Position-Based Machine Learning Propagation Loss Model Enabling Fast Digital Twins of Wireless Networks in ns-3

Eduardo Nuno Almeida, Helder Fontes, Rui Campos, and Manuel Ricardo
INESC TEC and Faculdade de Engenharia, Universidade do Porto
Porto, Portugal
{eduardo.n.almeida,helder.m.fontes,rui.l.campos,mricardo}@inesctec.pt

ABSTRACT

Digital twins have been emerging as a hybrid approach that combines the benefits of simulators with the realism of experimental testbeds. The accurate and repeatable set-ups replicating the dynamic conditions of physical environments, enable digital twins of wireless networks to be used to evaluate the performance of next-generation networks. In this paper, we propose the Position-based Machine Learning Propagation Loss Model (P-MLPL), enabling the creation of fast and more precise digital twins of wireless networks in ns-3. Based on network traces collected in an experimental testbed, the P-MLPL model estimates the propagation loss suffered by packets exchanged between a transmitter and a receiver, considering the absolute node's positions and the traffic direction. The P-MLPL model is validated with a test suite. The results show that the P-MLPL model can predict the propagation loss with a median error of 2.5 dB, which corresponds to 0.5x the error of existing models in ns-3. Moreover, ns-3 simulations with the P-MLPL model estimated the throughput with an error up to 2.5 Mbit/s, when compared to the real values measured in the testbed.

CCS CONCEPTS

• **Networks** → **Network simulations**; **Network experimentation**; Mobile networks; • **Computing methodologies** → **Machine learning**.

KEYWORDS

Propagation loss model, Machine learning, Digital twins, Wireless networks, ns-3

ACM Reference Format:
Eduardo Nuno Almeida, Helder Fontes, Rui Campos, and Manuel Ricardo. 2023. Position-Based Machine Learning Propagation Loss Model Enabling Fast Digital Twins of Wireless Networks in ns-3. In *2023 Workshop on ns-3 (WNS3 2023), June 28–29, 2023, Arlington, VA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3592149.3592150

1 INTRODUCTION

The development of next-generation networks requires the evaluation of their performance in realistic conditions. Experimental testbeds provide real results of the solution's performance, at the expense of cost and complexity of the set-up, as well as the testbed's limited availability. Network simulators, such as ns-3 [9], enable the development of repeatable and reproducible set-ups, which are relatively simple to create. However, the available networking models are generic and do not capture the specific characteristics of a given physical environment, especially in extreme scenarios with dynamic and unknown environment conditions.

In recent years, digital twins have been emerging as a hybrid approach that combines the benefits of simulators with the realism of experimental testbeds [10, 14]. Digital twins are composed of digital models that replicate the behavior of physical systems and the dynamic conditions of experimental environments. As such, they can be used to validate networking solutions and evaluate their performance in simulated environments that realistically replicate the dynamic conditions that characterize the physical environments. On the other hand, digital twins can be used in applications that require large quantities of data and interaction with the environment. One example are Reinforcement Learning algorithms, where agents are trained by applying actions on the environment and collecting observations that result from their actions, which would be very difficult to implement in experimental testbeds [3, 13]. Moreover, digital twins can be used to simulate scaled-up versions of experimental testbeds, with larger and more complex network topologies, that would entail technical difficulties to implement in the physical environment.

One of the key components of wireless networks' digital twins is the wireless channel model. Machine Learning (ML) techniques can be used to create accurate custom models of the wireless channel in a given physical environment, enabling the estimation of the channel quality [4, 5]. The ML models are trained with datasets of experimental received power or Signal-to-Noise Ratio (SNR), in order to learn and replicate the dynamic conditions of the environment without requiring predetermined models.

In [1], we presented the ML-based Propagation Loss (MLPL) model for ns-3. Unlike the trace-based simulation approach [6, 8], the MLPL model reproduces, in simulation, the experimental conditions measured in the physical environment for any network topology, node mobility and simulation duration. Using ML models trained with a dataset of network traces collected in an experimental testbed, the MLPL model estimates the radio propagation loss between two nodes for a given distance and time instant. This value is then used by ns-3 to calculate the received power at the destination node, considering the transmission power and the antenna gains of both nodes. Despite the precision of the MLPL model, the propagation loss is estimated considering the distance between both nodes, regardless of their absolute and relative positions. This

WNS3 2023, June 28–29, 2023, Arlington, VA, USA

ACM ISBN 979-8-4007-0747-6/23/06…$15.00
https://doi.org/10.1145/3592149.3592150

can lead to inaccurate estimations in complex environments with multi-path phenomena, where the propagation loss between the nodes can vary significantly for different relative positions, even though the distance is the same. Moreover, since both directions are treated equally, the MLPL model is unable to model asymmetric wireless channels, such as air-to-ground / ground-to-air wireless links [11].

This paper proposes the **position-based ML Propagation Loss (P-MLPL) model** for ns-3. The P-MLPL model is built upon the MLPL model, thus inheriting its advantages over the trace-based simulation approach, such as the flexibility of allowing any network topology, node mobility, offered traffic and simulation duration. Based on network traces collected in an experimental testbed, the P-MLPL model estimates the propagation loss suffered by packets exchanged between a transmitter and a receiver, considering the absolute node's positions and the traffic direction. This improves the model's precision, especially in complex environments, while also enabling the modeling of asymmetric wireless channels. The P-MLPL model is composed of the path loss and the fast-fading components, which are trained with the network traces collected in the experimental testbed. The total propagation loss estimated by the P-MLPL model results from the sum of both components. Therefore, the P-MLPL model enables the development of fast and more precise digital twins of wireless networks in ns-3. The created digital twins allow the validation of novel solutions and the evaluation of their performance in realistic conditions, as long as the collected network traces can be used to characterize the dynamic conditions of the wireless channel (physical twin). Furthermore, it allows experimental testbeds to be digitally scaled in simulation, allowing the use of larger and more complex network topologies in realistic conditions.

To improve the computational complexity of ns-3 simulations and mitigate the overhead due to the use of ML algorithms, an internal **cache is added to the P-MLPL model** to save the latest ML values calculated, thus avoiding repeated calculations. This cache provides a speedup in the computational performance of the P-MLPL model, thereby minimizing the duration of corresponding ns-3 simulations. This optimization is especially important, since the P-MLPL model needs to calculate the estimated propagation loss for every packet generated in ns-3, which can rapidly scale with the number of nodes and offered traffic.

The P-MLPL model is validated with a test suite developed for the ns-3 module. Additionally, we evaluate the precision and performance of the model in a specific physical environment, using the respective experimental network traces. Specifically, we evaluate the precision of the path loss and the fast-fading models, by comparing their estimates with the real experimental values. Also, we evaluate the precision of the throughput values measured in ns-3 when using the P-MLPL model, by comparing them with the corresponding real values, as well as with existing propagation loss models available in ns-3. Finally, we evaluate the computational performance of the ns-3 simulations.

The rest of this paper is organized as follows. Section 2 explains the P-MLPL model. Section 3 evaluates the P-MLPL model's estimation precision. Section 4 analyzes the P-MLPL model's performance in ns-3 simulations. Section 5 draws the conclusions and points out the future work.

2 POSITION-BASED ML PROPAGATION LOSS MODEL

This section presents the P-MLPL model, explaining its internal architecture, the structure of the ns-3 module and the dataset used to train the model. The P-MLPL model code is available in [2].

2.1 P-MLPL Model Architecture

The architecture of the P-MLPL model is shown in Figure 1. The P-MLPL ns-3 model is implemented in the `MlPropagationLossModel` class. This class is designed similarly to other propagation loss models in ns-3, by inheriting from the `PropagationLossModel` base class and overriding the `DoCalcRxPower()` virtual method. Internally, it is composed of the deterministic path loss ML model and the stochastic fast-fading model.

The path loss ML model uses a supervised learning algorithm to estimate the deterministic path loss of the wireless channel between a transmitter and a receiver. We use the ns3-ai module [15], available in the ns-3 App Store, to allow the use of external ML models developed with existing ML frameworks – for instance, Tensorflow, PyTorch or SciPy – without integrating them directly in ns-3. This module enables the exchange of data between ns-3 and an external computer process – in this case, the external ML model – via shared memory. We implement the `MlPropagationLossModelNs3AIDL` class to manage this interface, which inherits from the `Ns3AIDL` base class available in ns3-ai. To improve the computational complexity of the ns-3 simulation and mitigate the overhead of the ns3-ai module, an internal cache is added to the P-MLPL model to save the latest path loss ML queries. Whenever the P-MLPL model requires a new path loss value for a given pair of node positions, it first searches the value in its cache. If the value is found, it uses it immediately; otherwise, the P-MLPL model queries the path loss ML model for that value and saves it in the cache for future use. This mechanism avoids repeated queries to the ML model through the ns3-ai's shared memory, which are more expensive computational operations than retrieving values from memory.

The fast-fading component is modeled as a stochastic ergodic process that generates pseudo-random samples according to the empirical Cumulative Distribution Function (CDF) that fits the dataset samples representing the phenomena. The pseudo-random fast-fading samples are generated by a Random Number Generator (RNG) according to the fitted empirical CDF. To ensure repeatable and reproducible simulations, the fast-fading RNG is configured in ns-3, rather than in the external ML framework. This allows the RNG to be controlled by the same pseudo-random seed used in the ns-3 simulation, thus ensuring that the stream of pseudo-random samples are repeatable and consistent with the other RNGs in ns-3. Additionally, this design avoids the computational overhead associated with the shared memory mechanism of the ns3-ai module.

The total propagation loss for a given pair of node positions is calculated as the sum of the path loss for those positions and a pseudo-random sample from the fast-fading empirical CDF. When ns-3 requests the P-MLPL model to calculate the received power for a given packet sent by a transmitter to a receiver, the P-MLPL model returns the transmission power plus the calculated loss (path loss

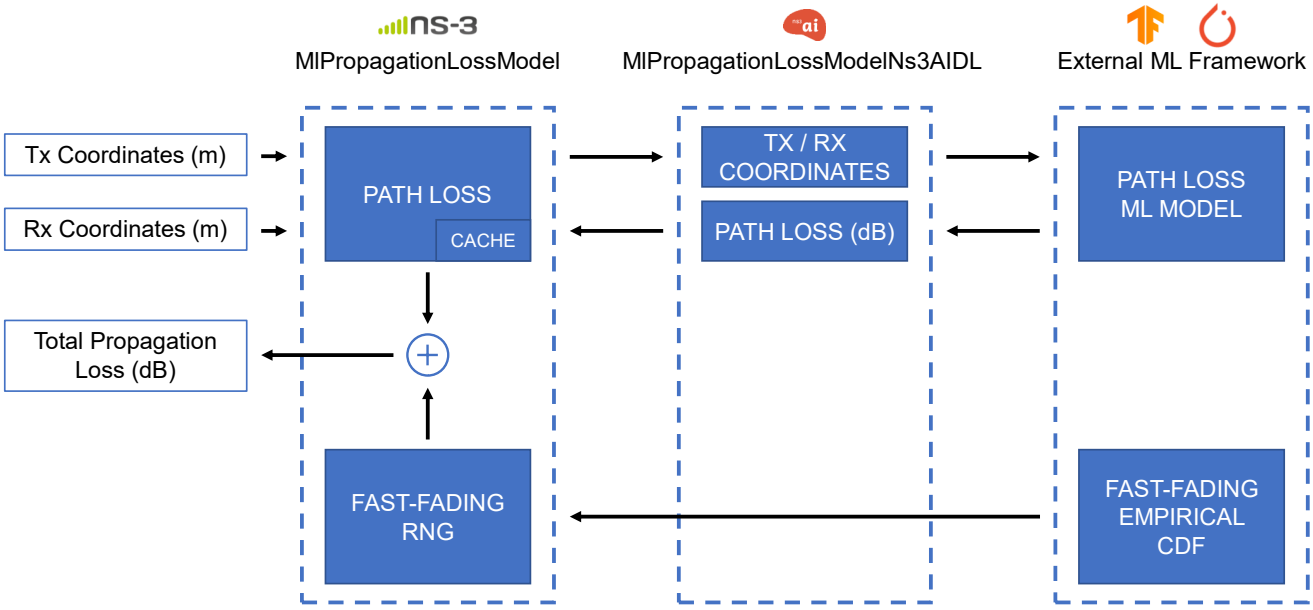


Figure 1: P-MLPL Model Architecture, Depicting the Proposed MlPropagationLossModel, and the Interactions with MlPropagationLossModelNs3AIDL (ns3-ai) and External ML Frameworks

plus fast-fading). This value is later added to the antenna gains by ns-3, depending on the wireless models configured in the simulation.

2.2 P-MLPL Module Structure

The P-MLPL module for ns-3 contains the P-MLPL propagation loss model, as well as a set of helper scripts to train the ML models and manage the interface with external ML frameworks via the ns3-ai module.

Specifically, two helper Python scripts are provided to the users of the model. The `train_ml_propagation_loss_model.py` Python script implements the training of the path loss ML model and the fast-fading empirical CDF, exporting the corresponding models as files that are read by the ns-3 simulation. This script works offline without requiring parallel ns-3 simulations and must be run once before attempting to start ns-3 simulations, so that the ML models are available to be used by P-MLPL.

The `run_ml_propagation_loss_model.py` Python script initializes the `MlPropagationLossModelNs3AIDL` model, which manages the shared memory that enables the exchange of data between the `MlPropagationLossModel` and the external ML models.

Additionally, the P-MLPL ns-3 module contains the developed test suite that validates the architecture and functionality of the P-MLPL model. There is also an ns-3 simulation example that shows how to use the P-MLPL model, which was used to generate the results in Section 4.

2.3 Deterministic Path Loss ML Model

The path loss component calculates the signal's propagation loss based on the distance between the transmitter and the receiver. It is based on a supervised learning model that estimates the deterministic path loss, in dB, for a given pair of transmitter and receiver positions.

We considered two supervised learning algorithms to train the path loss ML model: the eXtreme Gradient Boosting (XGBoost) algorithm and the Support Vector Regression (SVR) algorithm. These learning algorithms were selected due to the good results obtained when used with the MLPL model in [1]. The XGBoost algorithm was implemented using its own Python library, whereas the SVR algorithm was implemented with the SciPy library. The path loss ML model is trained with the path loss samples in the dataset.

2.4 Stochastic Fast-Fading Model

The fast-fading component is modeled as a stochastic random variable characterized by the empirical CDF that best fits the fast-fading samples in the dataset. After the fitting process, the empirical CDF is exported to a CSV file as a collection of (X, Y) pairs, where X corresponds to the fast-fading loss, in dB, and Y corresponds to the respective CDF percentile rank in $[0\%, 100\%]$.

In order to generate a stream of pseudo-random samples according to the fast-fading empirical CDF, an `EmpiricalRandomVariable` RNG is used by the P-MLPL model. When the P-MLPL model is created, the RNG is configured with the empirical CDF's (X, Y) pairs contained in the exported CSV file. Whenever a fast-fading sample is requested from the P-MLPL model, a new pseudo-random value is sampled from the RNG.

2.5 P-MLPL Propagation Loss Dataset

The training of the P-MLPL model requires a dataset of propagation loss samples collected in an experimental testbed. The description

of the dataset and the processing of its values are explained in the following sections.

2.5.1 Dataset Format. The P-MLPL dataset consists of a CSV file containing samples of the experimental propagation loss and the respective transmitter and receiver node positions.

The node positions are written in Cartesian coordinates (x, y, z), in meters, relative to a given origin point $(0, 0, 0)$. The origin point can be arbitrarily selected by the dataset creators, provided that all coordinates in the dataset are consistent with this reference. Moreover, ns-3 simulations using the P-MLPL model must use the same origin point in their mobility models, so that the propagation loss estimations are precise.

The propagation loss values can be provided in two formats. In the first format, the propagation loss values, in dB, are provided directly in the dataset. In the second format, the propagation loss is calculated indirectly using other metrics provided in the dataset, including the received signal power, or the SNR, along with the noise floor. To calculate the propagation loss, an additional JSON file must be provided, containing information about the wireless network used in the experimental testbed, such as the Wi-Fi standard, the transmission power, the antenna gains, the channel frequency and the channel bandwidth.

Apart from these mandatory fields, more information can be optionally added to the samples, including the instantaneous throughput measured by the receiver node. Although these fields do not influence the training of the P-MLPL model, they provide additional information that can be used to calculate other performance metrics related to the model. One example is the precision of the throughput measured in ns-3 simulations; this is analyzed in Section 4.2.

2.5.2 Dataset Outliers. To improve the training of the ML models, the outlier points in the dataset are first removed. For each group of samples of the same node positions, the standard z-score $z = (x - \mu)/\sigma$ is calculated for each sample x, where μ is the population mean and σ is the population standard deviation. We consider that a point is an outlier if its absolute z-score $|z|$ is greater than 5.

2.5.3 Decomposition of Propagation Loss into Path Loss and Fast-Fading. Since the dataset only contains the total propagation loss value, it is necessary to decompose it into the path loss and the fast-fading components. We assume that the fast-fading values follow a statistical distribution whose mean is 0 dB. This assumption enables the decomposition of the total propagation loss value into a deterministic path loss value plus a stochastic fast-fading random variable.

Thus, the path loss for each pair of node positions corresponds to the mean propagation loss value calculated using all samples of that pair of positions. The fast-fading values are calculated as the difference between the total propagation loss values and the path loss calculated for the corresponding node positions. The isolated path loss and fast-fading values are then used to train the corresponding ML models.

2.6 P-MLPL Module Test Suite

In order to validate the architecture and functionality of the P-MLPL ns-3 module, we developed a new test suite named `MlPropagationLossModelTest`. The objectives of the test suite are two-fold. On the one hand, the test suite validates the exchange of data between the P-MLPL model and the external ML framework via the shared memory mechanism provided by the ns3-ai module, which is managed by the `run_ml_propagation_loss_model.py` helper script. On the other hand, the test suite validates the correctness of the propagation loss values calculated by the P-MLPL model, which is the sum of the path loss ML model predictions and the samples drawn from the fitted fast-fading empirical distribution.

To that end, this test suite contains a set of *(node positions, propagation loss)* pairs taken from the example dataset contained in the ns-3 module. For each node position, the total propagation loss value calculated by the P-MLPL model is compared with the corresponding experimental value.

3 P-MLPL MODEL PRECISION

The precision of the P-MLPL model is evaluated in this section, including the precision of the path loss and the fast-fading models, as well as the overall overall propagation loss.

3.1 Experimental Dataset and Evaluation Methodology

In order to train the P-MLPL model and evaluate its precision, we created an experimental propagation loss dataset. This dataset was adapted from the larger dataset collected in the SIMBED project [12] (SubExp3 – run `08022019_11.04.35`), which consists of experimental wireless network traces collected using Fed4FIRE+ testbeds [7] in a warehouse environment. The experimental dataset used in this paper is available on the P-MLPL's GitLab repository [2], with the name `position-dataset-example`.

The set of positions of the transmitter and receiver nodes contained in the experimental dataset is shown in Figure 2. The transmitter node is fixed at position $(x_t, y_t, z_t) = (0, 0, 0)$, whereas the receiver node, at position (x_r, y_r, z_r), is moving away and towards the transmitter node in a straight line in the X-axis and the Y-axis. For each pair of transmitter and receiver positions $\{(x_t, y_t, z_t), (x_r, y_r, z_r)\}$, the dataset contains the SNR, noise floor and throughput measured by the receiver node. The dataset values were collected considering the network parameters in Table 1. It is worth noting that we used an effective antenna gain of -7 dBi for the antennas in both nodes. The gain of the antennas is a negative value since signal attenuators of 10 dB were used in-line with the 3 dBi antennas, to limit the signal's range in the warehouse.

As explained in Section 2.5, the dataset was pre-processed to remove the outliers, calculate the total propagation loss based on the SNR and noise values, and decompose the propagation loss into the path loss and fast-fading components. Then, the dataset was split in two sets: 1) the training set, containing 80% of randomly-selected samples; and 2) the test set, with the remaining 20% of samples.

3.2 Path Loss ML Model Precision

The path loss ML model was trained with the path loss values in the training set. As referred in Section 2.3, two supervised learning algorithms were considered for the path loss ML model: the eXtreme Gradient Boosting (XGBoost) and the Support Vector Regression (SVR).

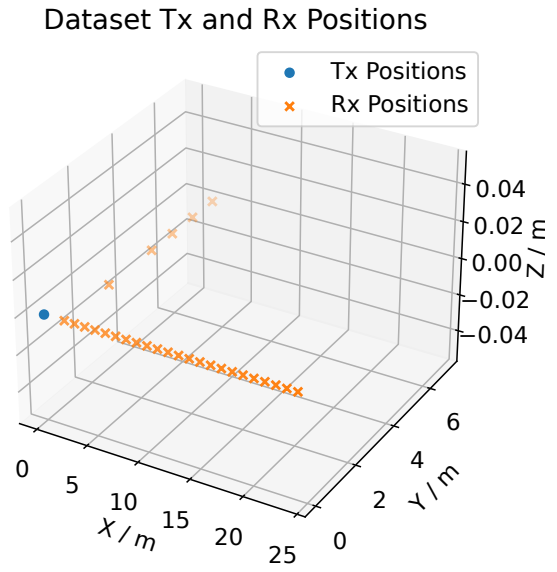


Figure 2: Set of Transmitter (Tx) and Receiver (Rx) Node Positions in the Experimental Dataset

Table 1: Network Parameters of the Experimental Dataset

Parameter	Value
Wi-Fi standard	IEEE 802.11a
Wi-Fi MAC	Ad Hoc mode
Tx power	1 dBm
Antenna gains	−7 dBi (3 dBi gain - 10 dBi attenuator)
Channel frequency	5220 MHz
Channel bandwidth	20 MHz
MAC rate adaptation	Minstrel
Application traffic	54 Mbit/s UDP constant bitrate
Packet size	1400 bytes

The precision of the path loss ML model was evaluated using the path loss values in the test set. The real path loss values P_i of the test set were compared to the respective predictions $\hat{P}_i$ by the ML model for the same set of node positions. The precision of the ML model was evaluated as the Mean Squared Error (MSE) calculated with all N samples of the test set, which is given by Equation (1).

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}\left(P_i - \hat{P}_i\right)^2 \tag{1}$$

Table 2 presents the MSE for the path loss ML models trained with the XGBoost and the SVR learning algorithms. The results show that the **SVR model achieved an MSE = 1.6 dB^2, resulting in very precise predictions of the path loss** for the set of positions in the test set. The XGBoost model achieved an MSE = 4.3 dB^2, resulting in precise predictions of the path loss, although less precise than the SVR.

Table 2: Path Loss ML Model Precision for Different ML Algorithms

ML Algorithm	Mean Squared Error (MSE)
XGBoost	4.3 dB^2
SVR	1.6 dB^2

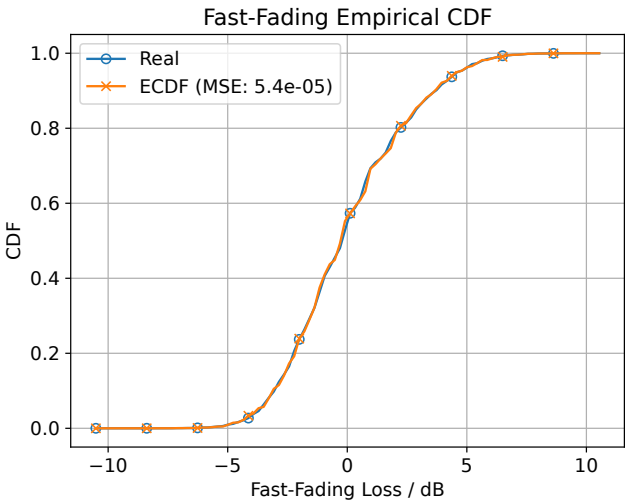


Figure 3: Fast-Fading Empirical CDF, Showing the Fitted Distribution, the Calculated MSE and the Original Data

3.3 Fast-Fading Distribution Precision

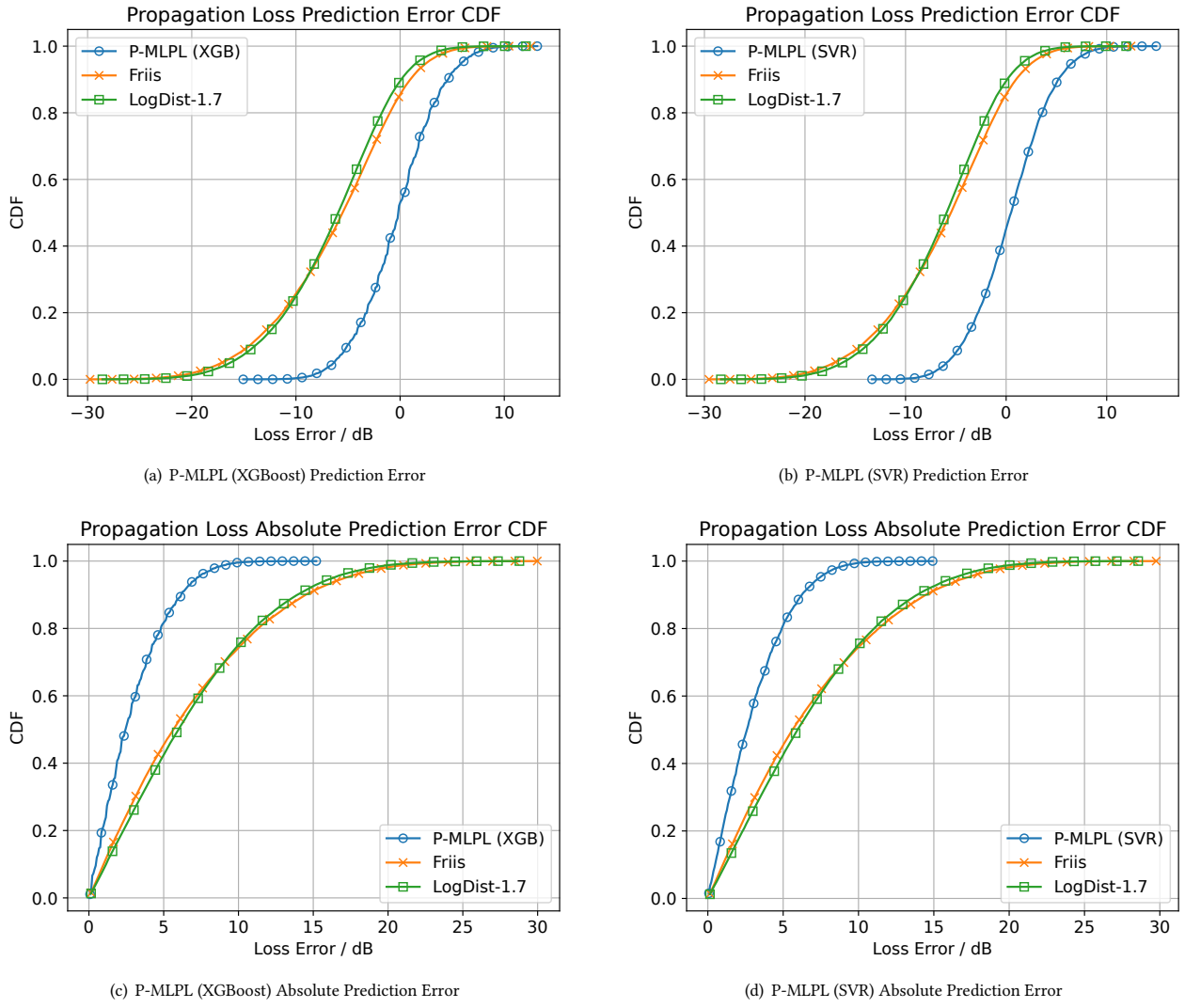
As indicated in Section 2.4, the fast-fading empirical CDF was fitted with the fast-fading values in the dataset. The precision of the fitted distribution was determined by comparing the CDF values of the fast-fading empirical CDF and the histogram of the original data. In this sense, we divided them into 30 bins and calculated the MSE between the CDF values of the fitted empirical CDF and the original data histogram.

The results of the fast-fading empirical CDF fitting are presented in Figure 3, showing the fast-fading empirical CDF, the real fast-fading values in the dataset, and the calculated MSE. It can be seen that the **fast-fading empirical CDF was perfectly fitted to the original data**, achieving an MSE = 5.4×10^{-5}. Thus, the empirical CDF provides a perfect representation of the fast-fading values in the dataset.

3.4 P-MLPL Model Precision

In addition to the individual evaluation of the path loss and fast-fading models, we also evaluated the precision of the full P-MLPL model. In this sense, we considered the two path loss ML models based on the XGBoost and SVR learning algorithms, which were combined with the best-fitted fast-fading distribution determined in Section 3.3.

The precision of the P-MLPL model was evaluated by comparing the propagation loss values in the test set to the corresponding estimations by the P-MLPL model. For each propagation loss value



(a) P-MLPL (XGBoost) Prediction Error

(b) P-MLPL (SVR) Prediction Error

(c) P-MLPL (XGBoost) Absolute Prediction Error

(d) P-MLPL (SVR) Absolute Prediction Error

Figure 4: CDF of the Real and Absolute Propagation Loss Prediction Errors by the P-MLPL Model, Compared to the Friis and the Log-Distance Models

L_i in the test set, we calculated the prediction error of the P-MLPL model $E_i^L = \hat{L}_i - L_i$ as the difference between the propagation loss predicted by the P-MLPL model $\hat{L}_i$ and L_i. Additionally, we calculated the prediction errors obtained with other propagation loss models available in ns-3, such as Friis and Log-Distance. We used the Log-Distance with path loss exponent $\gamma = 1.7$, which provided the closest match to the original data. We also added a Normal fast-fading distribution to both the Friis and Log-Distance models, with a mean $\mu = 0$ dB and a standard deviation $\sigma = 3$ dB that matches the distribution of fast-fading values in the dataset.

The CDF of the real and absolute prediction errors are shown in Figure 4. The results demonstrate that **the P-MLPL model is the most precise propagation loss model**, both when using the XGBoost and the SVR ML algorithms. In fact, analyzing the median value of the absolute error CDF, the P-MLPL model was able to predict the total propagation loss between the transmitter and the receiver with an error up to 0.5x than the baseline models. Moreover, the results show that the precision of the SVR path loss model is similar to the XGBoost model, with a median absolute error of 2.5 dB. In terms of the real prediction errors, the results reveal that half of the P-MLPL model predictions were either above or below the real propagation loss values, which matches the fitted fast-fading distribution. On the other hand, the Friis and Log-Distance baseline path loss models were too optimistic in this scenario, given that more than 90% of the predictions were below the real losses.

4 P-MLPL MODEL PERFORMANCE IN NS-3 SIMULATIONS

This section analyzes the performance of the P-MLPL model in the context of ns-3 simulations, in terms of the precision of the throughput obtained in simulation and the computational performance of the model.

4.1 ns-3 Simulation Set-Up and Parameters

To analyze the P-MLPL model performance, we used the experimental dataset described in Section 3.1. The set-up of the ns-3 simulation replicated the experimental environment where the dataset was collected, whose parameters are shown in Table 1. We used the ns-3.37 version configured with the default parameters, except the ones referred herein. We set the minimum RSSI preamble detection threshold to -90 dBm, so that the lowest data rates could be used, replicating the behavior of the experimental wireless networks cards.

In order to evaluate the precision of the throughput obtained in simulation using the P-MLPL model, we replicated the positions of the transmitter and receiver nodes available in the dataset. For each unique pair of node positions, the transmitter sent a UDP constant bitrate traffic of 54 Mbit/s to the receiver during 5 s, after an initialization period of 1 s to stabilize the Minstrel rate adaptation algorithm and the MAC queues. The generated traffic matched the traffic that was generated when collecting the dataset, which was selected to ensure that the connection is always fully loaded, as the offered load is always above the link capacity. Then, the average throughput received by the receiver node was calculated and saved.

In addition to the P-MLPL model, we performed the same test for the Friis and Log-Distance models, to compare the precision of the throughput obtained with the P-MLPL model against these baselines.

4.2 ns-3 Throughput Precision

After executing the tests explained in the previous section, we analyzed the precision of the throughput obtained in ns-3 simulations using the P-MLPL model. For each pair of node positions, we calculated the throughput error $E_i^T = \hat{T}_i - T_i$ as the difference between the throughput measured in ns-3 $\hat{T}_i$, and the respective real throughput T_i in the dataset.

The CDF of the real and absolute throughput errors are shown in Figure 5. The results demonstrate that the P-MLPL model was able to **accurately reproduce the experimental throughput** of the dataset. Analyzing the 90th percentile, both the P-MLPL models based on XGBoost and SVR predicted the throughput with an absolute error up to 2.5 Mbit/s. Although the XGBoost provided slightly better throughput precision than the SVR model, both ML models achieved higher precision compared to the Friis and Log-Distance models, which had absolute prediction errors up to 17.5 Mbit/s.

Moreover, these results confirm the conclusions drawn in Section 3. The P-MLPL model's increased precision in estimating the propagation loss translated into an increased throughput estimation precision. The optimistic estimations of the propagation loss by the Friis and Log-Distance models translated into optimistic throughput estimations.

4.3 P-MLPL Model Computational Performance

In order to evaluate the computational performance of the P-MLPL model, with and without caching propagation loss ML queries, we analyzed the duration of the ns-3 simulations using the P-MLPL model, as well as its memory usage. The duration of the ns-3 simulation corresponds to the time needed to finish the ns-3 simulation. The memory usage of the simulation consists of the memory used by the ns-3 simulation, in addition to the memory used by the ns3-ai framework and the external ML processes. The memory used by the ns-3 related processes was manually observed using the `htop` Linux tool.

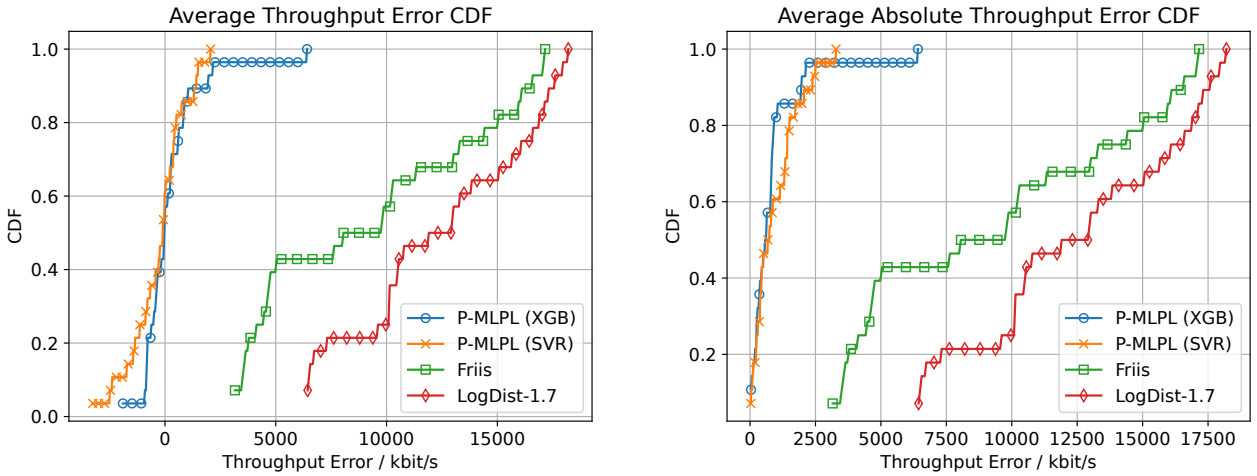
In this evaluation, we analyzed the computational performance metrics of the ns-3 simulations executed in Section 4.2. For each propagation loss model and cache configuration, we ran 10 ns-3 simulations runs using different seeds to calculate the 95% confidence intervals. The ns-3 simulations were executed one-by-one in an Ubuntu 22.04 with 8 CPU cores and 16 GB of RAM. The ns-3.37 simulator was configured and built with the *optimized* build profile.

The results of the tests can be seen in Table 3. The results show that, with the ML cache disabled, the P-MLPL model based on the SVR ML model was the fastest ML model, finishing a simulation run in 1 m 16 s on average. On the other hand, the P-MLPL model based on XGBoost was 25x slower than SVR, finishing a simulation run in 31 m 45 s on average. These results demonstrate that the **P-MLPL model based on SVR is a much faster alternative to the P-MLPL model based on XGBoost**, with similar propagation loss and throughput precision, making it the best choice for most of the scenarios.

With the ML cache enabled, the duration of the simulations reduced significantly. The P-MLPL model based on SVR reduced the duration of a simulation run to only 12 s on average, representing a speedup of 6.3x. Similarly, the P-MLPL model based on XGBoost reduced the duration to 13 s, representing a speedup of 147x. It is worth noting that the simulation set-up used in these tests benefits the caching technique, since the nodes adopt constant positions throughout each throughput monitoring period. Still, this condition is often used in many simulation scenarios, demonstrating the usefulness of this optimization.

In terms of memory usage, both ML models achieved similar results, with the P-MLPL model based on SVR using 184 MB of memory and the P-MLPL model based on XGBoost using 186 MB. Due to the simulation set-up used in these tests, the impact of the ML cache in terms of memory usage was negligible. In fact, since the dataset contains a limited set of unique positions, the size of the cache is negligible compared to the total memory used by the ns-3 simulation and the ML-related processes. In scenarios with more unique positions, the cache's size will increase proportionally to the number of positions observed in the simulation. Still, the effect of the cache in terms of simulation duration outweighs the increased memory usage.

To analyze the performance impact of the ML models in the overall ns-3 simulation duration and memory usage, we also executed the same tests with the baseline propagation loss models Friis and Log-Distance. Since these propagation loss models do not use external ML models nor caching, they provide reference values for ns-3 simulations without the ML overhead and cache



(a) Average Throughput Error

(b) Average Absolute Throughput Error

Figure 5: CDF of the Average Throughput Error Compared to the Real Throughput Collected in the Experimental Dataset

Table 3: Mean ns-3 Simulation Duration and Memory Usage, and 95% Confidence Intervals, for Different Propagation Loss Models and Cache Configurations

Propagation Loss Model	Simulation Duration		Memory Usage	
	Without Cache	With Cache	Without Cache	With Cache
P-MLPL (SVR)	1 m 16 s ± 3 s	12 s ± 2 s	184 MB ± 0 MB	184 MB ± 0 MB
P-MLPL (XGBoost)	31 m 45 s ± 28 s	13 s ± 1 s	186 MB ± 0 MB	186 MB ± 0 MB
Friis	19 s ± 2 s		34 MB ± 0 MB	
Log-Distance	27 s ± 1 s		34 MB ± 0 MB	

optimization. On average, when using the Friis propagation loss model, the ns-3 simulation run in 19 s, whereas when using the Log-Distance model, it finished in 27 s. In both models, the average memory usage of the ns-3 simulation was 34 MB.

These results demonstrate that, although the ML overhead increases the simulation duration, **caching ML queries not only mitigates this overhead, but also significantly improves the performance of the simulation**, even surpassing the performance of existing propagation loss models in ns-3 that do not use ML. Furthermore, the results also show that, although the Friis and Log-Distance models do not use external ML frameworks, the impact of repetitive calculations for the same set of node positions deteriorates the overall performance of ns-3 simulations. The application of the caching optimization technique to existing models in ns-3 is worthy of being explored in the future.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we proposed the position-based ML Propagation Loss Model (P-MLPL) to enable the creation of fast and more precise digital twins of wireless networks in ns-3, which is essential to develop and validate next-generation wireless networks solutions. Based on network traces collected in an experimental testbed, the P-MLPL model is able to estimate the propagation loss suffered by packets exchanged between a transmitter and a receiver, considering their absolute positions and the traffic direction. The propagation loss is calculated as the sum of the deterministic path loss plus a stochastic fast-fading value.

The results show that, for the scenario considered in this paper, the P-MLPL model can predict the propagation loss with a median error of 2.5 dB, which corresponds to 0.5x the error of the Friis and Log-Distance models available in ns-3. The increased precision in estimating the propagation loss translates into an increased precision of the throughput measured in ns-3, with an error up to 2.5 Mbit/s, when compared to the real values collected in the experimental testbed. Moreover, both the XGBoost and the SVR learning algorithms used to train the path loss model demonstrated similar estimation precision.

The use of an internal cache in the P-MLPL model to save repeated ML queries represented a speedup of up to 147x in the duration of a simulation run, with a similar memory usage. Despite the overhead of the ML queries via ns3-ai's shared memory, the computational performance of the cache-enabled P-MLPL model even surpassed the performance of existing propagation loss models in ns-3 that do not use ML. These results inspire the application of this optimization technique to the existing models in ns-3 to benefit from the same speedup.

As future work, we plan to submit the P-MLPL model to the ns-3 App Store. Additionally, we plan to explore the use of neural networks as an alternative to the XGBoost and SVR supervised learning algorithms. We also plan to evaluate the performance of the P-MLPL model in scenarios with different characteristics, including the presence of obstacles.

ACKNOWLEDGMENTS

This work has received funding from the European Union's Horizon Europe research and innovation programme under the Grant Agreement 101094831, project CONVERGE – Telecommunications and Computer Vision Convergence Tools for Research Infrastructures. The first author thanks the funding from FCT – Fundação para a Ciência e a Tecnologia under the PhD grant PD/BD/113819/2015.

REFERENCES

[1] Eduardo Nuno Almeida et al. 2022. Machine Learning Based Propagation Loss Module for Enabling Digital Twins of Wireless Networks in ns-3. In *Proceedings of the WNS3 2022 (WNS3 2022)*. ACM, 17–24. https://doi.org/10.1145/3532577.3532607

[2] Eduardo Nuno Almeida et al. 2022. *ML Propagation Loss Model for ns-3*. GitLab. https://gitlab.com/inesctec-ns3/ml-propagation-loss-model

[3] Eduardo Nuno Almeida, Rui Campos, and Manuel Ricardo. 2022. Traffic-Aware UAV Placement Using a Generalizable Deep Reinforcement Learning Methodology. In *2022 IEEE Symposium on Computers and Communications (ISCC)*. 1–6. https://doi.org/10.1109/ISCC55528.2022.9912770

[4] Eduardo Nuno Almeida, Kelwin Fernandes, Francisco Andrade, Pedro Silva, Rui Campos, and Manuel Ricardo. 2019. A Machine Learning Based Quality of Service Estimator for Aerial Wireless Networks. In *2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, 1–6. https://doi.org/10.1109/WiMOB.2019.8923217

[5] Gregor Cerar, Halil Yetgin, Mihael Mohorčič, and Carolina Fortuna. 2021. Machine Learning for Wireless Link Quality Estimation: A Survey. *IEEE Communications Surveys & Tutorials* 23, 2 (2021), 696–728. https://doi.org/10.1109/COMST.2021.3053615

[6] Renato Cruz, Helder Fontes, José Ruela, Manuel Ricardo, and Rui Campos. 2020. On the Reproduction of Real Wireless Channel Occupancy in ns-3. In *Proceedings of the 2020 Workshop on ns-3*. 41–48. https://doi.org/10.1145/3389400.3393754

[7] FED4FIRE+. 2018. *The Largest Federation of Testbeds in Europe*. https://www.fed4fire.eu/

[8] Helder Fontes, Rui Campos, and Manuel Ricardo. 2017. A Trace-Based ns-3 Simulation Approach for Perpetuating Real-World Experiments. In *Proceedings of the Workshop on ns-3*. 118–124. https://doi.org/10.1145/3067665.3067681

[9] Thomas R. Henderson, Mathieu Lacage, George F. Riley, Craig Dowell, and Joseph Kopena. 2008. Network Simulations with the ns-3 Simulator. *SIGCOMM Demonstration* 14, 14 (2008), 527.

[10] Latif U. Khan, Zhu Han, Walid Saad, Ekram Hossain, Mohsen Guizani, and Choong Seon Hong. 2022. Digital Twin of Wireless Systems: Overview, Taxonomy, Challenges, and Opportunities. *IEEE Communications Surveys & Tutorials* 24, 4 (2022), 2230–2254. https://doi.org/10.1109/COMST.2022.3198273

[11] Wahab Khawaja, Ismail Guvenc, David W. Matolak, Uwe-Carsten Fiebig, and Nicolas Schneckenburger. 2019. A Survey of Air-to-Ground Propagation Channel Modeling for Unmanned Aerial Vehicles. *IEEE Communications Surveys & Tutorials* 21, 3 (2019), 2361–2391. https://doi.org/10.1109/COMST.2019.2915069

[12] Vítor Lamela, Helder Fontes, Tiago Oliveira, José Ruela, Manuel Ricardo, and Rui Campos. 2019. *SIMBED - Offline Real-World Wireless Networking Experimentation Using ns-3*. Zenodo. https://doi.org/10.5281/zenodo.2634272

[13] Siyuan Li, Xi Lin, Jun Wu, Ali Kashif Bashir, and Raheel Nawaz. 2022. When Digital Twin Meets Deep Reinforcement Learning in Multi-UAV Path Planning. In *Proceedings of the 5th International ACM Mobicom Workshop on Drone Assisted Wireless Communications for 5G and Beyond*. 61–66. https://doi.org/10.1145/3555661.3560865

[14] Yiwen Wu, Ke Zhang, and Yan Zhang. 2021. Digital Twin Networks: A Survey. *IEEE Internet of Things Journal* 8, 18 (2021), 13789–13804. https://doi.org/10.1109/JIOT.2021.3079510

[15] Hao Yin, Pengyu Liu, Keshu Liu, Liu Cao, Lytianyang Zhang, Yayu Gao, and Xiaojun Hei. 2020. ns3-ai: Fostering Artificial Intelligence Algorithms for Networking Research. In *Proceedings of the 2020 Workshop on ns-3*. 57–64. https://doi.org/10.1145/3389400.3389404