Per Link Data-driven Network Replication Towards Self-Adaptive Digital Twins

Samir Si-Mohammed
ICube lab
CNRS / University of Strasbourg, France
Email: simohammed@unistra.fr

Fabrice Theoleyre
ICube lab
CNRS / University of Strasbourg, France
Email: fabrice.theoleyre@cnrs.fr

***Abstract*—Digital twins have recently emerged as a transformative paradigm in wireless networking, offering promising avenues to enhance network efficiency and adaptability. By digitally replicating the physical network, they enable advanced monitoring, analysis, and optimization—key enablers for next-generation wireless systems. Although extensive efforts have been made in modeling and simulating wireless networks, their accuracy often falls short in practical environments. Real-world conditions, such as physical obstructions and complex radio propagation phenomena, are particularly difficult to replicate without resource-intensive local measurement campaigns. To address this challenge, we propose a lightweight data-driven approach to modeling wireless networks. Specifically, we utilize a simple yet informative Key Performance Indicator—the Packet Delivery Ratio (PDR)—to train link-specific quality prediction models. By training the models individually for each link, we effectively capture network heterogeneity and achieve high prediction accuracy. Moreover, we introduce a dynamic and adaptive approach for continuously selecting the most relevant model for the prediction. Experimental results from a real-world deployment show that this approach achieves a high prediction accuracy over both short and mid-term horizons, highlighting the effectiveness of individualized, adaptive modeling in dynamic wireless environments.**

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

I. Introduction

Wireless networks have become an essential part of modern life, enabling omnipresent connectivity across personal, industrial, and critical infrastructure systems. From Wi-Fi and cellular networks to sensor deployments in smart cities, wireless technologies underpin a wide range of applications with different requirements. However, the performance of wireless networks is inherently dynamic. Radio links are subject to variable interference, physical obstructions, and mobility, leading to asymmetric, bursty, and lossy behaviors [1]. As a result, traditional static or rule-based management approaches often fall short in adapting to the fast-changing network conditions. Modern wireless systems require intelligent, data-driven solutions capable of reacting to real-time dynamics.

To this end, Digital Twins (DTs) are increasingly being adopted in wireless networking. A DT is a real-time digital replica of a physical system, continuously synchronized with its real-world counterpart. DTs offer powerful capabilities for system monitoring, configuration, and optimization by enabling the execution of *what-if* scenarios and forecasting future Key Performance Indicators (KPI) under different configurations. They have proven particularly effective in other domains such as Industry 4.0 and are now finding traction in networking contexts [2].

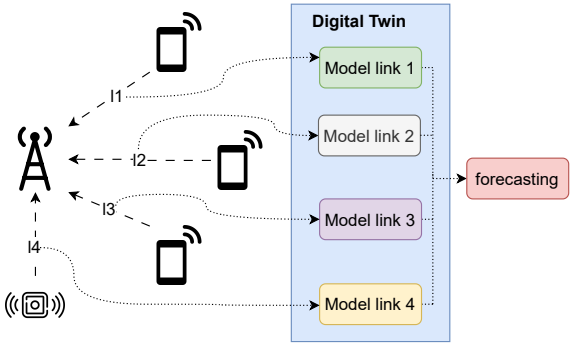


Fig. 1: Architecture of a DT for wireless networking

A major challenge in building effective DTs for wireless networks lies in the fidelity of the digital replica. Accurate modeling of the radio environment, especially radio propagation and link quality, is difficult due to the complexity and variability of wireless conditions. Simulation-based DTs often rely on approximate models (e.g., in ns-3 [3]), which may struggle to capture the heterogeneity of real-world wireless environments without significant manual calibration. More accurate alternatives, such as ray-tracing techniques [4], offer improved fidelity, but come at the cost of substantial computational resources. We argue that for DTs to be both accurate and practical, they should be grounded in data from real deployments to reflect the true variability of wireless links, while remaining lightweight enough for scalable use in dynamic scenarios.

Finally, another key challenge is the heterogeneity of wireless networks, where different links can exhibit vastly different characteristics due to hardware, topology, interference, and environmental factors. Yet, many existing approaches rely on holistic or global models that fail to reflect per-link variation [5]. This limits the precision of the DT as individual links often exhibit distinct behaviors that such models fail to capture.

In this work, we focus on the **replication** aspect of DTs for wireless networks, which is the core foundation upon which higher-level functionalities such as optimization, configuration testing, and predictive maintenance can later be built. Specifically, we propose an "agnostic" link model that can be adapted to any wireless network, *i.e.,* regardless of the underlying communication protocols or traffic types. Our method uses observed events (e.g., packet receptions) to train lightweight, individualized models for each link based on the PDR. To ensure robustness in dynamic environments, we introduce a dynamic selection mechanism that continuously identifies the most relevant model for accurate prediction. This link model can typically be part of the future DT, as illustrated in Figure 1.

The key contributions of this work are as follows:

1) We analyze the characteristics of radio links in a real network, highlighting the importance of per-link modeling;
2) we compare various forecasting techniques to evaluate their performance in predicting short- and medium-term behavior (a critical property of the DT);
3) we propose a self-adaptive method that dynamically selects the best model for each specific link.

II. Related Work

A. Ray-tracing for Network Digital Twins

In the context of wireless networking, DTs can be designed to model the radio environment with high precision. Significant recent efforts have been dedicated to ray-tracing-based approaches for Network DTs, which aim to replicate signal propagation in complex environments with high fidelity.

For example, NVIDIA Omniverse [6] offers a large-scale simulation platform for wireless systems, featuring ray-tracing capabilities to emulate radio wave propagation. Accurately synchronizing a DT with real-world conditions is particularly challenging and computationally expensive, as it requires full characterization of the physical (PHY) channel. To achieve realistic replay, detailed knowledge of the channel transfer function is necessary. A notable approach is taken by Colosseum [7], which emulates radio propagation using Software-Defined Radios (SDRs) interconnected via an Field-Programmable Gate Array (FPGA) fabric. This infrastructure relies on propagation models that are fine-tuned to match real-world conditions, necessitating precise and often difficult-to-obtain measurements from the deployment environment.

While these highly accurate approaches may be suitable for certain applications, we believe it is also essential to develop lighter DTs that can operate with less intrusive measurements.

B. Radio Link Quality Modeling

As highlighted in the Introduction, accurately replicating radio link quality is essential for building reliable DTs. To achieve this, link quality models play a key role in enabling predictive capabilities that can support tasks such as resource allocation. Numerous radio propagation models have been developed to address specific scenarios such as urban, vehicular, and tunnel environments [8]. However, these models require calibration with data from in-situ deployments to achieve realistic performance. More synthetic, data-driven models would be highly beneficial.

Also, wireless links are known to exhibit lossy, asymmetric [9], and bursty [1] behavior. Brun et al. [10] analyzed multiple testbeds and real-world deployments, showing that multipath fading is common and leads to time-varying link quality. Several metrics have been proposed to capture this variability [11], including the widely used Link Quality Indicator (LQI) and Received Signal Strength Indicator (RSSI). However, both LQI and RSSI are only loosely correlated with the actual Packet Reception Rate (PRR) [12]. As a result, measuring the average remains the most accurate method to directly estimate link quality.

After initially measuring link quality, a DT should rely on a predictive model. Sindjoung et al. [13] propose a link classification approach for IEEE 802.15.4 TSCH networks using traditional machine learning algorithms that combine RSSI and PDR. Their method is used for rerouting through *better* links.

Benadji et al. [14] adopt neural network architectures—including Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and Convolutional Neural Network (CNN)—to predict the overall evolution of PDR across the network. Their method predicts a global PDR, and is therefore particularly interesting to anticipate SLA violations. Cerpa et al. [15] compare Naive Bayes, Logistic Regression, and Artificial Neural Networks to predict packet losses. They predict the probability that the Packet Delivery Ratio (PDR) exceeds a given threshold value for the next time window, based on physical layer indicators (SNR, LQI, and RSSI) of recent transmissions. However, the SNR value is not reported systematically by commercial radio chips.

In our previous work [5], we addressed link quality prediction under various MAC configurations in IEEE 802.15.4 networks. Our model can predict KPI Indicators associated to a configuration even if it has not been evaluated previously, combining the KPIs for similar configurations. However, a fully functional DT also requires time-aware predictions. Finally, Almeida et al. [16] enhance the ns-3 simulator to construct a DT. The path loss is considered as the sum of a deterministic path loss (obtained with a ML model) and a stochastic fast-fading loss (according to well known Rician or Rayleigh distributions). Unfortunately, the path loss estimation is still imprecise and the study does not isolate the behavior of the different link categories.

Most of these studies assume a single, uniform model is applied across all links. However, given the existence of distinct categories of links with varying characteristics, we advocate for the use of per-link trained models to significantly enhance prediction accuracy. Table I summarizes key characteristics of existing link quality modeling approaches in the literature and contrasts them with our proposed contribution.

Work	Model Type	Granularity	Adaptivity	Prediction Horizon	Data	Limitation vs. our work
Brun et al. [10]	Empirical analysis	Per-link	×	N/A	Exp.	Descriptive only, no prediction
Sindjoung et al. [13]	Traditional ML classification	Per-link	×	/	Exp.	No dynamic/adaptive selection
Benadji et al. [14]	Deep learning (LSTM, GRU, CNN)	Global	×	Mid-term	Sim.	Ignores link variability, no per-link modeling
Cerpa et al. [15]	Traditional ML regression	Per-link	×	Short-term	Exp.	Focus on short-term only
Almeida et al. [16]	SVR and XGBoost	Global	×	Mid-term	Exp.	Single model for all links
Our contribution	Lightweight ML models	Per-link	✓	Mid-term	Both	–

TABLE I: Comparison of related work and our contribution on link quality modeling.

Fig. 2: Network deployment topology.

Parameter	Value
Number of nodes	11
Traffic data rate	1 packet per second
Traffic direction	Broadcast
Packet size	30 bytes
Operating system	Contiki-NG
Communication protocol	802.15.4
MAC Protocol	CSMA/CA

TABLE II: Considered parameters for our experimental setup.

III. Radio Link Characterization and Limits of Simulation Models

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 simulation models will provide a low accuracy in these conditions.

A. Experimental Setup

We conduct our experiments on the large-scale reproducible SLICES-FR testbed[1], where we deploy a wireless network consisting of nine nodes at the Grenoble site (see Figure 2). Each node features an M3-board microcontroller running firmware based on the Contiki Operating System[2]. The IoT devices communicate over an IEEE 802.15.4 network using the 6LoWPAN protocol, with each node broadcasting a 30-byte packet to all other nodes every second. The IEEE 802.15.4 protocol, commonly used in IoT applications, implements the CSMA/CA mechanism at the MAC layer. The parameters of our setup are detailed in Table II. It is important to note that our approach is designed to be generic, without relying on specific assumptions about the MAC layer, communication protocol, or deployment characteristics.

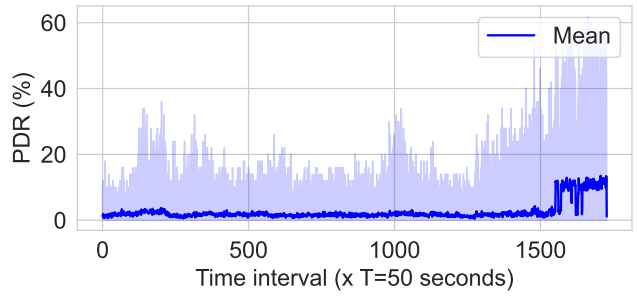
The network operated for 24 hours, during which time we collected time series data and statistics via a serial output connected to the SLICES-FR monitoring infrastructure, focusing on transmission and reception times. Our primary metric of interest is the PDR, defined as the ratio of successfully delivered packets to the total number of packets sent. It is important to note that our analysis treats links as asymmetrical, meaning the link from node A to node B is considered distinct from the link from node B to node A.

Each link between nodes is represented as a binary sequence, where a received packet is indicated by a 1 and a lost packet by a 0. Then, we aggregated these sequences to compute the number of successfully received packets over specific time intervals, set to $T = 50$ seconds for this study. This process generates time series data for each link, showing the count of correctly received packets in successive 50-second intervals.

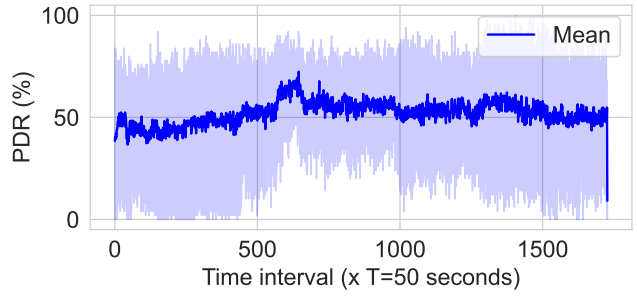
B. Links Heterogeneity

To capture the diversity of links and gain insights into their distribution across the network, we categorized them into four classes according to the mean PDR: (i) bad links, which exhibit a $PDR \leq 20\%$, (ii) average links, with $20\% < PDR \leq 60\%$, (iii) good links with $60\% < PDR \leq 75\%$ and (iv) excellent links ($75\% < PDR$). Figure 3 presents the mean time series for each of the four clusters, along with the minimum and maximum values observed within each cluster. This clearly highlights the heterogeneity within the data and emphasizes the distinct characteristics of each cluster.

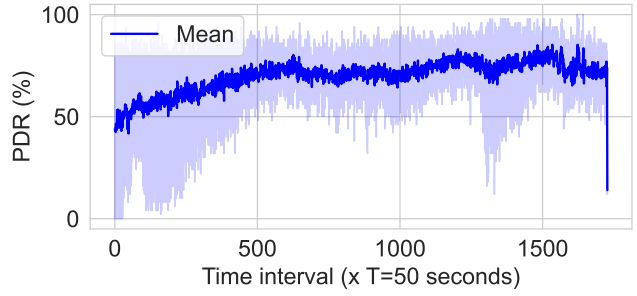
[1]https://slices-fr.eu/
[2]https://www.contiki-ng.org/



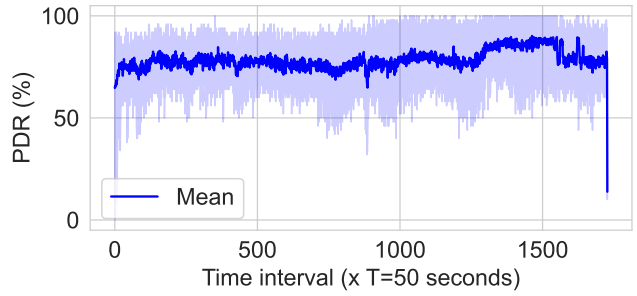
(a) Bad links (n = 27)



(b) Average links (n = 9)



(c) Good links (n = 8)



(d) Excellent links (n = 12)

Fig. 3: Clustered link time series: each subfigure shows the mean time series of a cluster, with a shaded region from minimum to maximum.

C. Are Simulations Accurate?

To highlight the limitations of using simulators to replicate the behavior and performance of real-world networks, we recreated the network setup in simulation. Cooja [17], a discrete-event simulator, is well-regarded for its flexibility as it allows real firmware to be run, effectively mimicking the behavior of actual wireless devices.

The 9 nodes run the same firmware (Contiki) in the testbed and in simulation. Cooja models wireless transmissions based on two user-defined parameters: P_{Tx} and P_{Rx}. These parameters are used to calculate the probability of a packet being successfully transmitted and received, respectively, provided the distance between the sender and receiver does not exceed a predefined threshold. In the testbed, we have 75 non-zero PDR links.

When a node A transmits a packet, P_{Tx} determines whether the transmission attempt succeeds. If successful, P_{Rx} then determines whether the packet is successfully received by node B. While this simplified model captures basic wireless behaviors, it falls short in replicating the nuanced and dynamic characteristics of real-world wireless networks, as explored in this study.

To compare the measurements obtained from the real experimental platform with those generated by the simulator, and to accurately model each link individually, we evaluate the experiments using a personalized calibration using the average PDR: this approach calibrates each link individually using the specific average PDR measured for that link on the real platform. For each link $(a \rightarrow b)$, the transmission probability is set as $P_{Tx}^{(a-b)} = \text{Average-PDR}^{(a-b)}$. This simulates a scenario where each link is calibrated based on its unique real-world performance.

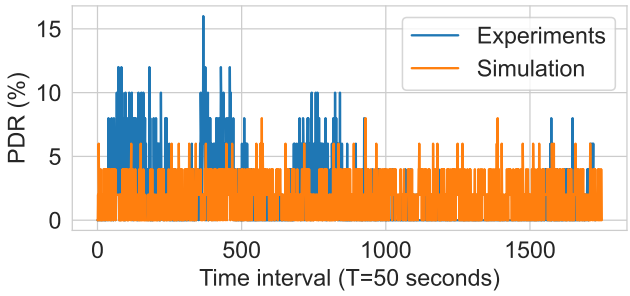
Figure 4 compares the actual PDR measurements from the real platform with those obtained from the simulator under the personalized calibration approach. We showcase a representative collection of wireless links, selected in different clusters. Key observations from the figure include:

- Real-world measurements show highly diverse behaviors, including sudden drops in PDR and consistently low PDR values throughout the deployment.
- Even calibrating each link with its specific average PDR is insufficient, as PDR evolves dynamically over time, influenced by temporal interferences and other environmental factors.

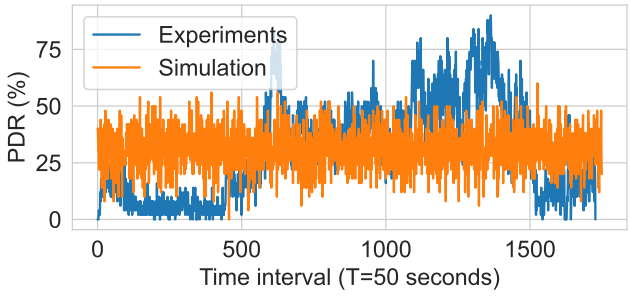
These findings underscore the limitations of simulators like Cooja in accurately replicating the complex and dynamic behaviors of real-world wireless networks. They emphasize the need for adaptive and more sophisticated modeling approaches to better capture the nuances of such environments.

IV. Can we provide link quality predictions?

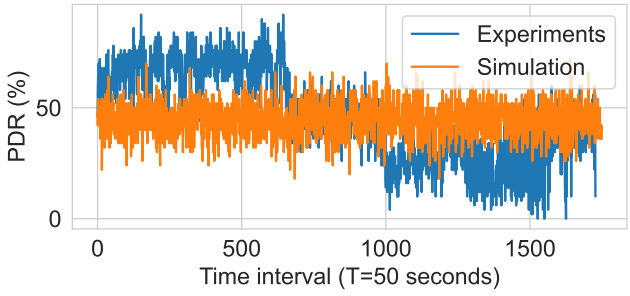
As observed earlier, wireless links within a network exhibit diverse behaviors. While some links maintain consistently high performance with minimal variation, others demonstrate bursty and unpredictable behavior. Consequently, using a single model to replicate all wireless links often yields suboptimal results. Additionally, applying computationally intensive models to links with straightforward, predictable behavior wastes resources.



(a) m3-150_m3-163



(b) m3-133_m3-153



(c) m3-166_m3-163

Fig. 4: Wireless links time series for three links from different clusters, where i_j represents the link from node i to node j, observed in real experiments and in the personalized simulator calibration.

A. Packet Delivery Ratio as Time Series

The objective of the DT is to predict, or forecast, the future values of a given performance metric for each link—specifically the PDR in our case. This task can be framed as a time series forecasting problem, which can be tackled using *e.g.*, regression or ARIMA models.

We analyze time series of PDR measurements, denoted as $[x_1, x_2, \ldots, x_n]$. During each time interval T, we calculate for each link the ratio of packets correctly received by the receiver. In our case, x_i represents the number of correctly received packets during the i^{th} interval ($T = 50s$ by default, as illustrated in Table III). We assume that the characteristics of a link are stationary during a time interval T, and that we collect enough measurements to reflect the actual PDR of the link. A low T value means a higher reactivity, while a high T value reflects a more accurate estimator. It is worth noting that while further optimization of this hyperparameter could potentially improve performance, it falls outside the scope of this paper.

To apply regression models, we preprocess the data using a sliding window technique. The DT mirrors the actual performance of the corresponding link: at any instant, it uses the last w values to make its predictions. By considering sequences in the models, we capture temporal patterns such as trends and seasonality, enhancing predictive accuracy for future PDR values. By transforming the time series problem into a supervised learning task, we define the regression model as follows:

$$x_i = f(x_{i-w}, \ldots, x_{i-1})$$

where f represents the predictive model.

B. Model training

We can apply regression models for forecasting at any instant to make predictions. These models are useful for their simplicity and efficiency in predicting outcomes based on input features. In the context of time series forecasting, these models are designed to predict a target variable x_w using one or more independent variables from previous time points $(x_1, x_2, \ldots, x_w)$. In our study, we consider the following lightweight 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 [18].
- **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 [19].
- **Decision Trees**: it 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 [20].
- **AdaBoost**: AdaBoost (Adaptive Boosting) works by combining multiple weak learners, typically decision trees to form a strong classifier [21].
- **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 [22].

Note here that we exclude neural network models due to their high complexity, and ARIMA models because they are better suited for stationary data. During the training phase, we aim to identify the best regression model and the optimal window size for each link. To achieve this, we allocate 33% of the data for training (approx. eight hours), representing the network deployment phase, to construct the models. Then, we

select the models giving the best predictions on that same training data, along with its best window size w. The choice of window size significantly impacts model performance: a small w may fail to capture long-term dependencies, leading to underfitting, while a large w can introduce unnecessary complexity and redundancy, increasing the risk of overfitting.

C. Continuous Predictions

We describe here the prediction process during the testing phase using a given regression model. The objective is to predict the evolution of the PDR for a given number of intervals. More precisely, we predict the PDR values for the next p time intervals recursively from the last observations, as follows:

At the interval i, we first predict the next value with the last w values. Then, we proceed iteratively: we use the predicted value to make the next prediction. We keep a fixed sequence with w elements and consequently remove the first value of the previous sequence. Thus, after predicting $\hat{x}(i+1)$, the next prediction is made using the inputs $(x_{i-w+1}, \ldots, x_i, \hat{x}(i+1))$, treating $\hat{x}(i+1)$ as an observed value. This process is repeated p times to generate a sequence of p predictions.

Let us call the set of predictions according to the different prediction steps (up to p) at the i-th interval $\hat{X}(i)$. The matrix $\hat{X}(i)$ represents the predicted values at different prediction steps. At each instant i, the model predicts up to p future values, forming a diagonal structure. The prediction matrix is structured as follows:

$$\hat{X}(i) = \begin{bmatrix} \hat{x}_{1,1} & \hat{x}_{1,2} & \cdots & \hat{x}_{1,p} & \cdots & \hat{x}_{1,i} \\ & \hat{x}_{2,2} & \cdots & \hat{x}_{2,p} & \cdots & \hat{x}_{2,i} \\ & & \ddots & \vdots & & \vdots \\ & & & \hat{x}_{p,p} & \cdots & \hat{x}_{p,i} \end{bmatrix}$$

where the elements in the matrix correspond to: (i) **rows:** the prediction step (how far ahead the model predicts) and (ii) **columns:** the time interval i. At each time interval i, the model takes the last w values from the real sequence (or previously predicted values) and generates a sequence of predictions recursively.

Considering that the last observed value is x_i, this process can be expressed as:

$$\hat{x}_{m,n} = f([v_{m-w,n-w}, \ldots, x_{m-1,n-1}]); \tag{1}$$

where $m \leq p, n \in \mathbb{N}^+$, and:

$$v_{k,j} = \begin{cases} x_k, & \text{if } k \leq i \\ \hat{x}_{k,j}, & \text{otherwise} \end{cases} \tag{2}$$

Figure 5 illustrates this process, showing how the $\hat{X}(i)$ matrix is constructed, with an example where $w = 3$ and $p = 4$. In practice, it is preferable to have $p \leq w$ to prevent the model from predicting more values than the historical data it relies on, thereby reducing the risk of divergence. However, this example is presented solely for illustrative purposes.

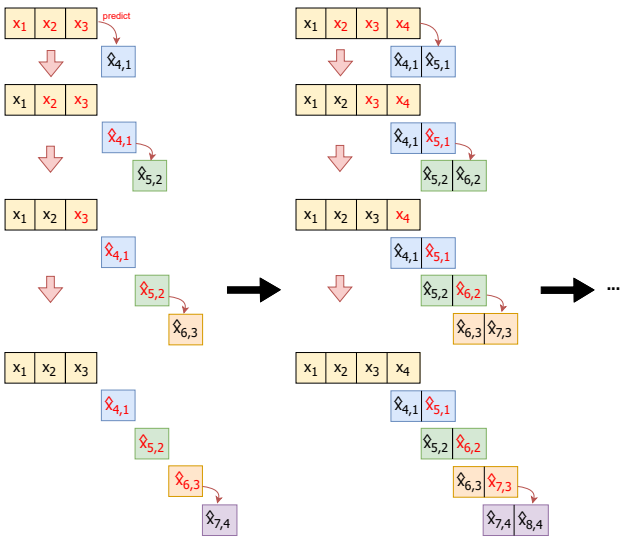


Fig. 5: Prediction matrix construction. We consider the example where $w = 3$ and $p = 4$. At each time step, the model utilizes the most recent observations along with previously predicted values (highlighted in red) to generate the next prediction. This process forms a sliding window that progressively shifts from real observations to the diagonal of the prediction matrix. Once a new observation becomes available, the process repeats, constructing a new diagonal, and continues iteratively.

As new data points arrive from the network, the model is updated dynamically—when a new value x_{i+1} is received at the $i+1$-th interval, the model is retrained using the most recent w values $[x_{i-w}, \ldots, x_i]$ as input and x_{i+1} as the target. This process continues iteratively throughout the deployment, ensuring that the model adapts to evolving network conditions.

D. Prediction strategies

We consider here the two following variants for the predictions during the deployment:

1) Variant 1: Fixed model: We consider that each link executes a specific **fixed** model. Our hypothesis is that the model achieving the best performance during training should also deliver satisfactory performance during deployment. Note that this assumes a relatively stable link quality.

This approach is described in Algorithm 1, and works as follows:

1) We store the new observation in a time series (line 3);
2) We use the new observation to refit the model: we can compute the error between this new observation and the corresponding predictions we made in the past (line 5);
3) We can also trigger a prediction: we recursively use our model to make p predictions. All the predictions are stored in $\hat{x}_{i,k}$ (lines 6-11)

Figure 6 illustrates the architecture of this approach, where each link is represented by a single model used for predictions during deployment.

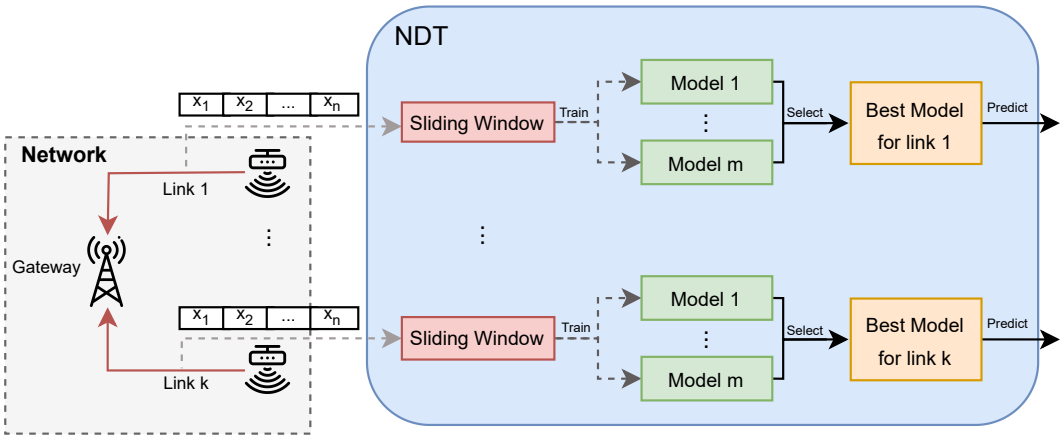


Fig. 6: Architecture of our approach. Each link is model separately according to the corresponding measured, and the best model is used for predictions.

Algorithm 1: Fixed Model

Input: w: Sliding window size
Input: p: Prediction step
Input: `model`
Output: At any instant, the next predictions for the PDR values $\hat{x}(*)$

$i \leftarrow 0$
while *a new observation* x_i *is measured* **do**
  /* Store the new observation */
  $x \leftarrow x \cup \{x_i\}$
  /* Refit the model comparing prediction and observation */
  if $i \geq w + 1$ **then**
    $\texttt{model}.fit[(x_{i-w}, \ldots, x_{i-1}), x_i]$
  /* Predict the next p values iteratively */
  if $i \geq w$ **then**
    $x_{input} \leftarrow (x_{i-w}, \ldots, x_i)$
    for $k \in [1, p]$ **do**
      $\hat{x}_{i,k} \leftarrow \texttt{model.predict}(x_{input})$
      /* Remove the first value of the sequence and insert the predicted value */
      $x_{input} \leftarrow x_{input} \setminus \{\text{first element}\}$
      $x_{input} = x_{input} \cup \{\hat{x}_{i,k}\}$
  $i \leftarrow i + 1$

Output: $\hat{X}(i)$

Algorithm 2: Adaptive Model

Input: w: Sliding window size
Input: p: Prediction step
Input: q: History window interval
Input: `models`
Input: `BestModel`
Output: At any instant i, the next predictions for the PDR values from the `BestModel`: $\hat{x}_{(*)}^{\texttt{BestModel}}$

$i \leftarrow 0$
while *a new observation* x_i *is measured* **do**
  /* Store the new observation */
  $x \leftarrow x \cup \{x_i\}$
  /* Enough observations */
  if $i \geq w$ **then**
    for *model in* `models` **do**
      /* Refit models + Predictions */
      $\texttt{model.fit}((x_{i-w}, \ldots, x_{i-1}), x_i)$
      $\hat{X}^{model}(i) \leftarrow$ Algorithm 1(w, p, `model`)
    /* Best models from prediction error */
    for *model in* `models` **do**
      $error^{\texttt{model}}(i)$ following Equation 3
    `BestModel` following Equation 4
  $i \leftarrow i + 1$

2) Variant 2: Adaptive Model: The previous approach presents limited accuracy for bursty links: the best model is selected and fixed for a specific link.

We propose here a self-adaptive approach for model selection, where the best-performing model is chosen dynamically at each ($R = 1$) iterations. Thus, we can here change the parameters of the model and the model itself. For this purpose, we execute in parallel all the different models. Then, we select continuously, at each time interval, the best model, according to the last predictions.

More specifically, at each interval, we compute the pre-

Parameter	Signifcation	Value
T	Time interval for the computation of number of received packets	50 seconds
w	Sliding window interval	$w \in \{3, 5, 10, 15, 20\}$
p	Prediction step	$p = 20$ intervals
q	History window interval	$q = 5$ intervals
R	Refit frequency	$R = 1$ interval

TABLE III: Signification and default value for the different parameters.

diction error of all models over the last q predictions. This enables us to retain only the model that has demonstrated the best recent performance. The underlying hypothesis is that the model performing best for the latest predictions should continue to yield satisfactory results in the subsequent predictions, up to a certain point.

To calculate the error of a model up to the i-th interval, for each step j from 1 to p, the predicted values $[x_{i-q,j}, \ldots, x_{i,j}]$ are compared against the actual values to compute step-wise errors for the last q intervals, where i is the current interval index. The overall prediction error up to the i-th interval is then calculated as the sum of Mean Squared Errors (MSE) over the last q intervals for the p different steps:

$$error^{model}\ \text{(i)} = \sum_{j=1}^{p} \left(\frac{1}{q} \sum_{k=i-q}^{i} \left(x_k - \hat{x}_{k,j}^{model} \right)^2 \right)^2 \quad (3)$$

The model with the lowest error is selected as the new `BestModel` for the next R intervals:

$$\texttt{BestModel} = \underset{m \in models}{argmin}\ (error^{\text{m}}) \quad (4)$$

As a result, predictions over the network's lifecycle may come from different models, ensuring that the system does not rely on a single model that may become inefficient over time.

Algorithm 2 proceeds in detail as follows:

1) The first part of the process is executed, once per model and per link to make the predictions, as previously (lines 6-7);
2) We compute the average error achieved with each model for the last intervals (lines 8-9);
3) We select for the next predictions the best model, *i.e.*, those minimizing the recent error (line 10).

V. Performance Evaluation & Validation

We evaluated our approach using the network described in Subsection III-A to validate it. To simulate the behavior of a real DT, we split the collected PDR measurements into two datasets: 33% (approximately eight hours of deployment) for model training and the remaining 66% (roughly sixteen hours) for testing. The testing dataset represents the projected future data that a real network would transmit to its DT.

For each link, we recorded the number of correctly received packets and generated time series representing the count of successfully received packets within successive $T = 50$-second intervals. To assess model performance, we used the MSE metric which quantifies the deviation between predicted and actual values during testing. The overall MSE for each model was determined by averaging the MSE values across all network links.

To sharpen our analysis, we varied the prediction horizon from 1 to 20 steps, testing each model's ability to forecast packet reception from 50 to 1000 seconds ahead. In this way, we assessed the models' effectiveness in both short- and medium-term forecasting.

A. One step prediction Error

Figure 7 compares the predictions achieved with the adaptive technique with the measurements observed for the considered links in Section III-C, with $p = 1$, which means that the models are used for one prediction only and are recalibrated at each interval. To facilitate interpretation, we evaluate performance using the Mean Absolute Error (MAE), noting that similar results were observed with the MSE metric. The results demonstrate that adaptive regression effectively captures the complexities of the experimental data across different link types, including highly dynamic links such as **m3-133˙m3-163** and **m3-166˙m3-163**.

B. Adaptive Model vs Fixed Model

We compare here the two approaches described in section IV-C:

fixed selects the best model at the end of the training phase. It remains fixed for the rest of the time;

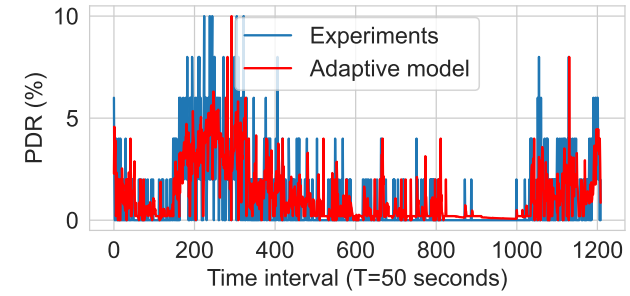
adaptive selects the best model at each iteration ($R = 1$), using the most recent q predictions (and their corresponding error).

Notably, both approaches use a separate model for each link, trained solely on observations from that specific link.

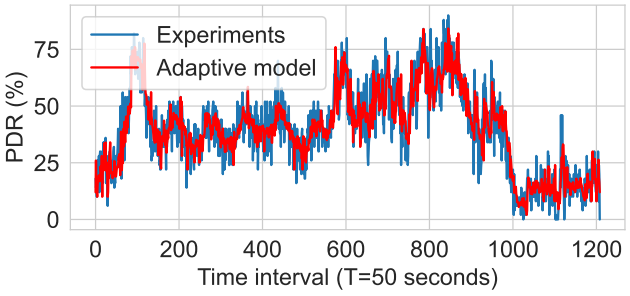
We identified in the preliminary section three distinct categories among the network links (cf. Figure 3), excluding the bad links: (1) excellent wireless links, (2) good performing links, and (3) average performing links. Therefore, we evaluated the behavior of fixed and adaptive approaches separately for the different link categories. We omit to consider the bad quality links due to their constant low PDR, which makes it easy to provide accurate predictions.

As Figure 8 shows, across all classes, the adaptive method consistently outperforms the single-model approach at every prediction step, demonstrating its ability to handle dynamic link variations and enhance prediction accuracy. Notably, the prediction error remains remarkably low at step 1, highlighting the method's effectiveness in short-term predictions, with errors increasing gradually as the prediction horizon extends.

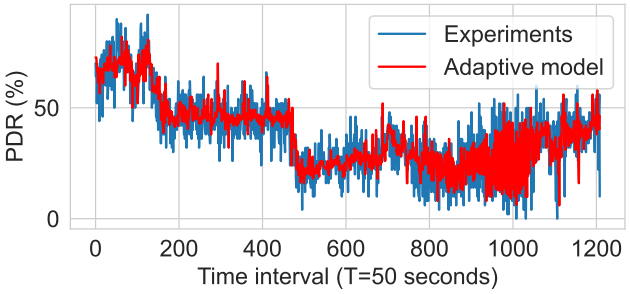
For average links, the method performs well in shorter prediction steps, maintaining errors between 6% and 8%. This is likely due to minimal variation over short intervals, allowing the model to retain high accuracy. As the prediction step increases, performance declines, but errors remain below 12%,



(a) m3-150_m3-163

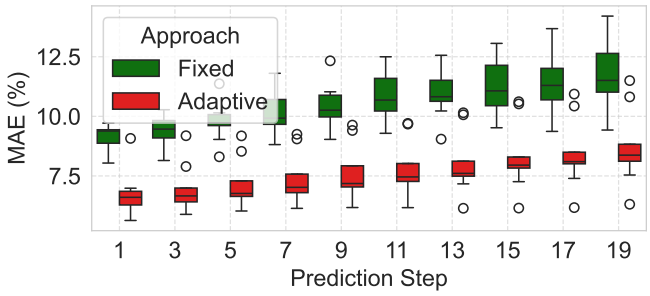


(b) m3-133_m3-153

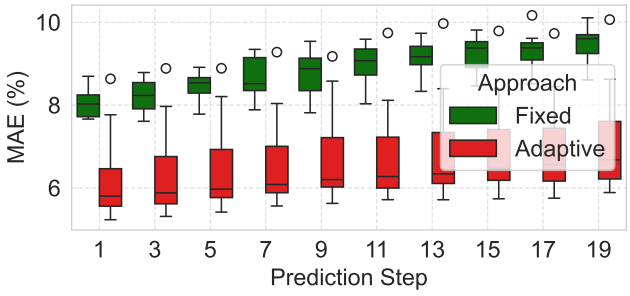


(c) m3-166_m3-163

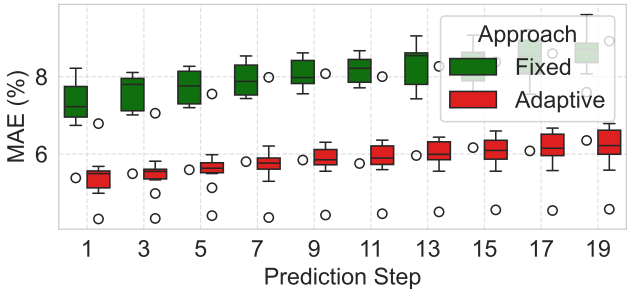
Fig. 7: Wireless links time series for three links from different clusters: Real Measurements vs. Adaptive Regression ($p = 1$)



(a) Average links (n=9)



(b) Good links (n=8)



(c) Excellent links (n=12)

Fig. 8: Prediction error of the adaptive method evolution on the different clusters.

demonstrating the method's resilience in sustaining reasonable accuracy over longer horizons.

For good links, both approaches perform even better. The adaptive method remains robust, particularly for short-term predictions, with errors around 6%. Even as the prediction step increases, errors stay within a manageable range, not exceeding 9%. This suggests the method effectively adapts to dynamic conditions while maintaining reliability.

For excellent links, performance improves further, with errors generally staying below 6%, even for higher prediction steps.

Overall, the method's robustness across all link categories highlights its versatility and potential for broader application in diverse network conditions. While the adaptive approach incurs higher computational costs than a single regression model, the simplicity and efficiency of the underlying models ensure that continual learning remains feasible. As a result, resource consumption remains well within acceptable limits for network management.

- Prediction time: In Table IV, we present the approximate time required, at a given instant and for a given link, to refit the model(s) and perform a prediction for both approaches, across the four cluster categories (Bad, Average, Good, Excellent), using a personal computer equipped with an Apple M2 chip. Each row reports the minimum, median, and maximum prediction times. From these results, two main conclusions can be drawn: (i) the prediction times for both approaches are low, demonstrating their capability to forecast the next $T = 50$s of transmissions in under one second—thus providing sufficient time for network-level decision making; and (ii) the adaptive model approach is clearly more time-consuming than the fixed (best) model, as it relies on multiple models (six

in this case) during both training and prediction phases. It is also worth noting that the high versatility observed in the *Bad* cluster is due to the heterogeneity of the links themselves: while some of them are stable and can be accurately modeled by *decision trees*, others are much more dynamic and require more time-consuming models such as *Extra trees*. Following this, assigning a distinct model to each link may become computationally expensive in large-scale networks, especially under high traffic conditions. One potential solution is to cluster links exhibiting similar behavior and use a shared model for each group, thereby reducing the computational overhead while preserving modeling accuracy.

TABLE IV: Prediction times statistics per cluster and model for the fixed and adaptive approaches.

Cluster	Approach	Min (s)	Median (s)	Max (s)
Bad	Adaptive	0.1705	0.3065	0.4768
	Fixed	0.0011	0.0810	0.1379
Average	Adaptive	0.5039	0.5562	0.5803
	Fixed	0.0017	0.0019	0.0031
Excellent	Adaptive	0.5229	0.5577	0.5953
	Fixed	0.0019	0.0026	0.0033
Good	Adaptive	0.5264	0.5640	0.6006
	Fixed	0.0017	0.0023	0.0029

VI. Conclusion & Perspectives

In this paper, we introduced a self-adaptive approach as the foundation of a DT for wireless networks, allowing an accurate and lightweight replication. Our method redefines wireless network modeling by constructing a collection of DTs, each specifically tailored to represent an individual radio link. This approach facilitates the dynamic, real-time selection of the best model for each link, using the most recent observations, through a rigorous comparison process. Our model presents a key adaptability to the diverse and fluctuating conditions of wireless networks, as our performance evaluation highlights. To promote transparency, reproducibility, and further research, we have made our dataset, implementation and documentation publicly and completely available to the community[3].

While current evaluation is limited to a small-scale testbed, it provides a necessary proof of concept. Future work will focus on large-scale experiments to evaluate scalability and robustness. Also, we plan to extend our approach to include additional metrics, such as latency and the number of consecutively lost packets, which are crucial indicators of network quality for critical applications [23]. Another priority will be enhancing the generalization capabilities of the DT to predict the performance of unseen links, considering environmental factors like the distance between the sender and receiver or the specific deployment area. Additionally, implementing a bidirectional communication channel between the physical network and its digital counterpart remains an important open challenge. Recent work [24] has demonstrated the feasibility of such links with latency guarantees around 14 ms, making this a promising direction for future integration. These advancements will contribute to a comprehensive DT that not only replicates the wireless network with high fidelity but also supports reliable *what-if* scenario analyses.

[3]**Code sharing:** To ensure reproducibility, we provide the source code, datasets and documentation at the following link: https://github.com/SamirSim/Wireless-Link-Quality-Prediction

References

[1] Kannan Srinivasan et al. The β-factor: measuring wireless link burstiness. In *Conference on Embedded Network Sensor Systems (Sensys)*, pages 29–42. ACM, 2008.

[2] Paul Almasan et al. Network Digital Twin: Context, Enabling Technologies, and Opportunities. *IEEE Communications Magazine*, 2022.

[3] Piotr Gawłowicz and Anatolij Zubow. ns-3 meets openai gym: The playground for machine learning in networking research. In *MSWIM*, pages 113–120. ACM, 2019.

[4] Roberto Pegurri, Francesco Linsalata, Eugenio Moro, Jakob Hoydis, and Umberto Spagnolini. Toward digital network twins: Integrating sionna rt in ns3 for 6g multi-rat networks simulations. *arXiv preprint arXiv:2501.00372*, 2024.

[5] Samir Si-Mohammed and Fabrice Theoleyre. Data-driven prediction models for wireless network configuration. In *AINA*. Springer, 2025.

[6] NVidia. Omniverse platform. https://docs.nvidia.com/aerial/aerial-dt/.

[7] Davide Villa et al. Colosseum as a Digital Twin: Bridging Real-World Experimentation and Wireless Network Emulation. *IEEE Transactions on Mobile Computing*, pages 1–17, 2024.

[8] F.A. Rodríguez-Corbo et al. Propagation models in vehicular communications. *IEEE Access*, 9:15902–15913, 2021.

[9] Marco Zúñiga Zamalloa and Bhaskar Krishnamachari. An analysis of unreliability and asymmetry in low-power wireless links. *ACM Trans. Sen. Netw.*, 3(2):1–34, jun 2007.

[10] Keoma Brun-Laguna, Pascale Minet, Thomas Watteyne, and Pedro Henrique Gomes. Moving beyond testbeds? lessons (we) learned about connectivity. *IEEE Pervasive Computing*, 17(4):15–27, 2019.

[11] Nouha Baccour et al. Radio link quality estimation in wireless sensor networks: A survey. *ACM Trans. Sen. Netw.*, 8(4), sep 2012.

[12] Alok Ranjan et al. Rssi or lqi: Insights from real-time deployments for underground sensing and applications. In *INFOCOM WKSHPS*, pages 1231–1236. IEEE, 2020.

[13] Miguel Landry Foko Sindjoung and Pascale Minet. Wireless link quality prediction in iot networks. In *PEMWN*, pages 1–6. IEEE, 2019.

[14] Hanane Benadji, Lynda Zitoune, and Véronique Vèque. Predictive modeling of loss ratio for congestion control in iot networks using deep learning. In *GLOBECOM*, pages 2814–2819. IEEE, 2023.

[15] Tao Liu and Alberto E. Cerpa. Data-driven link quality prediction using link features. *ACM Trans. Sen. Netw.*, 10(2), jan 2014.

[16] Eduardo Nuno Almeida et al. Machine learning based propagation loss module for enabling digital twins of wireless networks in ns-3. In *Workshop on ns-3*, pages 17–24, 2022.

[17] Fredrik Osterlind et al. Cross-level sensor network simulation with cooja. In *LCN*, pages 641–648. IEEE, 2006.

[18] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14:199–222, 2004.

[19] Jerome H Friedman. Greedy function approximation: A gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[20] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1986.

[21] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *ICML*, volume 96, pages 148–156. Citeseer, 1996.

[22] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63:3–42, 2006.

[23] Carlos J. Bernardos, Georgios Z. Papadopoulos, Pascal Thubert, and Fabrice Theoleyre. Reliable and Available Wireless (RAW) Use Cases. RFC 9450, IETF, August 2023.

[24] Clifton Paul Robinson et al. Twinet: Connecting real world networks to their digital twins through a live bidirectional link. In *GLOBECOM*, 2024.