Smart Integration of Network Simulation in Network Digital Twin for Optimizing IoT networks

Samir Si-Mohammed[a,b,*], Anthony Bardou[a], Thomas Begin[a], Isabelle Guérin Lassous[a], Pascale Vicat-Blanc[a,b]

[a]*ENS de Lyon; UCBL; CNRS; INRIA; LIP; F-69342; LYON Cedex 07; France*
[b]*Stackeo; Lyon; France*

Abstract

Network Digital Twin (NDT) and Network Simulation (NS) are two paradigms, leveraging virtual representations of networks, to help decision-making. These tools may seem similar and are often confused or opposed. However, their concepts differ, having their respective purposes and use cases. The goal of this article is to explore and clarify the specificities, the benefits and the limits of these two decision support tools, analyze how they complement each other and can be nicely combined. We argue that a smart integration of NS in NDT can ease, accelerate and strength decisions for network design, deployment, operations and evolution. Focusing on the domain of IoT solutions, where wireless networks are critical but complicated to configure for meeting specific application requirements, we show how NS, specifically discrete-event simulation, coupled with NDT can contribute to supports IoT architects and operators decisions throughout the life cycle of an IoT network. We analyse the different steps required to use NS in the context of NDT and examine how this helps remove NS barriers such as credibility and reliability. In particular, we show how NDT data enable to fine tune and customize the energy consumption models, making the simulation results more context-aware and insightful. Then, addressing the often-prohibitive simulation cost required for supporting decision, we propose to associate surrogate modelling to NS in the NDT. As surrogate models, we first introduce a simple ML (Machine Learning)-based surrogate model and illustrate this method with two IoT networks configuration optimization use cases. Secondly, we propose a Bayesian optimization approach based on Gaussian Processes as surrogate model to further accelerate the (re)configuration decisions. We show how this method enables to select simulation scenarios that converge rapidly to the optimal solution, and allows the NDT to timely perform the adaptation. The contribution of this article is threefold. It provides i) the first systematic analysis of the differences and potential synergies between NDT and NS; ii) a synthetic presentation of the integration of NS and associated decision algorithms within a NDT to unlock its accessibility and its credibility throughout the life cycle of an IoT solution; iii) a proposal for a smart and cost-efficient integration of simulation in NDT via surrogate modeling for reducing evaluation and optimization cost, paving the way to NDT-based dynamic adaptation and real-time optimization of IoT networks.

Keywords: Network Digital Twin; Network Simulation; Machine Learning; Internet of Things; Connected Solutions; Wireless Network, Configuration.

1. Introduction

The *Digital Twin* (DT) concept has been pioneered by NASA in 2012 and initially defined as an integrated multiphysics, multiscale simulation of a vehicle or system that uses the best available physical models, sensor updates, fleet history, etc., to mirror the life of its corresponding flying twin [1, 2]. Since then, it has been growing and continues to expand in the manufacturing industry but also in many other industrial domains such as transport, energy, utilities, buildings and more recently in health, biology, telecommunications and computer networks.

A "classical" Digital Twin maintains a continuously up-to-date digital representation of the physical world entities of interest in their environment, to provide holistic insights for optimal decision-making. Digital twins use historical and real-time data to represent the past, the present and predict possible futures.

Similarly, Network Digital Twins (NDTs) have recently emerged as the virtual representation of physical networks capable of collecting, storing and processing information from the real environment to represent and analyze their past, present and future behaviors. Currently, NDT is typically envisionned for network operations, optimization or capacity planning [3].

On the other hand, the networking community has relied for years on Network Simulation (NS) for supporting design decision-making in complex scenarios [4]. NS are computer programs that imitate the operations of network elements within a network for analyzing its performance and testing new architectures and protocols. NS used to be considered as safer, cheaper and more controllable than real world experimental network deployment [5]. It gives the possibility to test how a network will work before building it at scale, it can be used to find unexpected problems and to explore 'what-if' scenarios. As a software entity, NS can speed up or slow down the system behavior to see changes over long or short periods of time. NS is typically used during the design or the upgrade phases of a

*Corresponding author
Email address: samir.si-mohammed@ens-lyon.fr (Samir Si-Mohammed)

network.

As NDT and NS both leverage a virtual representation of a network and are utilized for decision-making, they may seem to be similar paradigms. However, their concepts differ, having their respective purposes and use cases. They present also their own drawbacks and limitations. We argue that they should not be confused or opposed but combined.

Therefore the goal of this article is to explore and clarify the benefits as well as the limits of these two network decision-support tools, and to analyze how they complement each other and can be efficiently associated for pushing their respective limits to deliver the better solutions for optimizing networks throughout their life cycle.

To examine this in depth, and as NDTs, dependant of the physical networks they mimic, make sense only in the context of real deployments, we consider the specific domain of the Internet of Things (IoT).

In IoT, wireless networks play a critical role but are complex to configure and optimize. Responsible for the connectivity of IoT devices to the Internet and for the transfer of collected data to computing resources for analysis, the IoT network is the cornerstone of any IoT solution. It often determines the viability of an IoT solution, in terms of QoS, but also economic profit and environmental impact. IoT end-devices are often small components with limited CPU, power and memory capacity but are supposed to last a long time. Ensuring a low energy consumption, good performance and reliable data transmission at the network level is therefore a priority. This often translates to determining the right network configuration for devices and appliances, but in practice, this is difficult and requires expertise.

Network administrators and engineers encounter a lot of the difficulties and obstacles when setting up and managing wireless networks. The network configuration challenges can be related to scalability (when the number of device increases and the infrastructure expands), security, interoperability, QoS, performance optimization, change management, compliance, remote management, etc. These challenges usually require skilled network operators with in-depth knowledge of networking technologies, protocols, and best practices. In the IoT domain, the presence of wireless network specialists all along the live cycle of an IoT deployment is relatively rare. In such context, the anticipation and automation provided by NDT technology could be of great help to simplify network configuration and mitigate some of the issues discussed above.

In this article, we study how the integration of NS within the NDT can contribute to supporting decisions of the whole IoT team, including IoT architects, throughout the life cycle of the IoT network.

To illustrate the proposed approach, we focus on the IoT network configuration decision and optimization. Each IoT network technologies such as LoRa, Sigfox, NB-IoT, LTE-M, Zigbee, 6LowPAN, can be configured differently, according to many configuration parameters. As example, for LoRa, these parameters are: spreading factor (SF) which determines the speed at which the signal frequency changes across the bandwidth of a channel; coding rate, indicating of how much of the data stream is actually being used to transmit user data; type of traffic determining whether the data is sent with or without an acknowledge. Each of these parameters can take various values, each potentially exerting a notable influence on the achieved performance. Thus, the configuration decision significantly and directly affects the operational performance of the solution.

Moreover, the needs of performance depend on the application: the same configuration can be efficient for one application and much less efficient for another context[6]. A reconfiguration can be also be required if the application characteristics evolve over time. Application-centric network simulation has proved to be powerful in this context [7] but may lack of precision and reliability. We examine how the combination with NDT can help remove NS barriers such as credibility. We show how NDT data enable to fine tune and customize the simulation models, such as energy consumption models, making the simulation results more context-aware and insightful.

However, it has been observed that, for a given IoT network and application context, the configuration parametric space is often large and that small changes to one parameter can lead to unexpectedly large effects on simulation outcomes, or vice versa. This means that the relationships between model parameters can be highly non-linear, requiring to run a large number of simulations before converging on the appropriate decision. However, performing these kinds of analyses can become both time- and cost-prohibitive specially in the constrained framework of an NDT. Surrogate modelling is used in engineering [8] for accelerating such compute-intensive simulation-based analyses. With the aim of lowering simulation cost required for supporting NDT decision, we explore two surrogate modeling methods, one based on simple Machine-Learning regression models, the other Gaussian Processes and Bayesian Optimization in the context of smart connected solutions and show how they accelerate and improve (re)configuration decisions.

The contribution of this article is threefold. It provides

- The first systematic analysis of the differences and potential synergies between NS and NDT;
- A synthetic presentation of the integration of NS within the NDT toolbox in the domain of IoT to make NS accessible and reliable throughout the lifecycle of an IoT solution.
- A proposal for a smart and cost-effective integration of NS in NDT via surrogate modelling and offline and online optimization for reducing the cost of simulation while improving decision, paving the way to dynamic adaptation of IoT networks via NDT.

The article is organized as follows: Related work and a comparative analysis of NS and NDT is developed in Section 2. Section 3 analyzes the benefits and challenges of the combination of NS and NDT. Section 4 describes the integration of NS within the NDT of an IoT network. Section 5 focuses on the acceleration of simulation via Ml-based surrogate modelling. Section 6 presents a GP-based surrogate modeling and Bayesian optimization approach for unlocking dynamic configuration adaptation via an NDT. Conclusions are established and perspectives proposed in Section 7.

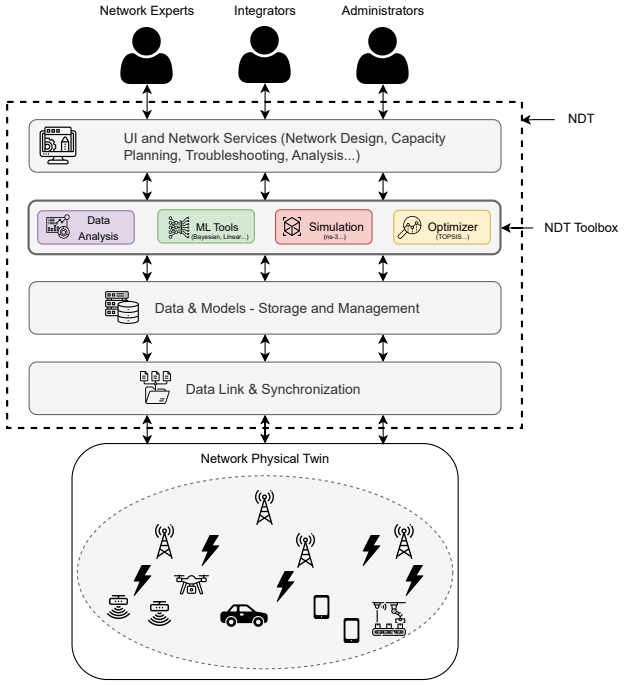

Figure 1: NS integration within NDT tool box

2. Related Work

2.1. Viewpoints on Simulation versus Digital Twin

Simulation has emerged in the 80's in the domain of aeronautics and is now largely used in many domains [9], while the concept of Digital Twin (DT), appeared in 2010 [2], is still maturing and not well defined [10]. As they both leverage digital models for decision-making throughout the life cycle of their physical counterpart, the two digital concepts are often confused or opposed. Consequently, in many sectors, questions around the real difference and convergence between Simulation and DT are raised.

For example, in [11], the industrial manufacturing community proposes different viewpoints: For Lendermann, DT is a new term for Modeling and Simulation (M&S), it just makes it **faster and more agile**. On the contrary, Shao considers that M&S help to understand **what may happen**, while DT enables to understand **what is happening or has happened**. He underlines that simulation is mainly used for design and offline optimization. He also interestingly notes that a starting point of DT solution approach is often to create a virtual simulation model.

For Shao et al. [12] the level of data integration between the digital twin and its physical counterparts is what creates the differences between digital model, digital shadow, and digital twin based. In summary, most of the offline simulation models are classified as digital models; simulation models that use near real-time sensor data as inputs are digital shadows; and simulation models that use near real-time sensor data as inputs and also control the physical counterparts by updating control parameters are DTs. This **connection and synchronisation with the physical counterpart** appears to be one of the main difference. In smart factory, the communication between the virtual simulation model and the real-world system is performed through an IoT solution.

On their side, Boschert and Rosen in [13], claim that M&S is established as a standard process in system development, e.g. to support design tasks or to validate system properties. For them, the main purpose of the DT is to make the information created by M&S during design and engineering also available and ready for evaluation during the operation of the system. This is nowadays often neglected, as data produced with M&S in design are often not accessible and reusable in operations.

According to [14], the DT integrates all data (test, operation data, etc.), models (design drawings, engineering models, analyses, etc.) and other information (requirements, orders, inspections, etc.) of a physical asset generated along its life cycle that leverage business opportunities. The role of the DT is to predict and optimize performance. To this purpose, simulation methods and/or data-based methods are used. DT provides an interface to different models and data in different granularities and keeps them consistent. As the complexity of the objects to be designed increases, involving an ever-growing number of disciplines and people, the need for consistency among these models increases. The DT as a common repository for all digital artefacts provides this consistency. Therefore the **persistence and consolidation of the physical entity information and data, throughout its lifecycle**, within the digital twin, is an other aspect, which differentiates it from simulation, which is a tool activated occasionally, only when needed.

For [15], the diverse applications of DT such as simulation, real-time monitoring, testing, analytics, prototyping, end-to-end visibility, can be broadly classified as sub-systems of DT (for example, a DT can be used for testing during prototyping, for real-time monitoring and evaluation, or for both). It is the **presence of all the components that makes a DT different from each of these individual methods**.

2.2. Network Simulation versus Network Digital Twin

Network simulation (NS) is the process of creating a virtual environment that mimics the behavior of a real network. It requires a specialized software, called simulator, to model and emulate network devices, protocols, and traffic patterns. Network simulation allows to test and evaluate the performance, scalability and behavior of a network before implementing changes or deploying new technologies. It helps identify potential issues, optimize network configurations, and validate network designs without impacting the production environment.

When defining and preparing the deployment of a network in a real environment, network experts use to leverage different network performance evaluation methods such as NS, analytical modeling [16] or experimentation [17] to have a long-term and large-scale perspective and make informed decisions. Researchers often consider NS as a better choice for in-depth evaluation. Indeed, NS provides good insights about the future performance of a network system and permits to test what-if scenarios at scale to trade-off cost, QoS and energy efficiency.

NS complements small scale proof of concept and pilot deployments when large scale assessment is required. In this case, NS is cost-effective since it prevents the massive deployment of real equipment.

Field network architects and operators rely on vendor data, benchmarks, prototyping but rarely on scientific tools, which are widely used by network researchers. The concept of NDT emerged recently, driven by the massive wave of development of digital twins in all economic sectors. The NDT paradigm is attracting interest from both the academic and industrial networking communities but is still a very immature concept and technology [18, 19].

Network standardization bodies are attempting to clarify NDT concepts, architecture and challenges [20], [21]. They define an NDT as a virtual representation of a physical network infrastructure. It provides a comprehensive view of the entire network, enabling better decision-making and planning. It comprises descriptive models of the network components, devices, configurations, and data-based models of behavior to predict the performance and future behavior of the actual network. A network digital twin collects and stores real-time data from the network for optimizing network operations. Different NDT architectures, encompassing numerous network services such as predictive maintenance, network diagnosis, efficient energy optimization, security management, optimized resource allocation and real-time network monitoring have been proposed [22].

Intent-Driven Network (IDN) has also been introduced by [23] as a concept that establishes a NDT connecting the physical network and business intent, and enabling efficient detection and resolution of network issues, prediction of future network conditions, and proactive elimination of network risks, thereby enhancing network reliability.

NDT researchers focus their work on the exploitation of the data provided by the physical twin via machine learning (ML) approach to enhance network operations. In [24], Alasaman et al. develop the technologies and research challenges involved in implementing a ML-based NDT. According to these authors, ML models can achieve similar accuracy to packet-level simulation, which are considered computationally expensive modeling tools, while keeping a limited execution cost similar to lightweight analytical models. This allows network operators to accurately control the network at much shorter timescales.

In [25] the authors dedicate the usage of NDT specifically on 5G networks security assessment, cyber security teams training and cyber risk evaluation. Their approach is based on Generative Adversarial Networks (GANs), deep neural networks able to learn from a set of training data and generate new data with the same characteristics as the training data. This way they generate synthetic flow-based network traffic to mimic both attacks and normal traffic without using NS.

For [26], NDT can help operators manage the IoT network as it has been developed.

In [27], a network simulator (ns-3) is used to reproduce a network testbed through network traces, representative of the dynamic conditions of the environment. It enables the creation of a digital model of the original wireless network environment, which allows the validation of novel solutions and the evaluation of their performance in realistic conditions in NS.

As conclusion, NS is largely used by the networking community, NDT is emerging as a promising technology for network management and optimization, and the coupling of the simulation models with the physical network has been fruitfully explored by network researchers. However, we observe that, despite this promises, the networking community seems to oppose NS and NDT and rarely propose to couple them.

As many engineering disciplines have identified the potential synergy of between simulation and digital twin, in the following section we propose to analyze the benefits and challenges of combining NS and NDT.

3. Combining Network Simulation and Network Digital Twin

3.1. The respective limits of Network Digital Twin and Network Simulation

Network Digital Twin are very recent and are currently focusing on helping decision making during the operation phase, by leveraging network monitoring data and ML-based observability tools [3, 27]. NDT limits and research challenges are :

1. **Prediction capabilities** : NDT relies on past data-based models; therefore it is difficult to anticipate the behavior of the network in future situations.

2. **Troubleshooting and root cause analysis capabilities** : NDT collect descriptive and historical data; therefore it is difficult to understand the failure or bottlenecks of the network in unforeseen and unplanned conditions.

3. **Data Integration and Management** : consolidating diverse data sources, large volumes, data quality can become rapidly an issue when deployments are large. Consequently, the NDT cost in terms of data management and processing can be important.

4. **Timely decision-making** : Real-Time Data Processing to provide up-to-date insights and efficient decision support is a difficult challenge for NDT. It requires scalable and efficient algorithms for processing streaming data, as well as fast optimization algorithms.

5. **Model Accuracy and Validation** : It is crucial to represent the physical network behavior faithfully. This implies the need to capture the complex dynamics of different network components, such as routers, switches, and communication links. Validation methods should also be explored to ensure the accuracy and reliability of the NDT models. This results in complexity for NDT design, model building and validation.

6. **Security and Privacy** of sensitive network information to protect NDT data from unauthorized access and cyber threats.

Approach	Network prototyping	NS	(current) NDT	NS-NDT
Purpose	Validation of feasability and hardware choices	Testing, evaluating, and validating designs and configurations	Real-time monitoring, Observability and Reconfiguration	Decision-support throughout the network lifecycle
Key phases	Design, Test and Development	Design and Deployment	Operations and Adaptation	Full life cycle (from Design to End-of-life)
Main use cases	Feasability test	Validation of network's properties, Capacity Planning, Behavior prediction, What-if analyses	Traffic and performance monitoring, Troubleshooting, Failure prediction, Network Control	From Design to Dismantling decisions
Scope	Focused and small scale network infrastructure	Focused on specific aspects or scenarios within the network	Comprehensive view of the entire network infrastructure, including devices, configurations, and operational data	High-level holistic view and to low-level focused analyses of the network
Main difference	No model, real experiments	No twinning (no real-time data from the actual network), simulation, predefined scenarios and traffic patterns	Twinning (incorporates real-time data), no simulation, real-time tests and changes	Twinning, simulation, real tests, real data
Implementation	Small scale physical network prototype	need to creating a virtual model from scratch with a simulator, defining network parameters and scenarios	Physical network data and information discovery, collection and integration into a digital model	Creation of a high-level simulation model, gradually detailed and enhanced by leveraging information discovered and collected by NDT (such as inventory, device types...).
Advantages	Precise, grounded but limited insights	Enable large scale exploration	Consolidated data-based decisions	Verification, update of simulation models, model reuse for operation, enhanced decision support
Challenges	Costly and time consuming	Complexity, cost, credibility	Limited prediction and troubleshooting capabilities, data collection and management, model validation	Reduced complexity and cost, increased capability and credibility

Table 1: Network Simulation and Network Digital Twin comparison

On their side, Network simulators have been designed by network experts targeting network researchers and programmers. NS like ns-3 [28] require network expertise and knowledge in C++ programming to design experiments, to run them but also to analyze and exploit the results. Simulation is a fair tradeoff between cost and accuracy, because it allows to have test performance at scale at a lower cost than real deployments, but can be greedy in execution time and its results reliability, questionable given the abstraction and assumption made on the various parameters for building the simulation model. To summarize, the main NS limitations are:

- **Accessibility** : for simulator development, deployment and learning,
- **Cost** : in terms of run time and computing power.
- **Credibility** : questioning how trustful the results are.

We observe that both NDT and NS are challenged on complexity, cost (time, energy, financial), model accuracy and validation. In the next section, we analyse how they can mutually compensate their respective flaws. In this paper, we do not address the security issues of NDT.

3.2. Benefits of combining NS and NDT

By deriving and adapting the observations made in other domains [13], we summarize the potential benefits of NS and NDT combination (NS-NDT) below:

1. **NS to NDT**

   (a) **Prediction capabilities extension** : NS can provide NDT with models and capabilities for predicting network traffic, performance as well as failure at large scale. NS can also accelerate and reduce the cost and footprint of NDT solutions by providing analytical models. Moreover, when data are sparse or difficult to gather from the real world (failures, attacks...), NS can generate synthetic data reflecting abnormal situations, for enhancing ML-based model creation to better detect anomalies, troubleshoot or secure a network. When particular values may not be accessible for a direct measurement, the simulation models build during the design phase can be reused yo obtain them.

(b) **Troubleshooting acceleration**: By using the simulation models, it is possible to interpret the NDT measurements in a different way, rather than just detecting deviations from the norm. Several modes of failure can be simulated for the current situation trying to reproduce the actual measurement signals. The comparison of the simulated signals with measured ones can help to troubleshoot and root cause the problem. Simulation completes monitoring of the network operation to give early warning of a failure or an attack.

(c) **Data collection reduction** : NDT can reuse NS models and results throughout the life cycle and minimize life-data collection reducing data integration and management. Together with data from online monitoring, the simulation models and operation history provided by the NDT are also the base for more flexible service planning. A comprehensive picture exists of the condition of the system which also eases the maintenance and upgrade operations before a failure occurs.

(d) **Lower Complexity for NDT building** : Simulation model created during the design phase can help build the initial NDT model, then simulation model reuse will enable continuous network improvement. As the NDT provides a smart view on the available system information, models and results from earlier life cycle phases are accessible also for users of different disciplines. Existing models can be slightly modified to adapt to the changing requirements or context.

(e) **Improved operation** : During the design phase, the network has to be planed to withstand a given workload and proved via simulations. However, additional information from the simulation can be deduced with little extra effort, like how "well" the design criteria are fulfilled. Network entities which do not really fulfil well these conditions are the best candidates to become bottlenecks and points of failure. The availability of this information allows improved operations, as the most important vulnerabilities are already known in advance.

2. **NDT to NS**

(a) **NS accessibility improvement** : Network representation and user-friendly interface in NDT can improve NS accessibility and its results exploitation by non network experts.

(b) **Lower Complexity for model building** : Descriptive data such as inventory, configuration files, collected on the deployed physical twin, can help defining initial simulation model parameters and updated dynamically.

(c) **NS credibility increase**: Data from real deployment and operation, collected as part of the NDT, can serve as verification input for the simulation models and lead to their continuous improvement. Descriptive data as well as life data measured and stored in NDT can strengthen NS reliability and validation, enhancing NS credibility.

3.3. Research challenges and opportunities for NS-NDT

NS-NDT can unlock significant opportunities for network optimization, predictive maintenance, resilience analysis, and intelligent decision-making in networking industries but introduces new research challenges and opportunities. We summarize them below:

1. **Problem specificity**: Each network domain such as IoT, telecommunication, Cloud but also energy or utilities have unique challenges and opportunities. There is no general solution that would fit all potential decision problems. Tailored NS-NDT solutions must then be explored in each context with industry-specific use cases and requirements.

2. **Human-Machine Interaction** : Intuitive user interfaces, visualization techniques, and decision support systems that facilitate effective human-machine interaction, enabling users to understand complex network behaviors, interpret NS-NDT insights, and make informed decisions.

3. **Scalability and Performance** : Small scenarios can be simulated quickly with NS, whereas larger scenarios require a longer execution time. In many NDT use cases, during network exploitation for example, fast response times are important. Handling effectively large-scale network infrastructures comprising thousands or millions of nodes, requires to optimize the computational and storage requirements of NS-NDTs.

4. **Dynamic Adaptation and Optimization** : Evolution of network operations in response to changing conditions and requirements, requires the development and integration of adaptive algorithms and optimization techniques in NS-NDT that leverage simulation and real-time data to dynamically reconfigure network resources, improve performance, and optimize energy consumption.

NS-NDT can be implemented by the integration of NS as a part of the NDT analysis toolbox used for knowledge extraction [29]. One of the key differentiators between conventional NS and NDT is that the latter has regular synchronization with the real-world entity. We argue that this difference creates a great opportunity for multiplying the benefits of both NS and NDT and can address the challenges listed above for NS-NDT development. Figure 1 illustrates the conceptual architecture we propose for the smart integration of NS with other analytics and optimization tools within the NDT.

In the following sections we develop the principles of the NS-NDT solution for the specific IoT network use case (challenge 1 and challenge 4) in Section 4 before exploring the scalability (challenge 2) in Section 5 as well as the dynamic adaptation and optimization aspect (challenge 3) in Section 6.

4. Integration of NS in NDT for IoT Network Optimization

We develop here the mutual benefits of the integration of NS within a NDT to support the decision-making process throughout the life cycle of an IoT network. In particular, we show how NDT benefits from NS providing the capability to easily evaluate and compare various configuration parameter settings for optimizing network performance. Then we study how NS benefits from NDT data providing real conditions data that can be used to increase NS reliability.

4.1. Background

The lifecycle of an IoT solution generally consists in five phases [30]: (i) **Design**, for defining and planning the digital system and the service components, (ii) **Development**, which involves software and hardware components building and integration, (iii) **Deployment**, where the devices are installed and their services (data collection, etc.) configured and activated, (iv) **Operations**, where the whole IoT chain, from sensor to the IoT platform in the cloud is exploited and maintained (v) **Decommissioning**, where the solution is terminated and the IoT system dismantled. Various stakeholders, such as (i) IoT architects and integrators, (ii) IoT solution developers, (iii) IoT security and solution operators can be involved throughout the life cycle of an IoT solution [31]. In practice, IoT teams do not necessarily comprise wireless network experts.

The performance and scalability of an IoT network, critically impacts the ability of the IoT solution to support the communication requirements and to potentially handle increased processing volumes in the future. During the design phase and before deploying a given IoT solution, IoT architects need to make a range of decisions. Simulation can greatly help them for various deployment scenario comparison and what-if analyses. For example, [7] shows how simulation helps making the decision for IoT network technology selection of an IoT network technology. Network simulation allows them to assess the performance of different technologies and select the most relevant one.

From the network perspective, IoT applications are mainly characterized by their traffic workload (number of sources, message size and period, etc.) and environment (indoor, outdoor, rural, urban, etc.) which can evolve over time [32]. These evolutions typically trigger adaptations of the IoT solution during the operations phase of its lifecycle. Simulation can help IoT operators defining the best configuration parameters settings before enforcing them within a device fleet via the IoT device management platform in charge of firmware updates. With network simulation they could also evaluate the effectiveness of potential changes or improvements in an operational environment before implementing them.

In the following, we formalize and investigate the configuration optimization problem of the IoT network of an application that can evolve over time.

4.2. NS-based approach for IoT configuration optimization

The simulation methodology for evaluating different configuration parameters and for selecting the best ones in the context of the integration of NS in the IoT network NDT is based on the methodology developed in [33]. This approach is divided into 4 steps as follows:

1. **Application modeling**, where the value of the application requirements, KPIs performance goals and weights are defined.
2. **Configurations Generation**, where what-if scenarios, integrating the application with the network technology configured with various parameter settings, are designed and created.
3. **Evaluation**, where the set of generated scenarios are instantiated then simulated and the KPIs of each what-if scenario are obtained. For the evaluation, a framework such as the one presented in [7] can be used.
4. **Selection**, where what-if scenarios are ranked and the best network configuration and topology are identified.

Application modeling. Modeling an IoT application requirements from the IoT network perspective consists in characterizing the load it imposes on the network over time. This load is a function of the number of communicating end-devices and the individual traffic they are exchanging. The communication requirements of an IoT application are abstracted by the end-devices, the workload they impose on the network and their physical environment as defined in Table 2 below. To simulate and study the scalability of an IoT network solution, the minimal and maximal values expected for the different selected parameters have to be specified.

Application abstraction parts	Parameters
End-devices	• Minimal number • Maximal number • Battery capacity (Amperes.hour) • Mobility model
Workload	• Traffic direction (downstream and/or upstream traffic) • Message size (bytes) • Minimal frequency (packets/second) • Maximal frequency (packets/second)
Environment	• Type (embodying the radio conditions) • Scope (meters) (maximum distance expected between two end-devices) • Expected lifetime (days)

Table 2: Application Modeling Parameters.

For the environment, two cases can be considered: Indoor or outdoor, where the latter can be either (a) rural, (b) suburban or (c) urban. Inspired by [34], a propagation (path loss) model is associated to each environment type to characterize this environment.

For the sake of simplicity, this work considers only static end-devices (no mobility model).

IoT Network configuration parameters. Each IoT technology can be characterized by a set of parameters that can be divided into generic and specific parameters. Generic parameters, such as maximum data rate, characterize the network technology. Specific parameters depend on each network technology. For instance, in the case of LoRaWAN, the specific parameters include the Spreading Factor (SF), the coding rate and the type of traffic (unconfirmed or confirmed). Some parameters are easily configurable by developers or by software (*e.g.,* SF for LoRaWAN) while others are less tunable or simply out of reach for the users (*e.g.*, the transmission power for LoRaWAN or MCS (Modulation and Coding Scheme) in Wi-Fi).

The generic parameters which are common to all the network technologies are: (i) The data rate, which is the theoretical maximal amount of data that can be sent per unit of time, (i) the frequency band, which is the frequency where the radio waves operate on and (iii) the topology type, which can be star or mesh. The number of gateways, which characterizes here the network infrastructure is considered as a common parameter for all technologies. We define a network configuration by a combination of values of these parameters (including the number of gateways).

Performance indicators definition. A Key Performance Indicator (KPI) is a metric to be evaluated and optimized. For an IoT scenario, we select the following KPIs:

- **Packet delivery**, which represents the amount of correctly received packets among all the sent ones.
- **Energy consumption**, which is the amount of energy consumed by the end-devices during the network deployment.
- **Packet latency**, which is the time that packets take to flow from the end-devices to the gateway.
- **Cost**, which is an estimation of the cost of deployment of the network. It represents in our case the purchase cost of the gateways.

Optimization problem formulation. The configuration decision can be formulated as an optimization problem as follows. Given,

- An IoT application A, with its set R of characteristics and communication requirements.
- A network technology T.
- A set C of possible configurations (parameters value combinations).
- A set K of key performance metrics or KPIs that characterize the behavior of an application A on a network technology T with C.

the decision problem consists in finding the configuration C_d in C of the network technology T that fits the application requirements R and provide the best performances for the application A, in terms of KPIs K.

The goal of the **Selection** step of the methodology proposed is to compare and rank the alternatives evaluated in the **Evaluation** step.

After KPIs values normalization, the results are ranked according to a score, obtained through a method derived from the TOPSIS MADM algorithm [35]. The ranking leverages KPIs weights, on the basis of their knowledge of the business context. The KPIs weighting is done using a vector of preference, more commonly named weights, in the form of $W = [W_1, \ldots, W_n]$ where $W_j \in \mathbb{R}$, $\sum_{j=1}^{n} W_j = 1$.

$$p_i = [p_{i1}, \ldots, p_{in}] \tag{1}$$

The positive ideal solution (best one) and the negative ideal solution (worst one) based on the range of estimated KPIs values are calculated. Then, a score S is given to each alternative depending on the Euclidean distances between the considered alternative and the positive and negative ideal solutions.

The output of the **Selection** step is the alternative that obtains the highest score S, according to this ranking.

4.3. Case study: Network Configuration Decision for a Precision Agriculture Application

In this case study, we consider the deployment of a precision agriculture solution in a given farm, using LoRaWAN. The IoT architect needs to adjust the network configuration parameters of this solution taking into account the specificity of the deployment. The precision agriculture system comprises humidity, temperature and PH sensors, which measure these metrics before sending them to a LoRaWAN gateway for further transmission and processing.

The application scenario is described in Table 3.

Application modeling	Parameters	Case A
End-devices	• Minimal number	200
	• Maximal number	200
	• Battery capacity (Amperes.hour)	2.4
Workload	• Traffic direction	Upstream
	• Message size (bytes)	50
	• Minimal frequency (packets/second)	0.001
	• Maximal frequency (packets/second)	0.001
Environment	• Type	Rural
	• Scope (meters)	8000
	• Expected lifetime (days)	N/A

Table 3: Application modeling of case A.

The IoT architect wants to explore and compare the various network configurations for the LoRaWAN settings within the end-devices. The considered parameters for LoRa network interface settings are:

- **SF:** Determines the speed at which the signal frequency changes across the bandwidth of a channel. The higher the spreading factor the lower the data rate.
- **CR:** An indication of how much of the data stream is actually being used to transmit usable data.
- **CRC:** An error-detecting code commonly used in networks to detect accidental changes in the transmitted data.
- **Type of traffic:** Determines whether the data is sent with or without an acknowledgement. It can therefore be confirmed (1) or unconfirmed (0), respectively.

The goal is to determine which SF to select as well as which CR (Coding Rate) and type of traffic (confirmed or unconfirmed) to use. Several network configurations are generated accordingly. The minimal and the maximal values are considered for each parameter.

Table 4 presents the KPI values of the various LoRaWAN alternatives. We conducted a comprehensive simulation for all the possible configurations, which took approximately 441 minutes (more than 7 hours). We show only some configurations and their corresponding KPIs and scores. Wee see the tremendous influence that parameters like the number of gateways on the KPIs. However, this comes with a higher cost. Based on these results, the algorithm elects LoRaWAN with 5 gateways, SF8, an unconfirmed traffic, 1 CR and a 0 CRC as the optimal alternative.

4.4. Exploiting NDT data to enhance NS credibility

As discussed in Section 3, simulation has always been challenged on its credibility [36]. To face it, parameter calibration and model validation are necessary to make sure that simulation returns accurate results. The connection of NS to the physical deployment within the framework of an NDT can address this limits by feeding the simulator with real data for parameter initialisation and calibration. Thus, we explore the merits of coupling simulation and life data in the context of NDT in the next section and see whether this produces more grounded data and simplifies model creation and validation.

We take here the example of the calibration of the energy consumption simulation model in IoT. Energy efficiency is indeed a critical aspect of IoT network technologies, as IoT devices generally operate on limited battery power or in energy-constrained environments. Therefore, insights related to the energy required for the communications and battery life-time duration estimates, delivered by NS, have to be highly reliable.

Energy consumption, which represents the rate at which energy is consumed over a period of time, can be measured on the overall network or on each IoT end-device. In this work, we define the energy efficiency ratio as the number of bytes that each transmitter can successfully transmit to the receiver using a single joule of energy. The higher this quantity, the more energy efficient the IoT network is. The battery lifetime gives an indication of the IoT system's lifetime without recharging batteries. Note that we focus here only on energy consumption due to the transmission costs. Sensing/actuating energy consumption is not considered in this study.

Energy consumption in discrete-event network simulators is often modeled as follows:

$$E = \sum_{i \in S} (\alpha_i \times t_i) \times V \tag{2}$$

where:

- E: Energy consumption in Joules,
- S: Set of different physical states (Tx, Rx, etc.),
- α_i: Current consumption of state i in Amperes,
- t_i: Total time passed in a state i in Seconds,
- V: Voltage in Volts.

The major problem with Equation 2 is that, in reality, the current consumption α_i of each state i is strongly tied to the type of equipment. Indeed, although several works associate network technologies to current consumption values (*e.g.,* [37], [38]), it is possible to find different equipment featuring the same network technology with different current consumption values (*e.g.,* in [39], [40]).

As the energy consumption is highly tied not only to the used network technology but also to the actual deployment, the simulation models can yield unreliable results. Calibrated energy consumption models would ensure that the simulated results align closely with the actual energy usage of the deployed IoT devices and networks. Thus, this will make the simulation more trustworthy for IoT architects. The real devices, instrumented and connected to a NDT, can deliver measurements of the actual energy consumption influenced by radio conditions, the nodes positions and the application workload. These data can then be injected within the simulator for continuous calibration.

In the following, we formulate the problem of refining simulation models regarding energy consumption. We assume that the NDT is connected to the physical network, with a subset of selected end-devices equiped with energy consumption sensors. These real measurements are used to calibrate the α_i values, as follows:

1. Take period measurements of the power consumption at a regular pace.
2. For fixed periods, calculate the energy consumed. To do so, one can use the integral of the power per time (seconds). The integral can be calculated using Thomas Simpson's method. Generate a dataset D_1.
3. For the same considered period, generate a trace of all the crossed states in the simulator and the corresponding times passed in each state. Generate a dataset D_2.
4. Merge D_1 and D_2, so that we have for each period the amount of time passed in each state in the simulator, and the consumed energy in the physical network.

Configuration					Key Performance Indicators				
nGW	SF	Traffic Type	Coding Rate	CRC	**Message Delivery** *Weight: 1* *Unit: %* *Goal: >90*	**Power Consumption** *Weight: 1* *Unit: mW*	**Message Latency** *Weight: 1* *Unit: ms* *Goal: <1000*	**Cost** *Weight: 1* *Unit: $*	**Score**
1	7	0	1	0	4.45	0.047	107.77	1000	0.774
1	7	1	1	0	6.25	0.032	107.77	1000	0.764
...									
5	7	1	1	0	100.0	0.033	107.77	5000	0.927
5	**8**	**0**	**1**	**0**	**99.0**	**0.082**	**195.07**	**5000**	**0.951**
5	8	1	1	0	100.0	0.057	195.072	5000	0.883
...									
40	12	0	4	1	100.0	0.14	3809.28	40000	0.315
40	12	1	4	1	100.0	0.23	3809.28	40000	0.241

Table 4: Precision Agriculture Case's Results.

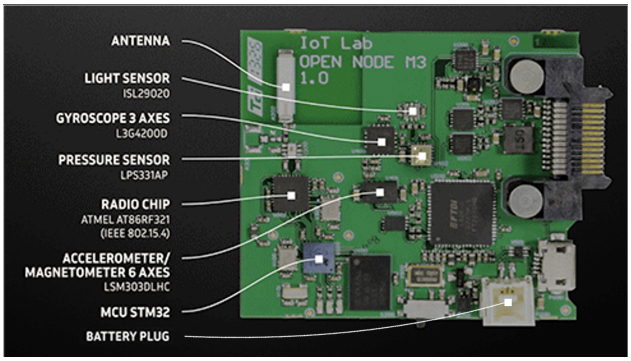

Figure 2: M3-board Micro-Controller Components [41].

5. Apply a linear regression to infer the coefficients by which the times passed in the different states must be multiplied to get the real energy consumed (after multiplying with the voltage that we consider fixed). These coefficients correspond to the calibrated α_i. Since the coefficients must be positive, we use the least square method for the linear regression.

The pseudo-algorithm of the solution is presented in Algorithm 1 in Appendix A.

Illustration: To illustrate the application of our method, we deploy the physical twin network in the FIT IoT-Lab [41] and use ns-3 as the network simulator. The end-devices are based on the M3-board micro-controllers (see Figure 2), with a firmware using the RIOT Operating System [42].

Use-case Description: The IoT devices transmit sensor data, like temperature, pressure, or vibrations via a IEEE 802.15.4 network (6LowPAN).
Some end-devices are instrumented and connected to the NDT platform collecting their energy consumption. These measurement are used to calibrate the simulator.

There are 50 sensors sending 100 bytes packets every second to a gateway. The sensors are separated by a distance of 200 meters. This deployment is described in Table 5.

Application modeling	Parameters	Case Study
End-devices	• Minimal number	40
	• Maximal number	60
	• Battery capacity (Amperes.hour)	2.4
Workload	• Traffic direction	Upstream
	• Message size (bytes)	100
	• Minimal frequency (packets/second)	1
	• Maximal frequency (packets/second)	2
Environment	• Type	Suburban
	• Scope (meters)	200
	• Expected lifetime (days)	N/A

Table 5: Application modeling of the case study.

Results: Table 6 shows the drawn current values of the simulation models before and after the calibration. As we can see, the values are clearly different for most of the NIC physical states.

To highlight this difference, Figure 3 displays the power consumption (in milliwatts) measured for one sensor 1) on the real deployment, 2) through the calibrated simulation models and 3) the default simulation models of ns-3. As we can see, the calibrated models manage to reproduce in a very accurate way the power consumption measured on the real deployment. It is also interesting to see that the power consumption estimated through default models is tremendously far from reality. As we said before, it is mainly due to the fact that default simulation models consider only the energy induced by transmission modules, and make abstraction of the energy consumption induced by the microcontroller processing unit and the firmware. For instance, it is worth noting that the used M3 board microcon-

State	Default Drawn Current value (mA)	Calibrated Drawn Current value (mA)
Tx	7	83
Rx	0.5	46
Tx-Busy	7	14
Rx-Busy	1.5	49
Trx-Switch	0.5	0.01
Trx-off	5×10^{-7}	5×10^{-7}

Table 6: Default and Calibrated Drawn current values for each state of the machine state used in ns-3 simulations to evaluate the power consumption of 802.15.4 communications.

troller consumed 14 mA at full power, to which we must add the radio chip which consumes 14 mA when transmitting and 12 mA when receiving, and other energy-consuming hardware as well [1].

In order to see whether the calibrated models can be used as a baseline for the prediction of future evolution of the network deployment, we test the same calibrated models for a different use-case as the one used for the calibration. This deployment consists in 60 end-devices (instead of 40), sending 2 packets (instead of 1) every second. The energy consumption is estimated without running again the calibration. As we can see in Figure 3, it is still close to the reality. The power consumption remains practically the same, despite the increased density and traffic workload.

By way of comparison, an experiment was undertaken involving the testing of an alternative protocol and operating system (CTP under Contiki OS [43]) featuring mesh traffic propagation encompassing 64 M3 nodes. The results revealed a higher consumption rate in comparison to the initial RIOT-based experiment.

As a side note, we would like to mention that the packet delivery observed in the simulation was also far from the real one. This suggests that, despite our efforts to replicate the deployment accurately in terms of the number and positions of the end-devices, the simulated radio environment differed significantly from the actual one. One possible explanation for this discrepancy is the presence of other active nodes in the FIT IoT-Lab platform that were in proximity to our deployed nodes. As a result, the interference caused by these additional nodes was not accounted for in the simulation, leading to differences in the packet delivery performance between the simulated and real environments. This confirms the benefit of coupling the simulator with the physical twin via its NDT to refine it and increase the accuracy of the simulation models.

Particularly, radio link quality estimation in simulation is a crucial, due to the substantial impact it has on critical performance metrics like packet latency and delivery in IoT networks. Hence, the accuracy of the simulation models heavily depends on how well they capture the real-world behavior of wireless channels. Calibration using real data from experiments would provide a means to validate and calibrate simulation models, ensuring that the simulated scenarios closely mirror actual conditions. However, to accurately reproduce a real radio environment in simulation is still an open challenge, although some recent works have emerged around this topic (*e.g.,* [44]).

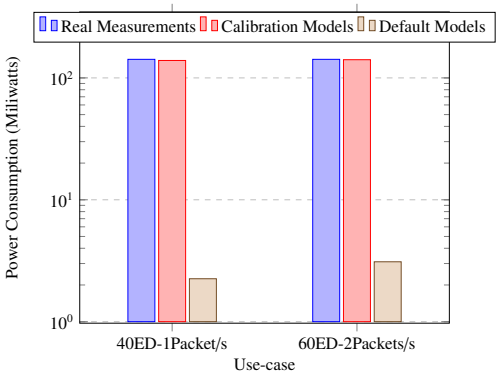

Figure 3: Energy Simulation Models Calibration Results.

4.5. *Integration of NS and Decision Support in NDT*

To give an overview of the user-friendliness potential of an NDT (challenge 2 of NS-NDT), we briefly present below the integration of NS in our Stackeo's NDT platform. Figure 4 displays the user interface presenting the conceptual network design of an IoT solution and Figure 5 gives the NDT dashboard view offered to the IoT operator.

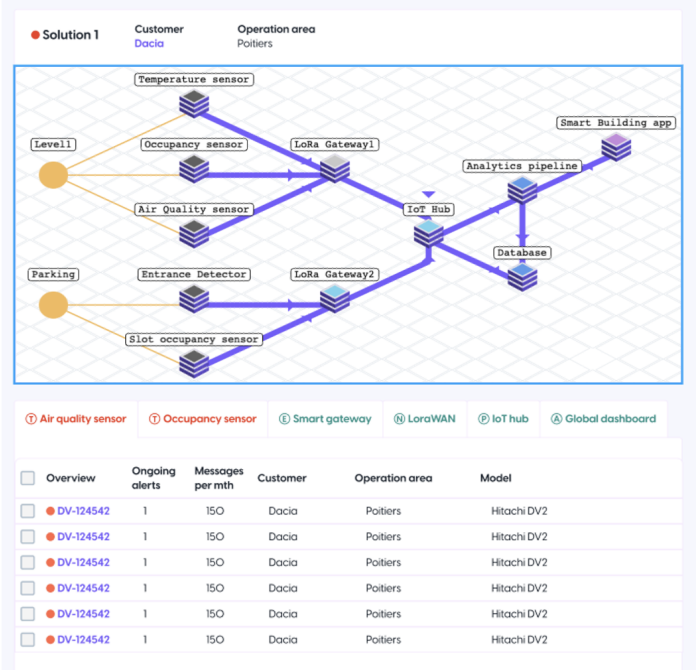

Figure 4: NDT UI for IoT network model design and exploitation in Operation

To perform the evaluation of the various configuration settings and making the decision, a network simulator based on ns-3 as well as an optimizer based on TOPSIS scoring algorithm has been integrated in the Stackeo's NDT platform and

[1] https://www.iot-lab.info/docs/boards/iot-lab-m3/

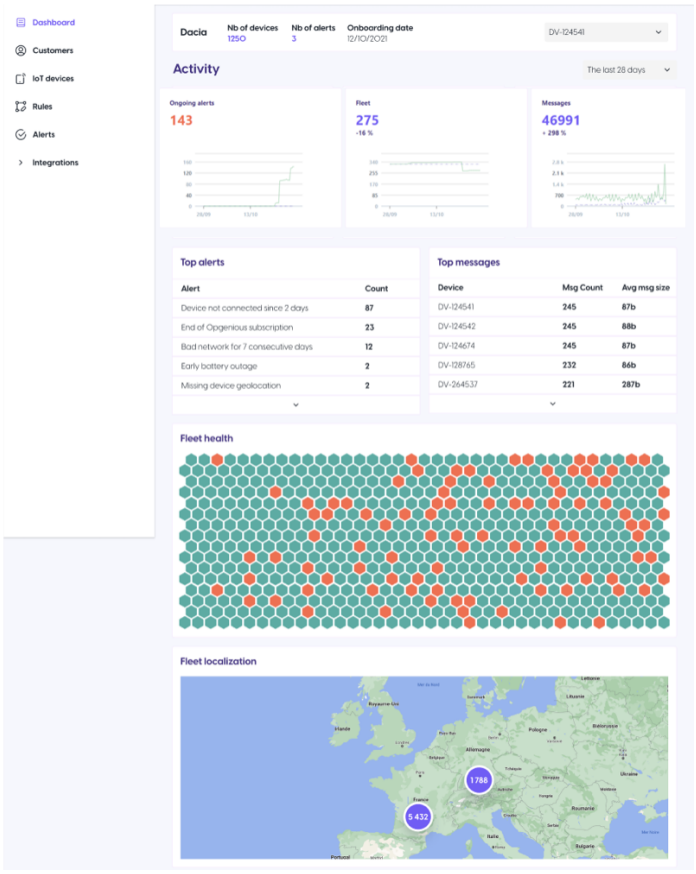

Figure 5: Overview of IoT NDT dashboard that highlights the status of the deployed IoT devices fleet and network activity

presented in [45]. During the design phase of an IoT solution, a simulation model can created and evaluated with the integrated NS tool following the HINT methodology presented above. The simulation model continue to evolve with the physical network deployment taking into account its real parameter values and is accessible to the IoT team for performance optimization decision support in operations.

In the previous section we have focused on the combination and integration of NS and MADM tools within a NDT for performance evaluation, but other analysis and indicators, related to security, financial environmental impact for example, can also be considered and evaluated. This can be done by adapting the model simulator and adding new parameters and KPIs to the decision support algorithm.

Regarding the environmental impact estimation, one way of evaluating it involves analyzing the environmental impacts at each stage of the end-devices' lifecycle. This naturally induces to clearly specify the type of end-devices composing the physical network, which is an information that is, by design, stored within an NDT as depicted in Figure 5. The related data for environmental impact of raw material extraction, manufacturing, use phase, and end-of-life disposal should then be collected in appropriate Life Cycle Analysis (LCA) database [46]. The evaluation of the footprint during the operation phase requires also the computation of the traffic volumetry which can

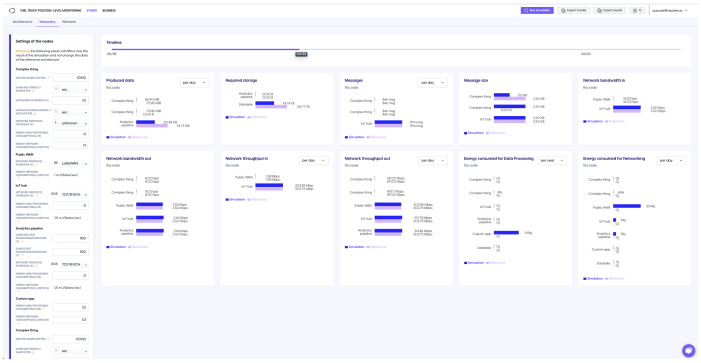

Figure 6: NDT UI for IoT network volumetry simulation results presentation

be done by simulation as illustrated in Figure 6.

This section explored the benefit of integrating NS within NDT to enhance NDT capabilities on one side and NS credibility and accessibility on the other side. In the next section, we address the problem of the NS scalability and cost issue in the NDT context where timely decision are needed. We propose to explore surrogate modelling [8] as an approach to increase the efficiency of the simulation process. In particular, we study a Machine Learning-based surrogate modeling methods (ML-SM), exploiting the previously executed simulations to predict the outcome of future ones preventing the NDT to perform unnecessary new simulation. We analyze if this helps deal with the execution time and cost problem that NS can bring to the NDT.

5. Surrogate modeling to lower NS costs in NDT

As detailed in the previous section and formulated in Subsection 4.2, NS can be used for testing various configuration settings and associated to a MADM method for selecting the best alternative. However, this benefit comes with the trade-off of increased time required for decision-making processes. Indeed, the number of possible configurations to test for a network technology can be huge. For instance, the Spreading Factor (SF) for LoRa, which has a great impact on the performance, can take 6 integer values between 7 and 12. Add to that the other configuration parameters of LoRa and their possible values, and the set of combinations becomes very quickly unmanageable. Yet, determining the appropriate values to use is a key issue for IoT architects when deploying their solution, especially since several KPIs have to be considered simultaneously (energy, throughput, latency, etc.). Even though simulation offers a better scalability compared to real experimentations, testing all the configuration combinations could become costly in terms of time and computing energy consumption. If we consider that there are n different parameters, where each one can take m different discrete values, a comprehensive evaluation would lead to $S = m^n$ simulations. Add to that the time of each simulation, this can quickly become overwhelming. The high number of the required simulations can make NS too computationally- and cost-prohibitive in the framework of

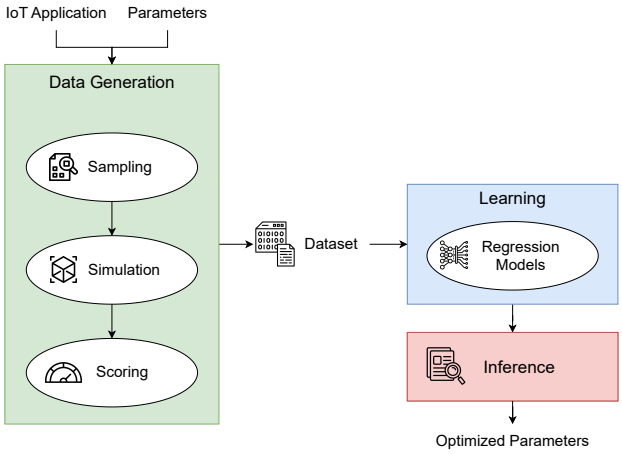

Figure 7: Step-by-step description of *COSIMIA*.

an NDT. There is then a need to lower the number of simulations to reduce simulation costs.

5.1. Principles

To mitigate the computational burden introduced by NS, surrogates models (SM) are often constructed using simulation outputs to approximate the response surface of the simulation model. Surrogate models are mathematically simple models (of coarse models) that map or regress the input–output relationships of a more complex, computationally intensive model (or fine model, here the simulation model).

Thus, the idea of our solution, named *COSIMIA*, is to leverage a surrogate modeling approach to reduce the exploration space (the whole set of possible combinations and simulations). The principle consists in selecting a subset of parameters combinations (ie. configurations), simulating the corresponding scenarios, calculating a score for each one, which reflects the goodness of this configuration, and building a regression model based on the scoring of each combination. Finally, to find the best configuration for a given context, an inference on the trained regression model is made instead of running exhaustive costly simulations. The successive steps are depicted in Figure 7, and we detail them in what follows:

5.1.1. Data generation

This step consists in selecting a subset of input parameters values to run a low number of simulations and calculate the resulting KPIs for each combination.

The configuration sampling is done as follows: For each parameter, we consider the minimal, middle and maximal value (*e.g.,* the values 7, 10 and 12 for SF). In practice, for unbound parameters, this may require to set a lower or upper bound. Then, these different configurations are evaluated using simulation, using ns-3 in our case. The resulting KPIs are then gathered from the simulation. Finally, a score is assigned to each parameters combination, representing the relevance of the configuration, using the TOPSIS MADM method as presented in Section 4. This way, and compared to an exhaustive search,

Input

Output

nGW	max_be	min_be	csma_backoff	frame_retries	success_rate	energy	latency	price	score
3	6	5	1	2	93.125	0.0330944	10.8	300	0.8737784824715121
3	6	6	1	2	90.0	0.0331656	15.76	300	0.8707352249067203
3	6	7	1	2	92.5	0.0336712	27.09	300	0.8700584575665402
3	7	0	1	2	93.75	0.0316469	3.91	300	0.8822267754254658
3	7	1	1	2	91.875	0.0319488	4.2	300	0.8791054386793002
3	7	2	1	2	79.0	0.0315099	4.44	300	0.6913738166711929
3	7	3	1	2	91.25	0.0316881	5.31	300	0.8799445643367099
3	7	4	1	2	85.2	0.0316415	7.14	300	0.6912018211361517
3	7	5	1	2	93.125	0.0330944	10.8	300	0.8737784824715121

Figure 8: Regression Models Input and Output.

considering n parameters with m possible values for each one of them, our approach requires at most 3^n simulations.

At the end of this step, we obtain a dataset composed by different configurations with an aossociated score based on its resulting KPIs.

5.1.2. Learning

The second step consists in building the surrogate model that gives the score of a given configuration, without using simulation. Here, we leverage Machine Learning, ML and more precisely Regression, which is a branch of ML, where algorithms are used to predict continuous outcomes from given input features. Several regression models have been proposed in the literature : Linear Regression [47], Gradient Boosting [48], Random Forests [49], Extra Trees [50], K-Nearest Neighbors [51] and Support Vector Machine [52].

The learning step of *COSIMIA* is achieved by feeding the generated dataset to a regression model, where the input variables are the configurations, and where the output is the score. Thus, we train the model to predict the score of a given configuration. Rather than selecting arbitrarily one regression model, we train the collection of regression models cited above and extensively used by the community. Each regression model can give different outputs (scores) for the same inputs and identify different configuration as the best choice (highest score) for a given context. We observe this in Figure 8 illustrating an example of a generated dataset, the inputs and the output of the different regression models used as training models.

5.1.3. Inference

Once the various surrogate models have been trained, a comprehensive inference is conducted for all the possible parameter combinations. For each one, instead of running the simulation, we use each trained models to calculate a score. Note that this comprehensive inference is made possible because the prediction of the score is fast, compared to the actual simulation.

5.1.4. Decision and validation

The decision step consists in comparing the different scores obtained via the different trained models. The configuration which returns the hightest score amongst all predictions is the one retained. Then simulation can be run to obtain the corresponding KPIs and verify the validity of the solution. As several surrogate models are employed and each gives a different output, users can run several simulations based on the associated

configurations rather than only one and then make their decision (for the regression model to use) on the simulated KPIs rather than on a predicted score. This lower the prediction error and gives more choice to the user.

An algorithm of COSIMIA for the configuration optimization is given in Algorithm 2 proposed in Appendix B.

5.2. Examples of Application

5.2.1. Case Study A: Smart Building with 6LoWPAN

In this section, we show the application of *COSIMIA* for the configuration optimization of the use case (see Table 5) presented in previous section. The considered configuration parameters for 6LoWPAN (802.15.4) are:

- **Number of frame retries (FR):** It is the maximum number of the retransmissions when there is no acknowledgement received before dropping the packet.
- **CSMA backoff (BE):** The number of times that the sensor stays in the backoff stage after unsuccessful channel sensing.
- **Maximal backoff exponent (MaxBE):** Maximal random interval before sensing the channel.
- **Minimal backoff exponent (MinBE):** Minimal random interval before sensing the channel.

Table 7 shows the improvement in execution time. Indeed, while the comprehensive simulation needs 1367 minutes to complete, only 26 minutes are needed for *COSIMIA* to be executed, which is equivalent to an improvement of a factor of 60. This is due to the fact that the configuration parameters specific to 6LoWPAN have on average high cardinalities. The proximity of the regression models is defined by the ratio of the score of the solution returned by each trained model to the optimal solution (returned by the comprehensive simulation). This proximity embodies the prediction error. We find that most of the ML techniques reach 99% of proximity. However, the linear regression struggles to exceed 85% of proximity, which corroborates the fact that the problem is not linear and the configuration parameters are interdependent.

Overall, *COSIMIA* has been able to return optimized configurations which are close from the optimal one returned by a comprehensive set of simulations. Moreover, this has been possible through a clear reduction of the simulation time, with a factor of 60 for this use case.

Impact of the sampling granularity. We investigate here the impact of the configurations sampling granularity on the performance of the method. A granularity of n means that we consider n values during the sampling phase. The average proximity is the mean of the proximities over all the tested regression models. We address here the tradeoff between prediction error and method complexity.

As expected, Figure 9 shows that the finer the granularity (in other words, the more points are taken for sampling), the more the number of simulation thus the longer the execution

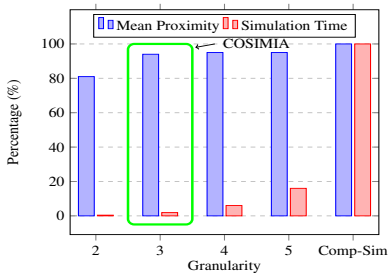

Figure 9: Impact of the sampling granularity of the solution.

time. Note that the "Comp-Sim" bar corresponds to the comprehensive simulation done to all the possible configurations. This is of course due to the fact that the subset of selected variables requires more simulations, which are themselves time-consuming. Regarding the proximity of the returned solution, it is around 80% when choosing only two points for the sampling (minimum and maximum values for each parameter). From granularity 3 (minimum, intermediate and maximum values as we recommend in *COSIMIA*) onward, the average precision immediately reaches 95% and remains stable at this value for the higher values of granularity.

5.2.2. Case Study B: Smart Agriculture using LoRa

We illustrate here the generality of *COSIMIA* and how it can be applied to an other use case and network technology, presented in Section 4.3. Recall that the IoT application pertains to a smart agriculture solution and is defined as follows: 200 sensors send 50 bytes packets every 600 seconds (10 minutes) to the gateways. The sensors are separated by a distance (deployment scope) of 8000 meters and are deployed in a rural environment.

Table 8 shows the results obtained for this case study. The comprehensive simulation gives the optimal solution for 5 gateways, a SF of 8, unconfirmed traffic, a CR of 1 and a CRC of 0. Determining this solution required no less than 441 minutes and 3840 simulations, while *COSIMIA* required 70 minutes and 480 simulations. This is equivalent to a reduction in simulation time by a factor of 6. Here also, the comprehensive simulation is conducted to validate the regression model outcome.

5.3. Synthesis

In this section, we have proposed a method, *COSIMIA*, combining simulation and ML-based surrogate modeling to reduce the number of simulations, accelerate and lower the cost of the configuration decision. The results indicate that *COSIMIA* obtains promising results on a couple of different examples, featuring different IoT applications with different network technologies. However, the approach has two major limitations, especially in the context of NS-NDT. First, the selection of configurations subset is arbitrary and makes the method an heuristic, which may not always guarantee optimal solutions despite its performance on the presented use-cases. The sampling approach used in the data generation step, employing min-mid-max values, assumes monotonic parameter influence on the TOPSIS score. This might lead to overlooking crucial values that

Trained Model	Best solution (configuration)	Corresponding simulated KPIs				Data generation		Score	Proximity
		Message Delivery *Weight: 1 Unit: %*	Energy consumption *Weight: 1 Unit: mW*	Message Latency *Weight: 1 Unit: ms*	Cost *Weight: 1 Unit: $*	Time *Unit: min*	Number of simulations		
Comprehensive simulation	[3,4,3,4,0]	92	0.03	5.56	300	1367	23040	0.8833	1
Gradient boosting	[3,5,3,5,3]	99.37	0.032	5.58	300	26	405	0.8818	0.99
Extra trees	[3,6,0,0,4]	93.75	0.031	3.91	300			0.8827	0.99
Random forest	[3,8,7,5,3]	100	0.033	31.16	300			0.8724	0.98
KNN	[3,8,7,5,6]	100	0.033	31.16	300			0.8724	0.98
SVR	[5,7,7,5,6]	100	0.03	30.15	500			0.8317	0.94
Linear regression	[10,8,7,5,7]	100	0.02	26.02	1000			0.70	0.79

Table 7: Results for Case Study A. The format of the solutions is the following: [NGW,MaxBE,MinBE,CB,FR].

can potentially lead to the optimal solution. Secondly, the data must be fully generated prior to training the regression models and utilizing them for configuration decisions. Thus, an important number of simulations is still required for each deployment. This could be deemed inefficient, particularly within the context of NDT-managed operations where the physical twin may experience frequent modifications. In such scenarios, an optimized configuration is expected to be swiftly applied upon request.

To overcome the limitations of this first approach we name *Offline COSIMIA*, we explore, in the following section, how surrogate modeling based on Bayesian optimization (*Online COSIMIA*) can be leveraged in NS-NDT for dynamic configuration.

6. Towards Online Optimization: Bayesian Framework

6.1. Principles

The resolution time of the optimization task, even reduced with the surrogate modelling based on regression approaches presented above in *Offline COSIMIA*, is still prohibitive specially when dynamic reconfiguration is required. Moreover, the arbitrary selection of explored configurations proposed prevent, to give any guarantee on the quality of the result. We explore here how Gaussian Process modelling (GP), which is one of the most commonly used surrogate models in Bayesian optimization, can help reduce the search time and improve result quality. We name this solution *Online COSIMIA*. Bayesian optimization, as a form of statistical learning, gradually learns from acquired data and maintains a probability distribution over the solution space. This facilitates both systematic exploration of diverse regions and exploitation of potential high-yield areas.

Given an objective function f, at each time step t, a Bayesian optimization algorithm exploits a surrogate model trained on data collected up to time t, that is $\mathcal{D}_t = (\boldsymbol{X}_t, \boldsymbol{y}_t)$ - the previous simulation results in our context - to predict the next configuration $\boldsymbol{x}_{t+1}$ to simulate. In most Bayesian optimization algorithms, a Gaussian Process (GP) is chosen as the surrogate model for f.

In this section, we describe how Bayesian optimization can be performed with a GP (Section 6.1.1). Then, we describe how Bayesian optimization can solve the optimization of the KPIs K under the TOPSIS scalarization (Section 6.1.2).

6.1.1. Gaussian Process as surrogate model for Bayesian Optimization

First, let us introduce formally the concept of GP. A GP is a stochastic process, that is, a collection of random variables $\{Y(\boldsymbol{x})\}_{\boldsymbol{x}\in C}$ indexed by a set C. Any stochastic process is a GP if and only if any finite collection $\{Y(\boldsymbol{x}_1), \cdots, Y(\boldsymbol{x}_k)\}$ has a joint multivariate Gaussian distribution. As such, a GP is fully specified by its mean function $\mu(\boldsymbol{x}) = \mathbb{E}[Y(\boldsymbol{x})]$ and its covariance function $k(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}[(Y(\boldsymbol{x}) - \mu(\boldsymbol{x}))(Y(\boldsymbol{x}') - \mu(\boldsymbol{x}'))]$. It is denoted $\mathcal{GP}(\mu(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'))$.

The idea of using a GP as a SM has been introduced by the pioneering work [53]. Naturally, it proposes to assume that f is a GP, with $\mu(\boldsymbol{x}) = 0, \forall \boldsymbol{x} \in C$ without loss of generality. As such, it is fully specified by its covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$. At time t, given the previously observed data $\mathcal{D}_t = (\boldsymbol{X}_t, \boldsymbol{y}_t)$, [53] shows that $f(\boldsymbol{x})|\mathcal{D}_t \sim \mathcal{N}(\mu_t(\boldsymbol{x}), \sigma_t^2(\boldsymbol{x}))$ with

$$\mu(\boldsymbol{x}) = \boldsymbol{k}(\boldsymbol{x}, \boldsymbol{X}_t)^\top \boldsymbol{K}_t^{-1} \boldsymbol{y}_t, \tag{3}$$

$$\sigma_t^2(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{k}(\boldsymbol{x}, \boldsymbol{X}_t)^\top \boldsymbol{K}_t^{-1} \boldsymbol{k}(\boldsymbol{x}, \boldsymbol{X}_t), \tag{4}$$

with the t-dimensional vector $\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{X}_t) = (k(\boldsymbol{x}, \boldsymbol{x}_i))_{\boldsymbol{x}_i \in \boldsymbol{X}_t}$ and the $t \times t$ Gram matrix $\boldsymbol{K}_t = \left(k(\boldsymbol{x}_i, \boldsymbol{x}_j)\right)_{\boldsymbol{x}_i, \boldsymbol{x}_j \in \boldsymbol{X}_t}$.

To perform the online optimization of the objective function f, a Bayesian optimization algorithm needs to decide which configuration $\boldsymbol{x}_{t+1}$ brings the best trade-off between the exploration of unknown regions of C and the exploitation of the previously explored data $\mathcal{D}_t$. Typically, this exploration-exploitation dilemma is addressed by using an acquisition function, $\varphi_t : \mathbb{R}^d \to \mathbb{R}$. φ_t exploits the information provided by the SM to quantify the benefits of querying a configuration $\boldsymbol{x}$ in terms of exploration and exploitation. It is used to determine the next configuration to query, by defining $\boldsymbol{x}_{t+1} = \arg\max_{\boldsymbol{x}\in C} \varphi_t(\boldsymbol{x})$. Several acquisition functions exist, such as Probability of Improvement [54], Knowledge Gradient [55], Expected Improvement [56] and GP-UCB [57].

Note that, under some regularity assumptions about f, a Bayesian optimization using a GP as a surrogate model and an acquisition function such as GP-UCB or Expected Improvement is guaranteed to find the optimal configuration (that is, $\arg\max_{\boldsymbol{x}\in C} f(\boldsymbol{x})$) asymptotically.

Model	Solution	KPIs				Data generation		Score	Proximity
		Message Delivery *Weight: 1* *Unit: %*	Energy consumption *Weight: 1* *Unit: mW*	Message Latency *Weight: 1* *Unit: ms*	Cost *Weight: 1* *Unit: $*	Time *Unit: min*	Number of simulations		
Comprehensive simulation	[5,8,0,1,0]	99	0.082	195	5000	441	3840	0.9518	1
Gradient boosting	[5,7,0,1,1]	89.5	0.093	112	5000	70	480	0.9468	0.99
Extra trees	[5,7,0,1,0]	89.5	0.048	107	5000			0.9469	0.99
Random forest	[5,7,0,1,1]	89.5	0.05	112	5000			0.9468	0.99
KNN	[5,8,0,2,1]	99	0.082	195	5000			0.9483	0.997
SVR	[10,7,0,1,1]	100	0.048	107	10000			0.9026	0.94
Linear regression	[1,7,0,1,0]	4.45	0.047	107	1000			0.7741	0.81

Table 8: Results for Case Study B. The format of the solutions is the following: [NGW,SF,Traffic-Type,CR,CRC].

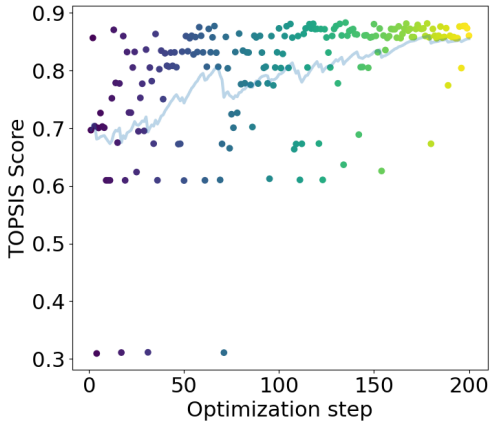

Figure 10: Performance of the online version of COSIMIA under 200 iterations for Case Study A. The blue curve shows the exponential moving average of the series. At the end of the simulation, the method recommends the configuration [3, 5, 4, 6, 4], obtains the KPIs [100, 0.032, 7.64, 300] and a TOPSIS score of 0.8833.

6.1.2. TOPSIS Scalarization

To address the optimization problem described in Section 4.2 in an online fashion, we propose to assume that $f(\boldsymbol{x}) = S(N_s(\boldsymbol{x}))$ is $\mathcal{GP}(0, k(\boldsymbol{x}, \boldsymbol{x}'))$. To find the next configuration to query, we use the GP-UCB acquisition function [57], defined by

$$\varphi_t(\boldsymbol{x}) = \mu_t(\boldsymbol{x}) + \beta_t^{1/2}\sigma_t(\boldsymbol{x}), \tag{5}$$

with $\beta_t \in O(\log t)$, with a closed-form provided in [57].

The proposed online optimization algorithm is described in Algorithm 3 in Appendix C, and Figure 10 reports the evolution of the optimization process for Case Study A. Observe that the online version of COSIMIA is able to progressively direct its search towards the most promising regions of the configuration space, without having to explore it thoroughly. Consequently, under 200 iterations only (or 22 minutes of time budget), the online version of COSIMIA converges towards a configuration that achieves an excellent TOPSIS score, equivalent to the one obtained by the configuration recommended by the exhaustive search.

6.2. Benefits of online learning with NS-NDT context

In this section, we introduced an online method for finding the best configuration in a very large configuration space that differs from the *Offline COSIMIA* proposed in Section 5. To conclude this section, let us discuss the pros and cons of both approaches, which are also summarized in Table 9.

Offline COSIMIA. The offline version of COSIMIA stands out for letting the user choose its regression model, offering her more flexibility. Although its data collection technique (described in Section 5.1.1) was designed as an empirical trade-off between the size of the collected data and its ability to represent the objective function f, this makes the offline version of COSIMIA very sensitive to the curse of dimensionality. In fact, to find the optimal configuration of d parameters, *Offline COSIMIA* requires $3^d \in O(2^d)$ simulations. Moreover, there may exist scenarios for which the data collection technique may not represent the objective function accurately. In such cases, *Offline COSIMIA* is very likely to recommend suboptimal configurations. That is the main reason why no theoretical guarantees can be provided, regardless of the chosen regression model. All in all, this makes *Offline COSIMIA* efficient at optimizing simple, low-dimensional objective functions.

Online COSIMIA. On the other hand, *Online COSIMIA* performs the data collection and the optimization of f at the same time. This allows the online version to exploit collected information to build a dataset specifically designed for the maximization of f. This ability makes *Online COSIMIA* asymptotically optimal. That is, given enough time budget, the optimal configuration will always be found and recommended, regardless of the scenario. Moreover, *Online COSIMIA* is less sensitive to the curse of dimensionality, as high dimensional Bayesian optimization techniques are able to optimize objective functions with 100+ dimensions very effectively [58, 59, 60]. However, these properties imply that *Online COSIMIA* must pay the computational overhead of learning from an increasingly large dataset, at each iteration. This additional computational toll makes *Online COSIMIA* unable to perform as much simulations as *Offline COSIMIA* in a given time budget.

Approach	Typical Usage	Code complexity	Simulation cost	Optimization Quality	User Flexibility
NS + TOPSIS	Design decision	Low	Prohibitive	Optimal	Low
NS + Offline COSIMIA	(Re)configuration decision	Medium	High	No guarantees	Medium
NS + Online COSIMIA	Online adaptation	Medium	Medium	Asymptotically optimal	Low

Table 9: Comparing Surrogate Modeling and optimization approaches.

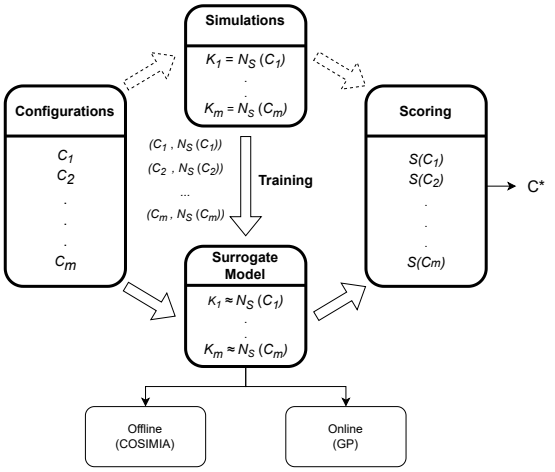

Figure 11: Surrogate Modeling versus Classical simulation to select a configuration

Figure 11 illustrate the principle of surrogate modeling to accelerate decision making.

7. Conclusion

In this article, we have presented the differences and relationships between simulation and digital twin, as discussed by the scientific community in general as well as in networking area. We have observed, that, contrary to other research domains, the network researchers tend to oppose the simulation to the digital twin paradigm. Taking into account the specific advantages but also the respective limits of NS and NDT, we have demonstrated that NS can be beneficially integrated within the NDT to strength decisions throughout the different phases of the lifecycle of a network. We have developed the principles of the NS-NDT solution for the specific IoT network use case and shown its feasability of this integration in the context of a commercial NDT platform. We then explored how NS can be enhanced in an NDT context thanks to field data collected via the link between the physical and digital twins to calibrate parameters and improve NS credibility. The criticality of the insights, such as battery lifetime estimation, these calibrations help to strength, corroborate the benefit of coupling the NS integration with NDT real-life data.

In our experiments, we have observed that measuring network parameters, such as energy consumption of the physical twin, can be complex and highly dependant on the specificity of the deployment and configuration (hardware, software). As future work, we plan to analyse in depth the sensitivity of the network parameter measurement in terms of location and frequency and propose a general method. Limiting the intrusiveness of this sensing on the physical twin as well as on the NDT data management plane is on our agenda. We also want to investigate the benefits of using NS-NDT for simulation propagation model improvement, mobility model refinement, and for IoT networks environmental impact prediction in their design, exploitation but also end-of-life phases.

References

[1] M. Shafto, M. Conroy, R. Doyle, E. Glaessgen, C. Kemp, J. LeMoigne, and L. Wang, "Nasa technology roadmap: Modeling, simulation," *Information Technology & Processing Roadmap Technology Area*, 2012.

[2] E. Glaessgen and D. Stargel, "The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles." Honolulu, Hawaii: American Institute of Aeronautics and Astronautics, Apr. 2012.

[3] Almasan, Paul et al., "Network digital twin: Context, enabling technologies, and opportunities," *IEEE Communications Magazine*, 2022.

[4] P. Barnes, "Challenges in Simulating Communication Systems: State of the art and open challenges in simulating network and communications systems." in *Proceedings of the 2022 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, 2022.

[5] C. Vallati, V. Omwando, and P. Mohapatra, "Experimental Work Versus Simulation in the Study of Mobile Ad Hoc Networks," in *Mobile Ad Hoc Networking*, 2013.

[6] Y. Chen, P. Yu, Z. Zheng, J. Shen, and M. Guo, "Modeling feature interactions for context-aware QoS prediction of IoT services," *Future Generation Computer Systems*, vol. 137, pp. 173–185, Dec. 2022. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167739X2200245X

[7] S. Si-Mohammed, T. Begin, I. Guerin Lassous, and P. Vicat-Blanc, "Adiperf: A framework for application-driven iot network performance evaluation," in *International Conference on Computer Communications and Networks*. IEEE, 2022.

[8] L. J. Hong and X. Zhang, "Surrogate-Based Simulation Optimization," in *Tutorials in Operations Research: Emerging Optimization Methods and Modeling Techniques with Applications*, 2021.

[9] M. Shafto, M. Conroy, R. Doyle, E. Glaessgen, C. Kemp, J. LeMoigne, and W. L, "NASA technology roadmap: modeling, simulation, information technology & processing," Tech. Rep., Apr. 2010.

[10] M. Perno, L. Hvam, and A. Haug, "Implementation of digital twins in the process industry: A systematic literature review of enablers and barriers," *Computers in Industry*, 2022.

[11] Taylor, Simon et al., "Using Simulation and Digital Twins to Innovate: Are we Getting Smarter?" in *Winter Simulation Conference*, 2021.

[12] G. Shao, S. Jain, C. Laroque, L. H. Lee, P. Lendermann, and O. Rose, "Digital twin for smart manufacturing: the simulation aspect," in *Winter Simulation Conference*. IEEE, 2019.

[13] S. Boschert and R. Rosen, "Digital Twin—The Simulation Aspect," in *Mechatronic Futures*, 2016.

[14] D. Hartmann and H. Van der Auweraer, "Digital Twins," in *Progress in Industrial Mathematics: Success Stories*, Cham, 2021.

[15] Sharma, Angira et al., "Digital Twins: State of the art theory and practice, challenges, and open research questions," *Journal of Industrial Information Integration*, 2022.

[16] S. Friedenthal, A. Moore, and R. Steiner, *A practical guide to SysML: the systems modeling language*. Morgan Kaufmann, 2014.

[17] Heidelberger and Lavenberg, "Computer performance evaluation methodology," *IEEE Transactions on Computers*, 1984.

[18] "Hype cycle for enterprise networking, 2023," 2023. [Online]. Available: https://www.forwardnetworks.com/what-is-a-digital-twin/

[19] "Who needs a network digital twin and why," 2022. [Online]. Available: https://www.networkcomputing.com/networking/who-needs-network-digital-twin-and-why#

[20] C. Zhou, H. Yang, X. Duan, D. Lopez, A. Pastor, Q. Wu, M. Boucadair, and C. Jacquenet, "Digital twin network: Concepts and reference architecture," *Internet Engineering Task Force*, 2021.

[21] ITU-T, "Digital twin network – Requirements and architecture," 2022. [Online]. Available: https://handle.itu.int/11.1002/1000/14852

[22] M. Kherbache, M. Maimour, and E. Rondeau, "Network Digital Twin for the Industrial Internet of Things," in *23rd IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, 2022.

[23] K. Hu, "Bridging networks and business intent to activate intelligence," 2018. [Online]. Available: https://www.huawei.com/en/huaweitech/publication/86/bridging-networks-business-intent

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

[25] Rebecchi, Filippo et al., "A digital twin for the 5g era: The spider cyber range," in *IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks*, 2022.

[26] M. Kherbache, M. Maimour, and E. Rondeau, "When Digital Twin Meets Network Softwarization in the Industrial IoT: Real-Time Requirements Case Study," *Sensors*, 2021.

[27] Almeida, Eduardo et al., "Machine learning based propagation loss module for enabling digital twins of wireless networks in ns-3," in *Proceedings of the 2022 Workshop on ns-3*, 2022.

[28] Sen, Argha et al., "An ns3-based energy module of 5g nr user equipments for millimeter wave networks," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2021.

[29] P. Öhlén, C. Johnston, H. Olofsson, S. Terrill, and F. Chernogorov, "Network digital twins–outlook and opportunities," *Ericsson Technology Review*, 2022.

[30] L. F. Rahman, T. Ozcelebi, and J. Lukkien, "Understanding IoT Systems: A Life Cycle Approach," *Procedia Computer Science*, 2018.

[31] K. Gremban and al., "Best practices for device configuration within an iot solution," 2022. [Online]. Available: https://learn.microsoft.com/en-us/azure/iot-hub/iot-hub-configuration-best-practices

[32] I. S. Udoh and G. Kotonya, "Developing iot applications: challenges and frameworks," *IET Cyber-Physical Systems: Theory & Applications*, 2018.

[33] S. Si-Mohammed, T. Begin, I. G. Lassous, and P. Vicat-Blanc, "Hints: A methodology for iot network technology and configuration decision," *Internet of Things*, vol. 22, p. 100678, 2023.

[34] M. Stoffers and G. Riley, "Comparing the ns-3 propagation models," in *2012 IEEE 20th international symposium on modeling, analysis and simulation of computer and telecommunication systems*.

[35] G.-H. Tzeng and J.-J. Huang, *Multiple attribute decision making: methods and applications*. CRC press, 2011.

[36] K. Pawlikowski, H.-D. Jeong, and J.-S. Lee, "On credibility of simulation studies of telecommunication networks," *IEEE Communications magazine*, vol. 40, no. 1, pp. 132–139, 2002.

[37] R. S. Sinha, Y. Wei, and S.-H. Hwang, "A survey on lpwa technology: Lora and nb-iot," *Ict Express*, vol. 3, no. 1, pp. 14–21, 2017.

[38] U. Noreen, A. Bounceur, and L. Clavier, "A study of lora low power and wide area network technology," in *2017 International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*.

[39] C. Gomez, J. C. Veras, R. Vidal, L. Casals, and J. Paradells, "A sigfox energy consumption model," *Sensors*, 2019.

[40] L. Casals, B. Mir, R. Vidal, and C. Gomez, "Modeling the energy performance of lorawan," *Sensors*, 2017.

[41] Adjih, Cedric et al., "Fit iot-lab: A large scale open experimental iot testbed," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. IEEE, 2015.

[42] E. Baccelli, O. Hahm, M. Günes, M. Wählisch, and T. C. Schmidt, "Riot os: Towards an os for the internet of things," in *2013 IEEE conference on computer communications workshops (INFOCOM WKSHPS)*.

[43] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki-a lightweight and flexible operating system for tiny networked sensors," in *IEEE international conference on local computer networks*. IEEE, 2004.

[44] E. N. Almeida, H. Fontes, R. Campos, and M. Ricardo, "Position-based machine learning propagation loss model enabling fast digital twins of wireless networks in ns-3," *arXiv preprint arXiv:2302.11539*, 2023.

[45] S. Si-Mohammed, Z. Fraoui, T. Begin, I. G. Lassous, and P. Vicat-Blanc, "Stacknet: Iot network simulation as a service," in *IEEE International Conference on Communications (ICC 2023)*, 2023.

[46] European Commission, "European platform on life cycle assessment (eplca)," 2020.

[47] G. A. Seber and A. J. Lee, *Linear regression analysis*. John Wiley & Sons, 2003.

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

[49] L. Breiman, "Random forests," *Machine learning*, 2001.

[50] Geurts, Pierre et al., "Extremely randomized trees," *Machine learning*, 2006.

[51] L. E. Peterson, "K-nearest neighbor," *Scholarpedia*, 2009.

[52] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, pp. 199–222, 2004.

[53] C. K. I. Williams and C. E. Rasmussen, "Gaussian processes for regression," in *Conference on Neural Information Processing Systems (NeurIPS'95)*, 1995.

[54] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global optimization*, 1998.

[55] S. S. Gupta and K. J. Miescke, "Bayesian look ahead one-stage sampling allocations for selection of the best population," *Journal of statistical planning and inference*, 1996.

[56] J. Mockus, "Application of bayesian approach to numerical methods of global and stochastic optimization," *Journal of Global Optimization*, 1994.

[57] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger, "Information-theoretic regret bounds for gaussian process optimization in the bandit setting," *IEEE Transactions on Information Theory*, 2012.

[58] D. Eriksson, M. Pearce, J. Gardner, R. D. Turner, and M. Poloczek, "Scalable global optimization via local bayesian optimization," in *Advances in Neural Information Processing Systems*, 2019.

[59] D. Eriksson and M. Jankowiak, "High-dimensional Bayesian optimization with sparse axis-aligned subspaces," in *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, 2021.

[60] A. Bardou, P. Thiran, and T. Begin, "Relaxing the additivity constraints in decentralized no-regret high-dimensional bayesian optimization," *arXiv preprint arXiv:2305.19838*, 2023.

Appendix A. Calibration Algorithm

Algorithm 1 Energy Consumption Calibration Algorithm

Inputs:
$R = [R_1, \ldots, R_k]$; Application requirements;
T: IoT network technology;
$Time$: Deployment time;
$P(t)$: Mesured power consumption at instant t;
$period$: Power consumption measurement period;
δ: Energy consumption calculation period;
N_S: Network Simulator;
P: Power consumption dataset;
D_1: Energy consumption dataset;
D_2: Physical states times dataset;
D: Final dataset;
Algorithm:
/* Initialization */
$D \leftarrow \varnothing$
/* Measurement */
while $t \leq Time$ **do**
  $P.insert(t, P(t))$
  $t \leftarrow t + period$
end while

/* Energy Calculation */
$t \leftarrow 0$
while $t \leq Time$ **do**
  $E \leftarrow \int_t^{t+\delta} M(t)\,dt$ /* Consumed energy between t and $t + \delta$ */
  $D_1.insert(E)$
  $t \leftarrow t + \delta$
end while

/* Physical State Times in Simulation */
$logs \leftarrow S(R, T, N_S).logs$ /* Logs of the simulated deployment (containing physical states and corresponding times) */
$t \leftarrow 0$
while $t \leq Time$ **do**
  $times \leftarrow logs[t, t + \delta]$ /* Physical state times between t and $t + \delta$ */
  $D_2.insert(times)$
  $t \leftarrow t + \delta$
end while

/* Regression */
$D \leftarrow merge(D_1, D_2)$
$LinearReg(times, E)$

return $LinearReg.coefficients$

Appendix B. COSIMIA Algorithm

Algorithm 2 COSIMIA Algorithm

Inputs:
$R = [R_1, \ldots, R_k]$; Application requirements;
T: IoT network technology;
$C = [c_1, \ldots, c_n]$; $c_i \in [a_i, \ldots, b_i]$; $c_i, a_i, b_i \in \mathbb{R}$; Configuration parameters;
$N_{ED} \in \mathbb{N}$; Number of end-devices;
$N_{GW} \in \mathbb{N}$; Number of gateways;
N_S: Network Simulator;
$K = [K_1, \ldots, K_m]$, $K_i \in \mathbb{R}$; KPI values;
S: Scoring function; $S : \mathbb{R}^n \rightarrow \mathbb{R}$;
M: Regression model;
D: Configuration samples dataset;
Algorithm:
/* Initialization */
$D \leftarrow \varnothing$
/* Sampling */
for N_{GW} in $[1, \frac{N_{ED}}{5}, 2]$ **do**
  for c_1 in $\{a_1, \frac{b_1+a_1}{2}, b_1\}$ **do**
    ...
    for c_n in $\{a_n, \frac{b_n+a_n}{2}, b_n\}$ **do**
      $K \leftarrow N_S(R, T, C)$
      $D.insert(C, K)$
    end for
    ...
  end for
end for

/* Scoring */
for (C, K) in D **do**
  $(C, K) \leftarrow (C, K, S(K, D))$
end for

/* Learning */
$M.learn(C, S(K))$

/* Inference */
$best_{config} \leftarrow [c_1, \ldots, c_n]$
$best_{score} \leftarrow 0$
for N_{GW} in $[1, \frac{N_{ED}}{5}]$ **do**
  for c_1 in $[a_1, b_1]$ **do**
    ...
    for c_n in $[a_n, b_n]$ **do**
      if $M.predict(C) > best_{score}$ **then**
        $C^* \leftarrow C$
        $S^* \leftarrow M.predict(C)$
      end if
    end for
    ...
  end for
end for

return C^*

Appendix C. Online COSIMIA Algorithm

Algorithm 3 Online COSIMIA Algorithm

Inputs:
$R = [R_1, \ldots, R_k]$; Application requirements;
T: IoT network technology;
$C = [c_1, \ldots, c_n]$; $c_i \in [a_i, \ldots, b_i]$; c_i, a_i, $b_i \in \mathbb{R}$; Configuration parameters;
$N_{ED} \in \mathbb{N}$; Number of end-devices;
$N_{GW} \in \mathbb{N}$; Number of gateways;
N_S: Network Simulator;
$K = [K_1, \ldots, K_m]$, $K_i \in \mathbb{R}$; KPI values;
S: Scoring function; $S : \mathbb{R}^n \rightarrow \mathbb{R}$;
G: Gaussian process;
Algorithm:
/* Initialization */
$\boldsymbol{X}_0 \leftarrow \varnothing$
$\boldsymbol{y}_0 \leftarrow \varnothing$
$t \leftarrow 0$
/* Optimization loop */
while some stopping criterion is not met **do**
  $\boldsymbol{x}_{t+1} \leftarrow \arg\max_{\boldsymbol{x} \in C} \varphi_t(\boldsymbol{x})$
  $K \leftarrow N_s(R, T, \boldsymbol{x}_{t+1})$
  $y \leftarrow S(K)$
  $\boldsymbol{X}_{t+1} \leftarrow \boldsymbol{X}_t \cup \{\boldsymbol{x}_{t+1}\}$
  $\boldsymbol{y}_{t+1} \leftarrow (\boldsymbol{y}_t, y)$
  $G.learn(\boldsymbol{X}_{t+1}, \boldsymbol{y}_{t+1})$
  $t \leftarrow t + 1$
end while
$i^* = \arg\max_{i \in [1,t]} y_i$
return $\boldsymbol{x}_{i^*}$