, has been presented. OpenNTN supports a wide range of scenarios, including dense urban, urban, and suburban environments, and allows flexible config-

uration of parameters such as antenna types, satellite altitude, and elevation angle.

Designed as a Python-based extension to the SionnaTM; framework, OpenNTN enables integration with existing communication system components, including those based on machine learning. Its open-source nature allows adaptation and extension of the framework for custom use cases beyond the current standards.

Validation of OpenNTN was performed through standard-aligned calibration, extensive unit tests, and comparisons with existing simulation results. The framework demonstrates high accuracy and flexibility in modeling NTN channels.

Future updates are planned to track evolving standards, including the anticipated addition of CDL and TDL models once their definitions are finalized.

REFERENCES

- [1] I. S. M. Hashim and A. Al-Hourani, "Satellite Visibility Window Estimation Using Doppler Measurement for IoT Applications," *IEEE Communications Letters*, vol. 27, no. 3, pp. 956–960, Jan. 2023.
- [2] F. Fourati and M.-S. Alouini, "Artificial intelligence for satellite communication: A review," *Intelligent and Converged Networks*, vol. 2, no. 3, pp. 213–243, 2021.
- [3] 3GPP, "Study on New Radio (NR) to support non-terrestrial networks," 3rd Generation Partnership Project (3GPP), Technical report (TR) 38.811, Oct. 2020, version 15.4.0.
- [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," Mar. 2022.
- [5] The MathWorks, Inc., "Model NR NTN Channel," Available at <https://de.mathworks.com/help/satcom/ug/model-nr-ntn-channel.html> (Aug. 2024).
- [6] Gigayasa Wireless Private Limited, "Gigayasa 5g toolkit," Available at <https://gigayasa.com/5g-toolkit/> (Nov. 2024).
- [7] G. Ferreira, "ns-3-ntn," Available at <https://gitlab.com/mattiasandri/ns-3-ntn>, Oct. 2022.
- [8] T. Düe and M. Vakilifard, "OpenNTN," Available at <https://github.com/ant-uni-bremen/OpenNTN> (May 2025).
- [9] T. Düe, M. Vakilifard, C. Bockelmann, D. Wübben, and A. Dekorsy, "An open source channel emulator for non-terrestrial networks," in *Advanced Satellite Multimedia Systems Conference/Signal Processing for Space Communications Workshop (ASMS/SPSC 2025)*, Sitges, Spain, Feb. 2025.
- [10] 3GPP, "Study on channel model for frequencies from 0.5 to 100 GHz," 3rd Generation Partnership Project (3GPP), Technical report (TR) 38.901, Apr. 2024, version 18.0.0.
- [11] M. Röper, B. Matthiesen, D. Wübben, P. Popovski, and A. Dekorsy, "Position Based Transceiver Design for Multiple Satellite to VSAT Downlink," *IEEE Open Journal of the Communications Society*, Oct. 2024.
- [12] ITU, "Ionospheric propagation data and prediction methods required for the design of satellite services and systems," Sep. 2016.
- [13] 3GPP, "Ue antenna assumption and beam modelling for ntn," 3rd Generation Partnership Project (3GPP), Nokia, TSG R1-1802551, Feb. 2018, 3GPP TSG RAN WG1 Meeting 92.
- [14] P. Kyösti, J. Meinilä et. al, "WINNER II channel models," *IST-4-027756 WINNER II, D. 1. 1. 2 VI. 2*, 2007.
- [15] 3GPP, "Solutions for NR to support Non-Terrestrial Networks (NTN)," 3rd Generation Partnership Project (3GPP), Technical report (TR) 38.821, Apr. 2023, version 16.2.0.
- [16] M. Sandri, M. Pagin, M. Giordani, and M. Zorzi, "Implementation of a channel model for non-terrestrial networks in ns-3," Available at <https://www.nsnam.org/workshops/wns3-2023/04-sandri-slides-wns3-2023.pdf>, Oct. 2022.
- [17] M. S. Carla Amatetti, Alessandro Vanelli-Coralli, "Report on unified and data driven air-interface for 6g-ntn," 6G-NTN, Technical report (TR) D4.1, May 2023, version 1.0.