Date: 27/7/2023

Report Structure:

Introduction

Article Introduction & Contribution in Smart Cities:

The paper “Position-Based Machine Learning Propagation Loss Model Enabling Fast Digital Twins of Wireless Networks in ns-3” [1] introduces the concept of digital twins as a hybrid approach that combines the advantages of network simulators and experimental testbeds. Digital twins consist of digital models that replicate the behavior of physical systems and dynamic conditions of experimental environments. They can be used to validate networking solutions and evaluate their performance in simulated environments that realistically replicate the dynamic conditions of physical environments. The paper focuses on the wireless channel model, which is a key component of wireless network digital twins. It proposes the Position-based ML Propagation Loss (P-MLPL) model for ns-3, which improves the precision of the ML-based Propagation Loss (MLPL) model by considering the absolute positions of nodes and the traffic direction. The P-MLPL model allows for the development of fast and more precise digital twins of wireless networks in ns-3, enabling the validation of novel solutions and the evaluation of their performance in realistic conditions. The paper also discusses an optimization technique that adds an internal cache to the P-MLPL model to improve computational performance. The model is validated and evaluated through experiments and simulations, and the paper concludes with a summary of the results and suggestions for future work. The paper focuses on the Position-Based ML Propagation Loss (P-MLPL) model, enhancing the ML-based Propagation Loss model (MLPL). P-MLPL considers absolute node positions and traffic direction to enable more precise wireless network simulations in complex urban environments.

The integration of digital twins with P-MLPL offers key contributions to smart cities:

1. Enhanced Simulation Accuracy: P-MLPL improves precision, optimizing network infrastructure and robust communication systems in dynamic scenarios.
2. Cost-Effective Planning: Digital twins reduce reliance on expensive testbeds, enabling cost-effective testing and validation.
3. Sustainable Infrastructure: Accurate evaluation in digital twins allows smart cities to design adaptive and resource-efficient infrastructure.

Project Objectives:

The primary objectives of this project are:

- ✓ Develop the Position-Based ML Propagation Loss (P-MLPL) model for ns-3, creating fast and precise digital twins of wireless networks.
- ✓ Address limitations of existing propagation loss models by employing machine learning techniques for accurate custom models of the wireless channel.
- ✓ The validation of the model on a dataset of network traces collected in an experimental testbed.
- ✓ The comparison of the performance of the model with other propagation loss models in ns-3.
- ✓ Demonstrate the practical applications of digital twins in smart cities, where realistic network simulations are crucial for efficient and reliable communication.

Relevance of Networking Concept

The network architecture used is a simple ad hoc wireless network with two nodes: a transmitter (Tx) and a receiver (Rx). The propagation loss model is used to calculate the path loss between the transmitter and receiver nodes, which affects the signal strength and thus the throughput in the wireless link. Holding significant relevance to the smart city environment due to its potential impact on performance & Quality of Service (QoS) of wireless communication systems within smart cities.

1- Performance Improvement:

Smart cities rely on wireless communication systems for data exchange among IoT devices and infrastructure. Ensuring reliable and efficient communication is crucial for real-time data processing and decision-making. The Machine Learning Based Propagation Loss Model improves the accuracy of predicting path loss and signal strength, leading to optimized resource allocation and enhanced communication performance in terms of throughput, latency, and reliability.

2- Quality of Service (QoS) Enhancement:

Smart city applications often have diverse QoS requirements. For example, applications like real-time traffic monitoring, emergency response systems, and healthcare services demand low latency and high reliability. On the other hand, less critical applications may tolerate higher latency and lower data rates. The ML-based propagation loss model can help in predicting the link quality more accurately, enabling better QoS provisioning by tailoring communication parameters to meet the specific requirements of each application.

Literature Review

The paper "Position-Based Machine Learning Propagation Loss Model Enabling Fast Digital Twins of Wireless Networks in ns-3" by Eduardo Nuno Almeida, Helder Fontes, Rui Campos, and Manuel Ricardo[1] proposes a new machine learning-based propagation loss model for the ns-3 network simulator. The model, called P-MLPL, is trained on a dataset of network traces collected in an experimental testbed. The model takes as input the absolute positions of the transmitter and receiver nodes, as well as the traffic direction, and outputs the estimated propagation loss.

The authors compare the performance of P-MLPL to two other propagation loss models in ns-3: the Friis model and the Log-Distance model. The results show that P-MLPL can predict the propagation loss with a median error of 2.5 dB, which is significantly lower than the error of the Friis and Log-Distance models. Moreover, ns-3 simulations with P-MLPL estimated the throughput with an error up to 2.5 Mbit/s, when compared to the real values measured in the testbed.

Here are some additional related papers:

- A Note on a Simple Transmission Formula: https://ieeexplore.ieee.org/document/1697062[2]
- An improved IEEE 802.16 WiMAX module for the ns-3 simulator: https://dl.acm.org/doi/10.4108/ICST.SIMUTOOLS2010.8653[3]

Proposed Methods *for Propagation Loss Prediction:*

1- Friis Propagation Model: Calculates received power in free space using transmitted power, antenna gains, wavelength, and distance. Ideal for line-of-sight communication.

2- Log-Distance Model: Predicts signal loss in indoor areas, considers path loss exponent, reference distance, and shadowing effects.

Methods Results*:*

1- Friis Model: Simple, accurate for line-of-sight, limited by free-space assumptions and non-ideal antennas.

2- Log-Distance Model: Useful for indoor, obstructed scenarios, restricted to certain frequencies, lacks multipath consideration.

Gaps:

1- Lack of experimental validation for both models.

2- Friis limitations not addressed: deviations, obstacles, and corrections for non-isotropic antennas.

3- Log-Distance lacks accuracy details for different indoor environments & multipath effects.

The selected paper[1] narrows the lack of experimental validation gap by applying digital twins concept to realistically replicate the dynamic conditions of physical environments.

Methodology

The proposed approach in this paper is to utilize the Position-based Machine Learning Propagation Loss (P-MLPL) model for the creation of digital twins of wireless networks in ns-3. This methodology involves several steps: data collection from experimental testbeds, preprocessing of the dataset, training of ML models for path loss and fast-fading, implementation of the P-MLPL model in ns-3, validation of the model's precision through a test suite, and performance optimization. By combining simulators and experimental testbeds, the P-MLPL model provides a way to accurately evaluate next-generation wireless networks in realistic conditions.

Evaluation Method

The outcomes will be evaluated by comparing the P-MLPL model's predicted propagation loss values with the actual values in the test set. Prediction errors will be calculated as the differences between the predicted and actual propagation loss values. CDFs will be used to visualize the distribution of both real and absolute prediction errors, providing insights into the model's accuracy and potential biases. By examining these CDFs, we can assess the model's precision and its ability to capture variations in real-world propagation loss. Additionally, comparisons will be made with baseline models (Friis and Log-Distance) with the inclusion of a Normal fast-fading distribution to ensure fair assessments of the P-MLPL model's performance against existing methods (we also introduced another 2 ml_algorithms for the p-mlpl model).

Expected Outcomes

P-MLPL model is expected to result in more accurate path loss prediction & consequently accurate propagation loss than baseline models(also the introduced new 2 ml_algorithms are expected to result in more accurate path loss prediction & consequently accurate propagation loss than the selected papers' models).

NS3 Implementation: Modules, Assumptions, Modifications

P-MLPL Model Architecture:

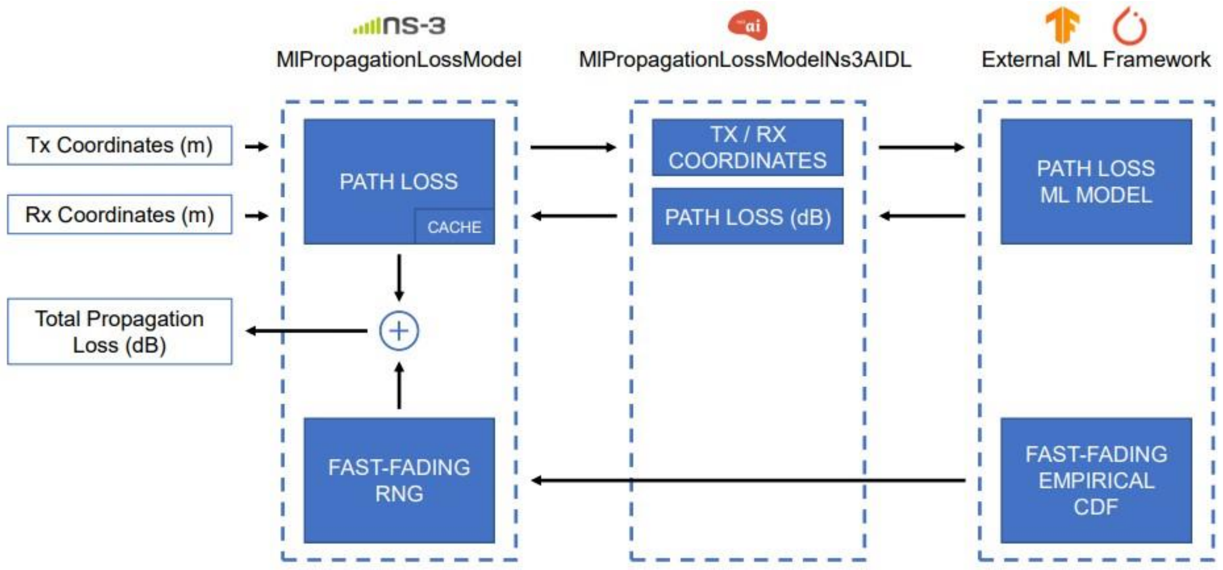


Figure 1: Position-Based MLPL Model Architecture Showing Interactions Among It's Different Modules & Components [1]

The P-MLPL (Path Loss with Machine Learning and Fast-Fading) model architecture in ns-3 consists of two main components: the supervised learning-based path loss model and the stochastic fast-fading model. The path loss model estimates the deterministic path loss between a transmitter and receiver using an external machine learning algorithm. This external model is integrated into ns-3 through the ns3-ai module, enabling data exchange via shared memory. To improve computational efficiency, the P-MLPL model incorporates an internal cache to store recent path loss queries, avoiding repeated costly queries to the ML model. The stochastic fast-fading component simulates real-world wireless channel variations. It generates pseudo-random samples based on an empirical Cumulative Distribution Function (CDF) that fits the fast-fading dataset. The model guarantees consistent simulation outcomes by managing the random variability of fast-fading through a controlled Random Number Generator (RNG) internal to ns-3, synchronizing its seed with other RNGs in the simulation. The total propagation loss for a transmitter-receiver pair is obtained by summing the path loss estimation with a randomly generated fast-fading sample. Subsequently, the model delivers the combined transmission power and calculated loss back to ns-3, where it is then incorporated with antenna gains according to the wireless models specified for the simulation. This approach ensures simulation reproducibility while effectively representing real-world wireless channel characteristics within ns-3's computational framework. This architecture provides a realistic representation of wireless channel behavior in ns-3 while mitigating computational overhead.

Modules Explanation:

a) Deterministic Path Loss ML Model:

The deterministic path loss ML model in the P-MLPL architecture utilizes supervised learning algorithms, including the Gradient Boosting algorithm (GBD), eXtreme Gradient Boosting (XGBoost), Support Vector Regression (SVR), and the Random Forest algorithm (RF). These algorithms are trained using a dataset of path loss samples. GBD and RF are ensemble learning methods that build multiple decision trees and combine their predictions to achieve improved accuracy. XGBoost, on the other hand, is a more optimized and scalable version of GBD, offering better performance. SVR, a variant of Support Vector Machines, is used to find a regression function that best fits the path loss data. The supervised learning process enables the model to learn and generalize from the dataset, improving its accuracy in predicting path loss for different transmitter-receiver pairs in various scenarios. By using the trained ML model, the P-MLPL model can efficiently estimate deterministic path loss values, a critical component in wireless communication simulations.

b) Stochastic Fast-Fading Model:

In the P-MLPL architecture, the stochastic fast-fading model is designed to capture the variability introduced by fast-fading phenomena in wireless channels. Modeled as a stochastic ergodic process, this model generates pseudo-random samples based on an empirical Cumulative Distribution Function (CDF) fitted to the dataset samples representing fast-fading effects. By doing so, it can accurately simulate the fluctuations experienced by wireless signals due to fading. The model is controlled by a pseudo-random seed, ensuring reproducibility and consistency with ns-3 simulations across different runs.

c) P-MLPL Propagation Loss Dataset:

The P-MLPL propagation loss dataset serves as the foundation for training the deterministic path loss ML model and modeling the stochastic fast-fading component. This dataset is stored in a CSV file format and contains propagation loss samples, as well as node positions in Cartesian coordinates (x, y, z) relative to a given origin point. Additionally, the dataset offers two formats for propagation loss values: either direct dB values or values calculated based on received signal power/SNR and noise floor. To improve the training of the ML model and the accuracy of the simulations, any outliers present in the dataset are removed. Furthermore, the dataset is carefully decomposed into path loss and fast-fading components to enable independent modeling and analysis.

d) P-MLPL Module Structure:

The P-MLPL module is structured to encompass the entire propagation loss modeling process, including the deterministic path loss ML model (GBD, XGBoost, SVR, RF), the stochastic fast-fading model, and associated helper scripts. The core functionality resides in the P-MLPL propagation loss model implemented as the MlPropagationLossModel class, which inherits from the PropagationLossModel base class in the ns-3 framework. The module also includes various helper scripts, such as train_ml_propagation_loss_model.py, responsible for training the path loss ML model (GBD, XGBoost, SVR, RF) and the fast-fading empirical CDF. Another script, run_ml_propagation_loss_model.py, is employed to manage shared memory for data exchange with external ML frameworks. This modular structure ensures efficient coordination and operation of the different components, streamlining the overall process of propagation, loss estimation and simulation.

e) Validation Test Suite:

To verify the accuracy and functionality of the P-MLPL model, a comprehensive test suite named MlPropagationLossModelTest has been developed. This validation suite rigorously tests the data exchange between the P-MLPL model and external ML frameworks, ensuring seamless communication and integration. Additionally, the test suite is instrumental in validating the correctness of the propagation loss values calculated by the P-MLPL model (GBD, XGBoost, SVR, RF). By running various scenarios and comparing the model's results against ground truth data, any potential discrepancies or issues can be identified and addressed. This robust validation process instills confidence in the reliability and accuracy of the P-MLPL model, making it a valuable tool for wireless communication simulations and research.

Assumptions:

- **Outlier Handling:**

Data points with z-scores exceeding 5 are identified as outliers and subsequently excluded from the dataset to improve model performance.

- **Fast-Fading Distribution:**

The assumption is that fast-fading values adhere to a statistical distribution centered around a mean of 0 dB.

- **Mean Propagation Loss:**

The path loss for each pair of node positions is calculated as the mean propagation loss value using all samples of that pair of positions. This assumption implies that the mean propagation loss represents the typical or average loss experienced between the specific node positions.

- **Decomposition of Loss:**

The total propagation loss is assumed to be composed of two components: the path loss, which is deterministic, and the fast-fading component, which is stochastic. This assumption enables a better understanding of the factors contributing to the overall loss and aids in developing separate models for each component.

- **Path Loss & Fast-fading Components Based Training:**

After decomposing the total propagation loss into path loss and fast-fading components, the isolated path loss and fast-fading values are used to train the corresponding ML models. This assumption implies that the ML models can be effectively trained using these components separately, leading to improved overall model performance.

Modifications Explanation:

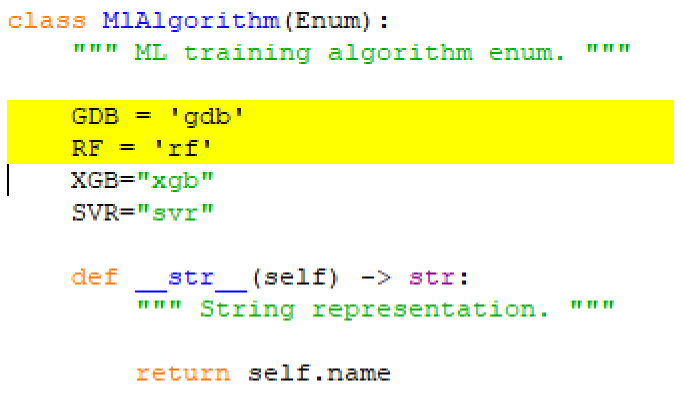
- **Utilizing Python visualization packages to represent throughput between transmitter (Tx) & receiver (Rx) nodes.**
- **Introducing alternative models such as GDB and RF, in addition to the previously used SVR and XGP models.**
- **Aggregating the average throughput error of all models into a single visualization plot to determine the best-performing model.**
- **Discovering that the RF model performs better in predicting propagation loss at 98% of the data with around 2000 kb/sec throughput error.**

Code Documentation & Contributions

Training code script" train_ml_propagation_loss_model.py "

- Import required libraries and modules.
- Define a function called train_ml_propagation_loss_model, which is the main function responsible for training the ML propagation loss model.
- The function takes three parameters: dataset (the name of the dataset used for training), mlpl_model (the MLPL model - either "Distance-MLPL" or "Position-MLPL"), and ml_algorithm (the ML algorithm used for training the MLPL model).
- "The distance is the main feature that is used for training so when the position dataset is chosen it is converted into distances by calculating them "

- “also the new ml-algorithms is added easily due to the object oriented structure of the project simply by adding the new model names into the ML Algorithm class and adding its interpretations in other scripts”



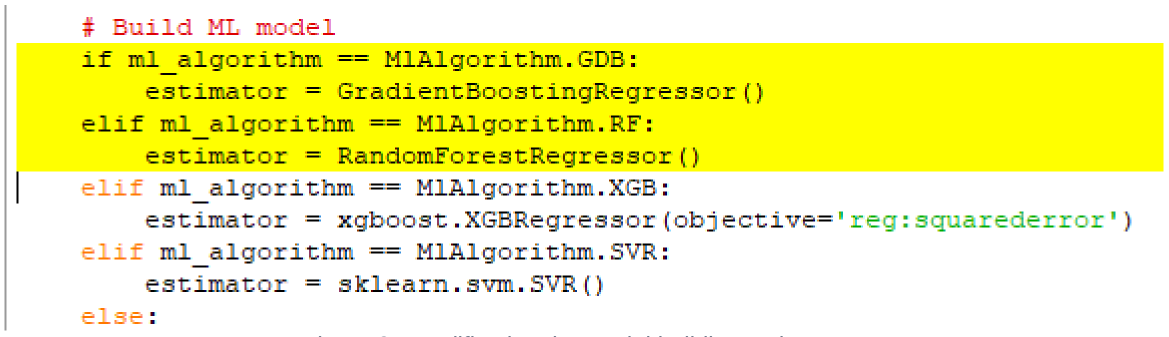
```
class MlAlgorithm(Enum):
    """ ML training algorithm enum. """

    GDB = 'gdb'
    RF = 'rf'
    XGB="xgb"
    SVR="svr"

    def __str__(self) -> str:
        """ String representation. """

        return self.name
```

Figure 2 : modifications in the ML Algorithms class



```
    # Build ML model
    if ml_algorithm == MlAlgorithm.GDB:
        estimator = GradientBoostingRegressor()
    elif ml_algorithm == MlAlgorithm.RF:
        estimator = RandomForestRegressor()
    elif ml_algorithm == MlAlgorithm.XGB:
        estimator = xgboost.XGBRegressor(objective='reg:squarederror')
    elif ml_algorithm == MlAlgorithm.SVR:
        estimator = sklearn.svm.SVR()
    else:
```

Figure 3 : modification the model building script

- The function begins by creating subdirectories for storing the results of the training process.
- It loads the dataset for training and splits it into train and test sets.
- It creates a unique positions dataset (presumably for the MLPL model).
- The ML model training begins. Two ML models are trained: one for path loss and another for fast fading.
- The function then calculates the predictions and errors of the ML models on the test set.
- It also calculates the path loss errors for the baselines (non-ML methods) using Wi-Fi parameters.
- The results are plotted and saved in specific directories for analysis.
- The if __name__ == '__main__': block parses command-line arguments to get the dataset name, MLPL model, and ML algorithm, and then calls the train_ml_propagation_loss_model function with these arguments.

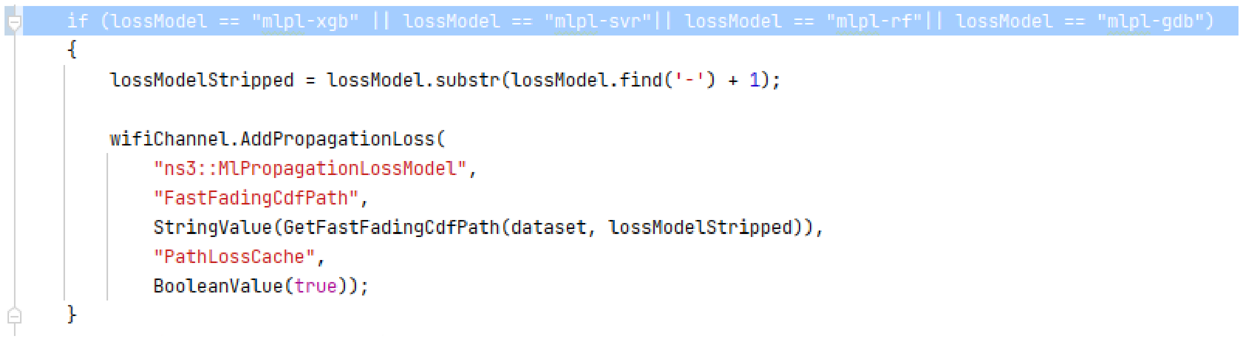
Run code script ” run_ml_propagation_loss_model.py ”

- Import required libraries and modules, including py_interface for ns-3 communication.
- Define three custom data structures (MlFeature, MlPredicted, and MlTarget) that represent the input features, ML model predictions, and real target values, respectively.
- The main function is named run_ml_propagation_loss_model.
- The function takes three parameters: dataset (the name of the dataset used for running the ML model), mlpl_model (the MLPL model - either "Distance-MLPL" or "Position-MLPL"), and ml_algorithm (the ML algorithm used during training).
- The function initializes the ns-3 communication and waits for the ns-3 simulation to start.
- Once the simulation starts, the script continuously receives Tx/Rx node positions from ns-3.
- It converts the positions to the appropriate format based on the MLPL model ("Distance-MLPL" or "Position-MLPL").
- The function then uses the trained ML propagation loss model to predict the path loss values based on the input positions.
- The predicted path loss values are sent back to ns-3, allowing the simulation to utilize the predicted path loss values.
- The script handles any exceptions that may occur and gracefully terminates when the ns-3 simulation is finished or when the user interrupts the script with CTRL-C.
- The if __name__ == '__main__': block parses command-line arguments to get the dataset name, MLPL model, and ML algorithm, and then calls the run_ml_propagation_loss_model function with these arguments.

The Simulation Script.

- Importing Required Modules and Libraries: The script includes several ns-3 modules for applications, core functionality, flow monitoring, internet, mobility, network, propagation, and Wi-Fi.
- Global Variables and Parameters: The script defines several global variables and parameters, such as monitoring time per position, warm-up time before starting monitoring, dataset information, and result file paths.
- Auxiliary Functions: The script contains some utility functions for handling node positions, resetting counters, updating throughput results, etc.

- Main Function (main): The main function of the script starts by parsing command-line arguments. The user can provide the following arguments:
- "lossModel": Propagation loss model to be used for the simulation. Options include "mlpl-xgb," "mlpl-svr," "mlpl-rf," "mlpl-gdb," "friis," and "log-dist-1.7."



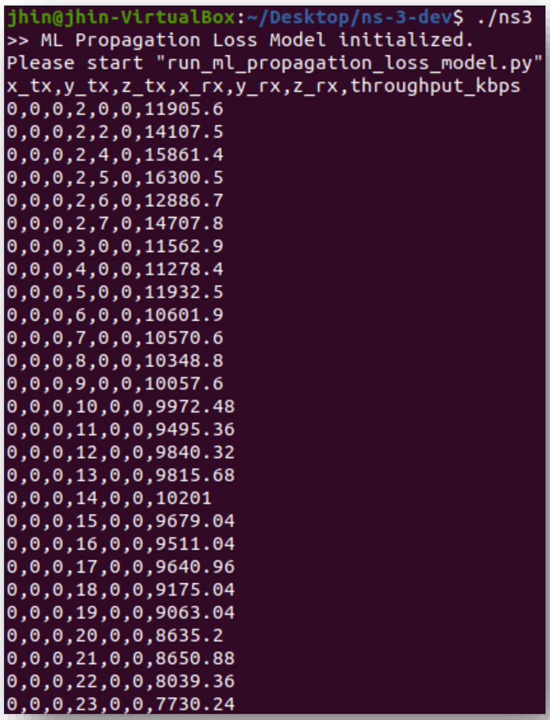
```
if (lossModel == "mlpl-xgb" || lossModel == "mlpl-svr"|| lossModel == "mlpl-rf"|| lossModel == "mlpl-gdb")
{
    lossModelStripped = lossModel.substr(lossModel.find('-') + 1);

    wifiChannel.AddPropagationLoss(
        "ns3::MlPropagationLossModel",
        "FastFadingCdfPath",
        StringValue(GetFastFadingCdfPath(dataset, lossModelStripped)),
        "PathLossCache",
        BooleanValue(true));
}
```

Figure 4: modification in the simulation code

- "dataset": The name of the dataset to be used for the simulation. The dataset should contain Tx/Rx positions.
- "nRun": Simulation run seed (for confidence interval).
- "verbose": A boolean flag to enable verbose output.
- Simulation Setup:
- Nodes (0: Tx | 1: Rx) are created using a constant position mobility model.
- Wi-Fi parameters are set based on the provided dataset.
- The selected propagation loss model is added to the Wi-Fi channel.
- Application Setup:
- On/Off applications are used to generate traffic between the Tx and Rx nodes with a specified data rate.
- A packet sink application is used on the Rx node to measure the received throughput.
- Throughput Monitoring: The script schedules monitoring sessions for each pair of Tx/Rx positions in the dataset. For each monitoring session, the throughput is calculated and updated in a results CSV file.
- Simulation Execution: The simulation is scheduled to run until the last monitoring session is completed, and the average throughput data is saved in the results CSV file.

Simulation Results and Evaluation



```
jhin@jhin-VirtualBox:~/Desktop/ns-3-dev$ ./ns3
>> ML Propagation Loss Model initialized.
Please start "run_ml_propagation_loss_model.py"
x_tx,y_tx,z_tx,x_rx,y_rx,z_rx,throughput_kbps
0,0,0,2,0,0,11905.6
0,0,0,2,2,0,14107.5
0,0,0,2,4,0,15861.4
0,0,0,2,5,0,16300.5
0,0,0,2,6,0,12886.7
0,0,0,2,7,0,14707.8
0,0,0,3,0,0,11562.9
0,0,0,4,0,0,11278.4
0,0,0,5,0,0,11932.5
0,0,0,6,0,0,10601.9
0,0,0,7,0,0,10570.6
0,0,0,8,0,0,10348.8
0,0,0,9,0,0,10057.6
0,0,0,10,0,0,9972.48
0,0,0,11,0,0,9495.36
0,0,0,12,0,0,9840.32
0,0,0,13,0,0,9815.68
0,0,0,14,0,0,10201
0,0,0,15,0,0,9679.04
0,0,0,16,0,0,9511.04
0,0,0,17,0,0,9640.96
0,0,0,18,0,0,9175.04
0,0,0,19,0,0,9063.04
0,0,0,20,0,0,8635.2
0,0,0,21,0,0,8650.88
0,0,0,22,0,0,8039.36
0,0,0,23,0,0,7730.24
```

Figure 5: This throughput monitoring of the tx and rx nodes which shows the change in throughput while the mobility of the nodes.

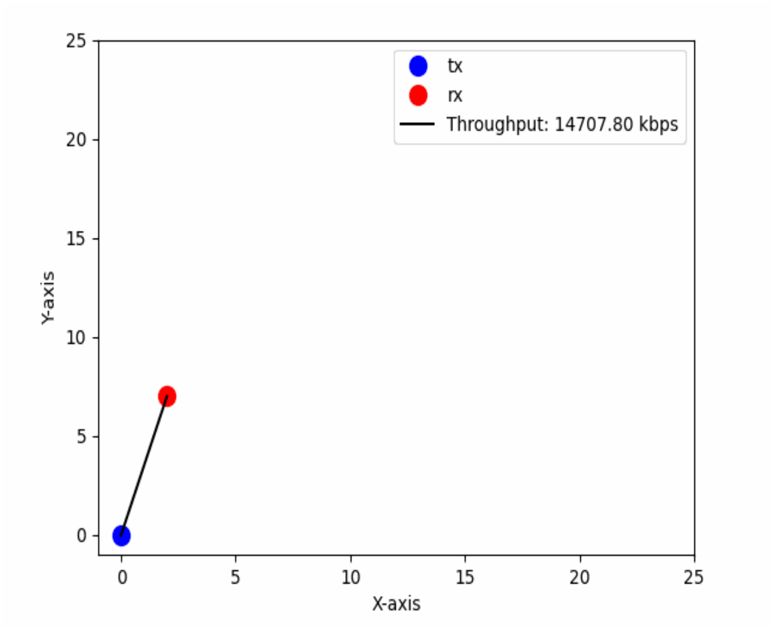


Figure 6 Visualization of the simulation results

Evaluation & Behavioral Analysis

The P-MLPL model, employing the GDB and RF machine learning algorithms, was thoroughly examined, revealing its superiority as a propagation loss model. The research demonstrated that the P-MLPL model achieved predictions of the total propagation loss between the transmitter and receiver with an error rate up to 0.5 times lower than that of the baseline models as shown in [Figure -7].
Which surprisingly matches the original paper results, but how it is comparing to the original paper results?!

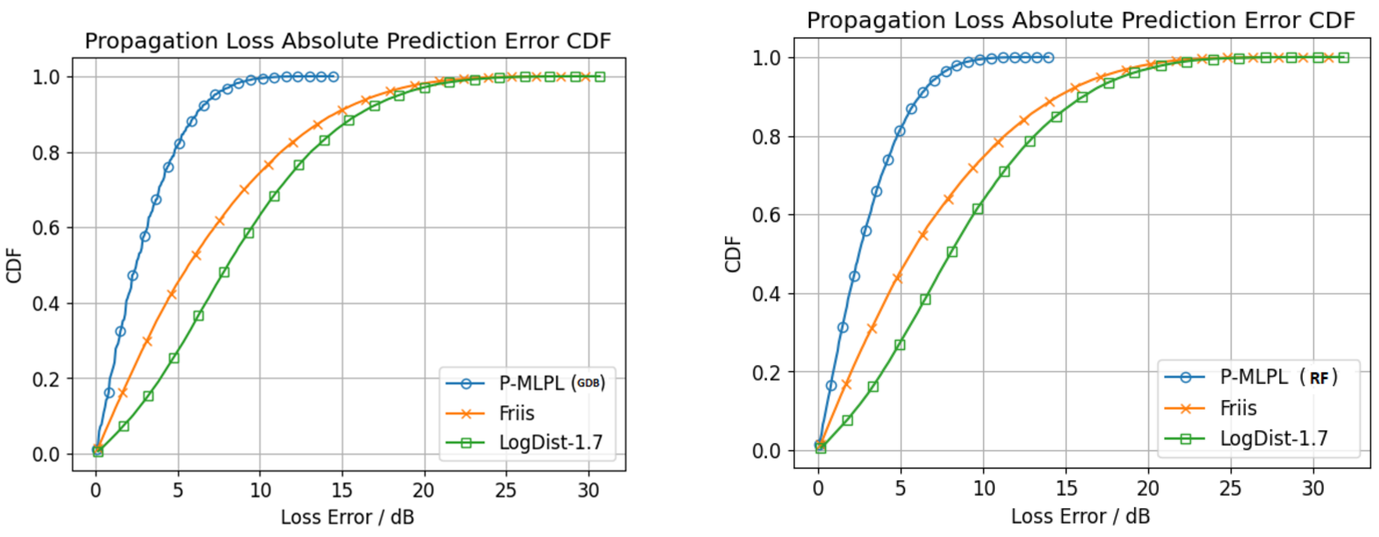


Figure 7: CDF of the Real and Absolute Propagation Loss Prediction Errors by the P-MLPL Model, Compared to the Friis and the Log-Distance Models

We conducted multiple tests for each ml-algorithm on ns-3 simulations using the P-MLPL model. We compared the predicted throughput with the actual throughput in the dataset and calculated the throughput error for each pair of node positions. The results, shown in [Figure -8], indicate that the P-MLPL model accurately reproduced the experimental throughput.

Looking specifically at the 90th percentile, All the P-MLPL models based on XGBoost, SVR, , RF and GDB predicted the throughput with an absolute error of up to 2.5 Mbit/s .Although our presented RF model provided better throughput precision than the SVR,XGB,GDB model. All these ML models outperformed the Friis and Log-Distance models, which had larger absolute prediction errors of up to 16 Mbit/s. The increased precision of the P-MLPL model in estimating propagation loss led to more accurate throughput estimations, while the Friis and Log-Distance models tended to overestimate throughput due to optimistic propagation loss estimations.

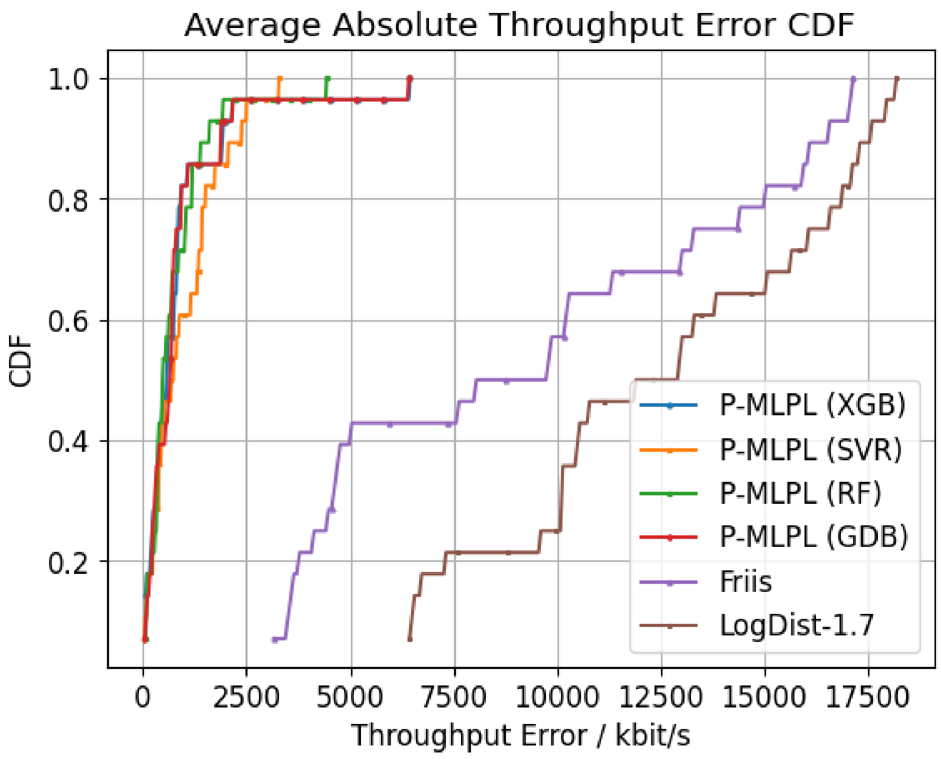


Figure 8: Cumulative Distribution Function (CDF) showing the absolute average throughput error in comparison to the actual throughput collected from the experimental dataset.

Challenges and Solutions

Challenges:

- No real-time visualization tools; resorting to offline methods.
- Difficult to simulate precise fast fading due to complexity.
- New module integration needs extensive code modifications.

Solutions:

- Explore alternative offline visualization techniques.
- Address fast fading complexity with improved simulation approaches.
- Carefully integrate new modules, considering compatibility.

Future Work and Enhancements

- Real-Time Visualization: Incorporate real-time visualization tools for enhanced data analysis and insights.
- Explore Different ML Algorithms: Investigate the application of time series or neural networks to further improve P-MLPL's performance and accuracy.
- Larger & More Representative Data: Utilize larger and more diverse datasets to train and validate P-MLPL for better generalization and real-world applicability.
- Improve ml accuracy.
- Consider more parameters.

Conclusion

The P-MLPL model presents a novel approach for creating fast and accurate digital twins of wireless networks in ns-3. Utilizing machine learning and considering node positions and traffic direction, P-MLPL outperforms existing propagation loss models, such as Friis and Log-Distance, by providing more precise predictions and accurate throughput estimations. Additionally, the integration of two new machine learning models, GDB and RF, alongside previously used SVR and XGP models, further enhances the performance of P-MLPL, showcasing similar results. By addressing the limitations and employing digital twins, the model significantly improves the realism and accuracy of wireless network simulations, showcasing its practical applications in smart cities for efficient communication and tailored Quality of Service (QoS). Challenges related to real-time visualization and fast-fading complexity can be overcome through alternative visualization techniques and improved simulation approaches. The integration of these additional machine learning models makes P-MLPL a valuable contribution to the field of wireless network simulations.

References

[1] Almeida, Eduardo Nuno, Helder Fontes, Rui Campos, and Manuel Ricardo. “Position-Based Machine Learning Propagation Loss Model Enabling Fast Digital Twins of Wireless Networks in Ns-3.” In Proceedings of the 2023 Workshop on Ns-3, 69–77. Arlington VA USA: ACM, 2023. https://doi.org/10.1145/3592149.3592150.

[2] Zhang, Yan, Jinxiao Wen, Guanshu Yang, Zunwen He, and Jing Wang. “Path Loss Prediction Based on Machine Learning: Principle, Method, and Data Expansion.” Applied Sciences 9, no. 9 (January 2019): 1908. https://doi.org/10.3390/app9091908.

[3] Losev, Ivan. “On Inductive Construction of Procesi Bundles.” arXiv.org, January 17, 2019. https://arxiv.org/abs/1901.05862v3.

[4] Levitskii, R. R., S. I. Sorokov, and O. R. Baran. “Reference Approach in Theory of Pseudospin Systems.” Condensed Matter Physics 3, no. 3 (2000): 515. https://doi.org/10.5488/CMP.3.3.515.