Towards Accurate, Data-Driven and Lightweight Digital Twins for Wireless Networks

Samir Si-Mohammed and Fabrice Theoleyre
ICube, CNRS / University of Strasbourg
Boulevard Sébastien Brant, 67412 Illkirch, France
Emails: simohammed@unistra.fr and fabrice.theoleyre@cnrs.fr

***Abstract*—Digital twins have recently emerged as a transformative approach to wireless networks, aimed at enhancing network efficiency. By replicating the physical network, they offer significant potential for the implementation of next-generation wireless networks. Digital twins make it possible to test decisions and evaluate their impact without disrupting actual network operations. However, digital twins are often based on simulations, and the accuracy of many physical (PHY) layer models is still low. This underlines the need for data-driven modeling with continuous calibration from the radio environment. In this paper, we propose a self-calibration approach where each radio link is represented by its own digital twin. Given the bursty nature of radio link quality, we introduce a self-adaptive method that selects the best prediction model on-the-fly for future short- and mid-term predictions. We validate our approach on a real network deployed in a testbed with very diverse links, on which we demonstrate the robustness of our self-adaptive model in predicting Packet Delivery Ratio (PDR) in both the short and medium term, giving the network sufficient time to anticipate and implement necessary reconfigurations.**

***Index Terms*—Digital Twins; Wireless Networks; Machine Learning; Regression; Data-oriented Modeling.**

I. Introduction

After having revolutionized Industry 4.0, Digital Twins (DTs) have recently gained much attention for networking [1]. A DT reflects in real-time the physical world and replicates digitally the state of a physical object. Inversely, any modification in the DT is replicated in the physical world. Unfortunately, radio links are known to be lossy, asymmetrical, and bursty [2]. Thus, estimating link characteristics in a wireless network DT is therefore of paramount importance.

Digital Twins are helpful to guide the network reconfiguration [3]. Since closed-formed equations often represent an unattainable goal, approximated models are used to explore alternative configurations In wireless networks, models (*e.g.*, routeNet-Erlang [4] Graph Neural Network (GNN) model) help to estimate per-flow performance metrics. The DT implements the so-called *what-if scenarios*: what would be the Key Performance Indicators when changing a specific parameter? Thus, the DT can help the network to operate permanently close to optimum.

The GNN models offer a good generalization property but poor observability for DT. Thus, simulation is extensively used in wireless networking [5]. With discrete-event simulation, we can reproduce per-packet action (transmission, collisions, etc.). Many radio propagation models have been proposed in the literature [6]. Unfortunately, simulations present poor accuracy and strongly depend on the PHY layer model [7]. If we want to use network simulation as a DT, we need to *synchronize* the DT with the real world. Measurements must *feed* the radio propagation models. Data-driven approaches such as Machine Learning are promising to self-tune radio propagation models [8]. However, acquiring PHY-layer metrics such as Signal Interference plus Noise Ratio (SINR) is challenging, and we prefer a technology-agnostic method.

Colosseum is a world-class research platform for 5G and beyond [9] that represents a pioneering piece of work in this direction. The platform is used for fast prototyping and evaluating 5G solutions in realistic conditions. Colosseum relies on a large-scale emulation platform: software radio nodes mimic the clients and the Base Stations. The system measures the channel response between each pair of nodes, and a Field-Programmable Gate Array (FPGA) is modified accordingly to digitally reproduce the initial situation. This platform is promising, but very expensive since it relies on dedicated and costly hardware that makes real-life deployments tricky.

Modern wireless networks support an increasing number of critical applications [10]. DT is a key enabler for predictive maintenance: with an estimation of the evolution of the Key Performance Indicators, we can predict when the system will behave below the expected standard. Forecasting QoS performance has received much attention in the past [11]. More precisely, the objective is to exploit a time series of measurements to predict the next values, for instance for a Wi-Fi network [12].

Ideally, we aim to develop an approach that enables the construction of an "agnostic" DT for any wireless network, *i.e.,* regardless of the underlying communication protocols or traffic types. This DT should accurately reflect real-time events occurring within the physical network, such as the reception of a packet, and be capable of predicting future events with a reasonable degree of certainty. Furthermore, the DT should maintain a balance between complexity and scalability, ensuring it remains practical and efficient for various network sizes and configurations.

In this paper, we propose to redesign the PHY model of the network simulator to create an accurate DT. Instead of relying on synthetic radio propagation models, we propose to feed the digital twin with measurements to tune automatically the DT models. The contributions of this paper are as follows:

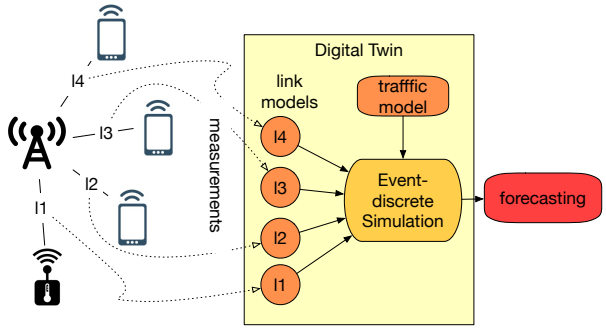

Fig. 1: Architecture of a Digital Twin for wireless networking

1) We analyze the characteristics of radio links in a real network and highlight their heterogeneity;
2) we compare different forecasting techniques and their ability to predict short and medium-term future (prediction property of the DT);
3) we propose a self-adaptive method that continuously selects the best model for a given link.

II. Filling the gap toward a real-time DT

A DT would be beneficial for its *what-if scenarios* in wireless networking. Measurements feed the digital twin (Fig. 1). More precisely, one model is maintained per radio link: it accumulates measurements for this specific link to train the model. Then, each link can be exploited to provide *what-if scenarios*, and more specifically for i) network simulation to predict the Key Performance Indicators (KPIs) in the short/mid-term, ii) generalization to predict the quality of an unused link based on its features only.

The DT must provide forecasting: it predicts the link quality in the future so that it can compute the expected KPIs of the network infrastructure. While exploiting synthetic radio propagation models in a network simulator may provide preliminary results, it cannot create a DT. Indeed, the digital twin is not synchronized with the physical devices.

We propose to design data-oriented models for each radio link, fed by measurements. Since PHY layer indicators are complex to obtain, we rely on a time series of packet success/loss (*i.e.*, a binary sequence). The model is calibrated for each radio link to predict the number of successes/failures in the next time window. We extend this approach to predict the PDR for many time windows. The accuracy decreases with distance from the prediction. Ideally, the network simulator will exploit the corresponding PDR and predict the probability that the Service Level Agreement (SLA) will not be respected.

In addition, a wireless network infrastructure needs to be reconfigured to operate close to optimum. However, a priori estimation of the quality of unused links is very challenging [13]. Broadcast packets can be exploited to infer the reliability of unicast transmissions, but ithey cannot consider multiple factors (*e.g.*, multi-rate, collisions in scheduled access).

The DT should be the cornerstone enabling generalization. It maintains a collection of models for all used radio links.

Parameter	Value
Number of nodes	11
Traffic data rate	1 packet per second
Traffic direction	Broadcast
Packet size	50 bytes
Operating system	Contiki-NG
Communication protocol	802.15.4
MAC Protocol	CSMA/CA

TABLE I: Considered parameters for our experimental setup.

We can therefore reuse these existing models, calibrated to the physical environment, to make inferences. The aim is to classify links according to their properties, and to make predictions about the expected characteristics of links for which we have not yet collected measurements.

III. Radio Link Characterization

The objective of this section is to present a concrete example of a wireless network and to thoroughly characterize the various wireless links within it. In particular, we highlight a very large heterogeneity of behaviors: simple simulations models will provide a low accuracy in these conditions.

A. Experimental Setup

We deploy a network using the FIT IoT-Lab platform [14]. Our experiment comprises 11 nodes (from node m3-2 to m3-12) located at the Grenoble site. Each node is equipped with an M3-board micro-controller, running firmware based on the Contiki Operating System [15]. The IoT devices transmit via an IEEE 802.15.4 network using the 6LoWPAN protocol. In this setup, every node broadcasts a 50-byte packet to all other nodes every second. The IEEE 802.15.4 protocol, widely used in IoT applications, employs the CSMA/CA method for its MAC layer. Table I lists all our parameters. We would like to emphasize that our approach is designed to be generic, making no specific assumptions about the MAC layer, communication protocol, or deployment characteristics.

The network was operational for 5 hours. We collect time series and statistics on the experiment through a serial output connected to the FIT IoT-Lab monitoring infrastructure, focusing on transmission and reception times. Our primary metric of interest is the PDR, which is the ratio of the total number of successfully delivered packets to the destination node divided by the total number of original sent packets. Each link between nodes is represented by a sequence of 0s and 1s, indicating whether a packet was received (1) or not (0). Since considering binary sequences for every packet is not representative of the network behaviour, we can use these binary elements to compute the number of correctly received packets over specific periods, which we set to 50 seconds for this study. Consequently, for each link, we generate a time series showing the number of correctly received packets for each 50-second interval. It is worth noting that the links are asymmetrical in our analysis (*i.e.,* the link from node A to Be is different than the link from node B to A).

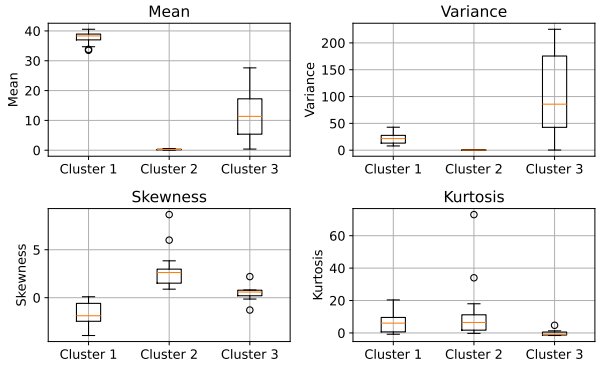

Fig. 2: Clusters statistics: Mean, variance, skewness and kurtosis.

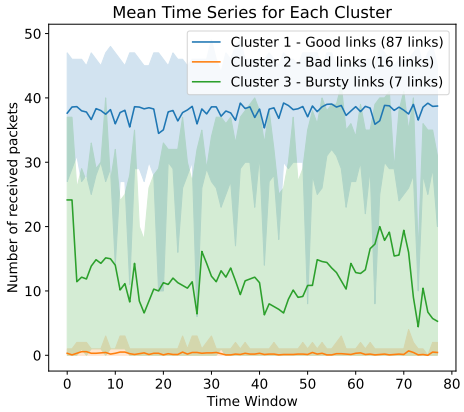

Fig. 3: Cluster time series: Each cluster is characterized by the mean of its constituent time series, with the range from the minimum to the maximum values providing a visual boundary around the mean.

B. Heterogeneity and Stationarity

To illustrate the diversity of the links and provide insight into their distribution within the network, we calculated various statistical metrics for each time series, including: tThe mean, the standard deviation, the variance, the skewness, which is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean, the kurtosis, which measures the "tailedness" of the probability distribution and helps in identifying periods of high volatility or extreme values in the time series [16], and the minimal and the maximal value. These metrics allow us to quantify the characteristics of the links effectively. Subsequently, we applied a clustering algorithm (namely KMeans [17]) to group these statistical metrics, to reveal potential similarities between different links.

Figure 2 illustrates the distributions of each parameter—mean, variance, skewness, and kurtosis—for all time series, categorized by different clusters. This highlights the heterogeneity and diversity within the data, and emphasizes the distinct characteristics of each cluster. Figure 3 displays the mean time series for each of the three clusters, along with the minimum and maximum values within each cluster. The first cluster consists of high-quality wireless links, characterized by good performance, although occasional drops in transmission may occur. This cluster encompasses 80% of all links. The second cluster comprises poorly performing links, where the number of correctly transmitted packets rarely exceeds five per window, representing 15% of the links. Finally, the third cluster captures links exhibiting highly dynamic and bursty behavior, with the number of correctly received packets fluctuating between 0 and 40, reflecting significant variability. This cluster accounts for the remaining 5% of links.

C. Are Simulations Accurate?

To illustrate the limitations of using simulators to replicate the behavior and performance of a real network, we implemented the exact same network on the Cooja simulator [18]. Cooja is a discrete-event simulator known for its flexibility, as it can run real firmware to mimic the behavior of actual wireless devices. Thus, it is a natural candidate for a DT. We simulated a network with the same number of devices, arranged identically to the real testbed, running the same firmware and traffic application. Cooja models wireless transmissions by using two user-defined variables (P_{Tx} and P_{Rx}) to determine the probability of packet reception, provided the distance between the sender and receiver does not exceed a certain threshold. If a node A transmits a packet, the model uses P_{Tx} to determine if the transmission attempt is successful. If the transmission is successful, the model then uses P_{Rx} to determine if the packet is successfully received by node B.

Given that we have 11 nodes and consider asymmetric links, there are approximately 110 different links for which we measure the evolution of the number of correctly received packets. To compare the measurements gathered from the real experimental platform with those from the simulator, and in order to model each link separately, we compare experiments with the two following approaches:

1) Global calibration with the average PDR: We calibrate the probability of packet transmissions of all the links in the simulator with the real average PDR percentage measured across the network in the real platform (*i.e.,* for each link $(a-b)$: $P_{Tx}^{(a-b)}$ = Average-PDR). It mimicks a simulation calibrated with an *average scenario*.
2) Personalized calibration with the average PDR: For greater precision, we calibrate each link separately with the average PDR value measured in the platform for that specific link (*i.e.,* for each link $(a-b)$: $P_{Tx}^{(a-b)}$ = Average-PDR$^{(a-b)}$). It mimicks a simulation where each link can be calibrated with its own measurements.

Figure 4 illustrates the differences between the actual PDR and those computed in the simulator with both types of calibration. We report the results for a few representative wireless links, each from a different cluster. Several observations can be made from the figure:

- The real measurements exhibit varied behaviors, includ-

ing sudden drops in PDR and consistently low values of PDR throughout the deployment,
- Using a single PDR for all links cannot capture the unique behaviors of each link, which display diverse characteristics,
- Calibrating each link with its own average PDR is insufficient, as PDR evolves over time, potentially due to temporal interferences and other dynamic factors in the environment.

These observations highlight the limitations of simulators like Cooja in accurately replicating real-world wireless network behaviors, emphasizing the need for adaptive and more nuanced modeling approaches.

IV. Can we provide link quality predictions?

As previously observed, the wireless links within a network can exhibit diverse behaviors. While there are very good links, with very small variations, we also identified bursty links. Thus, applying a single model to replicate all wireless links may lead to poor results. Moreover, executing complex models for links easily predictable represents a waste of computational resources. We propose to maintain a DT represented with a collection of DTs. Each radio link is connected to *its* digital twin (see Figure 1). The model is self-calibrated with the measurements received from this link. This decomposition offers several advantages:

- The ability to change the model for each link on-the-fly when deemed appropriate, based on the gathered measurements,
- improved troubleshooting of the network by precisely identifying which wireless link is underperforming,
- enhanced scalability, as adding a new link and its corresponding model does not impact the entire network.

A. Prediction model

The objective of the model is to predict (or forecast) the future values of a given performance metric for each link, specifically the PDR in our case. This problem can be approached as a time series forecasting task, which can be addressed using various methods, including specifically tailored models like Auto Regressive Integrated Moving Average (ARIMA) and classical regression models.

1) Auto Regressive Integrated Moving Average (ARIMA): this model is widely used for time series forecasting, particularly suitable for data that shows some degree of stationarity. They are effective in capturing the linear relationships in time series data. It combines three components: (i) **AutoRegressive (AR)** part, which uses the relationship between an observation and a number of lagged observations, (ii) **Integrated (I)** part which involves differencing the observations to make the time series stationary and (iii) **Moving Average (MA)** part which models the relationship between an observation and a residual error from a moving average model applied to lagged observations.

The ARIMA model is denoted as ARIMA(p, d, q), where:

- p is the number of lag observations included in the model (AR order),
- d is the number of times that the raw observations are differenced (I order),
- q is the size of the moving average window (MA order).

The general form of the ARIMA model is given by:

$$x_t = c + \sum_{i=1}^{p} \phi_i x_{t-i} + \epsilon_t + \sum_{j=1}^{q} \theta_j \epsilon_{t-j}$$

where x_t is the differenced series (if $d > 0$), ϵ_t is the error term, ϕ_i are the coefficients for the AR terms, and θ_j are the coefficients for the MA terms.

2) Regression Models: Regression models are useful for their simplicity and efficiency in predicting outcomes based on input features. In the context of time series forecasting, these models aim to predict a variable x_t using one or more independent variables from prior time points, $(x_1, x_2, \ldots, x_{t-1})$. In our study, we consider the following models:

- **Support Vector Regression**: SVR aims to find a hyperplane in a high-dimensional feature space that has the maximum margin from the training data points [19].
- **Gradient Boosting**: It builds a set of prediction models in a sequential manner. It combines these models to create a more through the reduction from each model of the errors made by the previous models [20].
- **Decision Trees**: A Decision Tree model splits the data into subsets based on the value of input features, creating a tree-like structure where each node represents a feature, each branch represents a decision rule, and each leaf node represents an outcome [21].
- **AdaBoost**: AdaBoost (Adaptive Boosting) works by combining multiple weak learners, typically decision trees to form a strong classifier [22].
- **Extra Trees**: It builds multiple decision trees using random subsets of the training data and features. It further randomizes the splitting process by considering random thresholds for each feature instead of searching for the best split point [23].

B. Training and Prediction setup

We consider time series of Packet Delivery Ratio measurements denoted as $[x_1, x_2, \ldots, x_n]$.. During a period T, we measure the ratio of packets correctly received by the receiver for each link (*i.e.*, x_i is the number of correctly received packets during the i^{th} time interval of $T = 50$ seconds in this case). While ARIMA models can be directly applied to this series, regression models require data preprocessing using a sliding window technique before prediction. This method works as follows:

1) **Window Selection**: Choose a window size w that defines the number of consecutive data points to be used as input features for the model.
2) **Sliding Window**: Slide the window across the time series data from the start to the end, creating overlapping sub-sequences. For example, the first window

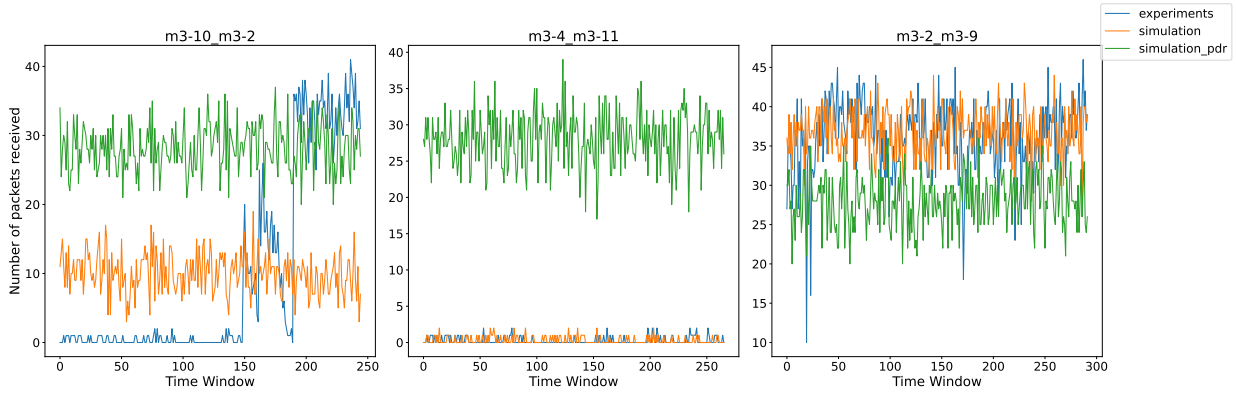

Fig. 4: Wireless links time series for three links from different clusters, where i_j represents the link from node i to node j.

would contain $[x_1, x_2, \ldots, x_w]$, the second window $[x_2, x_3, \ldots, x_{w+1}]$, and so on. This sliding window approach enables the models to learn patterns and dependencies from the historical data, such as trends, seasonality, and other temporal features, facilitating accurate forecasting of future PDR values.

3) **Feature and Target Extraction**: For each window, use the first w values as input features and the next value x_{w+1} as the target for prediction. This process transforms the time series into a supervised learning dataset, suitable for regression models.

In order to select the most suitable model for each link, we train various models using historical data. The model with the lowest prediction error, as measured by the Mean Square Error (MSE) on the training data, is chosen for the rest of the network deployment. This metric is also employed to determine the best window size w for each model. This process is formalized in Algorithm 1. For simplicity purposes, we assume in the algorithm that the window size w is already set. Note that for the ARIMA model, the training data is used to find the best combination of the parameters p, d, and q for accurate predictions.

Once the model and its parameters are chosen, we can use to make predictions for the next time interval: The model is continuously updated with new data from the network, and is used solely to forecast the next value in the time series. After each prediction, the model waits for the actual measured value from the network, which is then used to update the model.

Algorithm 1: Model Selection

Input: $X_{\text{train}} = [x_1, \ldots, x_n]$: Training dataset
Input: $X_{\text{test}} = [x_{n+1}, \ldots, x_m]$: Test dataset
Input: w: Sliding window size
Input: $Models = [Model_1, \ldots, Model_N]$: Forecasting models

```
/* Initialization                    */
D ← ∅
best_model ← None
best_error ← +∞
/* Creating Sliding Windows          */
index ← 0
while index + w < n do
    input ← ∅
    for i = 0 to w − 1 do
        input ← input + x_{index+i}
    D ← D + (input, x[index + w])
    index ← index + 1
/* Selecting the best model          */
D_train ← D[: 0.8] // First 80%
D_valid ← D[0.8 : 1] // Remaining 20%
foreach model ∈ Models do
    model.train(D_train)
    error ← MSE(model, D_valid)
    if error < best_error then
        best_model ← model
        best_error ← error
```

Output: $best\_model$

C. Adaptive Model

Finally, we propose and advocate for a self-adaptive approach to model selection at each iteration (see Figure 5). For testing, all candidate models are used to forecast the next value. After making predictions, each model is updated with the actual measurements, and the prediction error is computed for each model. The model with the lowest prediction error is selected for the next forecast, and this process continues iteratively. Consequently, predictions throughout the network lifecycle may come from different models, thereby avoiding reliance on a model that may become inefficient over time.

Note that the models can also be used for multi-step forecasting: Initially, the models predicts several future values in the time series. The predicted values are used for subsequent predictions: after predicting y_1, the next prediction is made using the input $(x_{n-w+1}, \ldots, x_n, y_1)$, treating y_1 as if it were the actual observed value. This process is repeated for p predictions. p is defined as the *prediction step*, which is the number of consecutive intervals for which the model makes predictions, before being refit with the ground truth values.

Algorithm 2: Adaptive Method

Input: $X_{\text{train}} = [x_1, \ldots, x_n]$: Training dataset
Input: $X_{\text{test}} = [x_{n+1}, \ldots, x_m]$: Test dataset
Input: w: Sliding window size
Input: p: Prediction step
Input: $best\_model$
/* Initialization */
$final\_predictions \leftarrow \emptyset$
$history \leftarrow [x_{n-w}, \ldots, x_n]$
$index \leftarrow 0$
while *The physical network is deployed* **do**
  /* Prediction with the current best model */
  for $i = 1$ **to** p **do**
    $history \leftarrow history[-w]$
    $prediction \leftarrow best\_model.predict(history)$
    $final\_predictions.append(prediction)$
    $history \leftarrow history.append(prediction)$
  /* Fitting the Model */
  $history \leftarrow [x_{n+index-w}, \ldots, x_{n+index+p}]$
  $D\_fit \leftarrow Sliding\_Window(history, w)$
  $best\_model.fit(D\_fit)$
  $g\_truth \leftarrow (x_{n+index+1}, \ldots, x_{n+index+p})$
  $error \leftarrow MSE(predictions, g\_truth)$
  foreach $model \in Models$ **do**
    /* Best model selection */
    $predictions \leftarrow \emptyset$
    for $i = 0$ **to** p **do**
      $history \leftarrow history[-w]$
      $prediction \leftarrow model.predict(history)$
      $predictions.append(prediction)$
      $history \leftarrow history.append(prediction)$
    /* Fitting the Model */
    $history = [x_{n+index-w}, \ldots, x_{n+index+p}]$
    $D\_fit = Sliding\_Window(history, w)$
    $model.fit(D\_fit)$
    $error = MSE(predictions, g\_truth)$
    if $error < best\_error$ **then**
      $best\_model \leftarrow model$
      $best\_error \leftarrow error$
  $index \leftarrow index + p$
Output: $final\_predicitons$

After every p predictions, the model is refitted with the most recent measurements. Specifically, the models update their parameters using the latest data points ($[x_{n-w}, \ldots, x_n]$) and incorporates the newly measured values up to x_{n+p} as the target. This process ensures that the models remain accurate and relevant by periodically incorporating the most recent data. This cycle of prediction and refitting continues indefinitely. This approach can be very useful for making mid- to long-term predictions.

V. Performance Evaluation & Validation

To validate our approach, we test it on the network described in Subsection III-A. To simulate the behavior of a real digital twin, we split the collected data (PDR measurements) as follows: 67% of the data are used for training the models (which corresponds to approximately to 3 hours of deployment), while the remaining 33% (remaining 2 hours) are reserved for testing and represents the theoretically upcoming data from the real network to the digital twin. For each link, we measured the number of correctly received packets, and accordingly generated the time series, representing the number of correctly received packets in successive 50-secondes intervals. We use the RMSE (Root Mean Square Error) metric to evaluate the performance of regression models and ARIMA. This metric represents the error between the forecasting and the actual measured values for the testing phases, for each 50-second interval. Note that the RMSE for each model is calculated as the average RMSE across all links in the network. We vary the prediction step in our evaluation from 1 to 10, to assess the ability of the models to correctly predict the number of correctly received packets up to 10 intervals (*i.e.,* 500 seconds).

A. Regression Models

We compare various regression models—SVR, gradient boosting, decision trees, AdaBoost, and extra trees— with the ARIMA models, each trained under the same conditions (*i.e.,* with the same data for each link). We evaluate their performance by computing the RMSE for predictions made over different prediction steps. Recall that the prediction step indicates the number of intervals predicted by the models before they are fit using actual measurements.

Figure 6 illustrates the RMSE of predictions for various regression models, alongside the ARIMA model. The regression models exhibit similar performance trends, with prediction errors increasing as the prediction step lengthens, yet even at a prediction step of 10, the errors stay within a range of 1 to 10. This error range, based on the RMSE, reflects the prediction error for correctly transmitted packets in 50-second intervals, ranging from 0 to 50. In contrast, the ARIMA model maintains relatively consistent performance across different prediction steps, but its accuracy is lower compared to the regression models, likely due to ARIMA's assumption of stationarity which may not adequately capture the dynamic nature of network links. The comparable performance of the regression models suggests they are effectively identifying key patterns in the data; however, this also implies potential limitations, as similar performance across diverse models might indicate the models are not fully leveraging unique data features or that the dataset lacks sufficient complexity to differentiate model capabilities. These observations underline the need for an adaptive approach to better address the non-stationary characteristics of the data and enhance predictive accuracy.

Interestingly enough, we also evaluated neural network architectures—specifically LSTM (Long Short-Term Memory), GRU (Gated Recurrent Unit), and RNN (Recurrent Neural

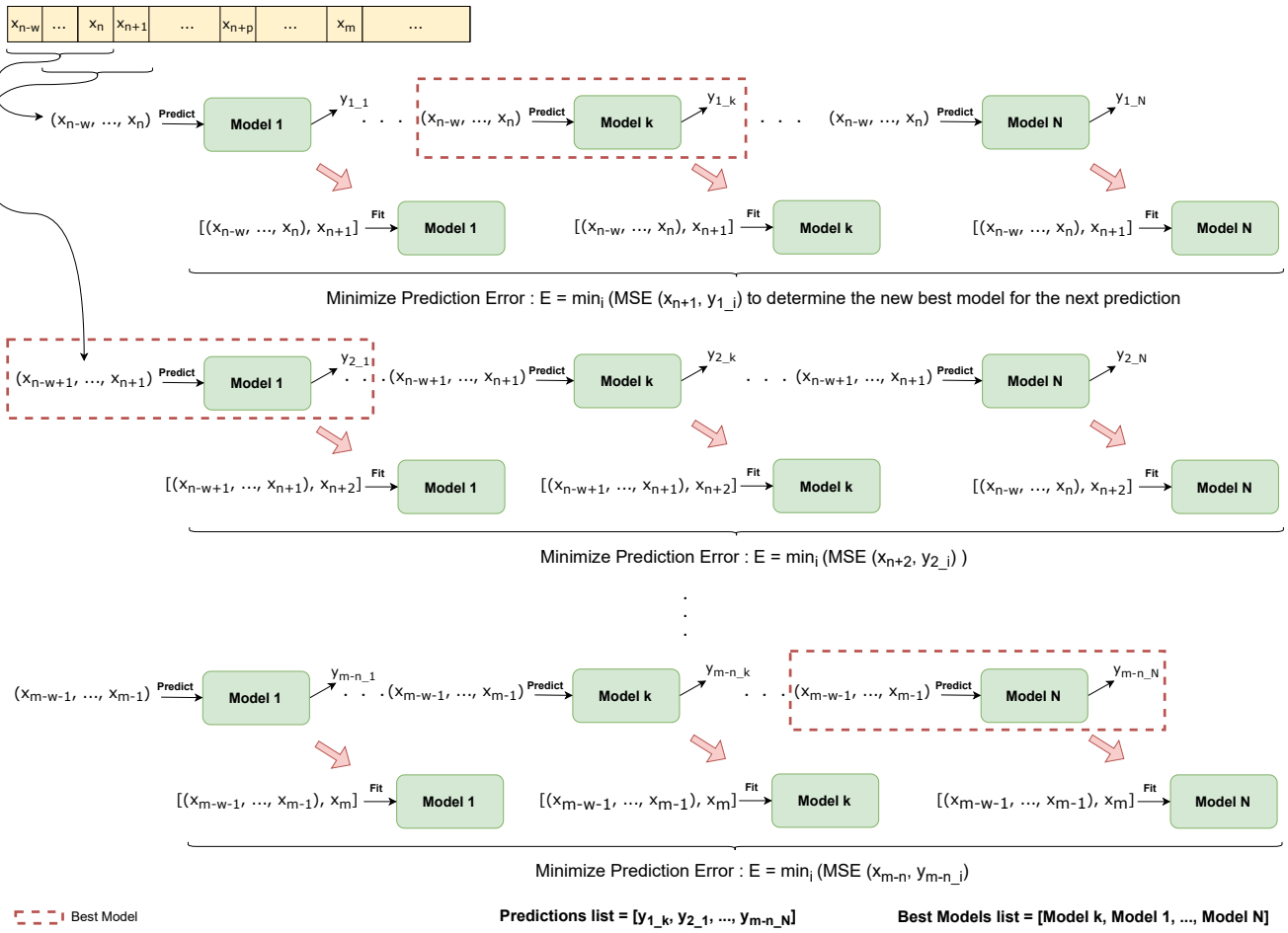

Fig. 5: Adaptive method functioning with a 1-step prediction. The first window of the testing data is used to predict the next interval by all the different models ($Model_1, \ldots, Model_N$). Each prediction y_{1_i} is then compared to the actual measured value x_{n+1}, and the model with the least error is selected for the next prediction. This process is repeated indefinitely along the network deployment, and the predictions retained are only those from the best models for each step ($[y_{1\_k}, y_{2\_1}, \ldots, y_{m-n\_N}]$).

Networks)—for forecasting. However, these models underperformed compared to the regression models, with RNNs showing slightly better efficiency. The neural networks were also more computationally intensive. This unexpected outcome likely stems from the limited training data available, and suggests that the dataset was insufficient to fully utilize the advanced capabilities of LSTMs, GRUs, and RNNs.

B. Adaptive Model vs Best Regression

Figure 7 compares the performance of our adaptive method to the use of a single, best-performing regression model for each link across the entire deployment. The adaptive method consistently outperforms the single-model approach at all prediction steps, demonstrating its strength in handling dynamic changes in link behavior and thereby enhancing prediction accuracy. Notably, the prediction error remains remarkably low at step 1, indicating the adaptive method's effectiveness in immediate, short-term predictions, and only increases slightly with the number of steps. This slight increase in error could

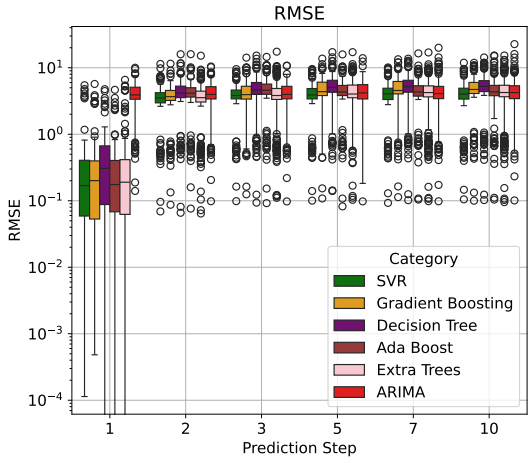

Fig. 6: Evolution of the RMSE for the Regression Models and ARIMA for prediction step going from 1 to 10 intervals.

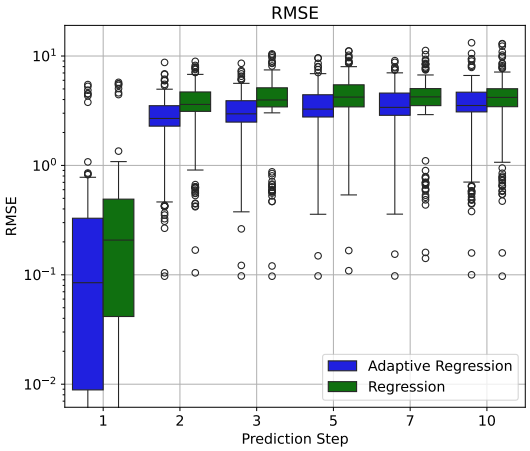

Fig. 7: Performance of the Adaptive Model method compared to the best regression model, according to the prediction step.

be attributed to the adaptive method's capacity to dynamically adjust to variations in link conditions.

To assess the efficiency of our method, we included the model's predictions for the considered links in the previous Figure 4. Thus, Figure 8 compares four sets of measures: (i) real experimental measures, (ii) simulation measures using a simulator calibrated with the average PDR of the network, (iii) simulation measures with each link calibrated to its specific average PDR, and (iv) adaptive method predictions with a prediction step of 1. The results show that adaptive regression captures the intricacies of the experimental data more effectively than both calibration methods of the simulator, even for different types of links, such as the highly dynamic **m3-10_m3-2** link.

***Performance Characterization*:** Recall that we identified three distinct clusters among the links in the networks (see Figure 3): The first cluster consists of high-quality wireless links, the second cluster comprises poorly performing links, and the third cluster captures links exhibiting highly dynamic and bursty behavior with significant variability.

To further evaluate the effectiveness of our adaptive regression method, we assessed its performance on the previously identified clusters. We displayed the distributions for each cluster in Figure 9. The results show that for the Cluster 0, the low dispersion in the data, which reflects stable performance with minimal packet loss, allows the model to predict outcomes with high accuracy. This stability and predictability in the data lead to lower prediction errors. The method's efficiency in this cluster suggests that it is well-suited to scenarios with reliable and consistent link performance.

For Cluster 1, the method demonstrates very good predictive performance for shorter prediction steps, with an error typically less than 1. This is likely because the limited variation within these short intervals allows the model to maintain a very high degree of accuracy. For higher prediction steps, while the performance does decline, the prediction errors still do not exceed 10. This resilience indicates that the method can maintain an acceptable level of accuracy over longer prediction steps.

Finally, for Cluster 2 where links are bursty, the method performs robustly, particularly for shorter prediction steps. As the prediction step increases, the model still manages to keep prediction errors within a manageable range, not exceeding 10 neither. This suggests that the method is adaptable and capable of providing reasonable predictions even in less stable environments.

The overall robustness of the method across all clusters highlights its versatility and potential for broader application in diverse network conditions. Finally, it is worth noting that although our adaptive method incurs a higher computational cost compared to using a single regression model, the simplicity and relative lack of complexity in the models ensure that this continual learning process does not consume excessive time or energy. This level of resource usage is acceptable within the context of network management.

VI. Related Work

Network Simulation: Performance evaluation of wireless networks has extensively relied on network simulation in the past [5]. We can also combine simulation and emulation [24] to execute the same code in simulators and real-life situations and to accelerate prototyping. Unfortunately, simulations present poor accuracy and strongly depend on the PHY layer model [7]. Still, other works such as [25] have used this synergy to calibrate the energy consumption in simulators using real measurements.

Radio models: Numerous radio propagation models have been proposed to cope with urban, vehicular [26], or tunnel [27] environments. Similarly, specific models exist for different applications such as *e.g.*, UAVs [28] and 6G [29]. However, these models need to be tuned with the in-situ deployments. The DT typically targets this synchronization between the physical world and the models of the DT.

More recently, Machine Learning based models have been proposed [30]. Measurements are used for the learning phase to predict the expected signal strength. Instead of relying on ray-tracing simulation, we could extend this approach by exploiting datasets with dual Wi-Fi/cellular nodes [31], LoRaWAN [32]. Almeida et al. [33] replicate a network testbed using network traces within a simulator and regression models, to effectively capture the dynamic conditions of the environment.

Closed-form expressions: Because network simulations were costly, synthetic models have been derived to compute performance metrics. Several models [34], [35] capture the impact of per-SF channel quality, multiple gateways, collision probability, and channel variation on the performance metrics. The accuracy is acceptable even for large-scale topologies, but they can only be used as a black box, for the whole cell. We cannot predict the impact of *e.g.*, a scheduling strategy or a station re-association.

Radio channel emulation: Colosseum proposes a DT relying on a large-scale emulation platform [9]. Software radio

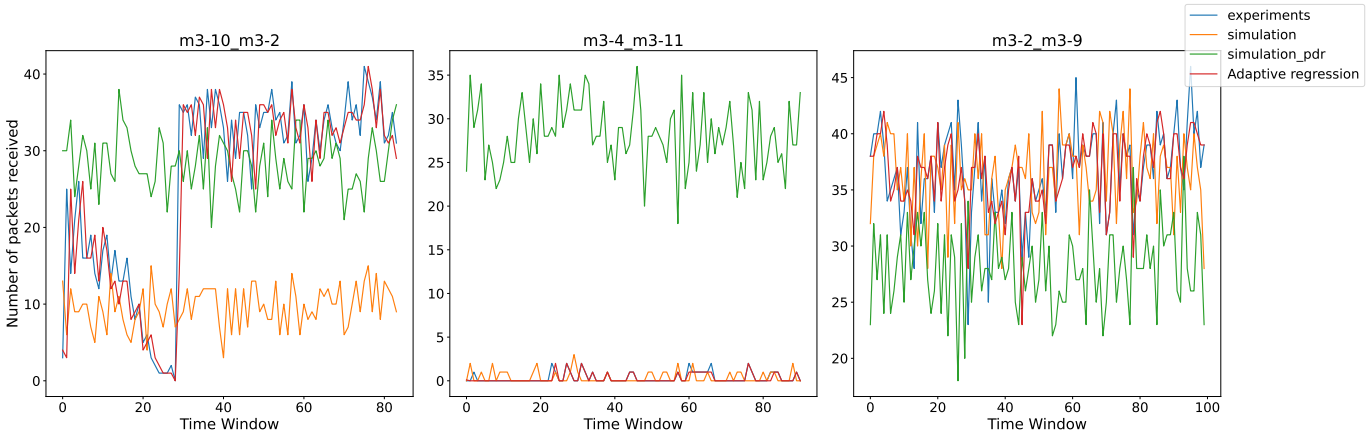

Fig. 8: Wireless links time series for three links from different clusters: Real Measurements vs. Adaptive regression.

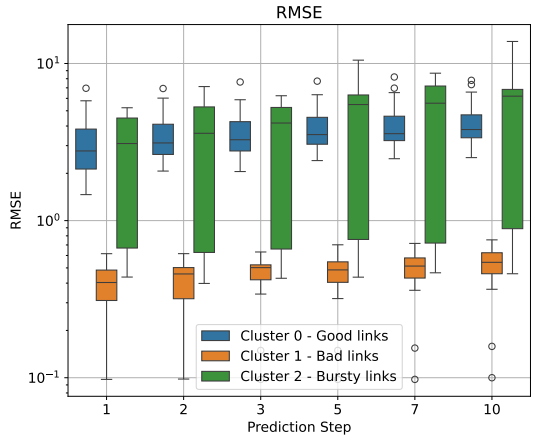

Fig. 9: Prediction error of the adaptive method evolution on the different clusters.

nodes represent the User Equipments and the Base Station, interconnected by a complex FPGA fabric. In particular, the FPGA emulates radio propagation, based on radio models tuned according to the real world. Thus, we need fine-grained measurements to mimic the channel transfer function, which is hard to obtain in real-life deployments.

Generalization and what-if scenarios: DTs are extensively exploited for their generalization property [36]. In particular, the so-called *what-if scenarios* can explore alternative configurations. Unfortunately, the existing what-if scenarios are particularly inaccurate in networking, even for non-wireless situations [37]. Almasan et al. [3] exploit the well-known routeNet-Erlang [4] GNN model to estimate per-flow performance metrics. They compute per-flow end-to-end performance metrics with a small set of configurations. Then, the GNN is used to infer the expected performance metrics in other configurations, exploring the most promising ones. Wang et al. [38] adopt the same GNN technique for network slicing management. Ak et al. [39] proposed a framework to generate synthetic data from a simulator for what-if analysis.

Realistic link quality: Numerous link quality metrics have been proposed in the literature [40]. Radio links are known to be lossy [10], asymmetrical [41], and bursty [2]. Cerar et al. [42] classify links according to their RSSI value and quality (low/medium/high). However, DT goes one step further since it needs to exploit a realistic estimation of the packet losses distribution.

Link quality prediction: Link quality prediction helps the DT to anticipate future situations, but is challenging because of the possible non-stationarity. Cerpa et al. [43] compare a Naive Bayes classifier, Logistic Regression classifier, and Artificial Neural Networks to predict the packet losses, based on the PHY indicators of the last W packets. However, this technique leads to very short-term predictions, *i.e.*, the next packet reception. In [44], the authors employ neural networks, including LSTM, GRU, and CNN, to forecast the evolution of the global PDR in the network. However, their approach is evaluated using data generated by the Cooja simulator, which may not fully capture the complexity and nuances of real-world data.

VII. Conclusion & Perspectives

In this paper, we have proposed a self-adaptive approach to be the core of the digital twin of a wireless network. This method involves creating a collection of digital twins, with each twin corresponding to a specific radio link. In this way, we can dynamically select the best model for each link. We tested our approach on a real-world wireless network characterized by high heterogeneity, demonstrating its high efficiency in replicating the network's PDR performance. Additionally, our method has shown the ability to forecast future events for long prediction intervals. Thus, the DT could be exploited later for network reconfiguration. Our dataset and implementation are freely available for reproducibility and reinterpretability.

As future work, we plan to evaluate our approach using other metrics such as latency and the number of consecutively lost packets, which are important indicators of network quality. Furthermore, we will focus on enhancing the generalization capabilities of our DT, aiming to predict the performance of

new links based on environmental information, such as the distance between the sender and receiver, or the specific area involved. All these features will then be integrated in a DT to both replicate the wireless network and provide accurate *what-if* scenarios.

REFERENCES

[1] M. Vaezi, K. Noroozi, T. D. Todd, D. Zhao, G. Karakostas, H. Wu, and X. Shen, "Digital Twins From a Networking Perspective," *IEEE Internet of Things Journal*, vol. 9, no. 23, pp. 23 525–23 544, Dec. 2022.

[2] K. Srinivasan, M. A. Kazandjieva, S. Agarwal, and P. Levis, "The β-factor: measuring wireless link burstiness," in *Conference on Embedded Network Sensor Systems (Sensys)*. ACM, 2008, pp. 29–42.

[3] P. Almasan, M. Ferriol-Galmes, J. Paillisse, J. Suarez-Varela, D. Perino, D. Lopez, A. A. P. Perales, P. Harvey, L. Ciavaglia, L. Wong, V. Ram, S. Xiao, X. Shi, X. Cheng, A. Cabellos-Aparicio, and P. Barlet-Ros, "Network Digital Twin: Context, Enabling Technologies, and Opportunities," *IEEE Communications Magazine*, vol. 60, no. 11, pp. 22–27, Nov. 2022.

[4] M. Ferriol-Galmés, K. Rusek, J. Suárez-Varela, S. Xiao, X. Shi, X. Cheng, B. Wu, P. Barlet-Ros, and A. Cabellos-Aparicio, "Routenet-erlang: A graph neural network for network performance evaluation," in *INFOCOM*. IEEE, 2022, pp. 2018–2027.

[5] K. Kritsis, G. Z. Papadopoulos, A. Gallais, P. Chatzimisios, and F. Theoleyre, "A tutorial on performance evaluation and validation methodology for low-power and lossy networks," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 1799–1825, 2018.

[6] J. Medbo, P. Kyosti, K. Kusume, L. Raschkowski, K. Haneda, T. Jamsa, V. Nurmela, A. Roivainen, and J. Meinila, "Radio propagation modeling for 5g mobile and wireless communications," *IEEE Communications Magazine*, vol. 54, no. 6, pp. 144–151, 2016.

[7] E. B. Hamida, G. Chelius, and J. M. Gorce, "Impact of the physical layer modeling on the accuracy and scalability of wireless network simulation," *SIMULATION*, vol. 85, no. 9, pp. 574–588, 2009.

[8] R. Adeogun, "Calibration of stochastic radio propagation models using machine learning," *IEEE Antennas and Wireless Propagation Letters*, 2019.

[9] D. Villa, M. Tehrani-Moayyed, C. P. Robinson, L. Bonati, P. Johari, M. Polese, and T. Melodia, "Colosseum as a Digital Twin: Bridging Real-World Experimentation and Wireless Network Emulation," *IEEE Transactions on Mobile Computing*, pp. 1–17, 2024.

[10] C. Bernardos *et al.*, "Reliable and available wireless (raw) use cases," IETF RFC 9450, https://datatracker.ietf.org/doc/rfc9450/, 2024.

[11] E. Ak and B. Canberk, "Forecasting quality of service for next-generation data-driven wifi6 campus networks," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 4744–4755, 2021.

[12] A. S. Colletto, S. Scanzio, G. Formis, and G. Cena, "On the use of artificial neural networks to predict the quality of wi-fi links," *IEEE Access*, vol. 11, pp. 120 082–120 094, 2023.

[13] R. T. Hermeto, A. Gallais, K. V. Laerhoven, and F. Tholeyre, "Passive link quality estimation for accurate and stable parent selection in dense 6tisch networks," in *International Conference on Embedded Wireless Systems and Networks (EWSN)*, 2018.

[14] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele *et al.*, "Fit iot-lab: A large scale open experimental iot testbed," in *IEEE WF-IoT*. IEEE, 2015, pp. 459–464.

[15] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki-a lightweight and flexible operating system for tiny networked sensors," in *29th annual IEEE international conference on local computer networks*. IEEE, 2004, pp. 455–462.

[16] L. T. DeCarlo, "On the meaning and use of kurtosis." *Psychological methods*, 1997.

[17] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

[18] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with cooja," in *conference on local computer networks (LCN)*. IEEE, 2006, pp. 641–648.

[19] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, 2004.

[20] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of statistics*, 2001.

[21] J. R. Quinlan, "Induction of decision trees," *Machine learning*, 1986.

[22] Y. Freund, R. E. Schapire *et al.*, "Experiments with a new boosting algorithm," in *ICML*. Citeseer, 1996.

[23] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine learning*, vol. 63, pp. 3–42, 2006.

[24] X. Wang, T. Shen, Y. Zhang, and X. Chen, "An efficient topology emulation technology for the space-air-ground integrated network," in *INFOCOM Workshops*. IEEE, 2023.

[25] S. Si-Mohammed, A. Bardou, T. Begin, I. G. Lassous, and P. Vicat-Blanc, "Ns+ ndt: Smart integration of network simulation in network digital twin, application to iot networks," *Future Generation Computer Systems*, vol. 157, 2024.

[26] F. A. Rodríguez-Corbo, L. Azpilicueta, M. Celaya-Echarri, A. V. Alejos, and F. Falcone, "Propagation models in vehicular communications," *IEEE Access*, vol. 9, pp. 15 902–15 913, 2021.

[27] A. Hrovat, G. Kandus, and T. Javornik, "A survey of radio propagation modeling for tunnels," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 658–669, 2014.

[28] A. Ahmad, A. A. Cheema, and D. Finlay, "A survey of radio propagation channel modelling for low altitude flying base stations," *Computer Networks*, vol. 171, p. 107122, 2020.

[29] D. Serghiou, M. Khalily, T. W. C. Brown, and R. Tafazolli, "Terahertz channel propagation phenomena, measurement techniques and modeling for 6g wireless communication applications: A survey, open challenges and future research directions," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 4, pp. 1957–1996, 2022.

[30] K. Ahmad and S. Hussain, "Machine learning approaches for radio propagation modeling in urban vehicular channels," *IEEE Access*, vol. 10, pp. 113 690–113 698, 2022.

[31] I. Aydin, F. Fund, and S. S. Panwar, "A data set and reference experiments for multipath wireless emulation on public testbeds," in *INFOCOM Workshops*. IEEE, 2023.

[32] E. Goldoni, P. Savazzi, L. Favalli, and A. Vizziello, "Correlation between weather and signal strength in lorawan networks: An extensive dataset," *Computer Networks*, vol. 202, p. 108627, 2022.

[33] E. N. Almeida, M. Rushad, S. R. Kota, A. Nambiar, H. L. Harti, C. Gupta, D. Waseem, G. Santos, H. Fontes, R. Campos *et al.*, "Machine learning based propagation loss module for enabling digital twins of wireless networks in ns-3," in *Workshop on ns-3*, 2022.

[34] V. Toro-Betancur, G. Premsankar, M. Slabicki, and M. Di Francesco, "Modeling communication reliability in lora networks with device-level accuracy," in *INFOCOM*. IEEE, 2021, pp. 1–10.

[35] D. Croce, M. Gucciardo, S. Mangione, G. Santaromita, and I. Tinnirello, "Lora technology demystified: From link behavior to cell-level performance," *IEEE Transactions on Wireless Communications*, vol. 19, no. 2, pp. 822–834, 2020.

[36] Y. Wu, K. Zhang, and Y. Zhang, "Digital twin networks: A survey," *IEEE Internet of Things Journal*, vol. 8, no. 18, pp. 13 789–13 804, 2021.

[37] L. Hui, M. Wang, L. Zhang, L. Lu, and Y. Cui, "Digital twin for networking: A data-driven performance modeling perspective," *IEEE Network*, vol. 37, no. 3, pp. 202–209, 2023.

[38] H. Wang, Y. Wu, G. Min, and W. Miao, "A graph neural network-based digital twin for network slicing management," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 2, pp. 1367–1376, 2022.

[39] E. Ak, B. Canberk, V. Sharma, O. A. Dobre, and T. Q. Duong, "What-if analysis framework for digital twins in 6g wireless network management," *arXiv preprint arXiv:2404.11394*, 2024.

[40] N. Baccour, A. Koubâa, L. Mottola, M. A. Zúñiga, H. Youssef, C. A. Boano, and M. Alves, "Radio link quality estimation in wireless sensor networks: A survey," *ACM Trans. Sen. Netw.*, vol. 8, no. 4, sep 2012.

[41] M. Z. n. Zamalloa and B. Krishnamachari, "An analysis of unreliability and asymmetry in low-power wireless links," *ACM Trans. Sen. Netw.*, vol. 3, no. 2, pp. 1–34, jun 2007.

[42] G. Cerar, H. Yetgin, M. Mohorčič, and C. Fortuna, "On designing a machine learning based wireless link quality classifier," in *International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2020, pp. 1–7.

[43] T. Liu and A. E. Cerpa, "Data-driven link quality prediction using link features," *ACM Trans. Sen. Netw.*, vol. 10, no. 2, jan 2014.

[44] H. Benadji, L. Zitoune, and V. Vèque, "Predictive modeling of loss ratio for congestion control in iot networks using deep learning," in *GLOBECOM*. IEEE, 2023.