Carl von Ossietzky
Universität
Oldenburg

Fakultät II - Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik

Online Meta-Modeling of Communication Networks in Cyber-Physical Energy System Simulations

Von der Fakultät für Informatik, Wirtschaft- und Rechtswissenschaften
der Carl von Ossietzky Universität Oldenburg
zur Erlangung des Grades und Titels

Doktorin der Naturwissenschaften (Dr. rer. nat.)

angenommene Dissertation

von Frau Malin Radtke

geboren am 21.04.1999 in Itzehoe

1. Gutachter:	Prof. Dr. Sebastian Lehnhoff Department für Informatik Carl von Ossietzky Universität Oldenburg
2. Gutachter:	Prof. Chenye Wu, Ph.D. School of Science and Engineering The Chinese University of Hong Kong (Shenzhen)
Tag der Disputation:	30. Juni 2026

Abstract

Energy systems are becoming increasingly interconnected, digitalized, and adaptive, forming Cyber-Physical Energy Systems (CPES) whose reliable operation depends on communication networks. This necessitates the joint analysis of energy and communication systems, for example in co-simulation environments. However, an inherent challenge in conducting such studies arises from the increasing complexity of the system. This challenge relates to the need to facilitate large-scale parameter studies and "what-if analyses" without a proportional increase in model complexity or computational effort. This is particularly relevant when considering the modeling of communication networks, which are often represented by either overly simplified analytical abstractions or by computationally expensive detailed simulations.

This thesis addresses these limitations by introducing an online-trained meta-model that approximates detailed communication network simulations during execution. By continuously assessing its prediction accuracy, the meta-model dynamically substitutes the detailed simulation once sufficient confidence is achieved, thereby reducing overall runtime. This is enabled by a structured, technology-agnostic abstraction of CPES communication scenarios and a discrete-event-compatible internal approximation model that supports online adaptation to scenario-specific communication behavior.

A thorough evaluation has been conducted, which demonstrates that the proposed approach offers a favorable trade-off between accuracy and computational efficiency. The meta-model has been shown to consistently outperform analytical communication models in delay approximation, while simultaneously achieving a significant reduction in execution time when compared to detailed simulations. This enables scalable and communication-aware CPES analysis without compromising on essential modeling fidelity.

Zusammenfassung

Die zunehmende Vernetzung und Digitalisierung von Energiesystemen resultiert in der Entwicklung von sogenannten cyber-physischen Energiesystemen (CPES), deren zuverlässiger Betrieb maßgeblich von Kommunikationsnetzen abhängt. Eine gemeinsame Analyse von Energie- und Kommunikationssystemen, etwa in Co-Simulationsumgebungen, ist somit erforderlich. Mit der Zunahme der Komplexität der Systeme ergibt sich jedoch eine zentrale methodische Herausforderung, nämlich die Durchführung großskaliger Parameterstudien und "Was-wäre-wenn-Analysen", ohne dass die Modellkomplexität oder der rechnerische Aufwand proportional zunehmen. Dies ist insbesondere bei Kommunikationsnetzen problematisch, die häufig entweder durch stark vereinfachte analytische Modelle oder durch rechnerisch aufwendige Detailsimulationen abgebildet werden.

In der vorliegenden Arbeit wird dieser Limitation durch die Entwicklung eines zur Ausführungszeit ("online") trainierten Meta-Modells begegnet, welches detaillierte Kommunikationsnetzwerksimulationen während der Ausführung approximiert. Das Meta-Modell evaluiert kontinuierlich seine eigene Prognosegenauigkeit und substituiert die detaillierte Simulation dynamisch, sobald eine hinreichende Güte erreicht ist. In der Folge wird die Gesamtlaufzeit der Simulation reduziert. Dies wird durch eine strukturierte Abstraktion von CPES-Kommunikationsszenarien sowie ein diskret-ereigniskompatibles internes Approximationsmodell ermöglicht, das eine Anpassung an szenariospezifisches Kommunikationsverhalten zur Ausführungszeit unterstützt.

Eine umfassende Evaluation hat ergeben, dass der vorgeschlagene Ansatz einen Kompromiss zwischen Genauigkeit und Recheneffizienz erreicht. Das Meta-Modell weist im Vergleich zu analytischen Kommunikationsmodellen eine höhere Genauigkeit hinsichtlich der Abbildung von Verzögerungszeiten in der Kommunikation auf und reduziert zudem die Ausführungszeit im Vergleich zu detaillierten Simulationen. In der Folge werden skalierbare CPES-Analysen unter Berücksichtigung von Kommunikationsverzögerungen ohne Verlust wesentlicher Modelltreue ermöglicht.

Wie jede Blüte welkt und jede Jugend
Dem Alter weicht, blüht jede Lebensstufe,
Blüht jede Weisheit auch und jede Tugend
Zu ihrer Zeit und darf nicht ewig dauern.
Es muß das Herz bei jedem Lebensrufe
Bereit zum Abschied sein und Neubeginne,
Um sich in Tapferkeit und ohne Trauern
In andre, neue Bindungen zu geben.
Und jedem Anfang wohnt ein Zauber inne,
Der uns beschützt und der uns hilft, zu leben.
Auszug aus Stufen von Hermann Hesse

Getreu diesen Zeilen - *Und jedem Anfang wohnt ein Zauber inne* - blicke ich auf die letzten Jahre zurück, die für mich weit mehr waren als die Fertigstellung einer Dissertation: Sie waren eine eigene Stufe auf meinem Weg, die ich nicht allein gegangen bin. Vielen Menschen möchte ich dafür von Herzen danken.

Mein besonderer Dank gilt meinem Doktorvater und Erstgutachter, Sebastian Lehnhoff, für die Betreuung dieser Arbeit und weit darüber hinaus für die Unterstützung auf meinem Weg. Danke, dass du meine Ideen stets ernst genommen und mir immer wieder die Möglichkeit gegeben hast, sie auch umzusetzen. I would also like to sincerely thank my second reviewer, Chenye Wu, for his valuable contributions to this work. Ebenso danke ich Astrid Nieße, die als Prüfungsvorsitzende die Disputation geleitet hat, und die mich darüber hinaus vielfältige Weise unterstützt hat. Mein Dank gilt zudem Friederike Bruns für ihre Mitwirkung als wissenschaftliche Mitarbeiterin in der Prüfungskommission.

Ein herzliches Dankeschön geht an meine gesamte Forschungsgruppe sowie an alle Kolleginnen und Kollegen, die diese Zeit fachlich wie menschlich bereichert haben.

Von ganzem Herzen danke ich meiner Familie, Mama, Papa und Stina - dafür, dass ihr mich unterstützt und auf jeder meiner bisherigen Stufen begleitet und immer an mich geglaubt habt. Der größte Dank aber gilt dir, Steffen, für all die Diskussionen, teils auch inhaltlicher Natur, und für die Ablenkung, die ich hin und wieder gebraucht habe, um mich danach wieder auf die Arbeit konzentrieren zu können. Danke.

Contents

Abbreviations

AE	Absolute Error
AMI	Advanced Metering Infrastructure
ANOVA	Analysis of Variance
BAN	Building Area Network
B	Byte
bps	bit per second
CBR	Constant Bit-rate
CDSB	Central Demand Supply Balancing
cocoon	coupled communication simulation with an online trained meta-model
CPS	Cyber-Physical System
CPES	Cyber-Physical Energy System
CSV	Comma-Separated Values
dB	Decibel
dBi	Decibel relative to isotropic radiator
dBm	Decibel relative to one milliwatt
DER	Distributed Energy Resource
DES	Discrete Event Simulation
DSL	Digital Subscriber Line
DWM	Dynamic Weighted Majority

ECDF	Empirical Cumulative Distribution Function
EWMA	Exponential Weighted Moving Average
FAN	Field Area Network
FES	Future Event Set
Gbps	Gigabit per second
GHz	Gigahertz
HAN	Home Area Network
IAN	Industrial Area Network
ICT	Information and Communication Technologies
ITT	Intention-To-Treat
JSON	JavaScript Object Notation
kbps	kilobit per second
km	kilometer
LV	Low Voltage
LTE	Long Term Evolution
m	meter
MAE	Mean Absolute Error
MAS	Multi-Agent System
Mbps	Megabit per second
MHz	Megahertz
MPLS	Multi-Protocol Label Switching
mph	messages per hour
mpm	messages per minute
mps	messages per second

MQTT	Message Queueing Telemetry Transport
ms	millisecond
MLP	Multi-Layer Perceptron
NAN	Neighborhood Area Network
NFR	Non-Functional Requirement
NRMSE	Normalized RMSE
ORKG	Open Research Knowledge Graph
OTN	Optical Transport Network
PLC	Power Line Communication
PMU	Phasor Measurement Unit
PP	Per-Protocol
QoS	Quality of Service
RF	Radio Frequency
RMSE	Root Mean Squared Error
RQ	Research Question
s	second
SCADA	Supervisory Control and Data Acquisition
SDN	Software-Defined Networking
SVR	Support Vector Regression
TCP	Transmission Control Protocol
trace	training data generation tool for combined communication and energy system scenarios
UQ	Uncertainty Quantification
WAC	Wide Area Control

WAN	Wide Area Network
WAMS	Wide Area Monitoring System
6LoWPAN	IPv6 over Low-Power Wireless Personal Area Network
WPAN	Wireless Personal Area Network

Nomenclature

	Communication Network Representation
d	End-to-end delay time of a message
$d_{\mathbf{real}}$	Measured end-to-end delay from the detailed communication simulation
$d_{\mathbf{pred}}$	Predicted end-to-end delay from the meta-model
$\mathbf{m}$	Message object
M	Set of message objects
t_s	Time at which a message arrives at a node
t_d	Time of message departure
p	Packet size of a message in Byte
ν	Network state
η	Node state
λ	Link state
t	Simulation time
G_t	System representation as graph at time t
N_t	Set of nodes in the graph at time t
L_t	Set of directed links (edges) in the graph at time t
n	Node (approximated communication endpoint) in the graph
ℓ_{ij}	Directed link from node n_i to node n_j
	Online Learning
c	Cluster of message objects

$\boldsymbol{\mu}(c)$	Centroid of cluster c
D	Distance matrix for message objects
sim	Simultaneity in message objects
$load$	Network load in message objects
i_{pupa}	Batch size in online learning
reg^c	Regressor trained on a message object cluster c
reg^{on}	Regressor trained online
X	Set of variables in clustering
δ	Distance threshold between clusters
α	Learning rate in online learning
$\theta_{butterfly}$	Threshold value for model substitution
f	Factor in substitution assessment
$S_{\mathbf{combined}}$	Combined confidence score for substitution assessment
w	Weights used for the weighting of factors f
$e_t^{(w)}$	Absolute prediction error of $d_{\text{w pred}}$ at message index t
$\hat{a}$	OLS slope of recent weighted prediction errors
$E_{\mathbf{max}}$	Expected maximum error threshold
$\overline{e}^{(w)}_{\mathbf{recent}}$	Mean absolute error of $d_{\text{w pred}}$ over the recent window
$d_{\mathbf{min}}$	Minimum distance from current message to any cluster centroid
$d_{\mathbf{max}}$	Maximum expected distance for cluster representativeness assessment
N_t	Node count in the communication graph at current evaluation
N_{t-1}	Node count in the communication graph at previous evaluation
	Evaluation Metrics and Performance Measures
$NRMSE$	Normalized RMSE of mean delay times

$RMSE$	Root mean squared error
MAE	Mean absolute error
$C_{\pm\sigma}$	Empirical coverage of delay values within one standard deviation of the mean
W	First-order Wasserstein distance between delay distributions
ET	Execution time of a simulation run
S	Substitution indicator of the meta-model
SC	Composite score combining accuracy and performance metrics
$r(\cdot)$	Rank of a configuration with respect to a given metric
$r_{\mathbf{max}}$	Maximum possible rank among evaluated configurations
	Scenario and Aggregation Parameters
n	Number of message objects evaluated in a scenario
r	Number of independent simulation runs per scenario
$\bar{d}_i$	Mean delay of message object i over multiple runs
$\mathbb{I}(\cdot)$	Indicator function, equal to 1 if the condition holds and 0 otherwise
	Trade-Off and Statistical Analysis
$\sigma^{\mathbf{m}}$	Standard deviation of the delay within model m
ϵ	Admissible absolute estimation error
$\epsilon_{\mathbf{cocoon}}$	Inherent approximation error of the meta-model
$\epsilon_{\mathbf{achievable}}$	Achievable estimation error under a fixed execution time budget
T	Available execution time budget for conducting simulation runs
$T_{\mathbf{critical}}$	Critical execution time
t	Mean execution time of one simulation run
$t_{\mathbf{cocoon}}$	Mean execution time of one simulation run using the meta-model

n	Maximum number of runs executable within time budget T
R	Required number of simulation runs to achieve error bound ϵ
$t_{\alpha/2,R_0}$	Critical value of the t-distribution for confidence level $1-\alpha$ and R_0 degrees of freedom

1 Introduction

> The single biggest problem in communication is the illusion that it has taken place.
>
> *George Bernard Shaw*

The term "Cyber-Physical System (CPS)" was first used in 2006 by Helen Gill [16] in the "NSF Workshop on Cyber-Physical Systems". The goal of the workshop was to frame a new research initiative to enable "rapid and reliable development and integration of computer- and information-centric physical and engineered systems" [16]. As the name suggests, CPSs combine communication and computational resources (*Cyber*) with interactions in the physical world (*Physical*), while capturing the inherent complexities of such integrated systems (*Systems*) [17, 18].
Today, CPSs form the backbone of many critical infrastructures [18]. This also applies to modern energy systems, which – driven by the increasing integration of distributed renewable resources – are evolving into complex, tightly coupled Cyber-Physical Energy Systems (CPESs). These systems comprise physical and control layers that are interconnected through communication networks [18, 19, 20, 21], as illustrated on the upper left in Figure 1.1. The transformation of energy systems to CPESs is enabled by Information and Communication Technologies (ICT), which provide high-speed and bidirectional communication infrastructures, advanced information processing, and distributed computing technologies indispensable for improving the efficiency, reliability, and sustainability of energy systems [21].

As a result, modern energy systems depend not only on the flow of energy but also on the reliable flow of information [20]. The combination of those creates significant interdependencies: the reliability and performance of energy systems increasingly depend on the underlying communication infrastructure, and vice versa [22, 23]. However, traditional approaches often neglect these inter-dependencies when analyzing energy systems, thus failing to capture the full range of cyber-physical dynamics [17]. In practice, cyber layer attributes, such as delay times or availability, directly affect the physical system, which

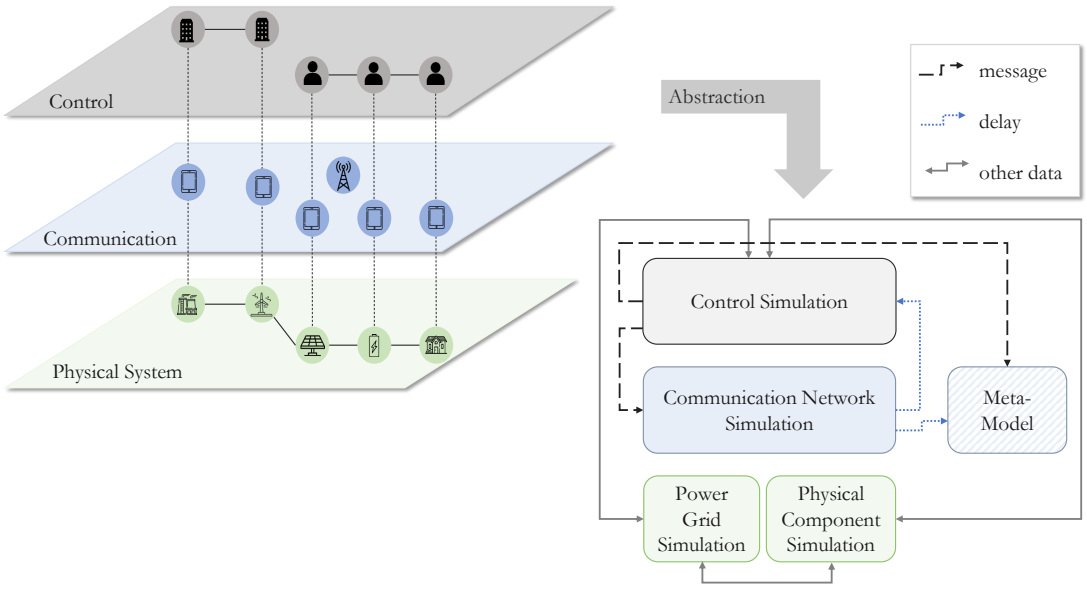


Figure 1.1: Problem Definition: The inter-dependencies between subsystems in CPES require joint consideration in simulations, which might become very complex with increasing system complexity.

can undermine stability and performance [17]. ICT has therefore become indispensable in enabling grid services such as state estimation, congestion management, and redispatch, which rely on sensors and advanced data processing for real-time monitoring and control [24]. While these technologies greatly enhance situational awareness and operational flexibility, they simultaneously increase the system's exposure to cyber and cyber–physical attacks [17, 21]. Such threats can endanger the reliable and efficient supply of electricity [19].

Beyond these challenges, the coordination and control of Distributed Energy Resources (DERs) pose further difficulties. A centralized energy system struggles to cope with the highly distributed nature of the modern grid, the requirement to operate in islanding mode, the intermittency of renewable sources, and the limited bandwidth available for communication [25, 26, 27]. In this context, Multi-Agent Systems (MAS) have emerged as a promising approach for managing distributed systems. They model the grid as a collection of relatively simple entities, so-called agents, which correspond to sources, loads, or other components and evolve within a shared environment [27]. Through interaction, agents can cooperate or compete to achieve local and global objectives, thereby providing a certain degree of distributed intelligence. Formally, a MAS consists of multiple interacting

computing elements, agents, that possess two essential capabilities: (1) a certain level of autonomy in deciding how to act in order to satisfy their design objectives, and (2) the ability to interact with other agents not only by exchanging data, but also through cooperation, coordination, or negotiation [28, 29]. Key properties of MAS include flexibility, fault tolerance, autonomy, responsiveness, pro-activeness, social ability, and scalability [26]. These characteristics make MAS particularly suitable for the requirements of modern energy systems. However, the integration of agent-based control into energy systems further increases the dependency on timely and reliable data exchange. The effectiveness of MAS relies on continuous communication between agents, accentuating the inter-dependencies between energy and communication networks, reinforcing the opportunities and vulnerabilities of CPESs.

The functionality, security and stability of CPESs can be enhanced by systematic testing in order to discover vulnerabilities, develop security countermeasures, and evaluate grid operation under fault or maliciously constructed scenarios [18]. However, due to their high complexity and interdisciplinary nature, formal analytical testing is not feasible. Experimentation on the real system is also not an option, which makes abstraction into simulation models essential [20]. As illustrated in the lower right of Figure 1.1, real world systems may be abstracted into simulation models that allow to capture essential properties of the energy and communication systems and to study their interactions. When investigating mutual dependencies between these systems, traditional methods often oversimplify interactions, which may neglect important dependencies. This includes communication-induced instabilities, where latency, data loss, and corruption directly influence grid dynamics. As such, realistic modeling of communication performance is essential to correctly capture the transition of grid services between normal, limited, and failed states [30]. Here, co-simulation offers a powerful approach, enabling the flexible and efficient investigation of dynamic, interdisciplinary interactions between communication and energy systems [23].

Despite these advantages, (co-)simulation of large-scale and coupled energy systems comes with considerable challenges. The sheer size and heterogeneity of the systems involved, together with the need to run custom solvers or execution environments, results in significant computational overhead [20, 31, 32]. Even as computational capacities increase, the pursuit of more accurate real-world representations continues to drive simulation codes toward greater complexity and higher costs [20, 32]. In principle, these problems can be mitigated by simplifying models, selecting faster numerical solvers, or providing more computational resources [23]. Yet, model simplification usually comes at the cost of accuracy, which motivates the search for alternative approaches that retain fidelity while improving

computational performance. In this context, meta-modeling offers a promising approach to reduce simulation costs while maintaining accuracy. In co-simulation environments, inputs and outputs are exchanged between different components (simulators), which makes it possible to observe the input–output behavior of a model and to train a meta-model based on this information. Such meta-models act as "models of models": instead of executing a computationally expensive simulation, they approximate its input–output responses and can thereby replace it at significantly lower computational effort [32].

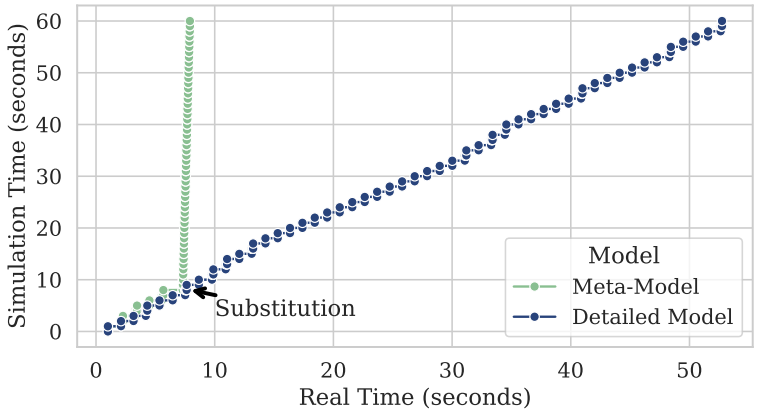


Figure 1.2: Comparison of simulation time advancement between the detailed communication model and the online-trained meta-model. Each data point represents a simulation event. After the substitution point, the meta-model advances simulation time at a substantially higher rate relative to real time, demonstrating the computational speedup achieved by replacing the detailed simulation.

Existing approaches to meta-modeling of communication network simulations often rely on offline models that are pre-trained with abstract traffic models such as Constant Bit-rate (CBR) message dispatch [33, 34, 35]. Nevertheless, communication applications in CPESs exhibit highly diverse and scenario-specific characteristics that cannot be adequately captured by such abstract, stochastic distributions [1]. A more flexible solution is therefore to train the meta-model online[1] during simulation execution, enabling it to dynamically adapt to the traffic characteristics of the scenario under investigation (the use of online learning has been, for example, motivated in [36] for the prediction of disturbance events in power grids). This online-trained meta-model could substitute the original communication network simulation to accelerate execution, as shown in Figure 1.2[2].

[1] For a historical overview on the spelling of *online* and *on-line*, please refer to Section A.

[2] The time behavior shown was obtained in a scenario (CBR 1 Mbps, five devices, 5G network) in the evaluation in Chapter 6.

1.1 Challenges and Research Gap

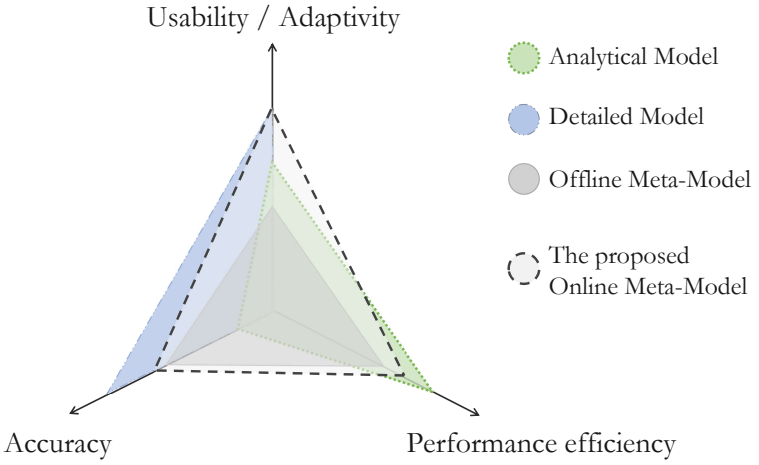


Figure 1.3: Classifying the proposed approach in terms of the trade-off between non-functional requirements (simplified illustration).

The developments outlined above highlight that a realistic representation of communication networks in energy system scenarios is indispensable. At the same time, several key challenges emerge. Detailed communication simulations are computationally very expensive, which makes them impractical for large-scale scenarios or extensive parameter studies. In co-simulation environments, where energy and communication models are coupled, these computational burdens quickly lead to limitations in terms of scalability and execution time.

A central challenge lies in the inherent trade-off: on the one hand, energy system simulations require sufficient accuracy to capture the relevant interactions between subsystems. On the other hand, models must remain efficient and adaptable to be applicable in co-simulation environments (see Figure 1.3). Traditional analytical models are efficient but oversimplify critical dynamics by relying on strong assumptions about network behavior (such as fixed delay distributions or independence of traffic flows) and therefore represent the lower bound in terms of modeling fidelity. In contrast, highly detailed network simulations offer accuracy but are hardly scalable.

Offline-trained meta-models have been proposed as one way to mitigate this trade-off. By learning from pre-generated input–output data, they approximate the behavior of detailed

simulations at significantly reduced computational cost. However, two major limitations restrict their applicability: (1) their validity is confined to the training data distribution, making them non-portable when scenarios deviate from the pre-generated set, and (2) they are typically implemented as black-box models that cannot be seamlessly integrated into co-simulation environments where multiple simulators exchange state information during runtime. As a result, offline meta-models lack the flexibility required for complex and dynamic CPES co-simulations.

This defines the research gap as illustrated in Figure 1.3: there is a lack of models for communication networks in CPES scenarios that are more accurate than analytical approaches, more efficient than detailed simulations, and at the same time adaptive and integrable across heterogeneous scenarios. This thesis addresses this gap by proposing the use of an online-trained meta-model, which approximates the observed input–output behavior of communication simulations during simulation execution. In this way, realistic, efficient, and adaptive representations of communication networks in CPESs can be achieved.

1.2 Research Goal

The research gap identified above forms the foundation of this thesis. To address it, several Research Questions (RQs) are defined, corresponding artifacts are stated, and relevant Non-Functional Requirements (NFRs) are derived.

1.2.1 Hypotheses and Research Questions

To summarize the research gap, there is a need for a method to approximate communication network simulations in energy system scenarios. Such a method should accelerate simulation execution and thereby enable extensive parameter studies of combined communication and energy system scenarios. Due to the high complexity and diversity of communication applications in CPESs, the model must adapt to the current scenario and be refined online during simulation execution.

This leads to the following overall RQ:

Research Question

How can online-trained meta-models be generated and integrated into discrete-event communication network simulations while preserving simulation fidelity?

Since CPESs consist of multiple subsystems and diverse applications, communication plays a central role in transferring information between distributed components. These appli-

cations range from simple broadcast communication to complex communication flows involving agent-based control structures and intelligent decision-making. This observation motivates the first hypothesis:

Hypothesis 1
The communication behavior of end devices in the energy system can be systematically described and simulated.

The assumption that this hypothesis is true means that end-device communication behavior can be captured as traffic models, which are subsequently embedded into energy-system-specific communication network models. Then, these scenarios can be simulated using a detailed communication network simulator. This leads to the first Sub-RQ:

Sub-Research Question 1

How can communication behavior in CPESs be abstracted into simulation-relevant input–output characteristics?

To address this question, the following artifacts are developed: (i) a conceptual description model for communication scenarios in CPES, (ii) an automated communication scenario generation tool implementing the defined traffic models, and (iii) datasets of communication network models and simulation data.

After describing and simulating realistic communication scenarios, the next step is to approximate the input-output behavior of the detailed simulation model with a meta-model. In turn, this meta-model should be integratable into a simulation environment to investigate mutual dependencies between communication and energy systems. Additionally, the meta-model should adapt to the current scenario during simulation execution. This leads to the second hypothesis:

Hypothesis 2
The response of the simulated communication network can be learned online by approximating the observed outputs.

If the assumption is made that the online approximation of the detailed simulation model is viable, then the second Sub-RQ can be formulated as:

Sub-Research Question 2

How can observations from discrete-event communication simulations be transformed into representations suitable for online training of communication meta-models?

This is supported by (i) a conceptual network description model and (ii) a software artifact in the form of an online learning algorithm.

A further question concerns the conditions under which the resulting meta-model can substitute the original simulation. This substitution should only occur if the meta-model is capable of approximating the original, detailed simulation with sufficient fidelity and if the operating conditions do not differ excessively in the present scenario. This leads to the third hypothesis:

Hypothesis 3
There is a threshold value above which a trained meta-model approximates the original simulation accurately (enough).

Assuming this hypothesis is true, the meta-model could be evaluated during simulation execution to determine its accuracy compared to the original simulation model. This is captured in Sub-RQ:

Sub-Research Question 3

What quantitative criteria can be used to evaluate whether an online-trained meta-model is a valid substitute for a detailed simulation model?

The work introduces (i) a conceptual threshold estimation methodology and (ii) a software artifact, the substitution threshold estimator, to systematically define and assess substitution conditions.

Finally, beyond proving the feasibility of substitution, it is necessary to assess the performance benefits in realistic scenarios. This motivates the fourth hypothesis:

Hypothesis 4
Substituting the original simulation with the meta-model speeds up execution while the accuracy of the results remains high.

To investigate this, the meta-model must be compared to alternative modeling approaches consisting of simple yet performant analytical models and detailed, high-fidelity yet complex

simulation models. The corresponding Sub-RQ addresses this comparison explicitly:

Sub-Research Question 4

Under which communication scenario characteristics does an online-trained meta-model achieve comparable validity to a detailed simulation model, using a channel model and static graph model as baselines?

This is analyzed using (i) a conceptual evaluation methodology and (ii) a benchmarking and simulation environment as a software artifact.

1.2.2 Non-Functional Requirements

The viability of an online-trained meta-model for CPESs simulations depends on a set of quality attributes that go beyond functional correctness. The NFRs below stem from the RQs and the gap analysis. For each requirement the *goal* (what must be achieved) and the *rationale* (why it matters) are retained.

NFR 1: Accuracy

Goal The meta-model must predict the observable behavior of the detailed communication network simulation model with a fidelity that exceeds that of simple analytical models.

Rationale When examining energy and communication systems in their combined form, the time lag in receiving information due to communication delays must be realistically represented in order to guarantee stable systems.

NFR 2: Performance Efficiency

Goal The meta-model must significantly reduce computational overhead compared to detailed, integrated communication simulations.

Rationale Co-simulations that couple power-system dynamics with high-fidelity communication network models (often requiring millisecond resolution) become infeasible for large-scale scenario exploration unless the communication part is computationally tractable.

NFR 3: Scalability

Goal The approach must remain tractable as the size of the communication topology or the message volume grows.

Rationale Future CPES test-beds involve a multitude of devices; the communication simulation must not become a bottleneck.

NFR 4: Adaptivity

Goal The meta-model must be deployable across heterogeneous simulation environments without extensive re-engineering and must be adaptable to diverse communication scenarios during simulation execution.

Rationale Researchers and practitioners use a variety of simulation environments and investigate diverse communication scenarios.

1.2.3 Limitations

The proposed meta-modeling approach addresses the previously introduced challenge: enabling efficient yet accurate communication network simulations in CPESs. While this approach reduces computational overhead and improves scalability, it also entails several limitations.

First, the meta-model approximates only the end-to-end delay as the primary performance indicator. If a researcher's focus lies on additional metrics (such as packet loss, jitter, or protocol-specific behaviors) the approach must either be extended accordingly or replaced by a detailed network simulation.

Second, meta-models are by design *models of models*: they approximate a detailed simulation that is itself an abstraction of the real system, introducing two layers of approximation. While the proposed online learning enables adaptation to the specific scenario under investigation (beyond the initial training data), generalization remains bounded by what is observed during pre-training and online updates. Consequently, out-of-distribution regimes and rare events (e.g., network faults, atypical congestion patterns, or cyber-attacks) may still be misrepresented if they are insufficiently encountered. Moreover, any inaccuracies or biases in the detailed simulation propagate to the meta-model and can compound the deviation from the real system.

Finally, the applicability of the approach depends on the complexity of the scenario. While analytical models may suffice for simple cases, highly complex or safety-critical studies still require detailed simulations to ensure reliability. To support practitioners, recommendations on when and how to apply the meta-modeling approach are presented in Section 6.6.

1.2.4 Reference to Own Preliminary and Accompanying Work

The research presented in this thesis is built on a coherent series of peer-reviewed journal publications and conference contributions. All of these works address different facets of the overarching goal – the efficient, accurate, and adaptive modeling of communication networks in CPESs – and together they constitute the technical backbone for the four

Sub-RQs and the four Hypotheses introduced earlier (see Subsection 1.2.1). A detailed list of the publications associated with this thesis can be found under *Own Publications* before the references section.

Early work on realistic end-device traffic [2] showed that detailed communication network simulations can be used to investigate inter-dependencies between energy and communication systems. Following on from this, a comprehensive simulation environment *cosima* [3] was built in which communication and energy system components can exchange data in coupled scenarios. This includes the integration of the communication network simulator *OMNeT++* [37] into the co-simulation framework *mosaik* [38]. In follow-up work, the agent framework *mango* [39] was coupled with *cosima* [4] in order to also be able to investigate intelligent, agent-based control strategies under the influence of realistic communication networks. In addition, a large-scale scenario was simulated in [5], in which agent-based control strategies were evaluated in a combined communication-energy system environment under communication network disruptions. These and other analyses of agent systems in CPESs under the influence of communication networks were systematized in [6].

A review of the state-of-the-art modeling approaches for communication networks in CPESs scenarios was presented in [7], establishing an overview that distinguishes analytical and detailed simulation techniques. The same line of work introduced the meta-model concept through a conference poster [8], which highlighted the potential of learning-based meta-models to replace costly simulations.

The methodology for constructing realistic communication scenarios (required to answer Sub-RQ 1) was formalized in [9]. In addition to this, the method was expanded and implemented in an automated scenario generation tool based on the developed communication traffic models in [1]. Concurrently, an extensive simulation data analysis was conducted in that context.

Addressing Sub-RQ 2 and Sub-RQ 3, the online approximation of communication-network simulations was introduced in [10]. The paper presents the incremental learning algorithm that continuously builds a meta-model of the detailed simulator and a threshold estimator that decides when the meta-model can safely replace the original model, thereby establishing a reliable substitution criterion. The exploitation of different modeling approaches in MAS analyses (required for Sub-RQ 4) was demonstrated in [11], where different modeling approaches (analytical, detailed simulation) were compared in the context of distributed control under communication impairments.

1.3 Big Picture and Thesis Outline

To structure the contents of this thesis, the big picture is introduced first. It is illustrated in Figure 1.4, which also relates the RQs to the resulting artifacts. This dissertation is organized into seven chapters.

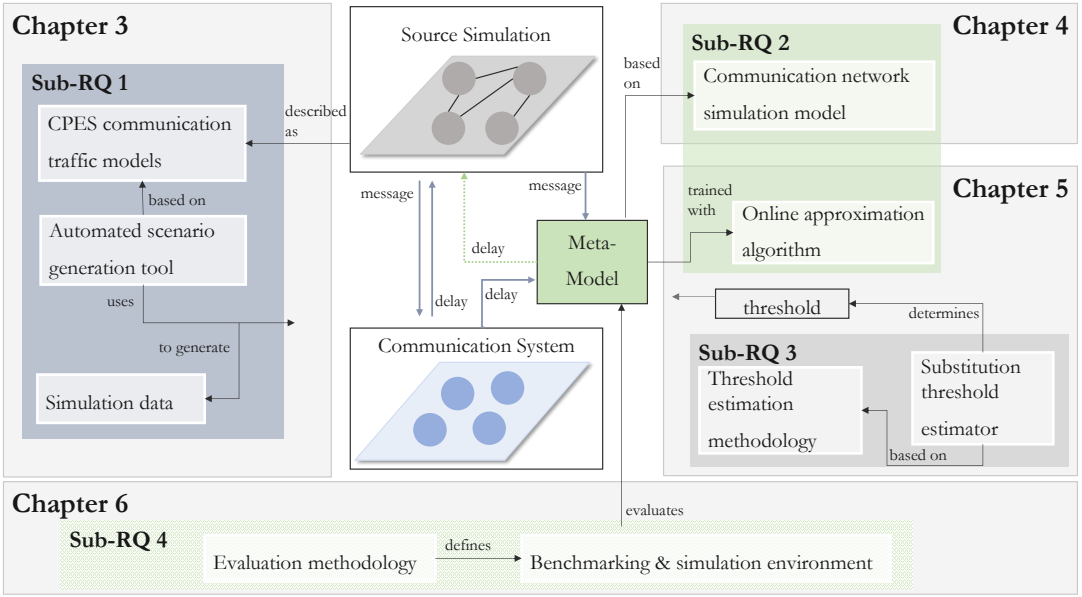


Figure 1.4: Outline of this dissertation: Chapters are organized according to the research questions and resulting artifacts.

Chapter 2 introduces the necessary foundations and related work. It covers the principles of Simulations and Meta-Models, as well as communication in CPESs and existing Communication Modeling Approaches.

Chapter 3 addresses Sub-RQ 1. Here, communication behavior in CPES is described as traffic models that serve as the basis for an automated scenario generation tool. Using this tool, simulation data is generated as an artifact, which is subsequently analyzed in the same chapter.

In Chapter 4, Sub-RQ 2 is tackled by designing a graph-based communication network simulation model that fulfills Discrete Event Simulation (DES) properties. This model forms the basis for the later meta-modeling approach.

Chapter 5 addresses Sub-RQ 2 and Sub-RQ 3. It presents the online approximation algorithm that trains the meta-model during runtime, based on pre-trained cluster regressors. A threshold estimation methodology is introduced, which determines when the meta-model can substitute the detailed simulation.

Chapter 6 evaluates the overall meta-modeling approach (Sub-RQ 4). An evaluation methodology and benchmarking environment are defined as artifacts. The approach is then assessed against alternative communication modeling approaches with respect to the Non-Functional Requirements.

Finally, Chapter 7 summarizes the main results and outlines future research directions.

2 Fundamentals and Related Work

> A complex system that works is
> invariably found to have evolved from
> a simple system that worked.
>
> *John Gall*

The analysis of CPESs necessitates methods that concurrently capture physical processes within the power system and the behavior of the supporting communication infrastructure. This chapter introduces the conceptual and methodological foundations for such analyses and positions the contribution of this thesis within the existing body of work. The focus is on simulation-based approaches, surrogate modeling with meta-models, and the specific characteristics of communication networks in CPESs, as well as on existing communication modeling approaches used in the energy domain.

First, Section 2.1 reviews fundamental concepts of systems, models, and simulations, with an emphasis on Discrete Event Simulation (DES) and co-simulation as the dominant paradigm for modeling communication networks alongside power systems. Section 2.2 introduces meta-models as computationally efficient surrogates for detailed simulations and discusses online learning and ensemble methods as mechanisms for adapting such models to changing conditions. Section 2.3 then summarizes key aspects of communication in CPESs, including network types and technologies, performance requirements, and representative applications that shape traffic patterns and quality-of-service needs. Building on these foundations, Section 2.4 surveys existing approaches for modeling communication networks in energy system studies and highlights their trade-offs in terms of realism, computational effort, and integration effort based on the own, previously published work in [7].

Finally, Section 2.5 synthesizes the main opportunities and challenges identified throughout the chapter and derives relevant factors for the work on the research questions: the need for communication models that preserve essential discrete-event characteristics and can be integrated into co-simulation frameworks for CPESs, while avoiding the computational cost of detailed simulations and inflexibility of offline-trained models.

2.1 Simulations

The analysis of CPESs requires methods capable of representing complex interactions between heterogeneous subsystems, such as physical energy systems and communication infrastructures. Direct experimentation on real systems is often infeasible due to cost, safety, or availability constraints. Therefore, simulation has become a central methodology for studying such systems [23].

This section introduces the fundamental terminology and concepts needed to understand simulation-based studies. First, basic definitions of systems, models, and states are provided, followed by a classification of systems and simulation models. Different ways to study a system are discussed, highlighting why simulation is often the method of choice when analytical solutions are not applicable.

Subsequently, DES is introduced as the dominant paradigm for modeling communication networks, including its mechanisms, components, and typical applications. Finally, co-simulation approaches are presented, which allow the coordinated execution of multiple domain-specific simulators. This is particularly relevant for CPESs, where continuous-time models of physical processes must be integrated with event-driven communication models.

2.1.1 Foundations of Systems and Models

Figure 2.1 illustrates the conceptual progression from real-world systems to models that are subsequently executed as simulation models. Within this context, several foundational concepts can be defined based on [40, 41, 42, 43] to provide a consistent terminology.

A *system* can be understood as a real-world entity, such as a local power grid or a communication network within a specific region. To enable systematic analysis, the system is conceptually separated from its environment by defining a *system boundary*. This boundary delineates which elements belong to the system and which belong to its environment, thereby establishing the scope of investigation. A system can be defined according to [40]:

Definition 1 (System)**.** *A system is a set of interrelated elements that are viewed as a whole within a specific context and are considered distinct from their environment.*

At its *boundaries*, the system can exchange *inputs* and *outputs* with the environment through defined interfaces.

A system may consist of multiple *system components* that can form subsystems or represent atomic units that cannot be further decomposed. The structure of a system emerges from the relationships among these elements. The properties of each system component are represented by constant and variable attributes, the so-called *state variables*. The collective values of these state variables at a given point in time define the overall

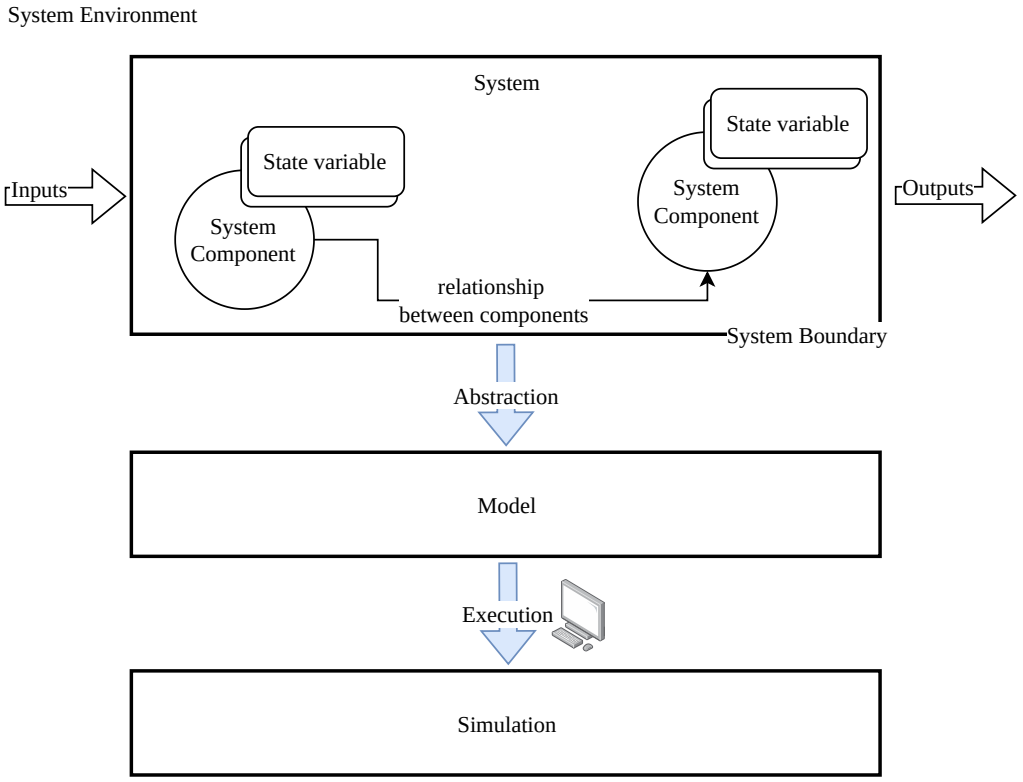


Figure 2.1: Conceptual flow from a real-world system to a simulation model. The diagram shows the delimitation of a system by its boundary, the construction of a model from system components, and the execution of that model to analyze system behavior for given inputs and outputs.

state of the system. Changes in one or more state variables lead to a corresponding *state change* or *state transition*. Thus, the system state provides a complete snapshot of the information required to describe the system's condition at a particular moment. Based on Law et al. [43], the system state can be defined as:

Definition 2 (System state)**.** *The system state is the collection of all state variables whose values at a specific point in time uniquely describe the condition of the system.*

As illustrated in Figure 2.1, systems may then be abstracted as models. With that, a model can be defined as [41]:

Definition 3 (Model)**.** *A simplified replica of a planned or existing system with its processes in another conceptual or concrete system.*

2.1.2 Classification of Systems and Models

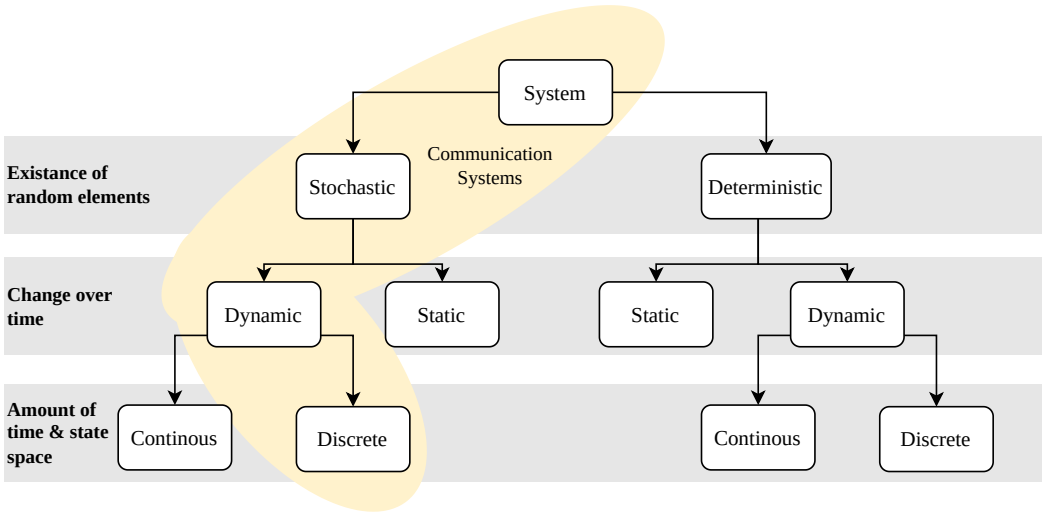


Figure 2.2: Hierarchical classification of systems based on randomness (stochastic vs. deterministic), temporal behavior (dynamic vs. static), and the nature of time and state spaces (continuous vs. discrete).

Figure 2.2 provides a hierarchical classification of systems and simulation models along several fundamental dimensions discussed in the literature. At the top level, systems are distinguished by the *existence of random elements*. A *deterministic* system contains no stochastic components, whereas a *stochastic* system incorporates randomness in its inputs or internal behavior [44]. Deterministic simulation models therefore produce identical outputs for identical inputs, while stochastic models generate random output realizations that must be interpreted as estimates of underlying performance measures [43].

A second axis differentiates between *static* and *dynamic* systems. This distinction refers to whether a system evolves over time. A *static* simulation model represents a system at a single point in time, while a *dynamic* model captures system behavior as it changes temporally [43, 44].

On the next level, the figure classifies systems by whether their time and state space are *continuous* or *discrete*. In discrete systems, state variables change instantaneously at separated points in time, while continuous systems involve state variables that evolve continuously with respect to time [43]. Analogously, continuous and discrete simulation

models reflect whether the modeled time progression or state changes appear continuous or occur in discrete steps [43]. This dimension relates directly to the general classification of systems based on the amount of time and state space considered [44].

Although not explicitly shown in the diagram, another common classification (*terminating* versus *non-terminating* systems) can also be applied. Terminating systems have a natural endpoint, whereas non-terminating systems conceptually operate indefinitely [44].

Communication systems are typically modeled as stochastic, dynamic, and predominantly discrete-event systems, since traffic processes, protocol interactions, and network state changes occur at distinct points in time and are subject to randomness.

2.1.3 Simulation as a Method

As depicted in Figure 2.1, *simulations* can be utilized as a technique to examine system behavior under specific inputs and outputs. In any case, a variety of methodologies exist for the study of a system, as evidenced by the work of Law et al. [43] and illustrated in Figure 2.3.

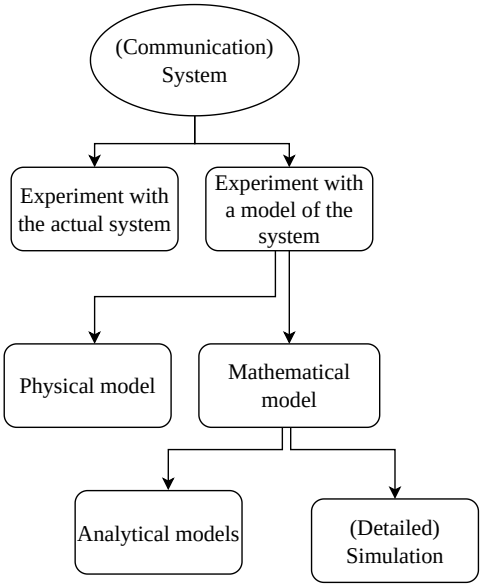


Figure 2.3: Ways to study communication systems: direct experimentation with the real system, modeling via physical or mathematical representations, and analysis using analytical models or detailed simulations (adapted from Figure 1.1 in [43]).

The study of systems can follow several methodological paths. In principle, one may

experiment directly with the real system, which yields valid and meaningful results. However, such an approach is often impractical in communication networks (especially in the context of CPESs), since real-system experimentation may be prohibitively costly, disruptive, or impossible when the system does not yet exist. Consequently, it is typically necessary to construct a model that serves as a suitable representation of the actual system.

Models themselves can take different forms. While physical or iconic models exist, they play only a minor role in operations research and system analysis. Instead, most models are mathematical, describing the system through logical and quantitative relationships that capture its essential structure and behavior.

Depending on the complexity of the model, one may attempt to obtain an analytical solution. Simple models may allow closed-form expressions for key performance measures. Yet many real-world systems (particularly communication systems with intricate interactions and stochastic processes) are too complex for exact analytical treatment. In such cases, the model must be investigated through simulation, meaning that it is executed numerically to observe how chosen inputs influence the resulting outputs and performance metrics. Complex systems can also be captured analytically, but this typically requires a higher level of abstraction, which may limit the level of detail and realism attainable in the resulting model. A detailed overview of modeling approaches for communication networks in CPESs is presented in Section 2.4.

In general, simulations provide a structured way to study the behavior of such models. According to Gutenschwager et al. [44]:

Definition 4 (Simulation model)**.** *A simulation model is a simplified but executable representation of reality that enables the experimental analysis of dynamic relationships within a system.*

With that, simulations can be defined as [44]:

Definition 5 (Simulation)**.** *Simulation is a problem-solving method in which controlled experiments are carried out on these models to gain insight into the behavior of the underlying system.*

Each *simulation experiment* consists of a targeted empirical investigation of the model's response, typically based on repeated *simulation runs*, where a run denotes one execution of the simulation model under a defined set of parameter values. Simulation methods themselves can be classified according to the mechanism used to advance simulation time. In the *fixed-increment time-advance* approach, time progresses in constant, predefined steps. In contrast, *Discrete Event Simulation (DES)* employs a next-event time-advance mechanism, where the simulation clock jumps from one event to the next, and state changes occur only when such events take place. [44]

2.1.4 Discrete-Event Simulation and Modeling

As previously stated, DES involves the instantaneous change in state variables at specific points in time, referred to as events [43].

Definition 6 (Event)**.** *An event is an instantaneous occurrence that may change the state of a system.*

Events occur at specific times during the simulation, represented by a *simulation clock*, which is a variable in a simulation model that gives the current value of simulation time. To ensure the continuous monitoring of the simulation time as it progresses, the definition of a *Time-Advance Mechanism* is necessary. Two approaches for advancing the simulation clock are proposed in [43]: the *next-event time advance* approach, which is utilized by all major simulation software, and the *fixed-increment time advance*, which is a special case of the former.

A DES model requires a clear specification of how the system evolves from one event to the next. To support this, several core components must be defined [42, 43]. The *system state* captures all variables needed to describe the system at any time, while the *simulation clock* maintains the current simulated time. Future state changes are organized through an *Event List* or *Future Event Set (FES)*, which stores scheduled events ordered by their time of occurrence.

The execution of the simulation is controlled by a set of routines. An *initialization routine* prepares the system state and event list at time 0. A *timing routine* identifies the next imminent event and advances the simulation clock accordingly. For each event type, an *event routine* specifies how the system state is updated when that event occurs. In practice, different conceptual approaches exist for organizing this scheduling logic. The *event-scheduling* approach explicitly processes events one by one as they occur. The *activity-scanning* approach describes the model through activities defined by start and end events that become active when their preconditions are satisfied. The *process-interaction* approach structures system behavior into processes, each representing a sequence of related events within the system [44].

These routines are orchestrated by the *main program*, which repeatedly processes events, updates the system state, checks termination conditions, and triggers reporting at the end of the simulation. Together, these components form the fundamental structure required to implement a discrete-event simulation model [42, 43].

Queuing models Queuing systems represent one of the most prominent application domains of DES. Simulation is frequently employed for their analysis, as queuing models

arise in a wide range of service-oriented and resource-sharing environments, including service facilities, material-handling systems, and telephone and communication networks [42]. In a typical queuing system, customers (understood as any entities requesting service) arrive over time, may join a waiting line, are eventually served by one or more resources, and then depart the system.

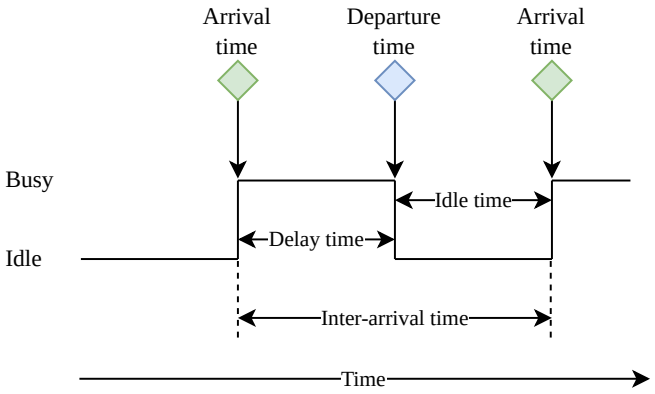


Figure 2.4: Illustration of fundamental timing relationships in a single-server queuing system, including arrival times, inter-arrival times, service start and completion times, customer delay (waiting time), and idle time (adapted from Figure 1.3 in [45]).

To support the quantitative evaluation of such systems, several key terms are commonly used. *Delay* or *waiting time* denotes the time an entity spends waiting for a resource. *Buffer occupancy* or *queue length* refers to the number of entities currently waiting. *Throughput* measures the number of completed units produced per unit of time, whereas *resource utilization* expresses the proportion of time a resource is busy relative to the total observation period. The *system loss rate* captures the rate at which entities depart the system without successfully completing their intended processing [46]. Figure 2.4 visualizes these core temporal relationships within a single-server queuing system. It depicts how entities arrive over time, may experience a delay before service begins, and how service completions determine periods of server activity and idle time. The inter-arrival times, delay times, and idle times shown in the diagram correspond directly to the performance measures used in the analysis of queuing systems.

Communication Simulation Frameworks A variety of communication simulation frameworks exist to model and analyze the behavior of networked systems at different levels of abstraction. Surveys by Vogt et al. [47] and Mets et al. [48] provide compre-

hensive overviews of tools used for both standalone communication-network simulation and co-simulation with power systems. Widely used frameworks include *ns-2/ns-3*, *OMNeT++*, *OPNET*, *GNS3*, and *NetSim*, each offering distinct modeling capabilities ranging from packet-level simulation to emulation and protocol evaluation. In practice, *ns-2/ns-3*, *OMNeT++*, and *OPNET* have emerged as the most common choices for research on communication networks and CPESs [20].

2.1.5 Co-Simulations

In CPESs, multiple models of different sub-systems (often based on different modeling paradigms) interact with each other [38]. Whereas classical simulation (as depicted in Figure 2.5) relies on a single model and a single solver, more advanced approaches differentiate between parallel simulation, in which one model is solved by several solvers in parallel, and hybrid simulation, which combines different modeling languages but still relies on a single solver [23]. *Co-simulation* extends these concepts by coupling multiple models with multiple solvers and coordinating their execution through a master algorithm. This coordination becomes particularly challenging in CPESs, as power systems are typically represented by continuous-time models, while communication networks are modeled using discrete-event simulation. Consequently, appropriate mechanisms for time synchronization and data exchange are required. Common synchronization strategies include master–slave coordination, time-step synchronization, and event-driven synchronization [20]. Vogt et al. [47] provide an overview of many CPES co-simulation frameworks.

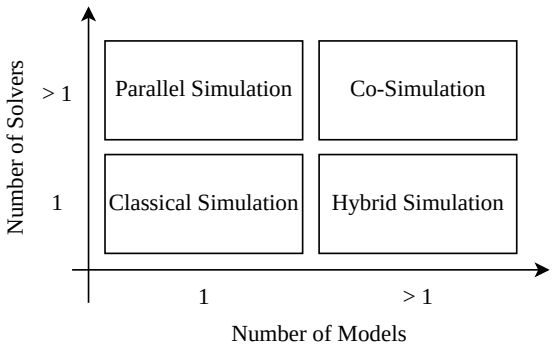


Figure 2.5: Comparison of simulation approaches: single-model simulations, parallel simulations with multiple solvers, hybrid simulations combining modeling paradigms, and co-simulations coordinating multiple models and solvers (adapted from Figure 5 in [23]).

2.2 Meta-Models

Computer simulations play a vital role in engineering, enabling the analysis and design of complex systems. However, detailed simulations that resolve fine-grained physical or cyber-physical processes are often computationally expensive and require substantial hardware resources and runtime [49]. This is especially challenging when many simulation runs are required, such as in design space exploration, parameter studies, or uncertainty analyses.

Meta-models, also known as surrogate models, are a solution to this challenge. They approximate the input-output behavior of detailed simulation models at a significantly lower computational cost [32, 50]. Conceptually, they act as "models of models". Rather than simulating the full internal dynamics of the underlying system, a meta-model directly predicts the output variables of interest as a function of selected input variables [32]. Once trained on data generated by the original simulation, such surrogates can be evaluated orders of magnitude faster. This enables tasks that would otherwise be infeasible, such as complex optimization, larger-scale simulation studies, or real-time performance prediction [32, 49].

Data-driven approaches are particularly prominent in this context. In this case, machine learning models serve as meta-models, learning the mapping from simulation inputs to outputs purely from examples, without explicit knowledge of the underlying equations. This approach enables flexible and efficient surrogate modeling, provided that representative training data is available. The idea of approximating complex simulation models with simpler representations is not new and can be traced back at least to early work on decision-tree-based transient stability assessment in power systems [51]. Meanwhile, machine learning-based surrogates have emerged as a valuable tool for accelerating the process of system dynamics and market processes [52]. These meta-models effectively substitute time-consuming simulations, providing rapid approximations that save significant time and resources [53]. In this work, meta-models are defined as follows:

Definition 7 (Meta-models)**.** *Meta-models are computationally inexpensive, data-driven approximation models trained on evaluations of a more detailed high-fidelity simulation to reproduce its input–output behavior.*

A central challenge in constructing meta-models is the amount and quality of training data required to achieve sufficient accuracy across the relevant operating space. Generating such data from detailed simulations may itself be costly. To alleviate this, Qian et al. [50] propose a two-phase learning approach: an initial, less detailed pretraining phase on cheaper data, followed by a refinement phase using more detailed simulations. This illustrates a general design trade-off between simulation fidelity, training cost, and surrogate accuracy that is also relevant for the meta-models considered in this work.

2.2.1 Types of Meta-Models

A variety of model classes can be used as meta-models, ranging from simple linear predictors to complex non-linear learners. The selection of a model depends on various factors, including the dimensionality of the input space, the amount of available data, requirements for interpretability, and the anticipated complexity of the underlying relationships.

Linear regression models approximate the relationship between a dependent variable and a set of independent variables by fitting a linear function. Despite their simplicity, they provide a useful baseline and are easy to interpret and analyze [53, 54]. When the true dependency is close to linear, they can offer accurate and robust predictions at minimal computational cost.

Support Vector Regression extends Support Vector Machines to regression tasks by constructing a function that deviates from the training data by at most a specified margin while maintaining maximum flatness. Through kernel functions, Support Vector Regression (SVR) can effectively capture non-linear relationships by mapping the input space into a higher-dimensional feature space, where a linear model is fitted [32]. This makes SVR a popular choice for surrogate modeling in settings with moderate data sizes and complex, but smooth, dependencies.

Artificial neural networks are flexible, layered models capable of representing highly non-linear mappings. By composing multiple affine transformations with non-linear activation functions, they can approximate complex functional relationships given sufficient network capacity and training data. Neural networks have been successfully applied as surrogates in various engineering domains, including energy systems, where they can replace detailed simulations by fast forward evaluations once trained [53]. Their main advantages are expressiveness and scalability to large datasets, while their drawbacks include higher training cost and reduced interpretability.

Decision trees partition the input space into regions via a sequence of hierarchical, axis-aligned splits and assign a (typically constant) prediction to each region. They are simple to implement, handle mixed data types, and offer high interpretability, as the decision path for any prediction can be inspected directly [54]. However, single trees tend to overfit and may yield unstable predictions under small changes in the training data.

Random forests mitigate some of the limitations of single decision trees by constructing an ensemble of trees, each trained on a bootstrap sample of the data and often with random feature sub-sampling at each split. The final prediction is obtained by averaging the individual tree outputs. This ensemble strategy yields models that are typically more robust, less prone to overfitting, and well-suited for high-dimensional feature spaces [53]. As a result, random forests are frequently used as off-the-shelf meta-models when non-linear

effects are expected and interpretability at the level of individual splits is not critical.

2.2.2 Online Learning

As will be discussed in more detail in Chapter 3, communication scenarios in CPESs exhibit highly diverse characteristics. Traffic patterns, network topologies, and operating conditions may differ substantially between scenarios and can change over time. Consequently, a meta-model that is trained once on a fixed dataset may fail to generalize to all relevant situations. In this thesis, the meta-model is therefore adapted *online*[1] during simulation to capture scenario-specific characteristics and to maintain accuracy as new conditions occur.

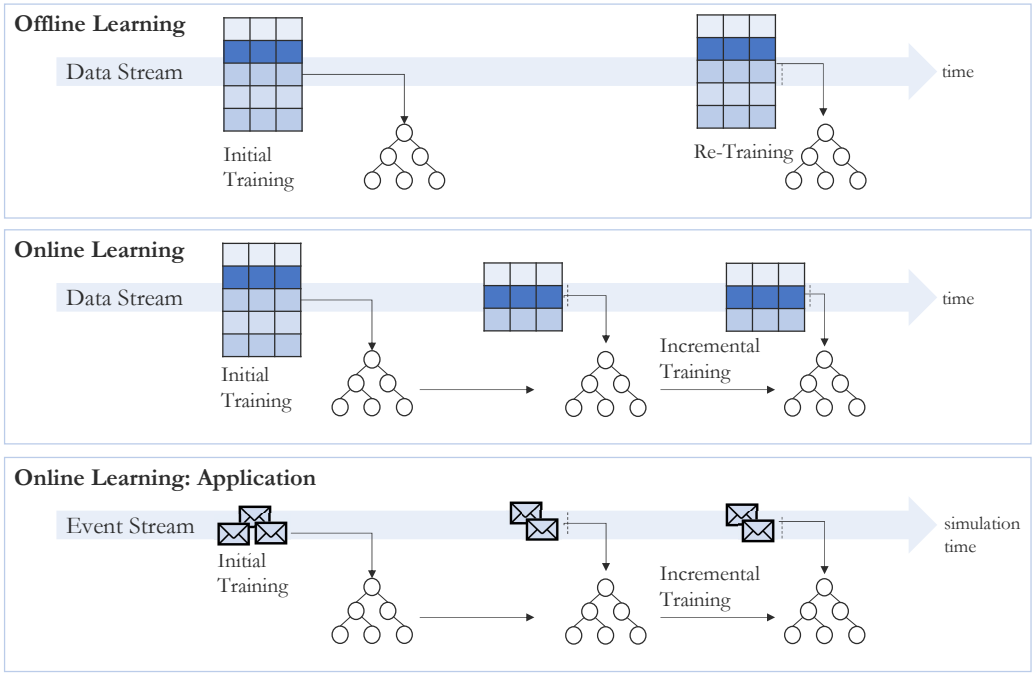


Figure 2.6: Comparison of offline and online learning paradigms: Offline learning trains a model once on a fixed dataset and requires retraining for updates, while online learning adapts continuously to incoming data streams.

Online learning has been studied extensively in the context of streaming data and big-data applications [55]. Streaming data that are generated continuously, are often only observed once, and may exhibit unpredictable and sometimes abrupt changes [55]. Such

[1] For a historical overview on the spelling of *online* and *on-line*, please refer to Section A.

concept drift (structural changes in the data-generating process) poses a challenge for conventional machine learning algorithms, which typically assume that training and test data are identically distributed [55]. Similar issues arise in CPESs simulations, where not all relevant events or operating conditions are observable during an initial training phase. Dynamic incremental learning allows a model to adapt to previously unseen events and operating regimes, thereby mitigating degradation in prediction quality and avoiding catastrophic forgetting when new scenarios are encountered [36].

Figure 2.6 summarizes the main differences between offline and online learning paradigms. In *offline* (or batch) learning, a model is trained on an entire, fixed dataset. This approach is well suited when the data distribution is stationary and the complete dataset is available in advance, but it becomes impractical for data streams due to memory limitations and the inability to react to drift [55]. Every substantial change in the data would require collecting a new dataset and retraining the model from scratch.

In contrast, *online* (or incremental) learning updates the model continuously as new data points arrive. Updates can be performed sample by sample or on small mini-batches and are often combined with mechanisms such as sliding windows that focus the model on the most recent observations [55]. This enables the learner to track gradual changes in the data distribution and to adapt efficiently without full retraining. Based on these explanations, online learning is defined as:

Definition 8 (Online learning)**.** *Online learning is a machine learning paradigm in which a model is updated incrementally as new data arrives, instead of being trained once on a fixed, static dataset.*

To cope with concept drift (e.g., in the context of this thesis changes in communication delay characteristics caused by network reconfiguration or varying traffic loads during a CPES simulation), various mechanisms have been proposed. Change-detection methods monitor the data stream or the model's predictive performance and trigger retraining, model updates, or a reset once a significant shift is detected [55]. Ensemble-based approaches maintain multiple models in parallel and compare their predictions [55]. These approaches will be presented in more detail in Subsection 2.2.3.

In this thesis, online learning is used to adapt meta-models of communication behavior during the course of a simulation. Here, the data stream corresponds to events occurring in simulation time (cf. Figure 2.6). New observations of communication network performance are incorporated into the meta-model incrementally, allowing it to remain accurate across heterogeneous and evolving communication scenarios without requiring the full cost of repeated high-fidelity simulations.

2.2.3 Ensemble Learning

Ensemble learning combines multiple predictors (classifiers or regressors) into a single model with the aim of achieving better predictive performance than any individual learner alone [56, 57]. This principle is often described as the "wisdom of the crowd": aggregated predictions from a diverse set of models tend to be more accurate and robust than those of a single model, provided that suitable mechanisms for combining their outputs are in place [56]. Beyond improved accuracy, ensembles can reduce generalization error, increase robustness to noise, and stabilize predictions in supervised learning tasks such as regression and classification [57]. An overview of the ensemble types considered in this thesis is given in Figure 2.7.

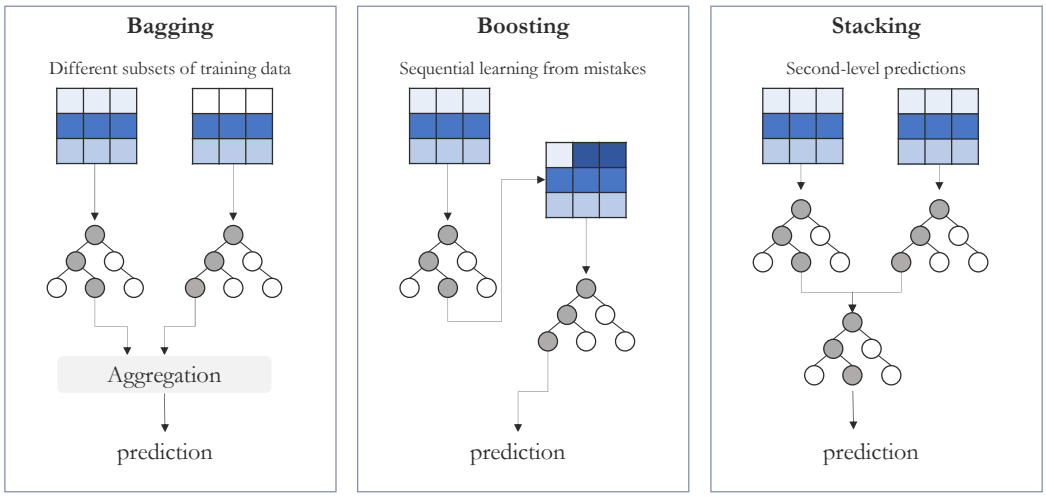


Figure 2.7: Overview of ensemble learning types: Bagging trains classifiers on different data subsets and aggregates predictions, Boosting builds sequential models correcting prior errors, and Stacking combines outputs of base classifiers with a meta-classifier.

Decision trees are frequently used as base learners in ensemble methods due to their relatively short training times, ability to handle mixed-type input features, and high inter-

pretability [56]. Regression trees extend decision trees to continuous targets by combining the tree structure with local regression models in the leaves. Overall, tree-based learners provide a favorable trade-off between predictive performance, computational cost, and interpretability, and they serve as a natural building block for many ensemble techniques.

In online learning scenarios, the distribution of the target variable and the underlying data-generating process may change over time. Such concept drift leads to a reduction in prediction accuracy if models are not adapted accordingly [56]. To cope with drift, algorithms must react quickly to changes, for example by comparing class or target distributions in older and more recent data, maintaining a forest of trees instead of a single model, or dynamically updating the ensemble. Dynamic approaches such as the Dynamic Weighted Majority (DWM) maintain a pool of base learners whose weights are adjusted according to their current performance; poorly performing trees can be removed and new ones added when the data distribution changes [56]. For ensembles to function effectively as "wise crowds", they should exhibit diversity of opinion, independence of individual learners, decentralization (each learner exploiting different information), and an appropriate aggregation mechanism.

Ensemble methods exploit the fact that different learners have different inductive biases and error profiles. By combining their predictions, overall robustness and accuracy can be improved [57]. Figure 2.7 illustrates three common families of ensemble techniques [58].

In *bagging* (left panel in Figure 2.7), multiple base learners are constructed by training each one on a different bootstrap sample of the training set (sampling with replacement). Predictions are then aggregated, typically by averaging (for regression) or majority voting (for classification). Random Forests are a prominent bagging-based ensemble of decision trees, additionally introducing randomness in feature selection at each split.

In *boosting* (center panel in Figure 2.7), base learners are constructed sequentially, with each new learner focusing on the instances that were incorrectly predicted by its predecessors. Errors of earlier models are emphasized by reweighting training samples, leading subsequent learners to correct them.

In *stacking* (right panel in Figure 2.7), the outputs of several base learners are used as inputs to a second-level model, the meta-learner. This meta-learner is trained to exploit the complementary strengths of the base models and to correct their systematic biases. Stacking is particularly useful when base learners are heterogeneous and capture different aspects of the data.

2.3 Communication in CPES

The operation of CPESs relies on a tight coupling between physical power system processes and the supporting communication infrastructure. Measurements, control signals, and market interactions are transported over heterogeneous networks that span customer premises, distribution grids, and wide-area backbones. The characteristics of these networks directly influence the stability, efficiency, and resilience of the overall system.

This section provides an overview of the communication aspects that are most relevant for this thesis. First, in Subsection 2.3.1 the main network types and technologies used in CPESs are outlined, emphasizing their hierarchical organization and typical deployment contexts. Next, the quantitative and qualitative requirements imposed by CPES applications are summarized in Subsection 2.3.2 and linked to standard performance indicators such as delay, jitter, throughput, and loss in Subsection 2.3.3. Finally, representative communication applications across power system domains are introduced in Subsection 2.3.4, forming the basis for the scenario-specific traffic models discussed later in Chapter 3.

2.3.1 Network Types and Technologies

Communication infrastructures in CPESs are typically organized in a hierarchical manner as displayed in Figure 2.8, reflecting the structure of the underlying power system. Many works provide detailed surveys of the corresponding network types and suitable communication technologies, including their advantages and limitations [48, 59, 60, 61, 62]. In the following, only a concise overview is given, focusing on the characteristics that are most relevant for this thesis.

Wide Area Networks (WANs) connect control centers, primary substations, and other critical infrastructure across medium- and high-voltage levels. They form the high-capacity backbone of the smart grid and must support long-distance data transmission with high reliability and low latency for applications such as Supervisory Control and Data Acquisition (SCADA), and wide-area monitoring. Typical coverage ranges from 10-100 km, with data rates from 10 Megabit per second (Mbps) up to the 1 Gigabit per second (Gbps) range [48, 59, 63]. Representative technologies include fiber-based Ethernet and optical transport networks, Multi-Protocol Label Switching (MPLS) backbones, as well as cellular and wireless solutions like WiMAX, 4G/5G, or satellite links in areas where wired infrastructure is not available [63].

Field Area Networks (FANs) and *Neighborhood Area Networks (NANs)*, often implemented as part of the Advanced Metering Infrastructure (AMI), provide communication within the distribution grid. They aggregate data from smart meters, protection and automation devices, and local controllers, and bridge customer premises networks with the

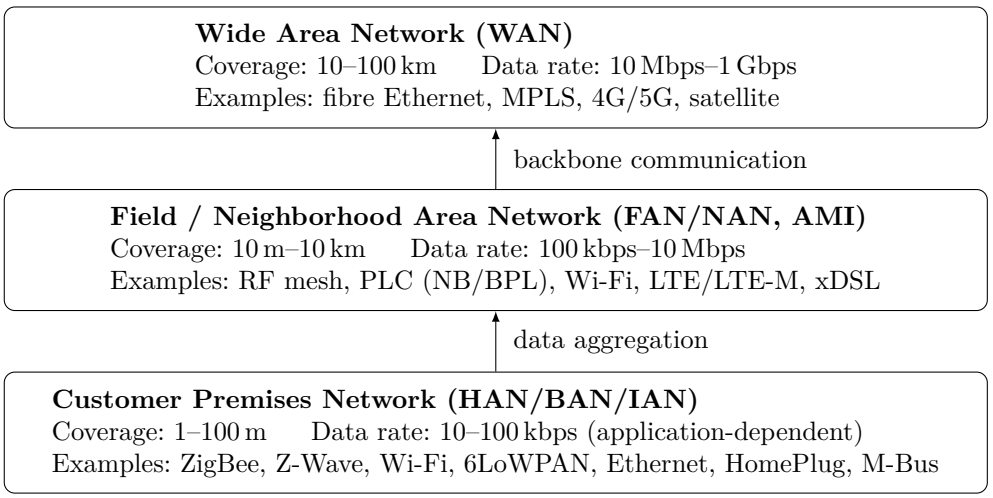


Figure 2.8: Network hierarchy in CPESs with characteristic coverage, data rate, and technology choices.

utility WAN. Coverage typically spans from a few tens of meters up to several kilometers (10 m-10 km), while required data rates lie between roughly 100 kbps and 10 Mbps, depending strongly on node density, topology, and application mix [48, 59, 63]. Common technologies include wireless Radio Frequency (RF) mesh networks, ZigBee [64] or Wi-Fi [65], cellular standards such as Long Term Evolution (LTE)/LTE-M [66] and dedicated LTE450 solutions [67] for rural areas, as well as wired Power Line Communication (PLC) [68] and xDigital Subscriber Line (DSL) on existing copper infrastructure [59, 63].

At the lowest level, *customer premises networks* connect smart meters, appliances, and local energy management systems within homes, buildings, or industrial facilities. These networks, typically referred to as *Home Area Networks (HANs)*, *Building Area Networks (BANs)*, or *Industrial Area Networks (IANs)*, cover distances from approximately 1 m to 100 m and usually require only low to moderate data rates on the order of 10-100 kbps, depending on the specific application [63]. They predominantly rely on low-power wireless technologies such as ZigBee [64], Z-Wave, Wi-Fi, or IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) [64], complemented by wired options like Ethernet [69], HomePlug, or M-Bus [59, 60, 61].

Across these network layers, three main categories of physical media are used: copper, fiber optics, and wireless links. Copper-based media remain widespread due to their low cost and ease of installation, and they underpin technologies such as PLC and xDSL [59]. Fiber-optic cables offer virtually unlimited bandwidth, immunity to electromagnetic interference, and high security, making them the preferred choice for backbone WANs and other critical links [59]. Wireless technologies transmit information via electromagnetic waves and enable

flexible deployment, especially in geographically challenging areas or where laying cables is not economical. They encompass short-range solutions for customer premises networks (e.g., ZigBee, Wi-Fi, Z-Wave) as well as wide-area systems (e.g., cellular, WiMAX, satellite) used in FANs, NANs, and WANs [59, 63].

2.3.2 Communication Requirements

The communication infrastructure of CPESs must support a wide range of applications, from real-time protection and automation to advanced metering, demand response, and market-related services. Consequently, numerous reports and survey papers have analyzed the communication requirements of smart grid applications and have proposed corresponding design targets for the underlying networks [17, 59, 60]. Following Parizad et al. [17], these requirements can be grouped into *quantitative* and *qualitative* categories. Quantitative requirements specify performance levels that the communication system must provide (e.g. latency, data rate, reliability), whereas qualitative requirements describe capabilities and functional properties such as scalability, interoperability, and security.

Quantitative Requirements *Delay (or latency)* requirements concern the time needed to transfer data from a source to a destination. Many smart grid applications are time-critical and involve (near) real-time data exchange, for example in protection schemes, voltage and frequency control, or distribution automation [59, 60]. For such applications, the communication system must guarantee sufficiently low end-to-end delay and jitter to ensure correct and timely operation.

In this thesis, delay is understood as the *end-to-end delay* experienced by a message along its communication path. For a packet that traverses several intermediate nodes, this delay is composed of multiple contributions at each hop:

- *Processing delay*, caused by protocol processing and packet handling at routers, switches, and gateways.
- *Transmission delay*, i.e. the time required to push all bits of the packet onto the link, determined by packet size and link data rate.
- *Propagation delay*, i.e. the time for the signal to travel across the physical medium (copper, fiber, or wireless), determined by distance and propagation speed.
- *Queuing delay*, incurred when packets wait in buffers due to contention for shared resources or transient congestion.

Here, a packet denotes a discrete unit of data into which a message is divided for transmission over a communication network, including protocol headers and payload, and forwarded independently along the communication path.

Data rate requirements specify the amount of data that can be transmitted within a given time interval. The heterogeneous nature of CPES applications leads to highly diverse data rate demands: protection signals or control setpoints typically require only small payloads, whereas Phasor Measurement Units (PMUs), high-resolution monitoring, or firmware updates entail much higher throughput [59, 60]. The communication infrastructure must therefore accommodate both low-volume, highly time-critical traffic and more bandwidth-intensive, but less delay-sensitive, services.

Reliability refers to the ability of the communication system to remain available and to deliver messages with a sufficiently low probability of loss. Since many energy system functions are mission-critical, the underlying networks must provide high availability, fault tolerance, and robust error handling so that information is delivered even in the presence of disturbances or partial failures [59, 60].

Qualitative Requirements *Scalability* is essential because millions of additional devices, such as smart meters, distributed energy resources, and controllable loads, are expected to be connected to the grid. The communication system must be capable of accommodating this rapid growth without disproportionate degradation of performance or excessive operational complexity [17, 60].

Interoperability is required to ensure that devices and applications using different communication standards and technologies can exchange information seamlessly. Given the coexistence of legacy systems, vendor-specific solutions, and emerging standards, interoperable interfaces and protocols are crucial for avoiding vendor lock-in and enabling end-to-end functionality across heterogeneous networks [17, 60].

Security requirements address both cyber security and privacy. Communication networks in CPESs transport grid-critical information as well as data related to consumer behavior and energy usage. They must therefore be resilient against a broad spectrum of attacks, including eavesdropping, manipulation, denial of service, and unauthorized access, while also protecting customers' privacy [17, 59, 60]. This implies the need for authentication, integrity protection, confidentiality, and appropriate access control mechanisms.

2.3.3 Communication Network Performance Indicators

The quantitative requirements outlined above are typically expressed in terms of measurable performance indicators. A stream of packets from a source to a destination is referred to

as a *flow*, and the needs of each flow can be characterized by four primary parameters: bandwidth, delay, jitter, and loss [70]. Together, these parameters determine the Quality of Service (QoS) provided to that flow. In addition, throughput is often considered as an operational measure of achieved performance.

- *Bandwidth:* The maximum rate at which data can be transmitted over a communication channel, typically measured in bit per second (bps). It represents the physical capacity of the link.

- *Throughput:* The actual data rate achieved by a flow, often lower than the available bandwidth due to protocol overhead, congestion, or retransmissions.

- *Delay (latency):* The time taken for a packet to travel from the source to the destination. In line with Subsection 2.3.2, this end-to-end delay comprises processing, transmission, propagation, and queuing delays accumulated along the path [71]. It is a key performance measure for time-critical and interactive applications.

- *Jitter:* The variation in packet delay or inter-arrival time, often quantified as the standard deviation of the delay. High jitter can degrade the performance of real-time applications such as voice or video streaming.

- *Packet loss:* The fraction of transmitted packets that fail to reach the destination. Causes include buffer overflows, transmission errors, or routing failures. Packet loss directly affects reliability and perceived quality, especially for protocols without loss concealment or retransmission.

2.3.4 Communication Applications

A broad range of communication applications in CPESs is surveyed in [62]. These applications are grouped into the major domains of the electric power system (generation, transmission, distribution, and customer) each with characteristic requirements regarding reliability, delay, data rate, and coverage.

In the author's own work [1], this classification is used to derive so-called *simple traffic models*, which capture typical communication patterns of selected applications and will be discussed in detail in Chapter 3. Here, Table 2.1 provides a concise overview of representative application categories and example use cases per power system domain.

Table 2.1: Communication applications mapped to power system domains (adapted from Figure 2.3 in [62]).

Power System Domain	Application Category	Example Application
Generation & Transmission	Wide Area Monitoring	State Estimation
	Wide Area Control	Wide Area Voltage Stability
	Wide Area Protection	Predictive Load Shedding
Distribution	Automation and Management	Meter Reading
	Customer Interaction & Pricing	Customer Information & Messaging
	Emerging Applications	Premises Network Administration
Customer	Automation	Home Automation

2.4 Communication Modeling Approaches

The previous sections discussed that communication infrastructures substantially influence the performance and robustness of CPESs. Depending on the study objective, different levels of abstraction are appropriate when representing the communication system, ranging from simple delay assumptions to detailed packet-level simulations. This section summarizes available modeling approaches, discusses their suitability for typical use cases, and motivates the development of meta-model-based approaches as an intermediate solution between analytical models and detailed simulations.

In an own, accompanying publication [7], a systematic literature review was conducted to analyze how communication networks are modeled in energy-system-related studies. The method used in this publication is illustrated in Figure 2.9. Based on 60 representative publications, communication applications were grouped by network type (premises networks, NANs, WANs, refer to Section 2.3 for details), different system perspectives were identified (component-specific, dyadic, holistic), and seven recurring objectives for jointly considering power and communication systems were derived (e.g., investigating functionality under communication influence, design optimization of the communication network, evaluation of real systems). The review then classified the concrete communication modeling approaches used and mapped them to these objectives and perspectives. The underlying database of this review was published by the author in the Open Research Knowledge Graph (ORKG) [12].

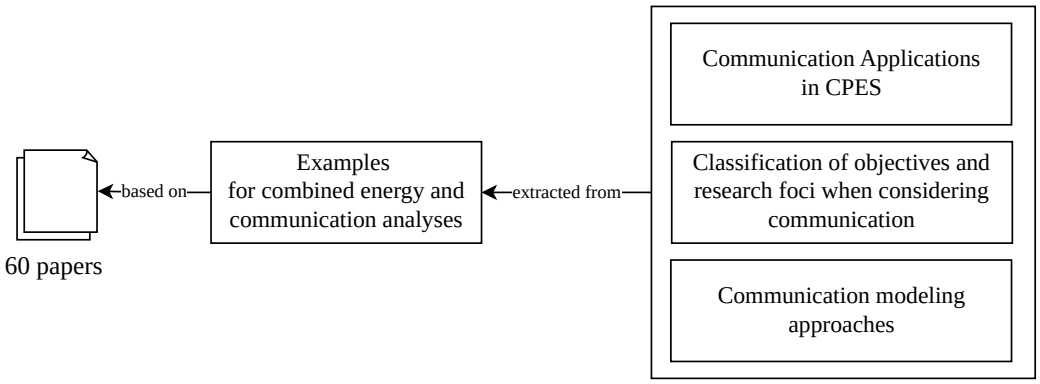


Figure 2.9: Illustration of the methodological approach for extracting communication modeling approaches from representative publications.

Starting from a (communication) system, its behavior can be studied either by direct experimentation with the real system or, more commonly in CPESs, by experimentation with a model (cf. Subsection 2.1.3). In the following, the focus will be on mathematical models, including *analytical models* and *(detailed) simulations*. Analytical models symbolically represent the equations to obtain closed-form results or bounds, while simulations numerically evaluate the same equations by executing the model in (simulated) time.

Under the umbrella of analytical models, several approaches can be distinguished. *Network calculus* provides deterministic bounds on delay and backlog based on abstract traffic and service curves and is used when worst-case guarantees are of primary interest. At a more application-oriented level, many energy-system studies represent communication effects by a direct *integration of performance KPIs*, where fixed or parameterized end-to-end delays, loss probabilities, or bandwidth constraints are attached to message exchanges in the power system model. More refined analytical descriptions include *queuing models*, which capture congestion effects in terms of arrival and service processes, and *graph-based models*, which represent the network as a graph whose edges are annotated with aggregate performance metrics. *Channel models* focus on the physical or link layer and describe error processes, fading, and time-varying delay on individual links. These analytical approaches are computationally inexpensive and comparatively easy to integrate into power system tools, but they rely on simplifying assumptions and typically do not capture the full protocol dynamics of realistic communication infrastructures.

On the simulation branch, *detailed simulations* of communication networks are implemented, for example, using frameworks such as *OMNeT++* or *ns-3* (cf. paragraph 2.1.4). They explicitly model packets, protocol stacks, queues, and scheduling mechanisms and can

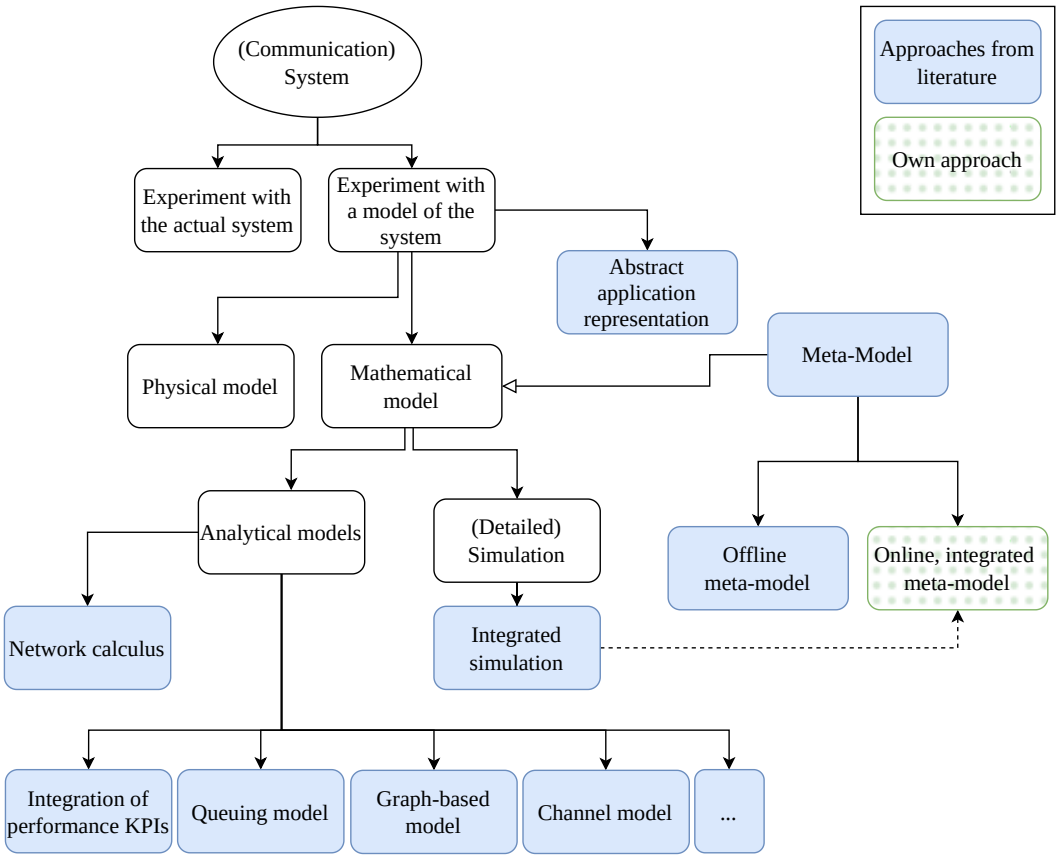


Figure 2.10: Conceptual classification of communication modeling approaches for CPESs. Approaches shown in blue are commonly used in the literature, while the dotted, green box highlights the online, integrated meta-model developed in this thesis (adapted from Figure 4 in own publication [7]).

represent heterogeneous technologies, traffic patterns, and dynamic network states. When coupled with power system simulations, they form an *integrated simulation* of the joint cyber-physical system. This allows for high-fidelity investigations of mutual dependencies between power and communication networks but entails substantial modeling effort, high computational cost, and additional complexity in co-simulation.

In parallel to these branches, Figure 2.10 shows two further abstraction layers. First, *abstract application representations* operate purely at the application layer and directly prescribe how messages are delayed, dropped, or manipulated without explicitly modeling the underlying network. They are useful when the focus is on application behavior (e.g., under cyber attacks) rather than on network mechanisms.

Second, *meta-models* act as surrogates for detailed simulations. They are trained to reproduce the input–output behavior of a more detailed model and can subsequently be evaluated much faster. Existing work in the communication networking community has predominantly considered *offline* meta-models, trained once on large datasets generated by detailed simulations. In this thesis, this idea is extended to an *online, integrated meta-model* (dotted, green box in Figure 2.10), which is integrated into the (power system or control) simulation and adapted during runtime.

In summary, the approaches in Figure 2.10 span a spectrum from simple analytical models with low computational cost but limited realism to detailed simulations with high fidelity but substantial computational and modeling effort. The meta-model concept aims at finding a trade-off in this spectrum, combining efficiency with sufficient accuracy for CPES studies.

Meta-Models for Communication Network Simulation As introduced in Section 2.2, meta-models approximate the input-output behavior of a high-fidelity model at a significantly lower computational cost. For communication networks, they typically predict performance indicators such as end-to-end delay, jitter, packet loss, or throughput based on a description of the network topology, traffic pattern, and protocol configuration. Within the classification of Figure 2.10, such models occupy the "Meta-Model" branch, with *offline meta-models* representing the state of the art and the *online, integrated meta-model* constituting the contribution of this thesis.

In existing approaches on offline meta-models, a detailed communication simulator is used to generate large training datasets for a variety of synthetic topologies and traffic scenarios, and supervised learning methods are employed to learn the mapping from scenario descriptors to performance metrics. Once trained, the meta-model replaces repeated simulator runs and can be queried directly within optimization or design loops. Table 2.2 summarizes a representative set of such contributions. While these works demonstrate that meta-models can effectively approximate detailed communication simulations, they also reveal three central limitations from a CPES perspective:

1. **Synthetic traffic models.** The meta-models are trained almost exclusively on synthetic traffic patterns (Poisson, on-off, CBR, etc.). These models are standard in networking research but do not adequately represent the heterogeneous, application-driven traffic found in CPESs, as will be detailed in Chapter 3.

2. **Expensive offline training.** Generating the required training datasets requires running large numbers of detailed simulations. This upfront cost is substantial and must

be repeated whenever the scenario, protocol stack, or relevant operating conditions change, which limits practicality for many CPES studies.

3. **Black-box integration.** Existing meta-models are typically used as external black-box predictors. They are queried for performance values but are not integrated as interpretable, co-evolving components of a joint power–communication simulation. This hinders the systematic analysis of mutual dependencies between both domains.

Table 2.2: Summary of related work on communication network simulation meta-models.

Reference	Network Model	Size	Traffic Model
[33]	Ring, star, scale-free topologies	3-15 nodes	Randomized intensities based on link capacities
[34]	Two real-world and multiple synthetic topologies	Up to 300 nodes	Poisson, on-off, constant bit-rate, auto-correlated and modulated exponentials, mixed
[35]	Power-law generated 5G network topologies	25-300 nodes	Exponential, deterministic, uniform, normal, on-off, PPBP
[72]	Three real-world topologies	25-300 nodes	Exponential, deterministic, uniform, normal, on-off, PPBP
[73]	Real-world topologies with 5G	Not specified	Fixed scenarios
[74]	Not specified	Not specified	Military logic, parameterized
[75]	Real-world street topologies with VANET	100 nodes	Random

To address these limitations, this thesis proposes the *online, integrated meta-model*, which is highlighted with a green, dotted box in Figure 2.10. Instead of relying on a purely offline training phase, the meta-model is updated during the course of a simulation using online learning techniques (Subsection 2.2.2). New observations of communication performance, generated by the simulation itself, are incorporated incrementally so that the model adapts to the actual, heterogeneous traffic patterns occurring in CPES scenarios. At the same time, the meta-model is embedded into the simulation loop as an integral component rather than

an external predictor, enabling a more transparent analysis of the interactions between power and communication systems.

2.5 Synthesis of Gaps & Positioning of this Work

The previous sections have introduced the methodological foundations (simulation and co-simulation), meta-modeling and online learning, as well as the characteristics of communication networks and their modeling in CPESs. Taken together, these fundamentals reveal both opportunities and challenges for representing communication behavior in energy system studies and motivate the research direction of this thesis.

The Figure 2.11 summarizes these aspects in terms of opportunities, challenges, and possible solution directions. From the simulation perspective (Section 2.1), discrete-event characteristics are essential for seamless integration of communication models into co-simulation environments, and combined energy–communication simulations are required to investigate mutual inter-dependencies in CPESs. Concurrently, detailed communication simulation can become a computational bottleneck in many-run studies with long horizons, multiple scenarios, or extensive parameter space exploration.

From the meta-modeling perspective (Section 2.2), surrogate models offer a way to reduce runtime by approximating the input–output behavior of detailed simulations. However, existing communication meta-models typically require large, costly training datasets and lack flexibility when new conditions occur. They are predominantly trained offline and used as external black-box predictors, which limits their suitability for dynamic, scenario-rich CPES analyses.

From the machine learning perspective (Subsection 2.2.2 and Subsection 2.2.3), online learning naturally copes with concept drift and streaming data, and ensemble methods provide robustness, drift handling, and "wise crowd" behavior via dynamic weighting. These concepts point to promising mechanisms for meta-models that must adapt to evolving communication conditions during simulation rather than being fixed after an offline training phase.

From the communication-systems perspective (Section 2.3), communication models must be sufficiently technology-agnostic and topology-aware to cover the diversity of CPES networks. Application diversity across power system domains leads to heterogeneous traffic patterns and performance requirements, which raises the bar for any surrogate that aims to replace detailed simulation: it must handle a wide range of traffic models and operating regimes while still providing meaningful performance indicators.

Finally, the review of communication modeling approaches (Section 2.4) highlights a systematic trade-off between realism and computational effort. Simple analytical abstractions

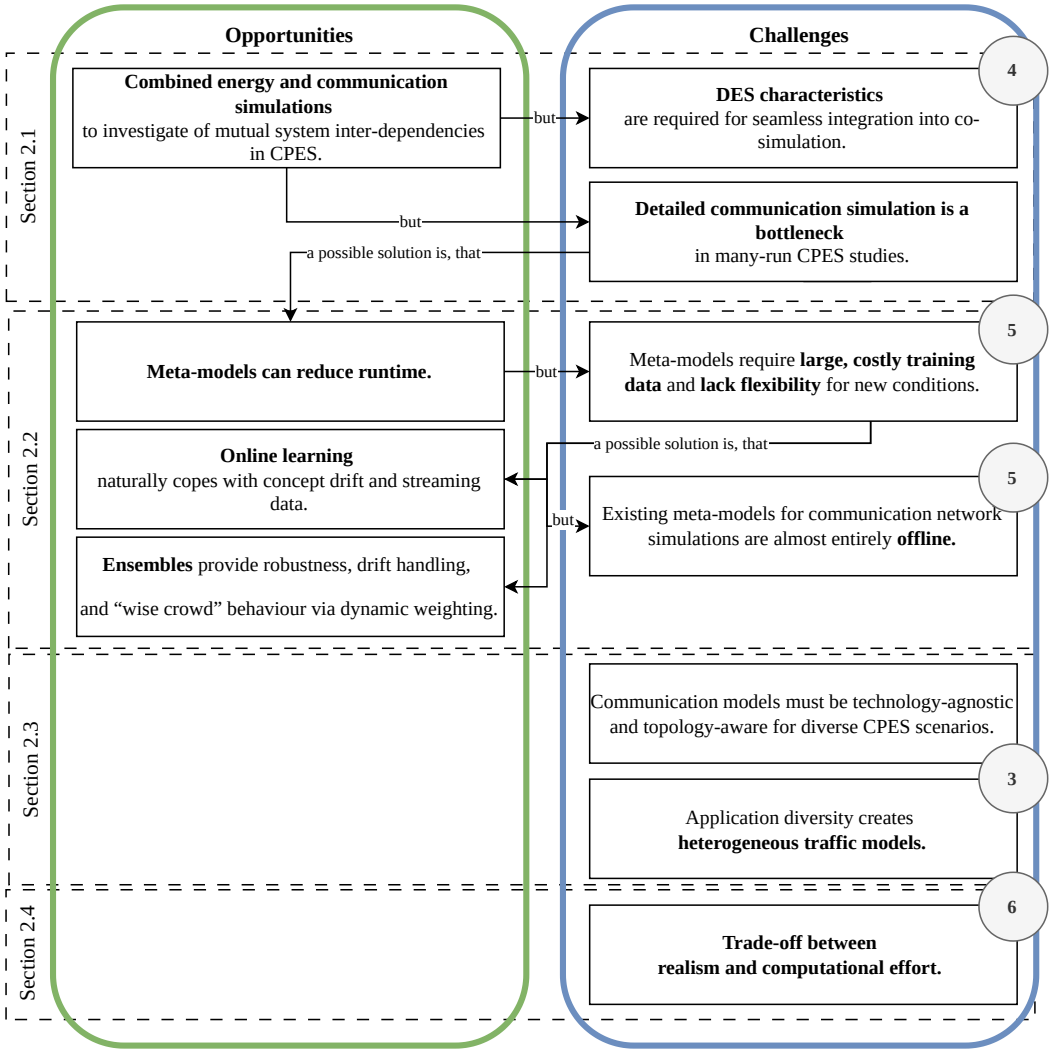


Figure 2.11: Synthesis of opportunities (in the left, green box) and challenges (in the right, blue box) for communication modeling in CPESs. In the illustration, the relevant sections from the fundamentals are marked with numbers on the left, while the chapters in which the points are mainly addressed are marked with gray circles on the right.

and KPI-based models are easy to integrate but risk missing important dynamic effects, while detailed DES-based simulators provide high fidelity at the expense of runtime and modeling complexity. Existing offline meta-models sit between these extremes but are not yet tailored to the specific requirements of CPESs and co-simulation.

Overall, the central research gap addressed in this thesis can be summarized as follows:

- There is a need for communication models that preserve essential DES characteristics and can be integrated into co-simulation frameworks for CPESs, without incurring the full computational cost of detailed packet-level simulation.
- Existing meta-models for communication network simulations are not directly applicable to CPESs, since they rely on synthetic traffic models, require extensive offline training, and act as non-integrated black-box predictors.

In response, this work develops an *online trained, integrated meta-model* for communication networks in CPESs. The model is designed as a DES-compatible component, employs online learning and ensemble techniques to adapt to heterogeneous and evolving traffic conditions, and aims to balance realism and computational efficiency in large-scale energy system studies.

3 Communication Scenarios in Energy Systems

> Great communication begins with connection.
>
> *Oprah Winfrey*

The primary objective of this work is to develop an online meta-modeling approach to approximate communication simulations in the context of CPESs. The design and evaluation of such an approach depends on a solid understanding of how communication in CPESs actually occurs in practice. The way the communication happens determines the relevant input features and the range of scenarios for which the meta-model must make accurate predictions. Consequently, this chapter addresses Sub-RQ 1 and therefore investigates how communication behavior in energy systems can be described, modeled, and simulated in a systematic and CPES-specific way.

Figure 3.1 provides an overview of how communication scenarios are generated. Starting at the top, the first step is to define how communication takes place in the scenario (*communication traffic model*), which is then combined with the communication network description (*communication network model*) to form a *communication scenario*. This scenario is then simulated in the coupled communication network simulator *OMNeT++* to automatically generate simulation data.

Communication in CPESs is driven by a wide variety of applications, ranging from periodic meter reading and customer information to time-critical wide-area control and complex, agent-based coordination schemes [1]. Each of these applications induces distinct temporal patterns, data volumes, communication modes, and organizational structures, and thereby imposes specific requirements on the communication network. To develop an automated scenario generation method, it is necessary to introduce precise terminology for the different building blocks of communication scenarios.

Definition 9 (Communication application)**.** *A communication application is a function or service in a CPES that relies on the exchange of information between distributed components in order to achieve a specific operational, control, or market-related objective.*

Definition 10 (Traffic model)**.** *A traffic model is an abstract representation of the communication behavior induced by a communication application. It specifies which entities exchange data, in which direction, with which temporal characteristics, and with which data volumes and communication modes.*

Definition 11 (Communication network model)**.** *A communication network model describes the structure and properties of the communication network in a CPES scenario. It includes the set of communication nodes, their interconnection topology, the assignment of communication technologies, and the relevant technology parameters.*

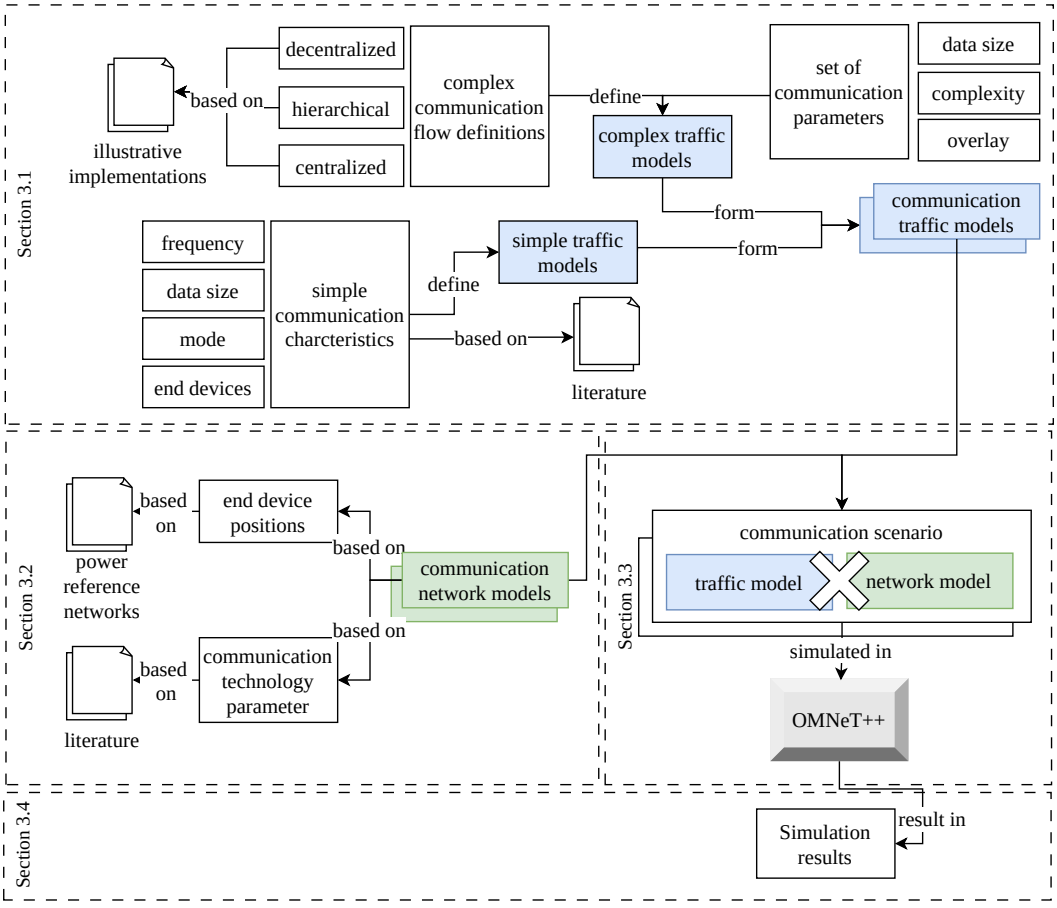


Figure 3.1: Overview of the data generation workflow for CPES-specific communication scenarios and simulation data used in this chapter.

In the scope of this work, a *communication scenario* is therefore understood as the combination of a traffic model and a communication network model, as illustrated in Figure 3.1.

This modular view allows communication behavior and communication infrastructure to be varied and combined systematically, which is essential for generating a diverse set of scenarios for simulation and subsequent meta-modeling.

Figure 3.1 summarizes the overall data generation workflow that is developed and used in this chapter. Starting from an abstract description of communication behavior, communication traffic models are derived (Section 3.1) and mapped onto a set of communication network models representing typical CPES environments (Section 3.2). The resulting communication scenarios are then simulated to generate data that reflect the diversity of communication behavior and network conditions relevant for CPES (Section 3.3). This simulation data forms the foundation for both the analysis of communication scenario diversity (Section 3.4) and the later training and validation of the proposed online meta-models.

The methodological foundations of the workflow presented in this chapter originate from the author's accompanying work. In particular, [9] introduces a generic description model for communication behavior in energy systems, which distinguishes between simple and complex traffic models and defines an automated process for generating representative communication scenarios and corresponding simulation data. Building on this contribution, the author further demonstrated in [1] how these scenarios can be systematically employed to analyze the impact of heterogeneous communication technologies and application characteristics on network performance. The present chapter consolidates, refines, and extends these results into a coherent scenario framework explicitly tailored to the requirements of online meta-modeling.

3.1 Communication Traffic Models

Starting with the upper part of Figure 3.1, the following describes how the communication traffic models are defined within the scope of this work.

Communication in CPESs results from a heterogeneous set of communication applications with differing objectives, time scales, and interaction patterns. Examples range from periodic meter reading and simple customer information services to wide-area protection schemes and multi-agent coordination in markets or distributed control [62]. Each of these applications induces characteristic temporal patterns, data volumes, and communication modes and therefore imposes specific requirements on the underlying communication network [1]. To study these requirements systematically and to generate representative input data for the online meta-modeling approach, the communication behavior of applications is abstracted into communication traffic models as defined in Definition 10.

Existing work on communication traffic modeling in the context of energy systems and related domains has largely focused on generic traffic abstractions and network-centric

studies. Typical approaches rely on stochastic traffic models such as Constant Bit-rate (CBR), Poisson, or on/off processes and apply them to synthetic or simplified network topologies in order to approximate communication load and evaluate network performance (cf. overview on related work in Table 2.2). While this is suitable for analyzing fundamental network phenomena, these models usually neglect the domain-specific characteristics of energy system applications and their interaction structures. Moreover, they often consider only a small number of fixed traffic configurations instead of systematically covering a broader space of application behaviors. In this work, communication traffic models are therefore derived on an application basis and explicitly tailored to CPESs.

To reflect the diversity of communication behavior while keeping the modeling effort manageable, two classes of communication traffic models are distinguished: *simple* and *complex* traffic models. Simple traffic models capture applications whose communication behavior can be described by a small set of aggregate parameters (e.g., monitoring applications). Complex traffic models, in contrast, are used for applications where the communication behavior is strongly shaped by the organizational structure and requires an explicit representation of communication flows between different system layers (e.g., decentralized demand supply balancing).

3.1.1 Simple Traffic Models

Simple traffic models represent communication applications whose behavior can be characterized by a few key parameters without explicitly modeling detailed message sequences. In this work, each simple traffic model is described by

- the *message frequency*, specifying how often messages are generated (e.g., periodically with a given interval or on demand),
- the *data size* of individual messages or ranges of payload sizes,
- the *communication mode*, such as unicast, broadcast and multicast, and
- the set of *end devices* involved as senders and receivers.

These parameters collectively define the temporal intensity and volumetric load imposed by an application on the communication network, as well as the directionality of the traffic.

Based on an overview of CPES applications that rely on communication networks by Cali et al. [62], a set of representative applications[1] is selected and mapped to simple traffic models (cf. [1]). The selection includes both customer-oriented and grid-oriented functions.

[1] The applications have been chosen to include different communication characteristics (e.g., regarding data sizes and frequencies).

Customer Automation is modeled with relatively small payloads on the order of 10-100 Byte (B), representing short control or status messages. *Automated Meter Reading* is included in two variants: an on-demand variant with packets around 100 B, and a scheduled variant with larger payloads in the range of approximately 1600-2400 B, reflecting the periodic transmission of more comprehensive meter data. *Storage applications* are represented with packet sizes of about 50 B, while Customer Information services are modeled with medium-sized messages of around 200 B. In addition, several *Wide Area Control* applications are covered, which are particularly relevant for transmission system operation. These include *Power Oscillation Monitoring*, *Voltage Stability Control*, *State Estimation*, and *Protection* functions. Their traffic is modeled with comparatively small packet sizes, typically in the range of a few tens of bytes (e.g., 10 B for oscillation monitoring, voltage stability control, and protection, and around 52 B for state estimation), consistent with phasor-based measurements and control commands.

3.1.2 Complex Traffic Models

For some applications in CPESs, communication behavior cannot be adequately described by a small set of aggregate parameters alone. This is particularly the case for applications that are realized as (or naturally lend themselves to) agent-based systems, in which autonomous entities with local objectives interact, exchange information, and adapt their behavior based on their environment and other agents. Agent-based realizations offer several advantages, including the ability to decompose complex control and coordination tasks into local decision-making, support scalability in large systems, and enable flexible, modular designs that can cope with heterogeneous assets and stakeholders. [76, 77]

Such applications typically involve distributed decision making, negotiation, or hierarchical coordination and therefore exhibit communication patterns that depend strongly on the organizational structure of the system and on the roles of the involved agents. Examples include market participation, demand–supply balancing, coordinated electric vehicle charging, fault detection and location, and distributed voltage regulation. [76]

For these applications, complex traffic models are defined that explicitly specify the communication flows between agents on different system layers and for different organizational structures. Following Yao et al. [78], a layered architecture for information exchange is adopted containing local, intermediary, and central agents:

- **Local layer**: Includes household agents, device agents, and other end-point devices.
- **Intermediate layer**: Contains aggregator agents and substation agents that facilitate communication between local and central entities.

- **Control layer**: Comprised of grid operator agents and control center agents responsible for system-wide decisions or central entities such as markets.

Yao et al. [78] also identify three different organizational structures:

1. **Centralized**: A central entity is responsible for information aggregation and decision making.

2. **Hierarchical**: An intermediary layer is introduced, where groups of local agents report state information to respective intermediary agents.

3. **Decentralized**: Direct communication occurs between agents without central coordination.

For each complex application and each organizational structure, abstract communication flows are derived from representative implementations in the literature. Table 3.1 summarizes these flows by listing, for each combination of application and structure, the sequence of message exchanges between layers (Local, Intermediary, Control). For instance, market participation under a centralized structure is modeled as local agents sending bids to a central market operator and receiving price signals and clearing results, whereas in a decentralized structure, local agents exchange trading proposals and negotiation messages directly. Similar distinctions exist for demand–supply balancing, electric vehicle management, fault detection and location, and voltage regulation. Table 3.1 includes literature references for each application and structure, demonstrating that all communication flows are derived from research on concrete implementations in various CPES contexts. It is important to note that not the full algorithmic logic of these applications was implemented, but the abstracted the communication flows were extracted from the referenced studies. This abstraction allows to capture the essential communication characteristics while maintaining a manageable simulation scope.

These flows were then parametrized using different:

- **Data sizes**: *small* (8-50 B), *medium* (50-200 B), *large* (200-1000 B), based on ranges from simple traffic models [62],

- **Optimization complexities**: reflected as the duration of hold times between message arrival and response: *immediate* (0-0.1 seconds (s)), *low* (0.1-1 s), *medium* (1-30 s), *high* (30 s-5 min),

- **Overlay topologies**: *ring*, *small-world*, *complete* as in [93].

Table 3.1: Communication flows in different organizational structures.

Application	Structure	Layer Communication Flow	Reference
Market Participation	Centralized	Control → Local: Market price signals Local → Control: Bids/offers Control → Local: Market clearing results	[79]
	Hierarchical	Control → Intermediate: Market information Intermediate ↔ Local: Bid collection and aggregation Intermediate → Control: Aggregated bids Control → Intermediate: Market results Intermediate → Local: Allocation decisions	[80]
	Decentralized	Local ↔ Local: Peer-to-peer trading proposals Local ↔ Local: Bilateral negotiation	[81]
Demand Supply Balancing	Centralized	Local → Control: Supply/demand data Control → Local: Allocation decisions	[82]
	Hierarchical	Control → Intermediate: Balancing request Intermediate → Local: Negotiation initialization Local ↔ Local: Peer negotiation Local → Intermediate: Negotiation results Intermediate → Control: Aggregated solution	[83]
	Decentralized	Control → Local: Balancing request Local ↔ Local: Peer negotiation Local → Control: Final solution	[84]
EV Management System	Centralized	Local → Control: Charging requirements Control → Local: Charging schedules	[85]
	Hierarchical	Control → Intermediate: Price and constraint information Local → Intermediate: Charging requirements Intermediate → Local: Charging schedules	[86]
	Decentralized	Local ↔ Local: Schedule exchange and coordination	[85]
Fault Detection and Location	Centralized	Local → Control: Measurement/fault indicators	[87]
	Hierarchical	Local → Intermediate: Measurement data Intermediate: Fault analysis	[88]
	Decentralized	Local → Local: Fault indicators to neighbors Local ↔ Local: Coordinated fault detection	[89]
Voltage Regulation	Centralized	Local → Control: Measurement data Control → Local: Voltage adjustment commands	[90]
	Hierarchical	Local → Intermediate: Measurement data Intermediate → Local: Voltage adjustment commands	[91]
	Decentralized	Local ↔ Local: Voltage state exchange Local ↔ Local: Coordinated adjustment information	[92]

By combining simple and complex traffic models, the resulting set of communication traffic models spans a broad range of communication behaviors observed in CPESs. This set serves as the starting point for the construction of communication scenarios in the Section 3.3, where the traffic models are mapped onto concrete communication network models and used to generate simulation data.

3.2 Communication Network Models

To define a comprehensive set of CPES-specific communication scenarios, the previously introduced traffic models must be combined with suitable communication network mod-

els. The communication network models presented in this section are designed to reflect the range of real-world communication systems outlined in Section 2.3, while remaining sufficiently structured and parameterized for systematic scenario generation.

Since the communication applications and the derived traffic models are grounded in CPESs, the locations of the end devices in the communication network are mapped to the geographical coordinates of the corresponding entities in the power system. To this end, communication network topologies are generated on the basis of the *SimBench-pandapower* reference networks [94]. Grid assets such as buses, loads, and substations are associated with communication nodes (e.g., meters, controllers, gateways), so that communication paths reflect realistic distances, clustering, and urbanization levels.

The resulting communication networks are modeled using the *OMNeT++* network simulator [37]. While other network simulators (for an overview refer to paragraph 2.1.4) could, in principle, be used analogously, *OMNeT++* was selected due to its comprehensive support for heterogeneous communication technologies and its extensive protocol libraries. Wireless and wired technologies are integrated via established *OMNeT++* libraries (*INET*[2] and *Simu5G*[3]), enabling a consistent configuration and reuse of models across scenarios.

A set of widely used (cf. e.g., [63]) communication technologies is considered (for details on communication technologies for CPESs refer to Section 2.3):

- **5G**: Used to represent high-capacity, low-latency cellular access with wide-area coverage.
- **LTE**: Models current-generation cellular networks with broad deployment, providing moderate to high data rates and good coverage.
- **LTE450**: Represents dedicated networks with sub-Gigahertz (GHz) cellular communication around 450 Megahertz (MHz), which offers improved coverage and penetration in rural or hard-to-reach areas.
- **WiFi (802.11/802.11ac)**: Used for short-range wireless access in customer premises and local environments, covering a wide range of data rates and device densities.
- **Ethernet**: Represents wired communication with high reliability and stable data rates.

Each technology is configured with realistic parameters, summarized in Table 3.2. For 5G networks, parameters following Nardini et al. [95] are implemented, including carrier frequency, transmit powers, antenna gains, and noise figures. LTE and LTE450 configurations

[2] `https://inet.omnetpp.org/`, last access: 2026-01-19
[3] `https://simu5g.org/`, last access: 2026-01-19

are based on the *INET* framework and prior work on combined control and communication simulations for CPESs [5], with appropriate adaptations for the lower carrier frequency and propagation characteristics of LTE450. WiFi configurations (IEEE 802.11 and 802.11ac) follow the parameter settings of the *OMNeT++* example models, representing typical low-power deployments in customer premises networks. Ethernet links are modeled with configurable link capacities and a finite queue capacity of 100 packets at intermediate nodes.

Table 3.2: Communication network parameters used in the simulation. Power levels are given in Decibel relative to one milliwatt (dBm); antenna gains in Decibel relative to isotropic radiator (dBi); noise figures in Decibel (dB).

Technology	**Parameter**	**Value**
5G	Carrier frequency	700 MHz
	UE Tx Power	26 dBm
	gNB Tx Power	46 dBm
	gNB antenna gain	8 dBi
	UE antenna gain	0 dBi
	gNB noise figure	5 dB
	Number of resource blocks	50
LTE	UE Tx Power	26 dBm
	eNodeB Tx Power	40 dBm
	Number of resource blocks	6
LTE450	Carrier frequency	450 MHz
	UE Tx Power	26 dBm
	eNodeB Tx Power	40 dBm
	Number of resource blocks	6
	Propagation model	Rayleigh
WiFi 802.11	Carrier frequency	2.4 GHz
	Bitrate	2 Mbps
	Tx Power	3 dBm
	Receiver sensitivity	-85 dBm
WiFi 802.11ac	Carrier frequency	5 GHz
	Bitrate	693.3 Mbps
	Tx Power	20 dBm
	Receiver sensitivity	-85 dBm
Ethernet	Packet queue capacity	100 packets

The various network types introduced in Subsection 2.3.1 are instantiated by combining these technologies with different topologies:

- **HANs** (used for the *Customer Automation* application) are modeled using an assumed in-home topology without explicit neighborhood structure. WiFi (802.11 and

802.11ac) and Ethernet are employed to represent short-range wireless access and wired in-building connections.

- **WANs** (used for *Wide Area Control (WAC)* applications) are based on *SimBench* medium-voltage networks with larger geographic extent and sparser node distributions. They are also modeled with 5G, LTE, LTE450, and Ethernet to capture a range of backbone and wide-area connectivity options.
- **NANs** (used for all remaining applications) are derived from selected *SimBench* distribution and mixed-level networks and are equipped with 5G, LTE, LTE450, and Ethernet technologies. They represent communication within the distribution grid and between customer premises and utility systems.

These technology choices reflect typical usage patterns reported in the literature [63, 96] and are supported by existing *OMNeT++* model libraries.

The *SimBench* networks are selected to cover diverse characteristics, including different urbanization levels (urban, semi-urban, rural), grid levels, and future network variants. For HANs, a single generic home topology is used and combined with multiple access technologies, as no explicit spatial neighborhood modeling is required. For NANs and WANs, different *SimBench* networks are used to generate scenarios with varying node densities, coverage areas, and path lengths. Wireless technologies are deployed with appropriately placed base stations and access points to ensure adequate coverage of the network area, depending on the size and spatial extent of the underlying power grid.

Overall, this modeling process yields the following set of communication network models:

- 3 HAN models, derived from 1 topology combined with 3 technologies.
- 12 NAN models, derived from 3 *SimBench* networks[4] combined with 4 technologies.
- 8 WAN models, derived from 2 *SimBench* networks[5] combined with 4 technologies.

As an example, Figure 3.2 shows one of the networks as a model in *OMNeT++*. The various end devices are named here as agent equivalents, but can also represent corresponding non-agent-based devices. A wireless network has been chosen as the technology in this example. The network with the given network size can be covered here by one antenna. These models form a modular and reusable basis for constructing communication scenarios by combining them with the traffic models from Section 3.1. In Section 3.3, these combinations are used to instantiate concrete scenarios and generate simulation data for subsequent analysis and meta-modeling.

[4] *SimBench* codes for NANs: `1-MV-urban-1`, `1-LV-semiurb4-1`, `1-MVLV-rural-all-2`
[5] *SimBench* codes for WANs: `1-MV-urban-1`, `1-MV-rural-0`

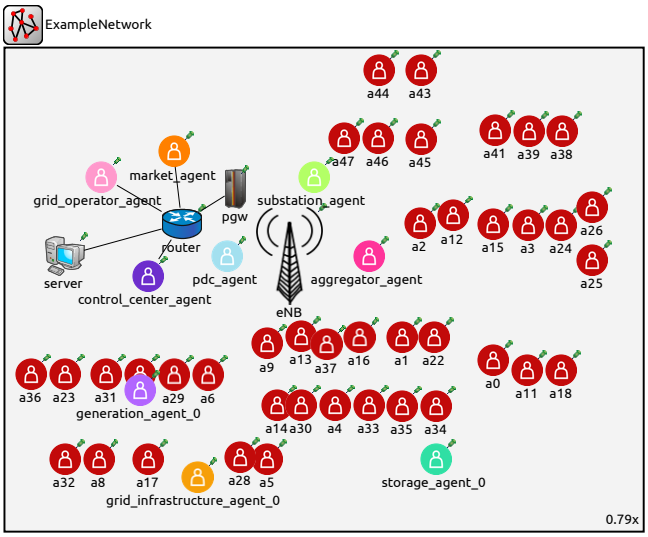


Figure 3.2: Visual representation of the communication network model in the *OMNeT++* communication simulator for the *SimBench* network with the code `1-LV-semiurb4-1`. (Icons prefixed with "a" represent household agents. The exact positions and numbers of the devices have been slightly modified for readability.)

3.3 Scenario and Simulation Data Generation

Based on the communication traffic models and communication network models introduced in Section 3.1 and Section 3.2, concrete communication scenarios are now generated according to the workflow summarized in Figure 3.1. Each scenario represents a specific combination of application-induced communication behavior and network conditions and serves as a single simulation run in the subsequent analysis as illustrated in the data generation workflow in Figure 3.3.

For simple traffic models, parametrization is carried out by varying the number of end devices and application-dependent parameters such as response times or request intervals. In this manner, several instances of the same type of application are generated, which may differ, for instance, in their level of parallelism or in their temporal patterns of activity. Complex traffic models are parametrized more extensively: the number of participating end devices, the data sizes (small, medium, large), the optimization complexity (reflected in different hold times between message arrival and response), and additional scenario-dependent parameters are varied systematically. Each of these parameter configurations

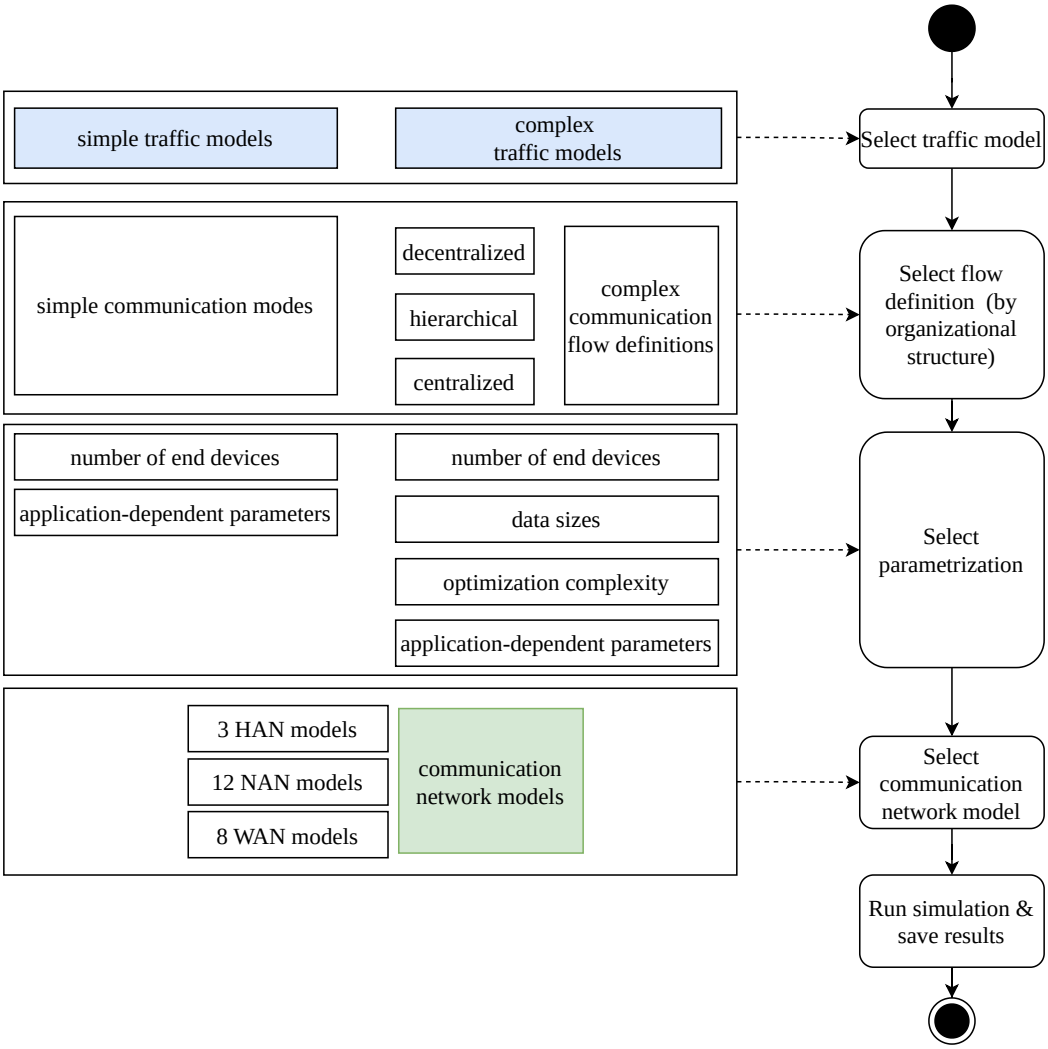


Figure 3.3: Simplified illustration of the scenario creation and data generation process.

is instantiated in three different modes introduced in Section 3.1 (centralized, hierarchical, decentralized). For decentralized structures, different overlay topologies for message dispatch are considered, resulting in distinct communication patterns even for otherwise similar application settings.

All resulting concrete traffic model instances are then combined with compatible communication network models from Section 3.2. Simple and complex applications that operate in customer premises are mapped to HAN models, distribution-level applications to NAN models, and wide-area control applications to WAN models. This systematic combination

yields a total of 4 673 distinct simulation scenarios. While this represents an extensive coverage of the modeled parameter space, it is important to note that the full space of possible communication scenarios in CPESs is effectively infinite due to the countless combinations of application parameters, network conditions, and organizational structures. The approach taken here does not aim at exhaustiveness but at a representative sampling of the communication behavior space in CPESs, enabling a meaningful analysis of traffic patterns and their impact on network performance.

The entire scenario generation and simulation workflow (cf. Figure 3.3) is implemented in Python in a repository called *training data generation tool for combined communication and energy system scenarios (trace)*[6]. Communication flows are specified using the predefined traffic model descriptions and translated into concrete message generation rules. For each scenario, the corresponding *OMNeT++* network description files (`.ned`) and initialization files (`.ini`) are generated automatically based on the selected *SimBench* network and communication technology configuration and written into the *OMNeT++* project structure. The simulations are then executed from within the Python script by invoking *OMNeT++* as an external process. After completion of each run, the relevant simulation results are extracted.

The resulting files contain both scenario-level metadata and message-level records. The metadata includes, among other attributes, the simulation start time, the communication application on which the traffic model is based, the applied organizational structure, the selected communication technology, and the corresponding parameter settings. The message-level records comprise all messages dispatched during the simulation, including a unique message identifier, the simulation time at which the message was sent, the sender and receiver nodes, the packet size in bytes, the reception time, and the resulting end-to-end delay in milliseconds (computed as the difference between reception and transmission time). These data form the basis for the subsequent analysis of communication scenario diversity and network performance in Section 3.4.

3.4 Simulation Data Analysis

The objective of this section is to analyze the simulation data generated from the communication scenarios introduced in Section 3.3. While the methodological details and selected results of this analysis have been published previously in own work [1], it is important in the context of this thesis to revisit the main findings with a specific focus on their implications for online meta-modeling. In particular, the analysis serves two purposes: (i)

[6] `https://github.com/OFFIS-DAI/trace`

it demonstrates the diversity of communication scenarios resulting from different CPES applications, parametrization, and organizational structures (Subsection 3.4.1), and (ii) it highlights how this diversity translates into markedly different behaviors of the modeled communication networks (Subsection 3.4.2). These observations motivate the choice of input features and requirements for the meta-model developed in later chapters.

3.4.1 Analysis of Communication Traffic Models

This subsection analyzes the communication traffic models across all generated scenarios, with a focus on four complementary aspects: (i) sender/receiver structures between system layers, (ii) the relationship between sender/receiver ratios and packet sizes, (iii) temporal message dispatch patterns, and (iv) the variability of traffic. Taken together, these viewpoints offer a concise but expressive description of the heterogeneous communication behavior observed in CPESs.

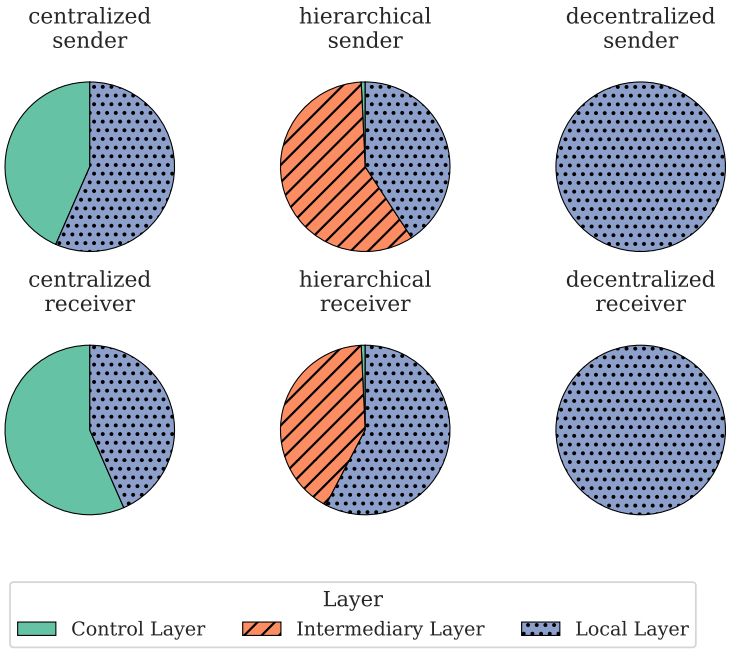


Figure 3.4: Distribution of messages by sender and receiver layers for the *Market Participation* application under different organizational structures.

Before analyzing temporal properties, the sender/receiver structures provide insight into how different system layers interact. Figure 3.4 shows the distribution of messages by sender and receiver layers for the *Market Participation* application across the three organizational structures introduced in Section 3.1. In the centralized structure, traffic is

dominated by bidirectional communication between Local and Control Layers, reflecting a direct interaction between local resources and central market or grid operators. In the decentralized structure, communication occurs almost exclusively within the Local Layer, illustrating the peer-to-peer nature of fully distributed market schemes. The hierarchical structure exhibits the most complex pattern, with the Intermediary Layer (aggregators) sending a large share of messages and the Local Layer receiving most of them. These differences demonstrate how organizational structure alone can shift the communication burden between system layers. This can affect which nodes and links are most critical from a communication perspective.

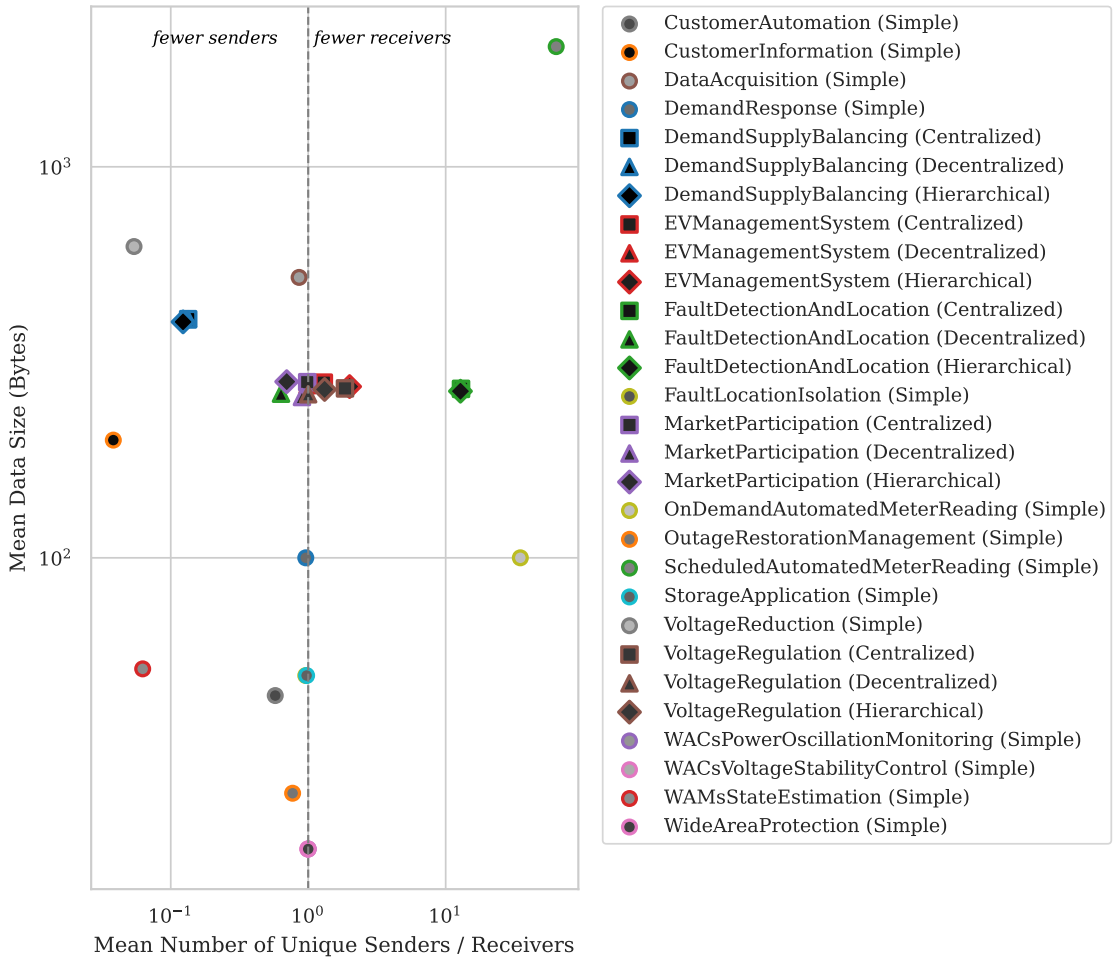


Figure 3.5: Ratio between the number of unique senders and receivers versus mean packet size (logarithmic scale) across all applications and organizational structures.

Beyond layer-wise distributions, the heterogeneity of communication roles can be cap-

tured by the ratio between the mean number of unique senders and receivers in a scenario. Figure 3.5 visualizes this sender/receiver ratio for all applications and organizational structures, plotted against the mean packet size on a logarithmic scale. Ratios close to 1 indicate balanced many-to-many communication, as in *Data Acquisition*, where control entities request measurements from multiple devices and receive corresponding responses. Ratios greater than 1 characterize scenarios with many senders and comparatively few receivers, typical of aggregation or fault reporting patterns such as *Fault Detection and Location*. Ratios below 1 occur when few senders communicate with many receivers, as in broadcast-oriented applications like *Customer Information*. The strong variation in packet sizes (from very small payloads for time-critical protection to large periodic transfers in *Scheduled Automated Meter Reading*) further underlines the diversity of traffic characteristics across applications. Organizational structures modify these patterns within the same application, for example shifting from broadcast-like to more balanced sender/receiver ratios when moving from centralized to decentralized realizations.

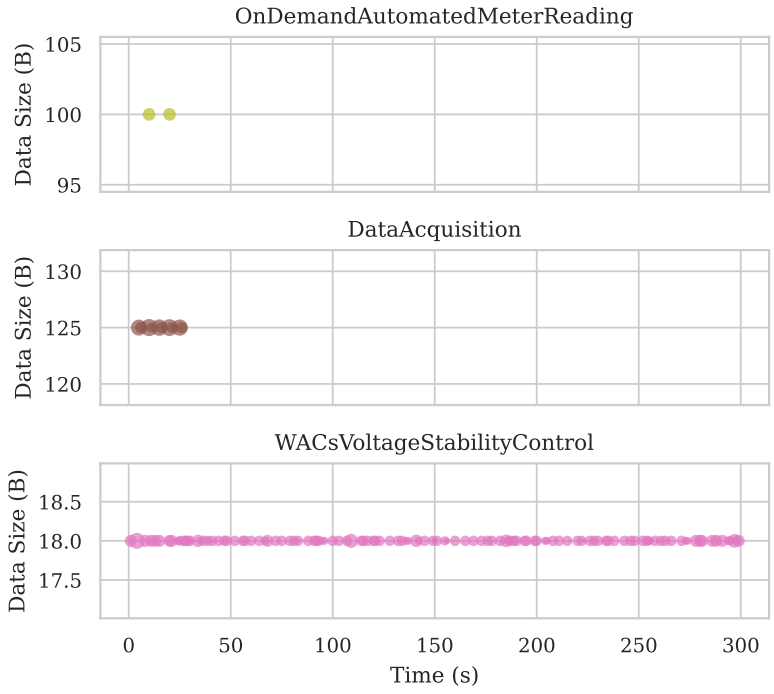


Figure 3.6: Messages and data sizes (in bytes) sent over time (in seconds) in three exemplary scenarios, illustrating that different application services exhibit fundamentally different communication behaviors. The marker size reflects the number of messages dispatched simultaneously with the same data size.

The temporal evolution of message dispatch further illustrates the heterogeneity of com-

munication traffic models. Figure 3.6 shows three exemplary scenarios. The *On-Demand Automated Meter Reading* application exhibits a sparse pattern with only a few messages at the beginning of the simulation, all with a fixed payload size. In contrast, *Data Acquisition* starts with a small number of requests sent from a central entity and evolves into a much higher message volume as responses accumulate over time, illustrating a growing load driven by increasing participation. The *WACs Voltage Stability Control* application features frequent, small-sized messages that are dispatched almost continuously, reflecting the requirements of time-critical, real-time control. These examples highlight that communication traffic in CPESs ranges from sporadic and event-driven to highly regular and dense, and that different applications can induce fundamentally different temporal load profiles.

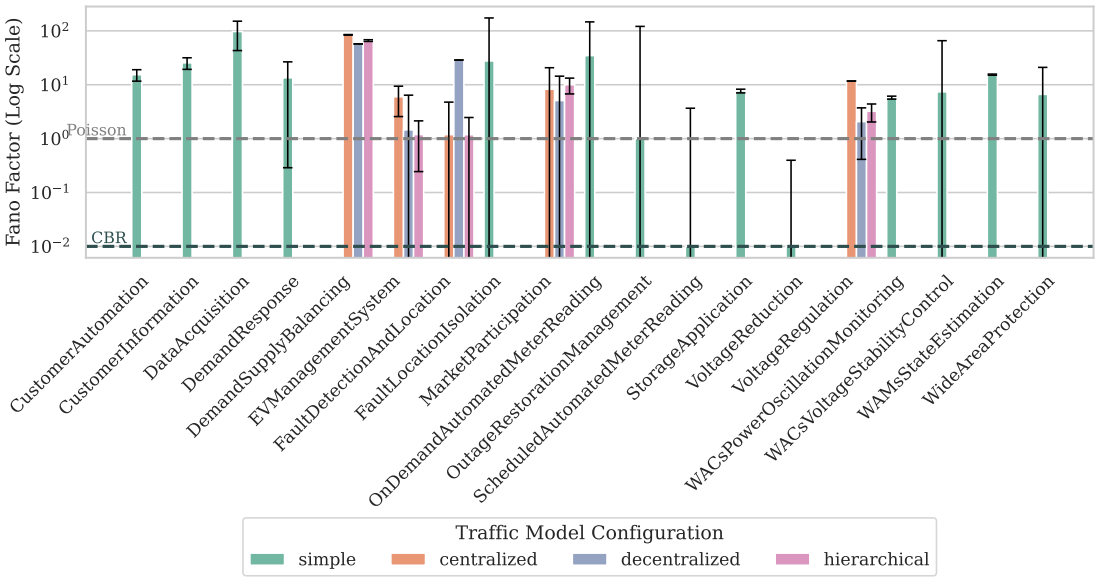


Figure 3.7: Distribution of Fano factors (logarithmic scale) by application and organizational structure, compared to reference values for Poisson and CBR processes.

To quantify the variability and burstiness of message dispatch in a compact form, the so-called *Fano factor* is used. The Fano factor is a statistical measure used to quantify the variability of counting processes, often used in neuro science [97]. This measure is used here to analyze the variability of communication traffic models. For a given time window of size w, let $N(w)$ denote the number of messages dispatched within that window. The Fano factor is then defined as the ratio of the variance $Var(N(w))$ to the expected value

$\mathbb{E}[N(w)]$ of the number of events dispatched in that time window:

$$F(w) = \frac{\mathrm{Var}(N(w))}{\mathbb{E}[N(w)]}. \tag{3.1}$$

In this work, $w = 1$ milliseconds (ms) is used to facilitate comparability between scenarios of different duration. A value $F(w) \approx 1$ corresponds to Poisson-like traffic with variance and mean of message counts being equal, $F(w) < 1$ indicates more regular, CBR-like behavior, and $F(w) > 1$ reflects super-Poissonian variability with bursty or clustered message arrivals.

Figure 3.7 summarizes the Fano factors across all applications and organizational structures on a logarithmic scale. The results reveal a wide spread of variability levels, ranging from nearly deterministic, CBR-like traffic to highly bursty patterns with Fano factors several orders of magnitude above one. Some applications, such as *Outage Restoration Management*, exhibit near-Poissonian behavior, while others, such as *Data Acquisition* or certain market-related and multi-agent coordination schemes, show pronounced burstiness. Organizational structures further modulate the variability of agent-based applications, with decentralized realizations often leading to more irregular, interaction-driven traffic. These findings demonstrate that standard assumptions such as Poisson or CBR processes (see Table 2.2 for an overview) capture only a small subset of the traffic behaviors observed in CPESs.

3.4.2 Analysis of Network Influences

The properties of communication traffic models are assumed to directly influence the load imposed on the underlying communication network and, thereby, to affect key performance metrics such as end-to-end delay and packet loss. In the following, the network behavior is therefore primarily analyzed through the lens of message delay times across the considered network technologies. A detailed, application-wise overview of mean and standard deviation of delay times for all applications and technologies is provided in the own, accompanying publication [1].

Across the simulated scenarios, delay characteristics vary substantially between applications and technologies, reflecting the interplay between application-induced traffic patterns and technology-specific resource handling. For example, applications with strong aggregation or reporting characteristics can exhibit pronounced delay variability, while more regular applications show comparatively stable delay behavior across technologies. In the simulation configuration, Ethernet is especially vulnerable during periods of heavy load because the buffers at intermediate nodes are limited, which can intensify queuing effects

when traffic is bursty.

To compactly and technology-neutrally relate traffic characteristics to network behavior, the Fano factor categories introduced in Subsection 3.4.1 are used as an indicator for variability in dispatch times. Following the categorization used in [1], three groups are distinguished: (i) CBR-like scenarios with $F(w) \approx 0$, (ii) Poisson-like scenarios with $F(w) \approx 1$, and (iii) super-Poissonian scenarios with $F(w) > 1$. These categories reveal a clear relationship between traffic variability and delay behavior across all considered technologies: CBR-like scenarios exhibit the lowest mean delays and the lowest delay dispersion due to predictable, evenly spaced arrivals that reduce contention and prevent sustained queue build-up. Poisson-like scenarios likewise maintain low mean delays because message arrivals are comparatively well distributed over time, lowering the probability of short-term overload peaks. In contrast, super-Poissonian scenarios induce markedly higher mean delays and substantially larger delay spreads, as clustered arrivals create transient periods of intense load that increase contention and queuing.

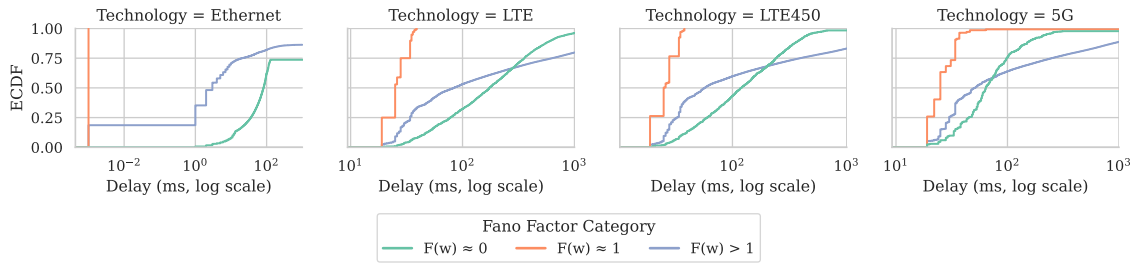


Figure 3.8: Delay distributions (logarithmic x-axis) across network technologies, stratified by Fano factor categories ($F(w) \approx 0$, $F(w) \approx 1$, $F(w) > 1$). Increasing burstiness is associated with heavier delay tails, particularly for Ethernet due to finite queuing capacity.

Figure 3.8 visualizes the relationship between dispatch-time variability and network delay by showing the Empirical Cumulative Distribution Functions (ECDFs) of end-to-end delay on a logarithmic x-axis, stratified by technology and grouped into three Fano-factor categories ($F(w) \approx 0$, $F(w) \approx 1$, $F(w) > 1$). The communication networks are modeled using different technologies, following the method described in Section 3.2. Within each technology, the curves exhibit a consistent ordering: moving from regular/near-constant dispatching ($F(w) \approx 0$) to Poisson-like traffic ($F(w) \approx 1$) and finally to bursty traffic ($F(w) > 1$) shifts the delay distributions to the right and reduces the slope of the ECDF, indicating both larger typical delays and increased dispersion. In other words, for a fixed quantile, the corresponding delay threshold increases with burstiness, and the gap between categories becomes most pronounced in the upper tail. Across technologies, the baseline

delay level differs substantially, but the qualitative effect of burstiness persists. Cellular technologies show broader distributions than Ethernet for many scenarios, while the bursty category consistently introduces heavier tails. This interpretation is consistent with the aggregate delay statistics from [1], where burst-prone applications can lead to very large mean delays and variability. Overall, the results highlight that traffic variability (captured by the Fano factor) is a key determinant of delay behavior beyond mean message rates. Technology-specific buffering and scheduling can mitigate the impact of moderate loads, but under bursty arrivals, queueing dynamics amplify tail latency, particularly in configurations with finite queue capacity.

3.5 Chapter Summary

This chapter presents the artifacts of Sub-RQ 1 and therefore established a CPES-specific framework to describe, generate, and analyze communication scenarios as the combination of an application-derived traffic model and a parameterized communication network model. Based on this modular scenario view, a large set of representative scenarios was generated and simulated, enabling a systematic assessment of how diverse CPES communication behavior can be and how this diversity translates into network performance differences.

Overall, the analysis shows that communication traffic models in CPESs are highly heterogeneous with respect to sender/receiver structures, payload sizes, temporal dispatch patterns, and variability (quantified via the Fano factor). Moreover, the same communication application can exhibit markedly different communication characteristics depending on its organizational structure (centralized, hierarchical, decentralized), which shifts communication burden across system layers and changes both interaction patterns and traffic variability.

The simulation results further demonstrate that the response of the communication network is strongly dependent on the application-induced traffic characteristics. In particular, dispatch-time variability and burstiness are closely linked to delay behavior and delay dispersion across technologies, indicating that network performance cannot be inferred from average traffic rates alone.

These findings have direct implications for the online meta-modeling approach developed in later chapters. First, the meta-model needs to embed information that captures both the current condition of the network and the key properties of the active traffic model, so that it can generalize effectively across different applications and organizational configurations. Second, it should be able to approximate not only scenarios with simple traffic models but also complex, interaction-driven traffic originating from agent-based and organizationally structured applications. Finally, the breadth of the generated scenario set provides both

the empirical foundation and the key insights required for the subsequent development of input features and for the later evaluation and comparative analysis in Chapter 6.

4 Communication Network Simulation Modeling

All models are wrong,
but some are useful.

George Box

In integrated CPES studies, communication behavior (e.g., end-to-end delay and congestion dynamics) constitutes an important part of the overall system response and must therefore be represented in the simulation. In Section 2.5, it was determined that replacing a detailed communication simulation with a learned approximation must address not only predictive accuracy, but also simulation compatibility. The approximation must retain the essential properties of DESs so that it can be integrated into a simulation environment (e.g., co-simulation), where time, events, and state transitions are managed in a strictly discrete-event fashion. This chapter specifically contributes to Sub-RQ 2 by detailing how simulation outputs can be represented as DES-conform observations (events and state updates) and processed online within a meta-model that itself constitutes an executable DES component.

In the own, accompanying publication [10], the core modeling approach for network simulation approximation has already been presented. This chapter revisits the same modeling foundations, but extends them by embedding the modeling choices into the overall methodological context of this thesis.

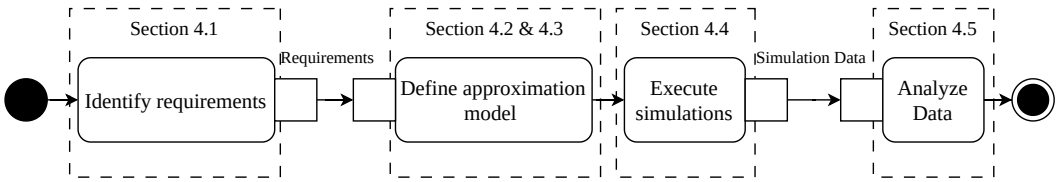


Figure 4.1: Presentation of the steps from identifying the requirements for an integrated simulation model to analyzing the simulation data.

The remainder of this chapter is structured as illustrated in Figure 4.1. Section 4.1

derives requirements for an integrated approximation model from DES theory. Section 4.2 introduces the graph-based internal representation used in the meta-model and motivates technology-agnostic modeling decisions. Then, Section 4.3 formalizes event semantics and time advancement, followed by Section 4.4, which describes the execution routine and interfaces in the coupled simulation. With that, Section 4.5 then analyzes state attributes and motivates the feature subset used throughout the remainder of this work, including a discussion of missing values. Finally, Section 4.6 summarizes the chapter.

4.1 Requirements for an Integrated Simulation Model

In Section 2.1, the foundations of simulations were introduced and a *system* was defined as a set of interrelated elements that are viewed as a whole within a specific context. Here, the system of interest is the *detailed communication simulation* that serves as the reference to be approximated. The meta-model is consequently designed as a simplified replica of this system, i.e., a *model* that reproduces those aspects of the system behavior that are relevant for integration into the surrounding simulation environment.

The detailed communication simulation consists of multiple interacting system components. In the communication network setting examined here, these components are considered communicating entities that take on the roles of sender and receiver during the exchange of messages. The approximation therefore must provide a representation that can capture the participating entities and their communication relations and evolve as new observations occur during runtime (details in Section 4.2).

In addition to representing the system's components, the model must also describe the system's current operational state. Each system component (i.e., each communication node) can be characterized by a set of state variables that, taken together, form a snapshot of the overall system at a given simulation time. This snapshot is required because the communication performance observed at the interface (e.g., message delay) is not solely a function of the current packet size, but also depends on the current network utilization and the recent communication history encoded in the state.

From the perspective of DES theory, it is crucial that this state changes only at discrete points in time. As discussed in Subsection 2.1.4, DESs are defined by instantaneous state transitions that occur at specific time instants, referred to as events. Hence, an integrated approximation model must expose a clear event definition that is compatible with the information available at its interface and a time advancement mechanism that preserves the causality and ordering constraints of discrete-event execution. The event definition as well as the time advancement strategy used in this work are formalized in Section 4.3.

Finally, the above elements must be combined into an executable routine that can

be embedded into a coupled simulation. Specifically, it is necessary to specify how the simulation is initialized, how events are processed, how state variables are updated from observations, and how outputs are produced. This ensures that the component remains DES-conform and interoperable. Based on the definitions introduced in the following sections, the resulting simulation execution routine is described in Section 4.4.

4.2 Graph-Based Network Model

The objective of this work is to approximate the behavior of a detailed communication simulation during execution time. Consequently, the approximation must be able to incorporate new observations as they occur, i.e., update its internal representation via discrete state changes triggered by events. This requires a modular and dynamically extensible representation of the communication system simulation within the meta-model.

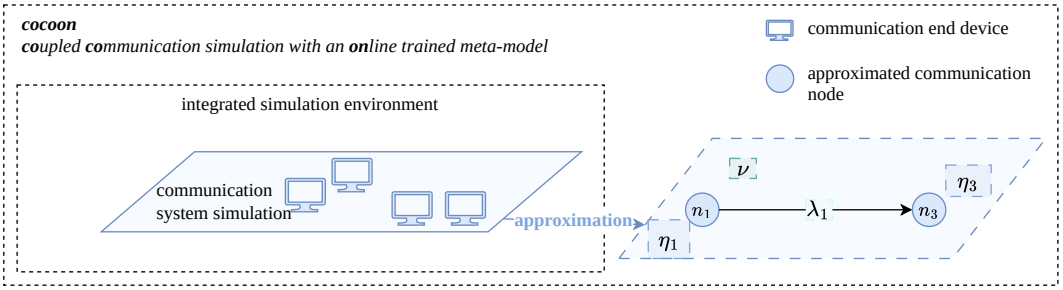


Figure 4.2: Simplified representation of the components in *cocoon*. On the left-hand side, part of the integrated simulation environment can be seen, in which the communication simulation is located. This system is approximated on the right-hand side by the graph of the meta-model.

For this purpose, the software artifact *coupled communication simulation with an online trained meta-model (cocoon)* was developed as part of this work. Figure 4.2 illustrates the conceptual separation between the detailed communication system simulation and its approximation. While the former provides the reference behavior, the meta-model in *cocoon* maintains an internal representation that is updated online and can be used to predict message delays in a DES-conform manner.

Within *cocoon*, the communication system simulation is represented as a directed graph

$$G_t = (N_t, L_t), \tag{4.1}$$

where N_t denotes the set of nodes and L_t denotes the set of directed communication links at simulation time t. A node $n \in N_t$ represents a communication endpoint observable at

the meta-model interface (e.g., a component that sends or receives messages). A directed link $\ell_{ij} \in L_t$ represents that messages have been observed from sender n_i to receiver n_j.

The graph is constructed online from observed message exchanges. When a message is observed with sender n_s and receiver n_r, the following updates are performed: (i) if $n_s \notin N_t$ or $n_r \notin N_t$, the corresponding node(s) are created and added to N_t; and (ii) if the directed link $(n_s, n_r) \notin L_t$, it is created and added to L_t. This observation-driven construction is essential for runtime approximation, as it allows *cocoon* to reflect changes in active participants and communication relations without requiring prior knowledge of the underlying network topology.

The graph representation abstracts from network-specific implementation details (e.g., protocol stacks, routing, medium access, or physical topology). Instead, it focuses on the interface-observable interaction structure and its temporal evolution. This design choice supports adaptivity across different detailed simulators and network models while retaining the ability to represent load-dependent behavior through state variables that are updated from observed events.

To capture the system state at a given point in time, state attributes are defined and assigned to the components of G_t. Since the system is represented as a graph, state is decomposed into three levels:

- a global *network state* $\nu(t)$ that represents the overall operating point of the communication system,
- *node states* $\eta_n(t)$ for each $n \in N_t$ that represent endpoint-specific conditions, and
- *link states* $\lambda_\ell(t)$ for each $\ell \in L_t$ that represent relation-specific conditions between communicating endpoints.

These state variables follow DES terminology (cf. Subsection 2.1.4) and are updated only at event times.

Table 4.1 summarizes the attributes considered at link, node, and network level. The attributes are designed to encode instantaneous, event-local information (e.g., current inter-arrival/inter-departure times) as well as aggregated information that captures recent dynamics (e.g., average delay, average idle time, utilization). In this way, the state provides a compact representation of the recent communication history that is required to model load-dependent delay behavior.

Node states are maintained from both sender and receiver perspectives to reflect that a node may participate in the system with different roles and potentially asymmetric conditions. Concretely, when processing a message from n_s to n_r, outgoing-related attributes are updated for the sender node n_s, while incoming-related attributes are updated for the

Table 4.1: Overview on state attributes.

State Type	Attribute
Link State	Resource utilization
	Throughput (bps)
	Average delay time (ms)
	Average inter-departure time (ms)
	Current inter-departure time (ms)
	Average inter-arrival time (ms)
	Current inter-arrival time (ms)
	Average idle time (ms)
	Current idle time (ms)
Node State	Average incoming delay time (ms)
	Average outgoing delay time (ms)
	Average inter-departure time (ms)
	Current inter-departure time (ms)
	Average inter-arrival time (ms)
	Number of messages sent simultaneously
	Average number of messages sent simultaneously
Network State	Average throughput (bps)
	Average delay time (ms)
	Average inter-departure time (ms)
	Average inter-arrival time (ms)
	Average idle time (ms)
	Average resource utilization
	Number of messages in transit
	Number of busy links
	Number of network nodes
	Number of messages sent at the current time

receiver node n_r. This distinction is used throughout the remainder of this chapter when formalizing event updates and time advancement.

4.3 Event Representation and Time Advancement

To preserve DES semantics at the meta-model interface, events are defined exclusively based on observable interactions. In the coupled setting considered in this work, the meta-model observes that a message is issued by a sender at a certain simulation time and that the message departs after an end-to-end delay that is either measured from the detailed communication simulation or predicted by the meta-model. Consequently, two event types are modeled:

- *message arrival events* at a node, which occur when the source simulation issues a message and the message becomes known to the communication component, and
- *message departure events* at a node, which occur when the message completes transmission and the corresponding end-to-end delay is realized.

This event definition ensures that state changes are instantaneous and occur only at discrete time instants, as required by DES.

Each observed message is represented as a message object

$$\mathbf{m} = (t_s, p, \eta(t_s), \nu(t_s), d), \tag{4.2}$$

where t_s denotes the simulation time at which the message arrival event is processed, p is the packet size (in B), $\eta(t_s)$ is the node state at the time of arrival, $\nu(t_s)$ is the network state at the time of arrival, and d is the resulting end-to-end delay of the message. It should be noted at this point that the link states are not taken into account here. This is because the attributes of this state type have not proven to be important for predicting delay times (see Section 4.5).

The states $\eta(t_s)$ and $\nu(t_s)$ in $\mathbf{m}$ are snapshots taken at the arrival time t_s. This is a deliberate design choice: the learning problem is later formulated as predicting the delay based on the overall system state at dispatch time. Capturing the state at t_s thereby establishes a consistent mapping between observations and targets, while maintaining DES-conform timing.

Since the goal is to approximate a detailed communication simulation, it can be distinguished between the realized delay obtained from the detailed simulator and the delay predicted by the meta-model:

$$d \in \{d_{\text{real}}, d_{\text{pred}}\}. \tag{4.3}$$

If the coupled setup executes the detailed communication simulator for a message, the measured delay d_{real} is used. In contrast, if the meta-model substitutes the detailed simulator, the delay is produced as a prediction d_{pred}. In both cases, the corresponding message departure time is defined as

$$t_d = t_s + d, \tag{4.4}$$

which yields a scheduled departure event at simulation time t_d.

Time advancement follows a next-event strategy. The simulation maintains a Future Event Set (FES) containing all scheduled but not yet processed events. At any point in time,

the simulation clock advances directly to the time stamp of the earliest event in this list, and all events scheduled for that same time stamp are processed before advancing further. In the coupled setting, the next time instant is therefore determined by either the earliest scheduled message departure or the next message arrival issued by the source simulation. This next-event principle preserves causal ordering and is consistent with standard DES execution routines [43].

4.4 Simulation Execution

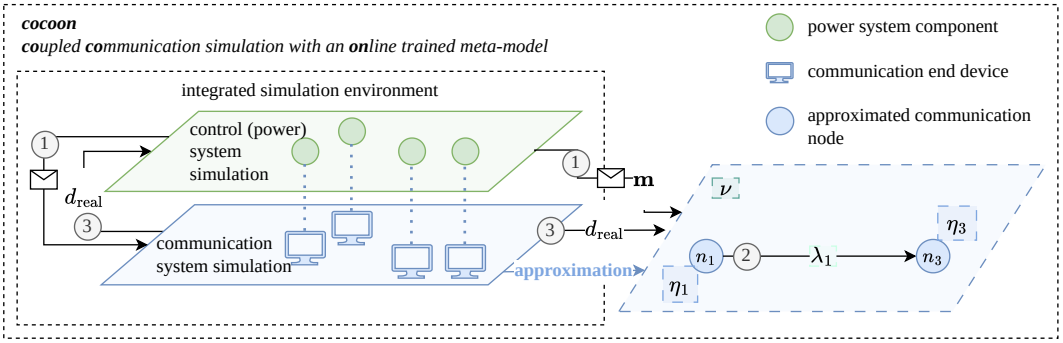


Figure 4.3: Coupled simulation execution and online approximation with *cocoon*.

Figure 4.3 illustrates the execution routine of the coupled setup and how the meta-model approximates the original, detailed communication simulation. The setup consists of three components: (i) the source (control/power system) simulation, in which messages are exchanged between system components (e.g., energy system entities), (ii) the detailed communication network simulation (e.g., *OMNeT++* configured with a selected network model), and (iii) the meta-model in *cocoon*, which maintains a graph-based internal representation and state variables as introduced in Section 4.2.

At the beginning of a simulation run, all components are initialized. The source simulation and the detailed communication simulation are initialized according to the scenario configuration. The meta-model is initialized with an empty directed graph and with an initial state object at network-level.

When the source simulation emits a message event (1), the corresponding information is forwarded to both the detailed communication simulation and *cocoon*. In the meta-model, this triggers the processing of a message arrival event and the construction of the message object as defined in Section 4.3.

Upon receiving the message information (2), the meta-model ensures that the internal graph representation matches the observation. If the sender and/or receiver nodes are not

yet present, they are created and inserted into the graph. If the directed link from sender to receiver does not yet exist, it is created as well. All affected graph components (sender node, receiver node, and the connecting link) are associated with their corresponding state objects, including the attributes listed in Table 4.1.

In the detailed communication simulation, the message dispatch is simulated and the resulting end-to-end delay d_{real} is obtained (3). This delay is returned to the source simulation to ensure consistent progression of the coupled execution. In addition, d_{real} is forwarded to the meta-model, where it is used to schedule the corresponding departure event and to update the relevant node- and network-level state attributes.

4.5 Analysis and Selection of State Attributes

The state representation introduced in Section 4.2 provides a broad set of candidate attributes at network-, node-, and link-level. Since these attributes later serve as inputs to the online learning component, an empirical screening step is performed to identify a subset that is informative with respect to the observed end-to-end delay.

For this analysis, previously generated communication scenarios (cf. Chapter 3) were transferred into the *cocoon* environment such that each message can be represented in the unified format $\mathbf{m} = (t_s, p, \eta(t_s), \nu(t_s), d)$, as defined in Section 4.3. To obtain comparable and robust statistics, scenarios were filtered to include only those with at least 50 messages. Moreover, messages with delay times above 3 000 ms were excluded, as such values typically indicate a Transmission Control Protocol (TCP) reconnect and constitute outliers with respect to the modeled communication dynamics. Due to computational burden, a random sample of 5 % of the remaining scenarios was selected for the attribute screening.

Based on this dataset, each candidate attribute was related to the corresponding message delay. As a measure of influence, the Pearson correlation coefficient between each attribute and the end-to-end delay was computed. Pearson correlation was chosen as a computationally efficient linear screening measure suitable for ranking candidate attributes. While Spearman's rank correlation would be more robust to the skewed delay distribution, the primary goal here is relative ranking rather than precise quantification of dependence, and the applied outlier exclusion (messages above 3 000 ms) reduces the influence of extreme values. In addition, the fraction of missing observations (NaN values) and structurally absent values (zeros) was determined for each attribute. Figure 4.4 summarizes both aspects by visualizing correlation values and missing-value shares side by side.

To focus the feature set on attributes that carry explanatory power, only attributes with an absolute correlation above 0.1 were retained as candidates here. The results indicate that several node- and network-level attributes are consistently correlated with delay, whereas

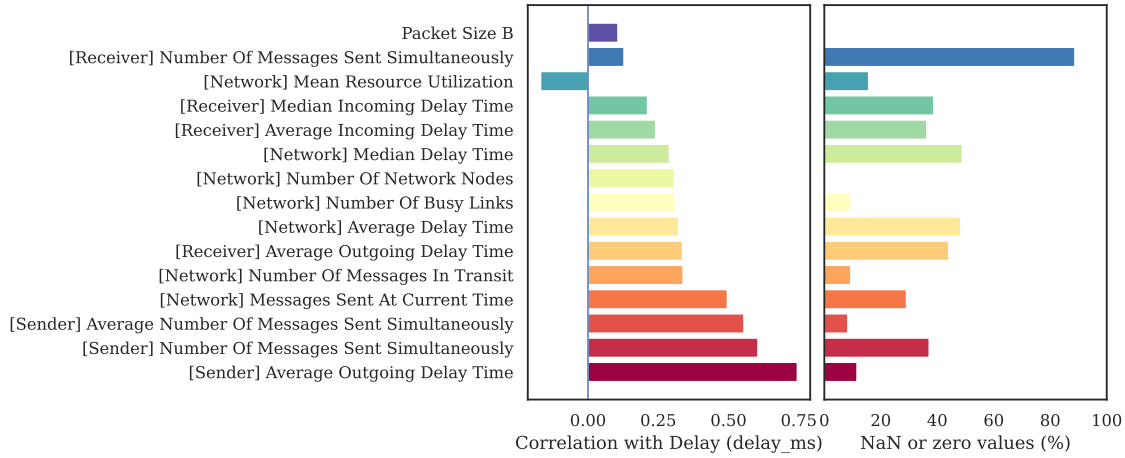


Figure 4.4: State-attribute screening for online learning: Pearson correlation between each candidate attribute and the observed end-to-end delay (left) and the share of missing observations (NaN or zero values) across the evaluated messages (right).

link-level attributes exhibit comparatively weak correlations in the evaluated scenarios. In particular, delay-related aggregates at sender and receiver level, as well as network-level load indicators (e.g., number of messages in transit or number of busy links), show the strongest correlations with the resulting end-to-end delay.

Consequently, the attributes from Table 4.2 were selected for subsequent training and evaluation.

At the same time, the missing-value analysis highlights an important practical limitation. Many aggregate attributes emerge only over the course of a simulation run because they require sufficient observations to become meaningful (warm-up effects). As a result, these attributes may be undefined or take default values immediately after initialization or when nodes/links are newly created. This effect is visible in Figure 4.4 through high shares of NaN or zero values for some attributes. Handling this warm-up behavior is therefore an integral part of the later online processing and is revisited when discussing model training and substitution.

4.6 Chapter Summary

This chapter addressed Sub-RQ 2 by formalizing how the input-output behavior of a detailed communication network simulation can be represented within the meta-model in *cocoon* while preserving essential DES properties. Building on DES theory, requirements for simulation compatibility were derived, emphasizing an explicit event definition, instan-

Table 4.2: List of attributes used, sorted by corresponding components. It is also specified whether the attributes are always available (and if so, a symbol is added for later use).

Component	**Attribute**	**Always Available**
Sender	average outgoing delay (ms)	×
	average incoming delay (ms)	×
	messages sent simultaneously $\eta_{\lvert\text{simultaneously}\rvert}$	✓
Receiver	average outgoing delay (ms)	×
	average incoming delay (ms)	×
	messages sent simultaneously $\eta_{\lvert\text{simultaneously}\rvert}$	✓
Network	average delay (ms)	×
	number of messages in transit $\nu_{\lvert\text{transit}\rvert}$	✓
	number of busy links $\nu_{\lvert\text{busy links}\rvert}$	✓
	number of network nodes $\nu_{\lvert\text{nodes}\rvert}$	✓
	messages sent simultaneously $\nu_{\lvert\text{simultaneously}\rvert}$	✓

taneous state transitions at discrete time instants, and a time advancement strategy that preserves causal ordering. Based on these requirements, a technology-agnostic, graph-based internal representation was introduced, in which communication endpoints and observed sender-receiver relations are constructed online as a directed graph and enriched with network-, node-, and link-level state variables.

To ensure executable integration into coupled (co-)simulation settings, the chapter then formalized event semantics and time advancement using message arrival and message departure events and a next-event execution principle. The resulting simulation routine specifies initialization, online graph updates from observations, event scheduling via predicted or realized delays, and state updates consistent with discrete-event execution. Finally, an empirical screening of candidate state attributes quantified both correlation with end-to-end delay and the prevalence of missing (NaN) or structurally absent (zero) values. This analysis motivated a reduced feature set dominated by node- and network-level load and delay aggregates, while link-level attributes were omitted due to limited explanatory power in the evaluated scenarios. The observed warm-up and missing-value behavior highlights a practical constraint for online operation and motivates explicit handling of undefined or default-valued attributes in later training and substitution steps.

5 Online Approximation of Communication Network Simulations

> Technology can become the "wings" that will allow the educational world to fly farther and faster than ever before – if we will allow it.
>
> *Jenny Arledge*

The previous chapter introduced the internal network representation used by the meta-model, i.e., a graph-based abstraction that captures the dynamically observed communication structure and its state during simulation execution. Building on this representation, this chapter addresses how the behavior of the detailed communication network simulation can be approximated *online* within *cocoon* while preserving the DES-conform event semantics required for integrated simulation.

In Section 2.5, two major challenges of existing meta-modeling approaches were identified. First, many general meta-modeling approaches rely on large and costly training datasets and show limited flexibility when conditions change, for instance due to different scenario parameterizations or shifts in communication behavior. This motivates online learning as a natural remedy: by continuously updating the model on streaming observations, it can cope with concept drift without requiring repeated offline retraining. Moreover, ensemble-based designs are a complementary means to increase robustness, as they can combine diverse predictors and adapt their weighting dynamically, thereby exploiting a "wisdom of the crowd" effect. Second, meta-models for communication network simulations are to date predominantly trained and used in an offline manner, which restricts their applicability in integrated simulation workflows where new conditions emerge during execution.

These observations directly motivate the focus of this chapter and relate to Sub-RQ 2, which targets the online training of a meta-model during execution time such that it can operate as a drop-in DES component. Concretely, the approximation is learned from message observations and state snapshots gathered during the ongoing simulation, enabling the model to adapt to scenario-specific traffic characteristics without requiring complete

retraining beforehand.

A second focus of this chapter concerns a central hypothesis of this thesis: that there exists a substitution threshold beyond which the online-trained meta-model approximates the detailed simulation sufficiently well to replace it entirely during execution. This threshold is the key mechanism that enables the meta-model to transition from an observation mode, in which it learns from the detailed simulation, to a substitution mode, in which it operates independently and thereby eliminates the computational overhead of the detailed simulation altogether. If such a threshold can be identified and reliably estimated at runtime, simulation execution can be accelerated substantially, which in turn enables larger parameter studies and broader sensitivity analyses under computational constraints. Crucially, this acceleration is achieved without requiring prior knowledge of the scenario, as the threshold is determined dynamically based on the observed approximation quality during execution. To analyze this hypothesis, Sub-RQ 3 is investigated, addressing how the meta-model can be evaluated with respect to accuracy and consistency criteria such that a defensible substitution decision can be defined.

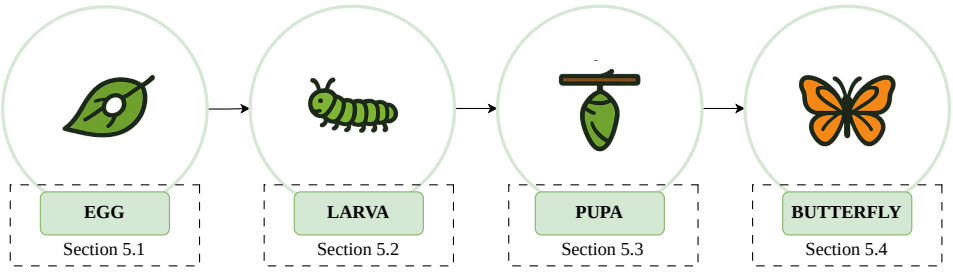


Figure 5.1: Four-phase online learning process used for runtime approximation and potential substitution of the detailed communication network simulation.

Building on the graph-based internal representation introduced in the previous chapter, this chapter presents an *online* approximation method that incrementally learns to reproduce the delay behavior of a detailed communication network simulation during runtime. The method follows a structured four-phase learning process, which was introduced in own, accompanying work [10] and is adopted and extended here in a thesis-consistent context. As illustrated in Figure 5.1[1], the four phases provide a controlled transition from prior knowledge to scenario-specific adaptation and, eventually, substitution. In order to illustrate the complex learning process, the analogy of butterfly metamorphosis was used, where each phase represents a distinct step in the development and refinement of the meta-model:

[1] *Note:* Icons in this figure were generated using ChatGPT v5.2 (OpenAI) (date: 2025-12-18, prompt: "Please generate four icons for the different phases in the metamorphosis of a butterfly"), then edited by the author.

- EGG PHASE (Section 5.1): offline initialization that derives prior predictors from historic samples to mitigate cold-start effects.
- LARVA PHASE (Section 5.2): warm-up phase in which prior predictors are applied while ground-truth delays from the detailed simulator are still observed to collect scenario-specific training data.
- PUPA PHASE (Section 5.3): online adaptation phase that trains a scenario-specific model on streaming data and combines it with the prior predictors (via dynamic weighting) to increase robustness under changing conditions.
- BUTTERFLY PHASE (Section 5.4): substitution phase in which the meta-model may replace the detailed simulation once a confidence criterion indicates sufficiently accurate and consistent approximation.

These different phases are described in more detail in the following sections.

5.1 Egg Phase - Offline Training on Clustered Message Objects

This phase sets the groundwork for the training process that follows. As discussed in Chapter 3, communication scenarios in CPESs display a wide range of characteristics. Consequently, a universal predictive model is inadequate to cover all possible scenarios.

By clustering the comprehensive set of simulation data (generated with the artifact *trace*) into distinct message object clusters based on their properties, it becomes possible to train regressors tailored to the specific requirements of each cluster. This enables accurate and efficient predictions adapted to the characteristics of the communication scenario at hand.

As displayed in Figure 5.2, this phase consists of the loading of training data, the clustering of message objects, and the pre-training of individual regressors on the clusters. In the following, the clustering of message objects based on their properties is described (Subsection 5.1.1). Building on these definitions, clusters in the simulation data generated with the method presented in Chapter 3 are analyzed. Subsequently, Subsection 5.1.3 presents a one-time comparison of different regressor types to select a suitable base learner for the EGG PHASE.

5.1.1 Clustering of Message Objects

Cluster analysis is a statistical method used to group classification objects into homogeneous clusters. According to Bacher et al. [98], good cluster solutions exhibit high intra-

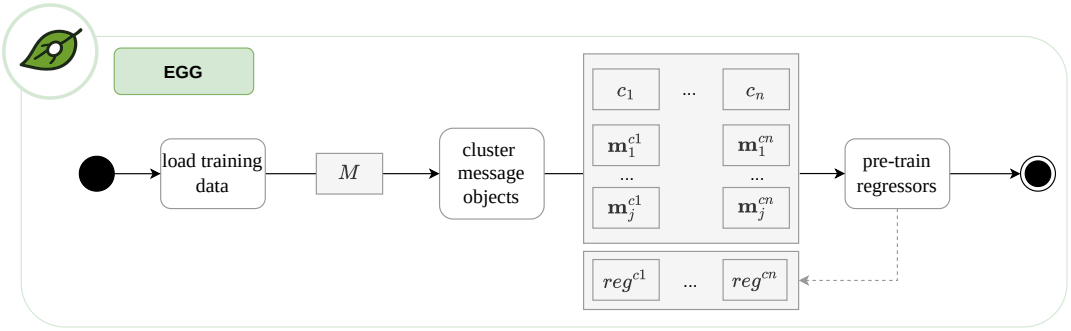


Figure 5.2: EGG PHASE (offline): training data is clustered and used to pre-train cluster-specific regressors that serve as prior predictors for online approximation.

cluster similarity while maintaining heterogeneity between clusters. They should also be interpretable, stable, and compact, with clusters having a meaningful minimum size.

Two primary clustering strategies exist according to von der Hude et al. [99]: *partitional clustering* and *hierarchical clustering*. Partitional clustering assigns objects randomly to clusters initially and reorganizes them iteratively. Hierarchical clustering, in contrast, successively divides objects into subsets or merges them into larger clusters. For this work, hierarchical clustering is chosen due to its flexibility and suitability for datasets with no predefined cluster count.

Distances between clusters are a crucial consideration in clustering. Common distance metrics include single-linkage (minimum distance), complete-linkage (maximum distance), average linkage (mean distance), and centroid distance (distance between centroids) [99].

In this work, centroid linkage is employed as it yields an explicit centroid representation per cluster, which can be reused efficiently during online operation: new observations can be assigned to the closest cluster by comparing to centroids, avoiding costly pairwise distance computations against all historical samples. This is particularly relevant in the proposed four-phase learning procedure, where cluster membership is needed as a lightweight runtime operation. The classification objects here are message objects rather than scenarios, as scenarios inherently encompass diverse messages, making them less suitable for clustering.

The clustering process is designed to group messages based on their similarities in key variables X that correlate with the target variable d_{real}. The selected variables (see Table 4.2) are:

$$X = \{\nu_{|\text{transit}|}, \nu_{|\text{busy links}|}, \nu_{|\text{nodes}|}, \nu_{|\text{sent}|}, p, \eta_{|\text{sent}|}\} \tag{5.1}$$

The scaled Euclidean distance is used to compute the pairwise distances between mes-

sages. Let $\mathbf{m}_i$ and $\mathbf{m}_j$ represent the feature vectors for messages i and j, scaled to have zero mean and unit variance. The distance $dist(\mathbf{m}_i, \mathbf{m}_j)$ is given by:

$$dist(\mathbf{m}_i, \mathbf{m}_j) = \sqrt{\sum_{k=1}^{K} \left(\frac{\mathbf{m}_{i,k} - \mathbf{m}_{j,k}}{\sigma_k} \right)^2} \tag{5.2}$$

Where:

- K: Number of variables in $\mathbf{X}$.
- $m_{i,k}$: Value of the k-th variable for message object i.
- σ_k: Standard deviation of the k-th variable across all messages.

The resulting pairwise distances form a distance matrix $\mathbf{D} \in \mathbb{R}^{|M| \times |M|}$, where M is the set of all message objects, and $|M|$ represents the total number of messages.

Agglomerative hierarchical clustering is applied to the distance matrix $\mathbf{D}$. The process begins with each message in its own cluster and iteratively merges the two closest clusters based on centroid linkage. Let c_p and c_q denote two clusters, and let $\boldsymbol{\mu}(c)$ be the centroid of cluster c (mean feature vector of its members). The centroid-linkage distance is then defined as:

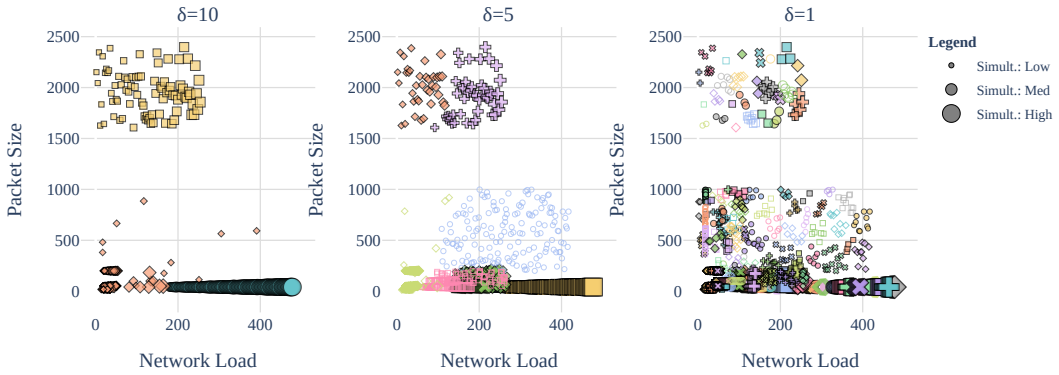
$$dist(c_p, c_q) = \|\boldsymbol{\mu}(c_p) - \boldsymbol{\mu}(c_q)\|_2 \tag{5.3}$$

Clusters are formed by setting a predefined distance threshold δ. At each iteration, the pair of clusters with minimal centroid distance is merged, and merging continues until the minimal inter-cluster centroid distance exceeds δ. The final cluster assignments $\rfloor = \{c_1, c_2, \ldots, c_N\}$ partition the set of message objects into N distinct clusters.

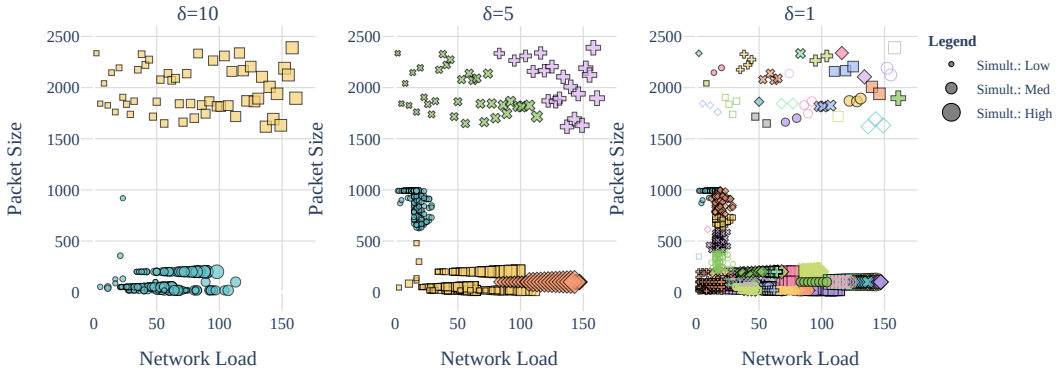
Each cluster c_n ($n \in \{1, \ldots, N\}$) contains messages with similar network states and node behaviors. Predictive models are then tailored to each cluster to improve accuracy and interpretability in specific operating regimes.

5.1.2 Cluster Analysis

In this subsection, an analysis of the resulting message object clusters is conducted. For this, the simulation data resulting from the analysis in Chapter 3 is used. Clusters are constructed for the technologies LTE, LTE450, 5G, and Ethernet and for a distance threshold $\delta = \{10, 5, 1\}$ to illustrate the clustering process. The scenarios are filtered as in Section 4.5 to only those with at least 50 messages and delay times below 3 000 ms. Again, a randomly



(a) Visualization of the clusters for different distance thresholds δ for LTE.



(b) Visualization of the clusters for different distance thresholds δ for Ethernet.

Figure 5.3: Clustering results for different communication technologies under varying distance thresholds δ. Each subplot shows message objects in the space of network load and packet size; marker color and shape indicates cluster membership and marker size encodes simultaneity level. Subplots are arranged left to right with decreasing granularity ($\delta = 10, 5, 1$). For both LTE and Ethernet, smaller thresholds resolve finer cluster structure within dense message regimes, at the cost of a larger number of pre-trained regressors in the EGG PHASE.

selected 5 % of the scenarios are used for clustering. The sample is selected in a way that ensures every communication application is represented in the communication traffic models, capturing diverse communication behaviors within the subset of messages.

To analyze the results of clustering between different technologies and distance thresholds, Figures 5.3a and 5.3b present a visualization of the message objects within their respective clusters. The visualizations illustrate how messages are grouped based on their network characteristics for different communication technologies.

Each subplot represents the clustering results for different distance thresholds (δ), demon-

strating how the number and compactness of clusters change based on the chosen threshold. The message objects are visualized in a 2-dimensional space, where the x-axis represents the overall network congestion, the y-axis represents the packet sizes of message objects and the size of the markers captures the concurrent message transmissions by summing messages sent at the current time and messages sent simultaneously by the sender. The values are computed as follows:

- x-axis (Network Load): $x = load = \nu_{|\text{transit}|} + \nu_{|\text{busy links}|} + \nu_{|\text{nodes}|}$
- y-axis (Packet Size): $y = p$
- Marker size (Simultaneity): $z = sim = \nu_{|\text{sent}|} + \eta_{|\text{sent}|}$

The impact of the distance threshold δ on the clustering results is clearly visible in the figure. A high threshold ($\delta = 10$) results in few or no distinct clusters, as most message objects remain unclustered due to the distance constraint. A low threshold ($\delta = 1$) leads to an excessive number of small, potentially meaningless clusters, as minor variations in network characteristics cause fragmentation. A medium threshold ($\delta = 5$) provides a compact and interpretable representation of the data, balancing cluster separation and internal cohesion.

Given these observations, a threshold of ($\delta = 5$) is selected for the subsequent analysis to ensure meaningful, well-separated, and compact cluster formations.

As previously stated, message objects have been selected for the clustering process as opposed to scenarios. This decision is further reinforced by the distribution of clusters by application, as illustrated in Figure 5.4 for LTE scenarios. It is evident that only certain applications, such as *Customer Information* and *Demand Response*, contain messages from a single cluster. However, numerous other applications, including *Demand Supply Balancing* and *Scheduled Automated Meter Reading*, exhibit a multitude of distinct clusters. To ensure that all messages in a scenario are processed with a regressor that is tailored to the specific properties of the message, it is necessary to cluster message objects at a higher level of detail rather than clusters of scenarios.

Figure 5.5 shows normalized cluster properties (network load, simultaneity, and packet size) alongside message delay for each communication technology. Values are normalized via Min-Max scaling; error bars indicate within-cluster standard deviation; delay is plotted on the secondary y-axis. Across all technologies, clusters with elevated network load and simultaneity consistently exhibit higher message delays, while packet size shows little systematic relationship with delay. This pattern is most pronounced in LTE and LTE450, where distinct high-load clusters (e.g., LTE Cluster 5, LTE450 Cluster 6) coincide with

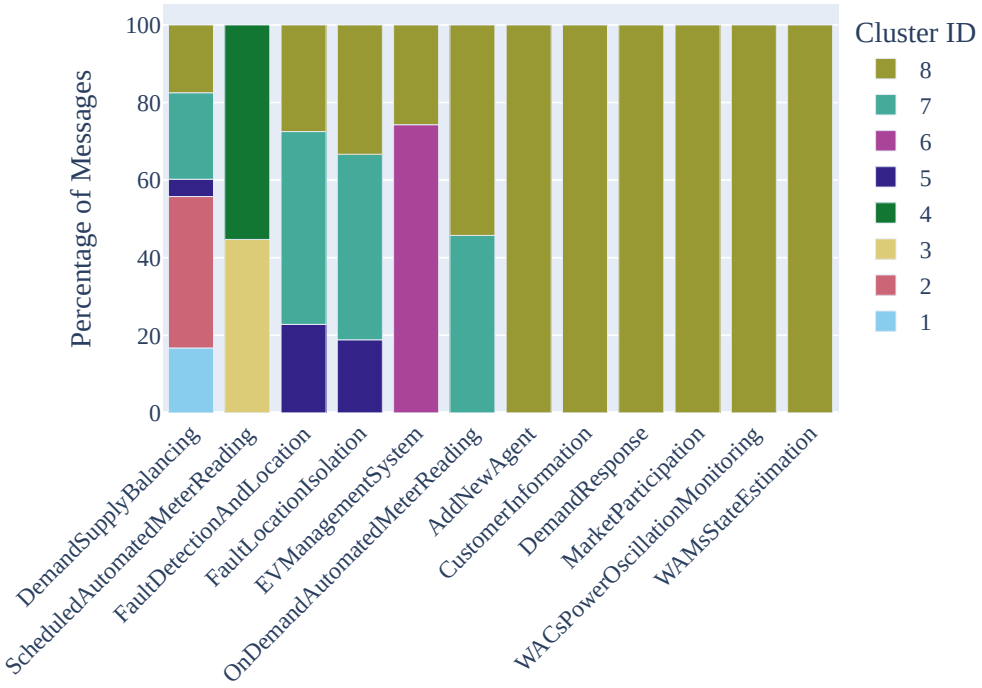


Figure 5.4: Cluster distribution by application for scenarios with LTE communication network models.

clear delay peaks, and is also visible in 5G. For Ethernet, load is more evenly distributed across clusters, resulting in comparatively stable and low delay values throughout.

These results show how the previously performed clustering of the message objects firstly successfully classifies the messages with regard to their properties and secondly also differentiates the messages with regard to their behavior in the simulated network. In the subsequent phases of online learning, this makes it possible to classify the message objects even before the actual delay time is observed, thus enabling more targeted prediction.

5.1.3 Cluster Pre-Training

Within the EGG PHASE, the objective of cluster pre-training is to obtain a set of cluster-specific regressors that provide message-property-dependent delay predictions during the subsequent online phases. However, the selection of a suitable regressor type is not performed repeatedly. Instead, different model families are compared once in this section to

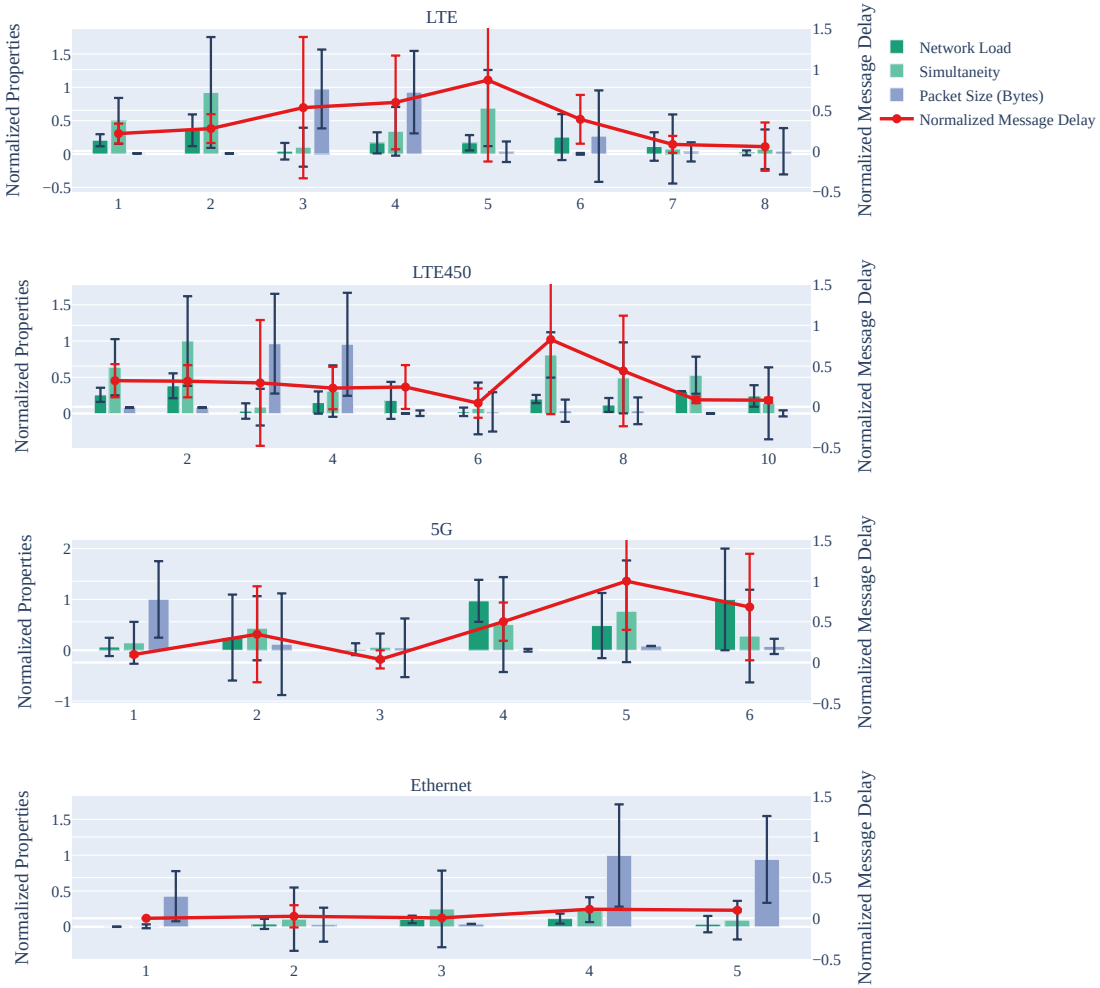


Figure 5.5: Cluster properties compared with message delays (normalized).

identify a base learner that offers a favorable trade-off between prediction accuracy and computational cost in the considered data. The selected regressor type is then used consistently in the EGG PHASE: for each cluster, a separate regressor instance is trained to yield message-specific predictions.

For the one-time model selection, four regression model families are evaluated using their default configurations as provided by the *scikit-learn*[2] library in Python: a Decision Tree Regressor, a Random Forest Regressor, a Linear Regression model, and a simple

[2]version 1.6.1, `https://scikit-learn.org/stable/`, last access: 2025-12-19

Multi-Layer Perceptron (MLP) regressor. These model families were selected to cover a representative range of learning biases and model complexities, ranging from linear relationships to nonlinear and ensemble-based approaches, while being well-established and computationally efficient. In addition, they offer complementary methodological properties, such as the high interpretability of decision trees, the improved predictive performance and robustness of ensemble methods, and the ability of neural networks to capture complex nonlinear dependencies. Training and testing are conducted independently per cluster of the data set introduced in Subsection 5.1.2, using a 70/30 split (training/testing).

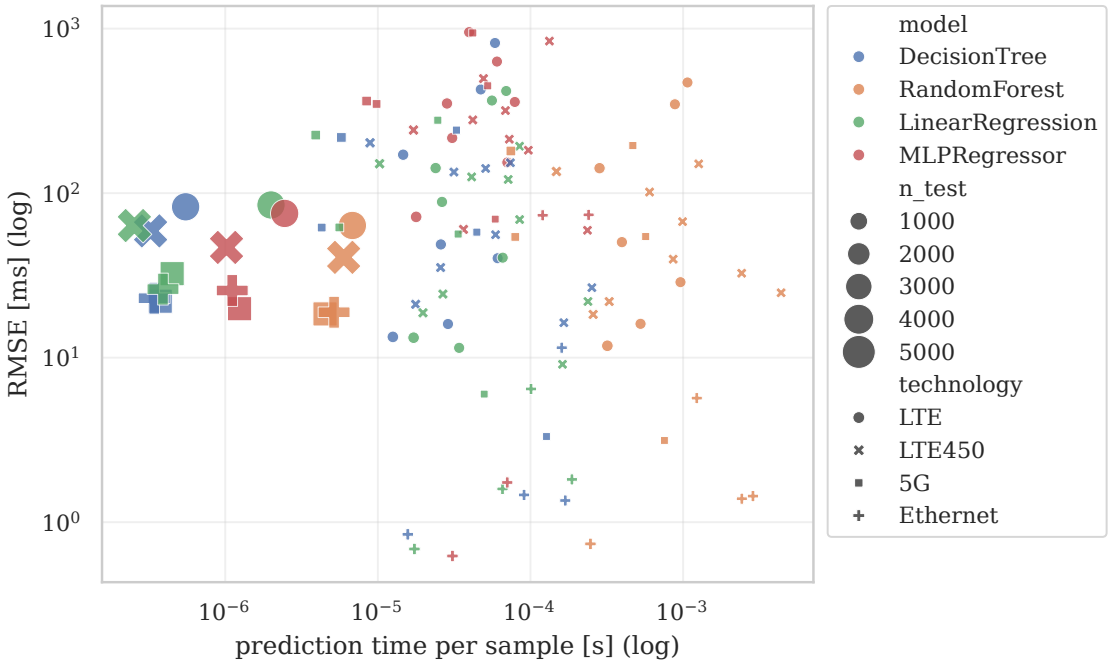


Figure 5.6: Trade-off between prediction errors (as RMSE) and prediction time per sample for different models (Decision Tree, Random Forest, Linear Regression, MLP Regressor) and data sets (different clusters and technologies).

Figure 5.6 visualizes the trade-off between prediction errors of message delay times (measured as Root Mean Squared Error (RMSE)) and prediction time per sample. Cluster instances are shown with different marker sizes, where larger markers correspond to larger clusters, and marker styles distinguish the considered network technologies. A clear separation is visible with respect to inference speed: the fastest predictions are located on the left of the log-log plot and are dominated by Linear Regression and Decision Tree models, whereas Random Forest predictions are shifted notably to the right, indicating higher per-sample computational cost. At the same time, accuracy is strongly regime-dependent:

within a fixed prediction-time band, the RMSE varies substantially across clusters and technologies, reflecting that delay behavior changes with the underlying message-object properties and network conditions. Overall, Random Forest models tend to populate a "slower but often lower-error" region, while the largest clusters are predominantly located in the fast-runtime region. The MLP regressor exhibits acceptable performance mainly for larger clusters but degrades for small clusters, indicating limited robustness under data sparsity.

Based on this one-time comparison, the Decision Tree Regressor is selected as the base learner in this work. It provides a robust accuracy-runtime compromise and, crucially, enables training many lightweight cluster-specific models. Consequently, in the EGG PHASE the chosen regressor type is used to pre-train one decision tree per cluster, yielding message-specific priors that are subsequently refined by online learning and dynamic weighting in later phases. For completeness, Chapter 6 also includes a Random Forest Regressor as an overall baseline in the integrated setting.

Finally, the focus of this work lies on the overall system design and the online learning and substitution algorithm rather than on identifying an optimally tuned regressor for each application. More specialized model selection and hyper-parameter optimization may further improve approximation quality in specific deployments and are therefore considered a direction for future work.

5.2 Larva Phase - Simulation Warm-up

The LARVA PHASE phase marks the beginning of the simulation warm-up process, where the original communication simulation is observed to approximate its input-output behavior. At this stage, it is essential to warm up the simulation before collecting meaningful data because of the lack of knowledge about the behavior and states inherent in the simulation.

The simulation warm-up phenomenon is a well-documented issue in simulation theory, as described by Mahajan et al. [100]. Most simulation models start in an empty and idle state, taking some time to reach a steady state, during which they are considered to be in a transient state. This problem is known as the *initialization bias* or *start-up problem*. To overcome this issue, simulation models are typically run for a specified warm-up length, after which observations are recorded. Various methods exist to determine this warm-up length, including graphical, statistical, and heuristic approaches.

In online learning, a similar challenge arises as when recording results from simulation models. However, since the meta-model needs time to be trained online, conventional warm-up length determination methods can be bypassed. Instead, a substitution threshold is defined, after which the online-trained model substitutes the original simulation (for details

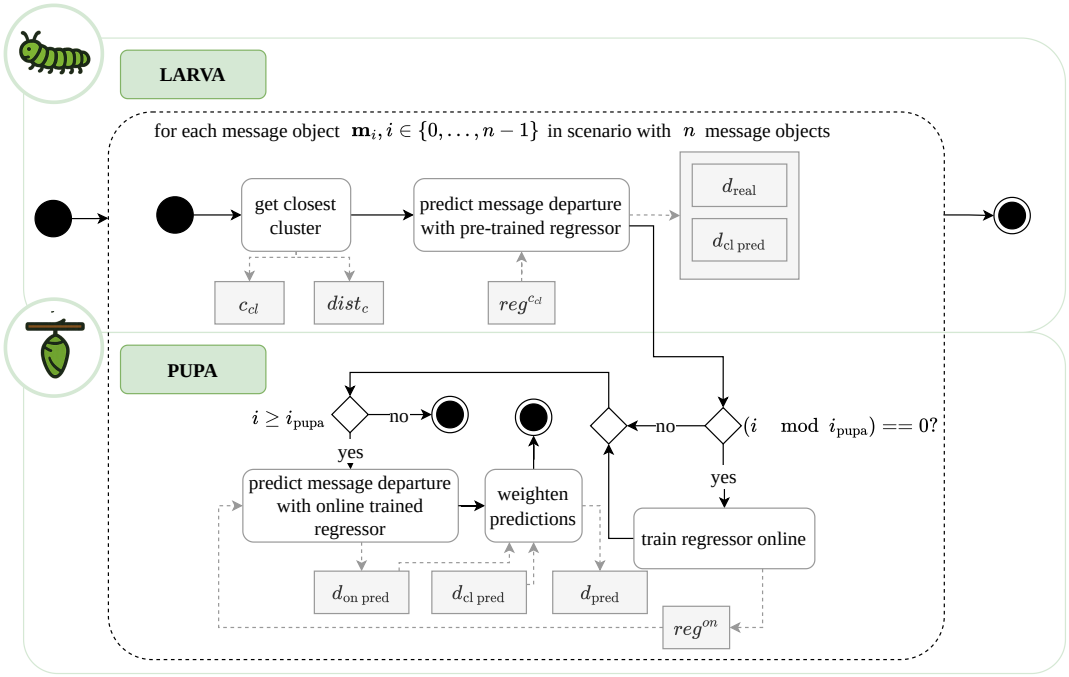


Figure 5.7: LARVA PHASE and PUPA PHASE within the online training algorithm.

refer to Section 5.4).

During the LARVA PHASE, the simulation is warmed up by iterating over messages in the FES, as shown in Figure 5.7. For each message object $\mathbf{m}_i$ in the FES, its feature vector is compared to the centroids of all clusters obtained in the EGG PHASE. The message object is assigned to the cluster c_{cl} whose centroid has the smallest Euclidean distance $dist_c$ to the message's feature vector. Subsequently, the cluster-specific regressor associated with this nearest cluster $reg^{c_{cl}}$ is used to predict the delay $d_{\text{cl pred}}$ for $\mathbf{m}_i$.

An illustrative example of this and the following PUPA PHASE will be provided in Subsection 5.3.1.

5.3 Pupa Phase - Scenario-specific Online Training

In the PUPA PHASE, an additional regressor is trained online on the current scenario data. As shown in the lower part of Figure 5.7, this phase is only conducted for messages $\mathbf{m}_i$ in a scenario after the index i of the message is above some pre-defined constant i_{pupa}. This is the case because a certain amount of information about the current scenario must already be available. This constant i_{pupa} is also used as a kind of batch size in the course

of the scenario, after which the online regressor reg_{on} is re-trained.

This prediction of the online trained regressor reg_{on} is combined with the prediction of the pre-trained regressor $d_{\text{cl pred}}$ by weighting these values. This weighting depends on the previous error of the regressors. Here, the most recent predictions should be given more weight, as an increasing amount of information becomes available as the scenario progresses.

The Exponential Weighted Moving Average (EWMA) [101] for prediction errors is computed recursively as:

$$EWMA(t) = \alpha \cdot e_t + (1 - \alpha) \cdot EWMA(t - 1) \tag{5.4}$$

where e_t is the absolute prediction error at time step t and $\alpha \in (0, 1]$ is a smoothing factor that controls the influence of recent errors versus past errors. The initial condition is set as $EWMA(0) = e_0$.

This smoothing is applied separately to the prediction errors of the cluster-based and online-trained regressors to track their recent accuracy.

The weighted prediction $d_{\text{w pred}}$ is then calculated as follows:

$$d_{\text{w pred}} = \frac{d_{\text{on pred}} \cdot EWMA_{\text{cl}}(t_s) + d_{\text{cl pred}} \cdot EWMA_{\text{on}}(t_s)}{EWMA_{\text{cl}}(t_s) + EWMA_{\text{on}}(t_s)} \tag{5.5}$$

with $EWMA_{\text{cl}}(t_s)$ being the exponentially weighted moving average of absolute errors until simulation time of the message arrival t_s for the cluster-based regressor, and $EWMA_{\text{on}}(t_s)$ being the exponentially weighted moving average of absolute errors for the online-trained regressor until the time t_s.

This formulation adaptively balances the two regressor predictions based on their recent accuracy, ensuring that the final, weighted prediction prioritizes the model with lower errors over time.

5.3.1 Illustrative Example

An example scenario was selected according to the following criteria:

1. The scenario contains message objects from different clusters.

2. The scenario contains a medium number (≈ 150) of message objects.

The remaining selection was random. The resulting example scenario is the communication application *Fault Detection and Location* with 158 message objects and the technology 5G.

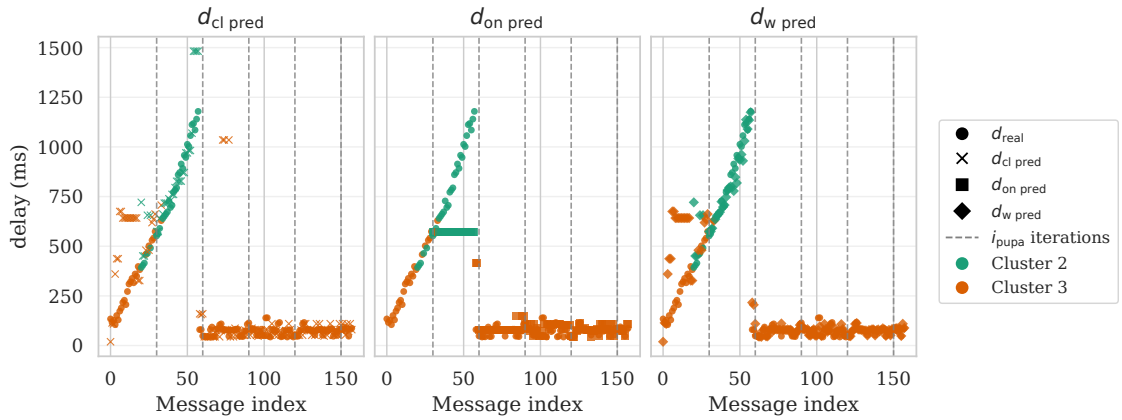


Figure 5.8: Predictions $d_{\text{cl pred}}$, $d_{\text{on pred}}$ and $d_{\text{w pred}}$ compared to real values d_{real}.

Larva phase In the left part of Figure 5.8 the predictions $d_{\text{cl pred}}$ of the cluster-specific regressors are compared to the real values d_{real}. The message objects in the selected example scenario are assigned to either cluster 2 or cluster 3. It can be seen that the predictions are approximately correct, but some dynamics in the delay times cannot be predicted. In addition, the prediction results do not improve over the course of the scenario because the regressors are pre-trained on the historical data from the clusters and are not refined during the scenario execution.

Pupa phase The previously selected example scenario has now been expanded to include the online training of a regressor on the scenario data. The batch size constant i_{pupa} was chosen in a way that multiple learning iterations can be observed in during the simulation process with $i_{\text{pupa}} = 30$. The iterations after i_{pupa} message objects is indicated in Figure 5.8 as dotted, gray lines.

The message departure is then predicted with the online trained regressor reg^{on}, resulting in an additional prediction $d_{\text{on pred}}$. This prediction value is visualized for the example scenario in the middle part of Figure 5.8. The data shows that no predictions $d_{\text{on pred}}$ are made until the message object with the index i_{pupa} is reached. The online regressor is then trained for the first time, but initially makes poor predictions. However, after a few iterations an improvement in the predictions can be seen.

The resulting, weighted predictions $d_{\text{w pred}}$ for the exemplary scenario are plotted in the right part of Figure 5.8. It can be seen that the weighted predictions reflect the real values relatively accurately, especially in the later stages of the scenario. In addition, at the beginning of the scenario, the cluster-based prediction is initially weighted higher so that information can first be accumulated for the online trained regressor.

5.4 Butterfly Phase - Simulation Substitution

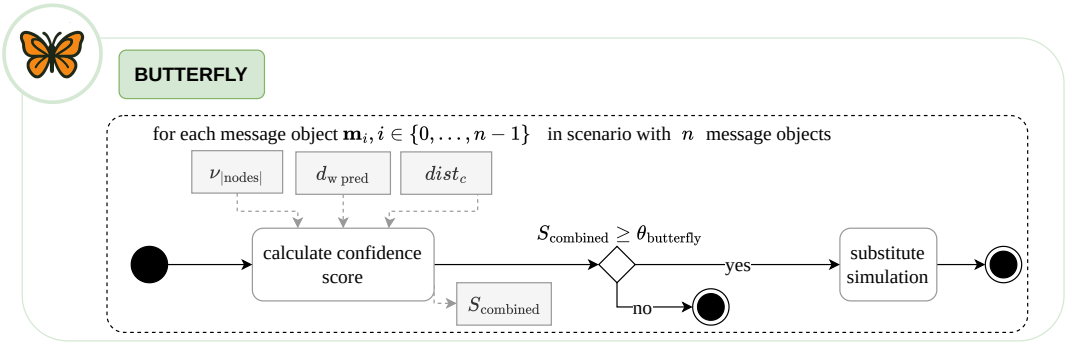


Figure 5.9: Butterfly phase in the online training algorithm.

After the initialization and adaptation phases (Egg phase, Larva phase, Pupa phase), the meta-model has accumulated scenario-specific observations and continuously refined its predictors. The Butterfly phase formalizes the transition from "assisted" approximation (where ground-truth delays are still produced by the detailed simulator) to *simulation substitution*, i.e., using d_{pred} as the effective delay time for subsequent messages and thereby replacing the detailed communication network simulation during runtime.

A substitution decision must be conservative: replacing the simulator too early can introduce systematic timing deviations that may accumulate over the course of a simulation run, since uncertainties of co-simulated subsystems propagate and compound across simulation components [102]. At the same time, delaying substitution unnecessarily reduces the achievable speed-up.

For this reason, substitution is not triggered by a single metric (e.g., a point estimate of error). Relying solely on a single indicator such as prediction accuracy has been identified as insufficient in online learning settings, since it fails to capture the full range of conditions under which a model may degrade [103, 104]. Instead, a multi-criteria confidence assessment is employed to capture complementary aspects of approximation reliability. This assessment considers not only whether the current error is small, but also whether it is improving, whether the current operating regime is covered by the trained priors, and whether the underlying network representation is sufficiently stable.

The substitution decision in the Butterfly phase combines four weighted factors to determine when the meta-model has achieved sufficient accuracy and stability to replace the original communication simulation:

1. **Error Trend Factor** f_{trend} *(is the approximation improving?)*

2. **Error Level Factor** $f_{\textbf{level}}$ *(is the current approximation sufficiently accurate?)*
3. **Cluster Distance Factor** $f_{\textbf{cluster}}$ *(are the current message object properties similar to those seen during pre-training?)*
4. **Network Topology Factor** $f_{\textbf{topology}}$ *(is the observed communication network topology stable?)*

This selection reflects typical failure modes of online approximation and mirrors the multi-source nature of uncertainty in co-simulation systems, where parameter uncertainty, input attribute uncertainty, and output attribute uncertainty constitute distinct and complementary sources of unreliability [102]. For instance, a low mean error alone can be misleading if errors are currently increasing (regime change), if predictions are made far outside the distribution of the pre-trained clusters (out-of-distribution), or if the network graph is still evolving rapidly (new nodes/edges leading to different congestion behavior). Consequently, the weighted aggregation of complementary factors yields a single confidence score that can be compared to a substitution threshold.

The error trend factor f_{trend} evaluates whether the meta-model is converging in the recent past. Let $e_t^{(w)}$ denote the absolute prediction error of $d_{\text{w pred}}$ at message index t, i.e., $e_t^{(w)} = |d_{\text{actual}} - d_{\text{w pred}}|$. Let $\hat{a}$ denote the slope obtained by fitting a linear function to the sequence of the i_{pupa} most recent such errors $\{e_{t-i_{\text{pupa}}+1}^{(w)}, \ldots, e_t^{(w)}\}$ via ordinary least squares. A non-negative slope indicates stagnation or degradation of predictive quality and therefore yields a confidence contribution of zero. A negative slope increases the confidence up to a saturation level, which prevents overly strong influence of short-term fluctuations:

$$f_{\text{trend}} = \begin{cases} 0.0 & \text{if } \hat{a} \geq 0 \\ \min\left(1.0, \frac{|\hat{a}|}{50.0}\right) & \text{if } \hat{a} < 0 \end{cases} \tag{5.6}$$

The error level factor f_{level} captures the current magnitude of approximation errors. It is computed from the mean absolute error of $d_{\text{w pred}}$ over the recent window of i_{pupa} observations, normalized by an expected maximum error bound $E_{\max}$. The normalization maps errors to a unit interval and yields higher confidence if the observed error level remains well below the maximum tolerated error:

$$f_{\text{level}} = 1.0 - \min\left(1.0, \frac{\overline{e}_{\text{recent}}^{(w)}}{E_{\max}}\right) \tag{5.7}$$

where $\overline{e}_{\text{recent}}^{(w)} = \frac{1}{i_{\text{pupa}}} \sum_{j=t-i_{\text{pupa}}+1}^{t} e_j^{(w)}$ is the mean absolute error of $d_{\text{w pred}}$ over the recent window, and $E_{\max}$ is a configurable expected maximum error threshold. In contrast to

f_{trend}, which is sensitive to changes, f_{level} serves as an absolute quality gate. Both are required because a decreasing error trend can still occur at an unacceptably high error level.

The cluster distance factor f_{cluster} addresses representativeness with respect to the pre-training data. Even if errors appear small in a short recent window, substitution can be risky if the current messages are far from the pre-trained regimes (e.g., a new traffic pattern or a rare operating condition). Therefore, f_{cluster} measures how close the current message's feature vector $\mathbf{m}_c \in X$ is to the closest cluster centroid obtained in the EGG PHASE:

$$f_{\text{cluster}} = 1.0 - \min\left(1.0, \frac{d_{\min}}{d_{\max}}\right) \tag{5.8}$$

where $d_{\min} = \min_k \|\mathbf{m}_c - \boldsymbol{\mu}(c_k)\|_2$ is the minimum Euclidean distance to any cluster centroid $\boldsymbol{\mu}(c_k)$, and $d_{\max} = 100 \cdot \delta$ is a maximum expected distance derived from the clustering distance threshold δ configured in the EGG PHASE. Intuitively, this factor penalizes out-of-distribution situations: when $d_{\min}$ is large, confidence decreases even if recent errors are temporarily low.

The network topology factor f_{topology} captures the stability of the underlying communication graph representation. Substitution is most meaningful once the communication structure has largely stabilized. Frequent changes in node participation typically imply changing routing and congestion dynamics, which can invalidate learned mappings. This concern is particularly relevant in co-simulation contexts, where simulators with high dependency on other components require additional care to ensure that approximation quality is not compromised by structural change [102]. Hence, f_{topology} decreases confidence when the node count changes strongly between consecutive BUTTERFLY PHASE evaluations:

$$f_{\text{topology}} = 1.0 - \min\left(1.0, 2 \cdot \frac{|N_t - N_{t-1}|}{N_{t-1}}\right) \tag{5.9}$$

where N_t and N_{t-1} denote the number of nodes in the communication graph at the current and previous BUTTERFLY PHASE evaluation, respectively. The factor of 2 implies that a relative node count change of 50 % or more reduces confidence to zero. While node count is a simple metric, it is computationally lightweight and reflects a key prerequisite for substitution: if the set of participating nodes is still changing, the meta-model should continue observing ground truth to remain reliable.

The final substitution decision is based on a weighted combination of the four factors:

$$S_{\text{combined}} = w_{\text{trend}} \cdot f_{\text{trend}} + w_{\text{level}} \cdot f_{\text{level}} + w_{\text{cluster}} \cdot f_{\text{cluster}} + w_{\text{topology}} \cdot f_{\text{topology}} \tag{5.10}$$

where $w_{\text{trend}}, w_{\text{level}}, w_{\text{cluster}}, w_{\text{topology}}$ are user-configurable weights that sum to 1.0. The weights and the threshold $\theta_{\text{butterfly}}$ can be configured to reflect application-specific requirements, as the appropriate balance between conservativeness and speed-up is inherently dependent on the research question and system under study [102].

The substitution threshold is reached when:

$$S_{\text{combined}} \geq \theta_{\text{butterfly}} \tag{5.11}$$

where $\theta_{\text{butterfly}}$ denotes the confidence threshold for meta-model substitution and can likewise be configured to control the conservativeness of the substitution decision.

5.5 Chapter Summary

This chapter introduced the online approximation approach implemented in *cocoon* for replacing detailed communication network simulations during runtime while preserving DES-conform event semantics as an answer to Sub-RQ 2. Motivated by the research gap that existing meta-models are predominantly offline and often require large, costly training data, the chapter established an online learning algorithm that continuously adapts to streaming observations and naturally supports changing conditions.

The method follows a structured four-phase procedure that provides a controlled transition from prior knowledge to scenario-specific adaptation and, eventually, simulation substitution. In the Egg phase, historical message objects are clustered via hierarchical clustering with centroid linkage. The distance between two clusters is defined as the Euclidean distance between their respective centroids - the mean feature vectors of the message objects they contain - yielding explicit centroids that enable the efficient runtime assignment of new messages to their closest cluster. Cluster-specific regressors are then pre-trained to provide message-property-dependent priors. A one-time comparison of several model families (Decision Tree, Random Forest, Linear Regression, MLP) demonstrated the trade-off between prediction accuracy and computational overhead; based on this trade-off, the Decision Tree Regressor was selected as the base learner that is subsequently used consistently to train one regressor per cluster.

In the Larva phase, the simulation warm-up is performed by processing messages while still observing ground-truth delays from the detailed simulator; cluster membership is determined by nearest-centroid distance and the corresponding cluster regressor provides the initial prediction. In the Pupa phase, an additional scenario-specific regressor is trained online after a configurable batch size i_{pupa}, and its prediction is combined with the cluster-based prediction using error-adaptive weighting based on EWMA, thereby

implementing an ensemble-style mechanism that emphasizes the currently more accurate predictor.

Finally, the BUTTERFLY PHASE defined the substitution mechanism and its underlying hypothesis: if the approximation becomes sufficiently accurate and stable, the detailed simulation can be replaced to accelerate execution. By this, the Sub-RQ 3 has been answered. To make this decision robust, substitution is formulated as a multi-criteria confidence assessment combining (i) error trend, (ii) error level, (iii) distance to the closest cluster centroid (representativeness), and (iv) topology stability; these factors are aggregated via user-configurable weights and compared to a configurable threshold $\theta_{\text{butterfly}}$.

Building on these algorithmic components, the next chapter evaluates the proposed online approximation method (Sub-RQ 2 and Sub-RQ 3) in comparison to alternative communication modeling approaches.

6 Evaluation and Comparative Analysis

> Everything should be made as simple as possible, but not simpler.
>
> *Albert Einstein*

This chapter addresses Sub-RQ 4 by evaluating the proposed online-trained communication meta-model with respect to its performance in comparison to established communication modeling approaches. In particular, the analysis focuses on the extent to which the meta-model achieves its intended goal of reducing computational complexity relative to detailed communication simulations while maintaining higher fidelity than simplified analytical models.

To this end, the communication modeling approaches introduced in Section 2.4 are applied to a set of representative communication scenarios derived from combinations of traffic models and network models as defined in Chapter 3. These scenarios reflect the diversity of communication behaviors observed in energy system applications and provide a consistent basis for comparison across modeling paradigms.

The evaluation follows a comparative perspective and assesses the different approaches with respect to selected NFRs. In particular, accuracy, performance efficiency, and scalability are examined in detail, with adaptivity discussed as a secondary criterion mainly in the presentation of the simulation environment. The analysis thereby complements the methodological contributions presented in the previous chapters and provides empirical evidence for the practical trade-offs between detailed simulation, analytical modeling, and online meta-modeling approaches, as motivated by the limitations and open gaps identified in prior work.

6.1 Simulation Environment

The concepts introduced in the preceding chapters are instantiated within a unified simulation environment that serves as the basis for the comparative evaluation. The aim is to assess different communication models under identical conditions, while isolating the impact of the model from the application.

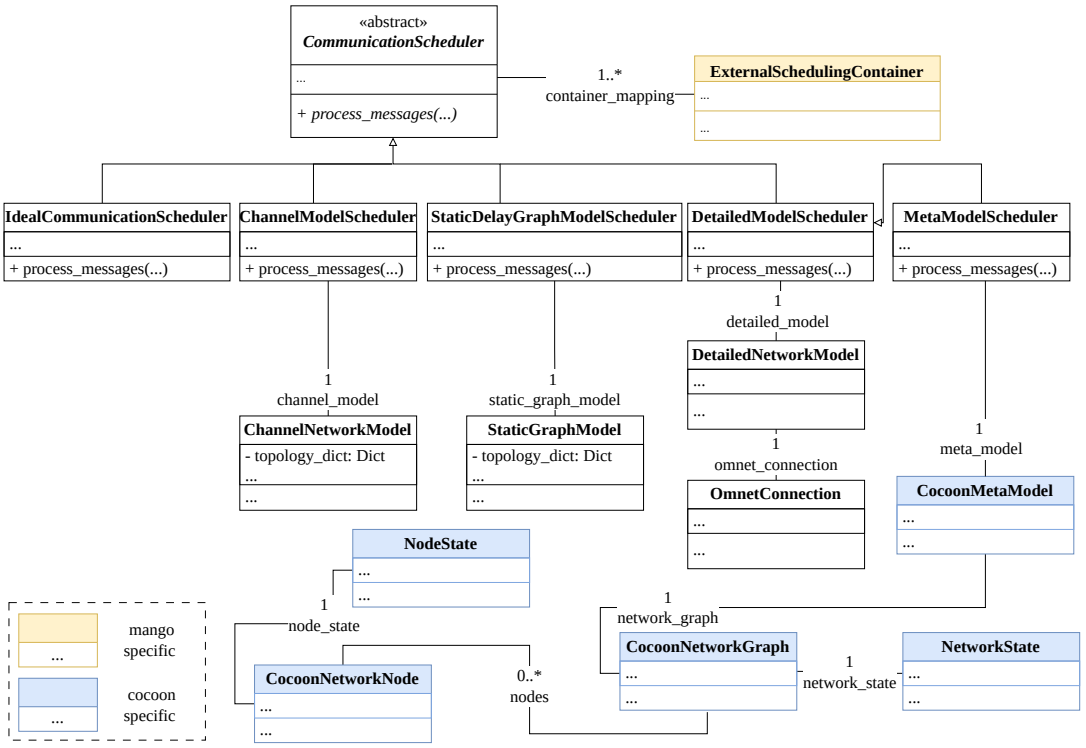


Figure 6.1: Overview of the evaluation setup and scheduler architecture. The simplified diagram illustrates the hierarchical relationships between different communication models and schedulers.

To demonstrate the applicability of the proposed online meta-model, it is integrated into the agent-simulation framework *mango* [39]. The framework is implemented in Python and provides predefined abstractions for agent design, such as roles that encapsulate agent capabilities and behaviors. Agents are assigned to so-called containers, which are responsible for managing message exchange both within and across containers. This abstraction reflects real-world deployments, where individual containers can be mapped to physical devices and communication is realized via concrete protocols such as Message Queueing Telemetry Transport (MQTT) or TCP.

For the purpose of communication-aware simulation, *mango* offers `ExternalScheduling` containers, a specialized container type that allows the implementation of custom scheduling logic. While time in *mango* typically advances in real time, this container type enables an event-based advancement of simulation time. Only the execution of individual container steps is bound to real time, whereas inter-container communication and scheduling are handled in abstract simulation time. This behavior is essential for integrating communication

models that follow discrete-event simulation principles, as such models inherently rely on the ability to advance simulation time in response to discrete events rather than in continuous real-time increments. Without this capability, communication delays computed by an external simulator could not be correctly reflected in the agent simulation, since real-time execution would decouple the timing of message arrivals from the simulated agent behavior. The event-based scheduling mechanism thus ensures that causality between communication events and agent reactions is preserved throughout the simulation.

An overview of the resulting evaluation setup and scheduler architecture is shown in Figure 6.1. All communication modeling approaches are encapsulated in dedicated scheduler implementations that inherit from a common abstract base class, `CommunicationScheduler`. This design ensures a modular architecture in which different communication models can be exchanged transparently without modifying the simulated scenario. Each scheduler internally maintains a distinct model instance that realizes the functionality of the respective communication modeling approach - details of these models are discussed in the following subsection. The mapping of *mango*-specific containers to the scheduler is provided explicitly, enabling the scheduler to mediate all message exchanges.

The overall scheduling logic proceeds as follows. The simulation starts at time zero and proceeds iteratively until a termination condition is met. In each iteration, all containers are stepped, returning outgoing messages and the next scheduled agent activities. Messages are then processed by the selected communication model, and, if required, waiting for communication simulation results is handled explicitly, which is particularly relevant for detailed network simulations. Simulation time is advanced to the next event, determined either by the earliest scheduled message arrival or the next agent activity. If no future events remain or the predefined simulation duration is reached, execution terminates.

This implementation ensures that the scheduling logic and communication modeling are fully decoupled from the agent behavior and scenario definition. As a result, arbitrary *mango* scenarios (and thus a wide range of communication traffic patterns) can be evaluated consistently across all considered communication modeling approaches.

The design of the simulation environment also serves to demonstrate that the proposed meta-model satisfies the NFR of *adaptivity*. By integrating the meta-model into *mango* solely via the abstract `CommunicationScheduler` interface, it can be deployed without modifications to agent implementations or scenario definitions. This integration shows that the meta-model is not tied to a specific simulation framework or communication setup, but can be incorporated into heterogeneous environments that support event-based scheduling while adapting to different communication scenarios during execution.

6.1.1 Communication Modeling Approaches

Multiple communication modeling approaches are implemented within the evaluation setup to enable a systematic comparative analysis with respect to the defined NFRs across a diverse set of communication scenarios. This comparison is motivated by the trade-offs identified in Section 2.4: analytical models offer high computational efficiency at the cost of reduced fidelity, whereas detailed communication simulators provide accurate, technology-specific representations but incur substantial computational overhead.

Based on these considerations, two analytical models (a channel model and a static delay graph model) and one detailed communication model are selected as reference models for comparison with the proposed online-trained meta-model implemented in *cocoon*. Offline-trained meta-models are intentionally excluded from the comparison. While such models can achieve high predictive accuracy in narrowly defined settings, they typically rely on extensive, scenario-specific training data and fixed feature representations that must be generated prior to simulation. In coupled power, control, and communication system simulations, this assumption is difficult to satisfy, as communication behavior emerges dynamically from device interactions, control logic, and scenario evolution. More importantly, offline-trained meta-models are usually formulated as static input–output predictors ("black box") without an explicit notion of internal simulation state or event handling. As a consequence, they cannot be integrated as simulation models that participate in event-based scheduling and time advancement, but instead act as external predictors detached from the simulation loop. This prevents their transparent integration into heterogeneous co-simulation environments and violates the NFR of *adaptivity* targeted in this work.

In contrast, the selected models span the relevant spectrum of communication modeling approaches that can be integrated transparently into heterogeneous co-simulation environments: analytical models provide lightweight, adaptable abstractions with minimal integration effort, while the detailed communication model represents the upper bound in fidelity at the cost of computational complexity. All approaches are integrated uniformly into the simulation environment through a common abstraction, enabling a controlled and meaningful comparison focused on the trade-off between accuracy, performance efficiency, scalability, and adaptivity. As illustrated in Figure 6.1, each communication modeling approach is realized by a concrete scheduler that inherits from the abstract `CommunicationScheduler` class and internally maintains a corresponding model instance. This design ensures consistent scheduling behavior while allowing the internal logic of each communication model to differ.

The detailed communication model relies on a discrete-event network simulation executed in *OMNeT++* [37]. The interaction between *mango* and *OMNeT++* follows a structured

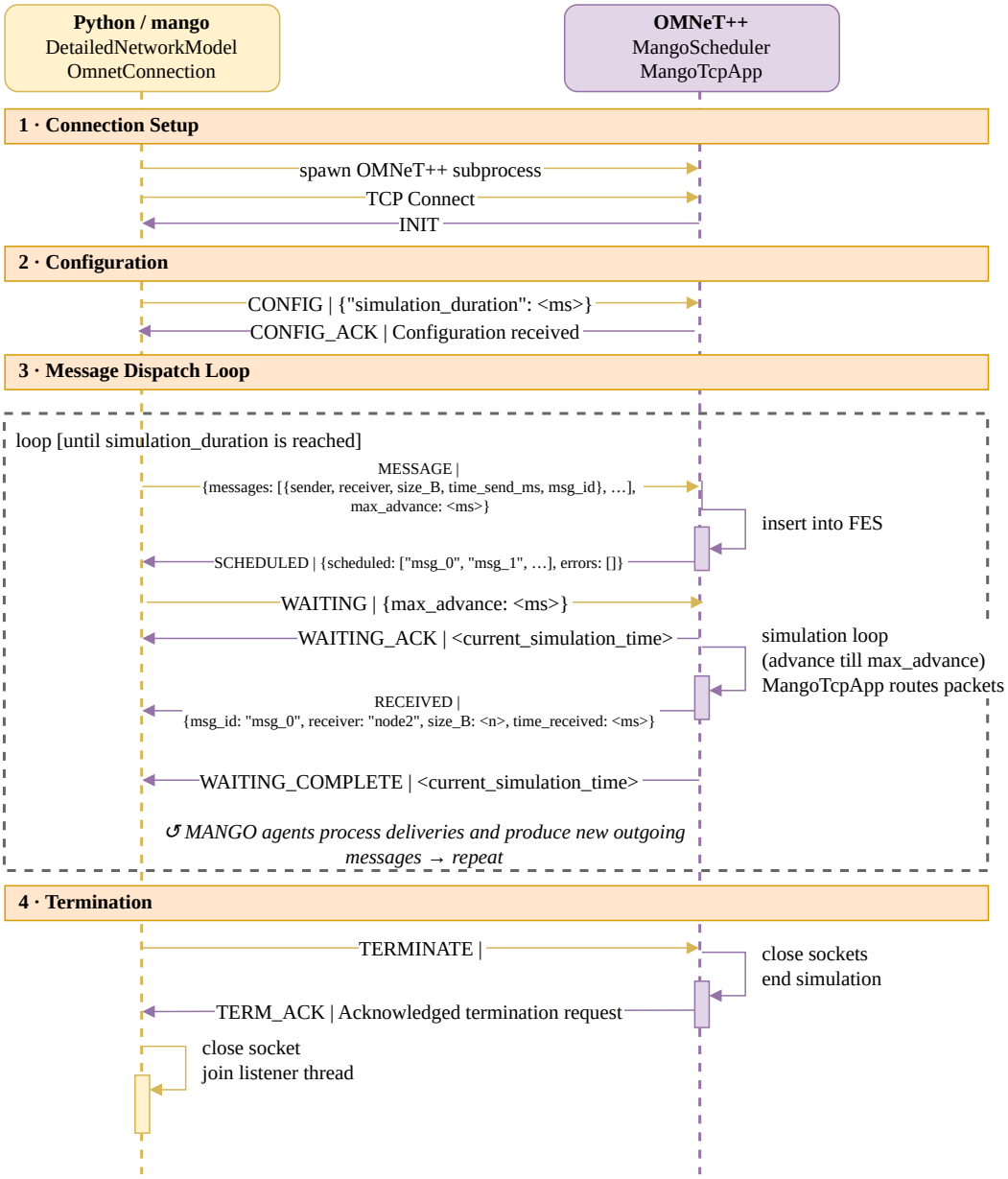


Figure 6.2: Sequence diagram of the communication between the *mango* Python environment and *OMNeT++* for co-simulation of detailed network behavior.

process, as illustrated in Figure 6.2. During initialization, the required `C++` project is built and the simulator is launched as a separate process, establishing a TCP connection through

which all subsequent communication is coordinated. The simulation duration is then transmitted via a `CONFIG` message and acknowledged by *OMNeT++* before execution begins. Throughout the simulation, an `OmnetConnection` is maintained via TCP sockets. In each iteration, message arrival events generated in *mango* are forwarded to *OMNeT++* as JSON-encoded `MESSAGE` objects, where message dispatch and network traversal are simulated up to the specified `max_advance` time horizon. Upon message reception, the resulting delay information is transmitted back to *mango* as `RECEIVED` events and integrated as message departure events, after which a `WAITING_COMPLETE` signal synchronizes simulation time advancement. Once all events are processed, a `TERMINATE` message concludes the simulation and sockets are closed. This approach enables high-fidelity modeling of concrete network technology dynamics.

The channel model represents an analytical approximation derived from detailed *OMNeT++* simulation models. Its construction follows a three-step process: first, the network topology is extracted from the network description files, including nodes, positions, and connections, with wireless links added automatically where applicable. Second, model parameters are fitted based on detailed simulation outputs, including transmission rates, processing delays, jitter, and propagation speeds. These parameters are stored in a `topology_dict` that captures the essential network characteristics. Finally, this representation is used by the `ChannelModelScheduler` to compute message delays during simulation.

The static delay graph model is generated as a post-processing step from detailed simulation results. For each network technology, unicast message exchanges are simulated between a set of nodes and the resulting delay distributions are analyzed for the different network models under consideration. A larger graph is then constructed, where end-to-end delays between nodes are defined by the mean observed delays; if no direct measurements are available, a global mean delay is applied. This model captures static delay characteristics without accounting for dynamic network effects.

The `MetaModelScheduler` builds upon the detailed model by inheriting from the `DetailedModelScheduler`. This is required because, during the initial learning phases (Larva phase and Pupa phase), the meta-model depends on the same functionalities as the detailed simulation to observe and learn communication behavior. Beyond these phases, the scheduler extends the detailed model by approximating (and substituting) the observed simulation behavior using the online-trained meta-model, thereby reducing computational complexity while preserving key delay characteristics.

6.2 Experimental Setup

This section describes the experimental setup used to evaluate the proposed communication meta-model and to compare it with alternative communication modeling approaches. It specifies the underlying communication network model, the evaluation metrics, the considered communication scenarios, and the meta-model training configurations. In addition, a step-based evaluation methodology is introduced to structure data generation, hyperparameter tuning, and comparative validation in a systematic and reproducible manner.

6.2.1 Communication Network Models

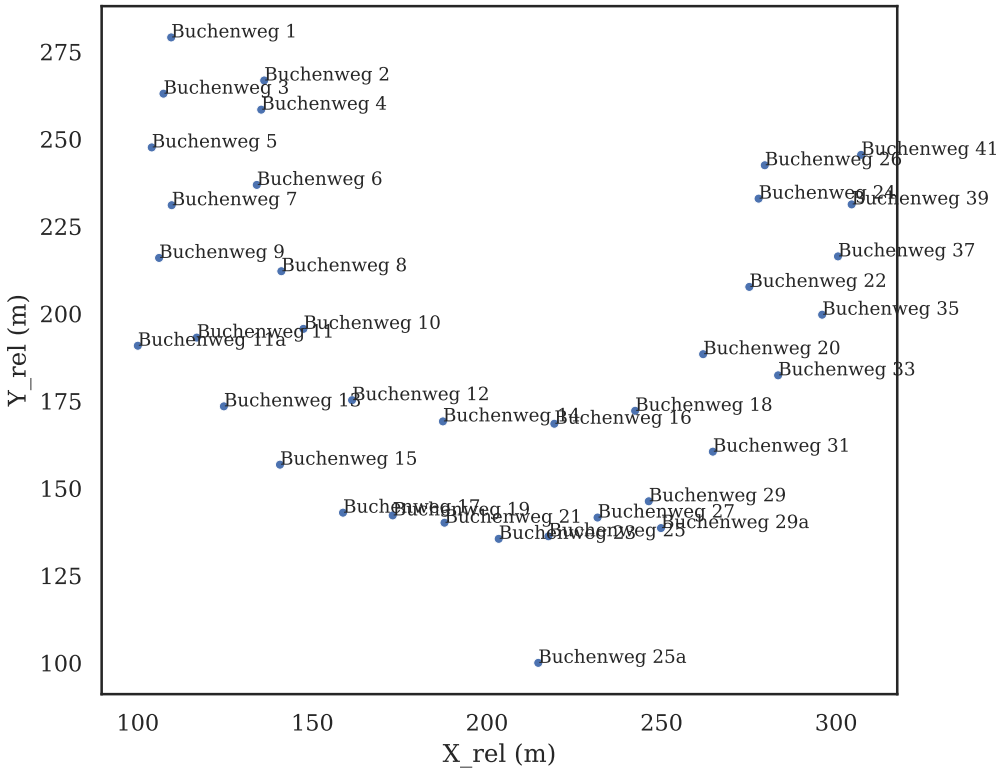


Figure 6.3: Geographic distribution of end devices and network components used to derive the communication network topology.

As discussed in Section 2.3, communication infrastructures in CPESs can differ significantly depending on the application context and network level. While multiple communication network models were introduced in Chapter 3, the evaluation presented in this chapter is restricted to a single communication network topology (combined with different communication technologies) in order to reduce confounding effects and to isolate the

impact of the communication modeling approaches.

The selected communication network topology is derived from a real-world power network. Specifically, a future Low Voltage (LV) reference network of the German city of Bremerhaven as proposed in [5] was used as the basis for the geographic placement of end devices as illustrated in Figure 6.3. The modeled LV network of the street *Buchenweg* was selected to represent a dense set of smart meters or comparable communication units typically found in NANs or FANs. In addition, a set of further devices was placed at randomly selected locations in the remaining network to represent other DERs or sensing components. This results in an overall network topology with 50 nodes representing different device types and communication relationships.

A central control entity was positioned at the geometric center of all end device locations and connected to a core router, representing communication over the WAN. Following the clustering approach proposed in the literature, end devices were grouped into spatial clusters, and one access router was placed for each cluster. These cluster routers are connected to the central router as part of the core network, while each end device is connected to its closest router based on geographic distance. This structure reflects the typical separation between access networks (NAN/FAN) with lower data rates and the higher-capacity WAN. For wireless communication scenarios, the same clustering and routing approach was applied. In addition, one antenna per network was placed to ensure sufficient coverage of all wireless end devices.

With regard to the selection and parameterization of technologies in *OMNeT++*, the same design decisions were made as in Chapter 3, and the technologies LTE, LTE450, 5G, Ethernet and parameterization as shown in Table 3.2 were selected.

6.2.2 Metrics

Different metrics (Table 6.1) are used to evaluate the communication modeling approaches across the considered scenarios. Accuracy-related metrics are evaluated relative to the detailed communication simulation, which serves as the reference baseline. Performance-related metrics are evaluated independently, with the ideal communication model providing a theoretical lower bound for execution time and scalability. A key challenge in the evaluation arises from the non-deterministic nature of detailed communication simulations. Due to stochastic effects in traffic handling, scheduling, and network behavior, repeated simulation runs yield distributions of delay times rather than identical results. Consequently, direct point-wise error measures between two modeling approaches are not meaningful. To address this, accuracy is assessed using distribution-aware metrics that operate on aggregated statistics over multiple runs.

Table 6.1: Overview on metrics used in evaluation.

NFR	**Metric**	**Abbreviation**
Accuracy	NRMSE for mean delay times	$NRMSE$
	Empirical Coverage ($\pm\sigma$)	$C_{\pm\sigma}$
	Wasserstein Distance	W
Performance	Execution Time in s	ET
(Functionality)	Substitution Success	S
-	Score (combines other metrics)	SC

This distributional perspective is further motivated by the observation of Steinbrink and Lehnhoff [102] that delay distributions in smart grid co-simulation are generally non-Gaussian, and that likelihood bounds (such as 5th/95th percentile intervals) provide more informative characterizations of output uncertainty than mean and standard deviation alone. While the metrics employed in this evaluation do not require explicit distributional assumptions, they are designed to capture complementary aspects of the delay distribution: $NRMSE$ targets the central tendency, $C_{\pm\sigma}$ assesses whether the dispersion of the comparative model aligns with the variability of the baseline, and W measures distributional similarity in a geometry-aware sense. Together, these three metrics approximate the information content of a full likelihood bound comparison without requiring a parametric distributional model.

An overview of the data structure underlying the accuracy evaluation is illustrated in Figure 6.4. For a fixed communication scenario and modeling approach, each message object is observed over multiple runs, resulting in a distribution of delay times. The following metrics are derived from these distributions.

The **Normalized RMSE (NRMSE)** evaluates the deviation of mean delay times between a comparative model and the detailed baseline (denoted as `model:m` and `model:det` in Figure 6.4). For each message object, the mean delay over all runs is computed for both models. The root mean squared error between these per-message means is then calculated and normalized by the average mean delay of the detailed model. Let

- m denote a communication modeling approach,
- $i \in 1, \ldots, n$ index message objects,
- $j \in 1, \ldots, r$ index independent simulation runs,

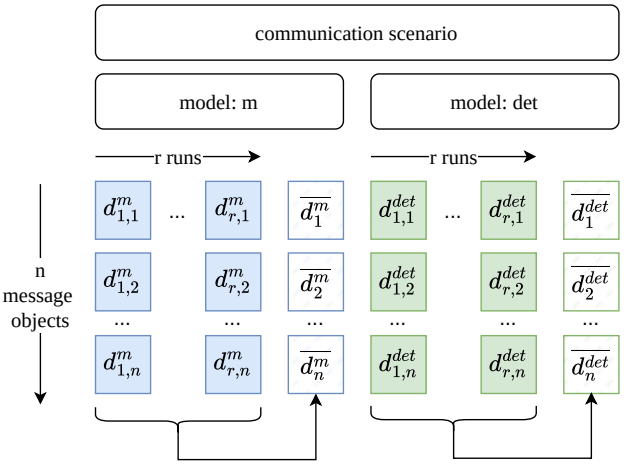


Figure 6.4: For a fixed communication scenario, each communication modeling approach is executed for r independent runs. For each of the n message objects, the resulting delay times $d^m_{j,i}$ form a distribution per model m.

- $d^m_{j,i}$ be the delay time of message object i in run j produced by model m.

For each message object i and model m, the empirical mean and standard deviation is defined as:

$$\overline{d_i^m} = \frac{1}{r}\sum_{j=1}^{r} d^m_{j,i}, \qquad \sigma_i^m = \sqrt{\frac{1}{r-1}\sum_{j=1}^{r}\left(d^m_{j,i} - \overline{d_i^m}\right)^2}. \tag{6.1}$$

The detailed communication simulation is denoted by $m = \text{det}$.

With that, the $NRMSE$ for the delay times in a scenario with model m can be calculated as:

$$NRMSE = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\overline{d_i^m} - \overline{d_i^{\text{det}}}\right)^2}}{\frac{1}{n}\sum_{i=1}^{n}\overline{d_i^{\text{det}}}} \tag{6.2}$$

This metric captures systematic bias in expected delay behavior.

The **empirical coverage** measures how well the delay distributions of a comparative model m align with the variability observed in the detailed model det. For each message object, the mean and standard deviation of the baseline delay distribution define an interval. The fraction of delay samples produced by the comparative model that fall within this interval is computed and then averaged across all message objects. The empirical coverage

is calculated as:

$$C_{\pm\sigma} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{r}\sum_{j=1}^{r}\mathbb{I}\left(\left|d_{j,i}^{m} - \bar{d}_{i}^{\text{det}}\right| \leq \sigma_{i}^{\text{det}}\right)\right) \tag{6.3}$$

$\mathbb{I}(\cdot)$ is the indicator function, yielding 1 if the condition holds and 0 otherwise.

This metric reflects whether the comparative model reproduces not only expected values but also realistic dispersion. It is conceptually related to the likelihood bound assessment recommended by Steinbrink and Lehnhoff [102]: whereas their framework evaluates whether output distributions fall within explicit percentile bounds (e.g., 5th/95th quantiles), $C_{\pm\sigma}$ operationalizes the same question using the empirical mean and standard deviation of the baseline as interval boundaries. Under Gaussian delay distributions, the two approaches are approximately equivalent, as the $\pm\sigma$ interval corresponds to the central 68% probability mass. For non-Gaussian distributions $C_{\pm\sigma}$ remains a well-defined and interpretable coverage measure, though it does not directly correspond to a fixed probability level. The Wasserstein distance W, introduced below, complements $C_{\pm\sigma}$ by capturing full distributional discrepancy without relying on interval boundaries.

To directly compare entire delay distributions, the first-order **Wasserstein distance** [105] (also known as Earth Mover's Distance) is computed for each message object between the baseline and comparative model distributions. This metric quantifies the minimal effort required to transform one distribution into the other and captures differences in shape, spread, and location. The reported value is the mean Wasserstein distance across all message objects:

$$W = \frac{1}{n}\sum_{i=1}^{n}\mathcal{W}_1\left(\left\{d_{j,i}^{m}\right\}_{j=1}^{r}, \left\{d_{j,i}^{\text{det}}\right\}_{j=1}^{r}\right) \tag{6.4}$$

The first-order Wasserstein distance is well suited for comparing empirical delay distributions, as it defines a true metric on probability measures and remains finite even for distributions with non-overlapping support [106]. In contrast to divergence-based measures, it explicitly accounts for the underlying metric structure of the delay variable, thereby reflecting differences in location, dispersion, and tail behavior in physical time units. For one-dimensional empirical distributions, the Wasserstein distance admits an intuitive interpretation as the average amount of delay mass that must be transported to align two distributions, making it particularly appropriate for assessing discrepancies between stochastic communication models [107].

Performance efficiency is measured using the **total execution time** (ET) of a simulation run. In addition, for the meta-model, **substitution success** (S) is recorded as a functional

indicator denoting whether the meta-model successfully replaced the detailed simulation during execution.

To support a compact comparison across configurations, a **composite score** (SC) is computed by ranking configurations with respect to the individual metrics. Lower values of NRMSE, W, and ET, and higher values of $C_{\pm\sigma}$ are preferred. The score is defined as

$$SC = [(r_{\max} - r(NRMSE)) + r(C_{\pm\sigma}) + (r_{\max} - r(W)) + (r_{\max} - r(ET))] \cdot S, \quad (6.5)$$

where $r(x)$ denotes the rank of a configuration with respect to metric x (with 1 being best), and $r_{\max}$ is the maximum possible rank. Multiplication by S ensures that configurations in which substitution did not occur are assigned a score of zero.

6.2.3 Communication Scenarios

The evaluation employs a diverse set of communication scenarios to reflect the heterogeneity of real-world communication behavior in CPESs. The selection of scenarios is motivated by prior simulation data analysis and by a systematic investigation of communication traffic diversity, as presented in Section 3.4 and in the own publication [1]. In these works, the Fano factor $F(w)$ was used to quantify the variability of communication traffic and to distinguish between traffic patterns with regular, random, and bursty characteristics.

Based on these findings, three classes of traffic models are considered in the evaluation: CBR ($F(w) \approx 0$), Poisson traffic ($F(w) \approx 1$), and complex, bursty traffic patterns modeled using Central Demand Supply Balancing (CDSB) ($F(w) > 1$). These classes represent increasing levels of variability and burstiness and are chosen to cover the range of communication behaviors observed in CPES applications, from highly regular monitoring traffic to event-driven and agent-based interactions with clustered message dispatches.

These traffic classes are realized through explicit message dispatch mechanisms within the agent-based simulation framework. For CBR traffic, message sending is implemented via periodic scheduling, where sender agents emit messages at fixed intervals and optionally broadcast them to multiple receivers. This results in strictly regular inter-arrival times and serves as a baseline for highly predictable monitoring and reporting traffic.

Poisson traffic is generated by sampling exponentially distributed inter-arrival times, with all message send events scheduled at simulation start based on a fixed rate parameter. This produces memory-less message arrivals with controlled stochastic variability while preserving a well-defined average message rate. Fixed random seeds are used to ensure reproducibility across simulation runs.

Complex, bursty traffic patterns modeled using CDSB are implemented as event-driven

interaction sequences between agents. In these scenarios, incoming messages may trigger delayed responses, conditional follow-up messages, or cascades of additional communication events. The resulting message streams exhibit temporal clustering, non-uniform inter-arrival times, and feedback-driven bursts that depend on agent state and control logic rather than on exogenously defined schedules.

The specific traffic parameterizations listed in Table 6.2 are chosen to represent comparable average message rates while systematically increasing temporal variability and burstiness. For CBR traffic, three message rates are selected to reflect typical periodic monitoring and reporting intervals in CPES applications. The Poisson configurations mirror these average rates and are instantiated with different random seeds to assess robustness with respect to stochastic variability. The CDSB parameterizations combine message rate, processing delay, and response probability to emulate event-driven and feedback-based communication patterns commonly found in agent-based control and coordination scenarios.

The traffic models are further parameterized by different message frequencies, scenario durations, and numbers of end devices in order to capture both small-scale and more demanding communication scenarios. Scenario durations (one to thirty minutes) are chosen to ensure that, in combination with the selected message rates, a representative number of message transmissions[1] is observed for all traffic models while keeping the evaluation computationally tractable. Longer durations, such as multi-hour or day-long scenarios, would lead to disproportionately increased execution times for the detailed communication simulation, as discrete-event network simulators operate at millisecond resolution and must process time advancement and internal events continuously over the entire scenario horizon, particularly for wireless technologies. Since the evaluation focuses on message-level behavior and resulting delay distributions, rather than on long-term background dynamics, the selected durations provide sufficient statistical representativeness without introducing unnecessary computational overhead. It should be noted that this design choice may exclude scenarios in which approximation-based approaches, including the proposed meta-model or analytical methods, would exhibit even more pronounced performance advantages over detailed simulation.

Similarly, the number of end devices is varied up to a maximum of 50 to analyze scalability effects while avoiding confounding initialization and background simulation costs. In detailed network simulations, all modeled devices contribute to simulation complexity regardless of whether they actively transmit messages, and network initialization over-

[1] Depending on traffic configuration, scenario duration, and system size, the evaluated scenarios comprise on average between approximately 600 and 6 000 message transmissions per run, with peak values approaching 30 000 messages in communication-intensive configurations.

head increases rapidly with network size. The selected range of 5 to 50 devices therefore allows systematic investigation of scaling behavior in terms of communication modeling accuracy and performance, while ensuring that the comparison remains focused on the communication interactions.

In addition to traffic characteristics, different communication network models are applied. Both wired and wireless communication technologies are considered in order to reflect typical deployment contexts in CPES, such as access networks and wide-area communication infrastructures. By combining traffic models with different network technologies, the evaluation captures the interaction between traffic variability and network-induced effects on message delays. An overview of all factors and their respective stages considered in the

Table 6.2: Overview on factors and stages in evaluation.

Symbol	**Factor**	**Stages**
CM	Communication Model	Ideal Communication
		Detailed Model
		Channel Model
		Static Graph Model
		Meta-Model *cocoon*
CN	Communication Network Model	LV-based topology with LTE
		LV-based topology with LTE450
		LV-based topology with 5G
		LV-based topology with Ethernet
T	Traffic Configuration	CBR (1 mps)
		CBR (1 mpm)
		CBR (1 mph)
		Poisson (1 mps, seed 1)
		Poisson (1 mpm, seed 1)
		Poisson (1 mps, seed 2)
		Poisson (1 mpm, seed 2)
		CDSB (1 mpm, 5 s, 50 %)
		CDSB (5 mpm, 30 s, 75 %)
		CDSB (10 mph, 60 s, 25 %)
S	Scenario Duration	one minute
		ten minutes
		thirty minutes
ND	Number of Devices	5
		10
		50

evaluation is provided in Table 6.2. Each communication scenario is defined by a specific combination of communication model, network technology, traffic model, scenario dura-

tion, and number of devices. This systematic variation enables a structured comparison of communication modeling approaches across a representative subset of the communication behavior space in CPES, while keeping the overall evaluation tractable.

6.2.4 Meta-Model Training Configurations and Hyper-Parameter

The online-trained meta-model implemented in *cocoon* exposes several hyper-parameters that influence clustering behavior, online learning dynamics, and the substitution decision between detailed simulation and approximation. These hyper-parameters are systematically varied in the evaluation in order to assess their impact on accuracy and performance efficiency across different communication scenarios. An overview of all meta-model–specific factors and their respective stages is provided in Table 6.3.

The cluster distance threshold δ (C-DT) controls the granularity of the initial clustering used during the pre-training phase (EGG PHASE), thereby influencing how sensitively the meta-model distinguishes between different communication states. The batch size i_{pupa} (C-IP) determines how frequently the online regressor is retrained during simulation execution and thus represents a trade-off between learning speed and computational overhead. The learning rate α (C-LR) affects the responsiveness of the error smoothing mechanism used for weighting predictions, while the butterfly threshold $\theta_{\text{butterfly}}$ (C-BT) governs the confidence required to fully substitute the detailed simulation with the meta-model. In addition, different substitution priority configurations (C-SP)[2] are evaluated by varying the weighting of error level, error trend, cluster distance, and topology stability in the substitution decision.

Beyond hyper-parameter variation, different test–training splits (C-TTS) are applied to reflect realistic deployment and reuse scenarios of the meta-model. The *parametrization split* represents cases where only minor variations (such as the message frequency) in traffic parameters are introduced. The *traffic load split* evaluates generalization across different message volumes and load levels. The *technology split* reflects deployment across different communication technologies, such as transitioning from wireless to wired networks. The *scale split* investigates generalization from small validation setups to larger system sizes with more end devices. Finally, the *traffic model split* assesses the ability of the meta-model to adapt from simpler traffic patterns to more complex communication behavior.

6.2.5 Step-Based Method

The evaluation follows a step-based methodology designed to progressively investigate different aspects of the proposed meta-model, ranging from data generation and internal

[2] error level: $w_{\text{level}=0.4}, w_{\text{trend}=0.2}, w_{\text{cluster}=0.2}, w_{\text{topology}=0.2}$, error trend: $w_{\text{level}=0.2}, w_{\text{trend}=0.4}, w_{\text{cluster}=0.2}, w_{\text{topology}=0.2}$, none: $w_{\text{level}=0.25}, w_{\text{trend}=0.25}, w_{\text{cluster}=0.25}, w_{\text{topology}=0.25}$

Table 6.3: Overview on meta-model specific factors and stages in evaluation.

Symbol	Factor	Stages
C-DT	Cluster Distance Threshold	1
		3
		5
C-IP	Batch Size i_{pupa}	50
		100
		150
C-LR	Learning Rate α	0.1
		0.5
		0.9
C-BT	Threshold Value $\theta_{\text{butterfly}}$	0.1
		0.5
		0.9
C-SP	Substitution Priority	error level
		none
		error trend
C-TTS	Test-Training Split	parametrization split (pa)
		traffic load split (tr)
		technology split (te)
		scale split (sc)
		traffic model split (tm)

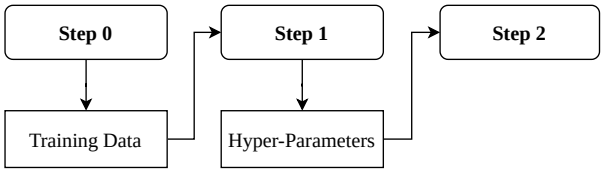


Figure 6.5: Step-based evaluation methodology illustrating the separation between data generation (Step 0), hyper-parameter tuning and robustness analysis (Step 1), and comparative validation across communication modeling approaches (Step 2).

optimization to comparative validation against alternative communication modeling approaches. Each step is associated with a specific experimental design and a clearly defined objective.[3]

[3] All simulations were executed on a virtualized Linux system running Ubuntu 22.04.5 LTS (kernel 5.15.0). The system was equipped with an Intel® Xeon® Gold 6248R CPU (3.00 GHz), providing 4 CPU cores with single-thread execution (no simultaneous multithreading), and 7.8 GiB of RAM. The virtual machine was hosted on a VMware hypervisor.

Step 0: Training Data Generation Step 0 establishes the empirical foundation of the meta-model by generating a representative and diverse training dataset. The objective is to expose the meta-model to a broad range of communication behaviors, covering different network technologies, traffic patterns, and system scales. This step focuses exclusively on data generation and does not include comparative evaluation.

In this step, only the meta-model in training mode is used (CM: 1 level). All four communication network technologies are considered (CN: 4 levels). A comprehensive set of traffic patterns is employed (T: 13 levels), including the traffic models listed in Table 6.2 and additional unicast message exchange configurations. Scenario duration is determined implicitly by the traffic configuration (S: coupled to T), and three system sizes are evaluated (ND: 3 levels).

Step 1: Hyper-Parameter Tuning and Robustness Analysis Step 1 shifts the focus from data generation to meta-model refinement. Using a Face-Centered Central Composite Design [108], this step investigates the influence of hyper-parameters on prediction accuracy, stability, and substitution behavior. In addition, different test–training splits are evaluated to assess robustness under varying deployment assumptions.

The evaluation compares the detailed simulation and the meta-model (CM: 2 levels). Two representative communication technologies are selected to contrast wired and wireless settings (CN: 5G, Ethernet) and a reduced set of representative traffic patterns is used (TS: CBR 1 mps, Poisson 1 mps, CDSB 1 mpm with 5 s processing delay and 50% response ratio). A medium-sized system is considered (ND: 10 devices). All meta-model–specific hyper-parameters (C-DT, C-IP, C-LR, C-BT, C-SP) are varied according to Table 6.3, and three test–training splits are applied (C-TTS: parametrization, scale, technology).

Step 2: Comparative Validation and Generalization The final step evaluates the external validity of the meta-model in comparison to alternative communication modeling approaches. A Fractional Factorial (screening) design [108] is used to analyze the impact of key scenario factors on the defined NFRs, namely accuracy, performance efficiency, scalability, and adaptivity.

All communication modeling approaches are included (CM: Ideal, Detailed, Static Graph, Channel Model, Meta-Model). All four communication technologies are evaluated (CN: LTE, LTE450, 5G, Ethernet). Six representative traffic models are considered, covering regular, stochastic, and bursty behavior. System size is varied to assess scalability (ND: 5, 10, 50 devices), and scenario duration is increased (S: five and ten minutes). All test–training splits are applied (C-TTS: all stages), while the remaining meta-model hyper-parameters are fixed to the optimal configuration identified in Step 1.

6.3 Hyper-Parameter Optimization (Step 1)

In total, Step 1 comprises 726 evaluated scenarios, including predominantly meta-model configurations and a number of six detailed reference configurations. Each scenario was executed with five independent simulation runs to account for the non-deterministic behavior of the communication models.

Following the approach of Banks et al. [42], a number of simulation runs R required to estimate the mean delay with a significance level of $\alpha = 0.05$ and an allowed error of $\epsilon = 0.05$ can be derived using Equation 6.6:

$$R \geq \left(\frac{t_{\alpha/2,R_0} \cdot \sigma}{\epsilon} \right)^2, \tag{6.6}$$

where $t_{\alpha/2,R_0}$ denotes the critical value of the t-distribution with R_0 degrees of freedom.

Although multiple evaluation metrics are considered in Step 1 (cf. Table 6.1), the standard deviation σ in Equation 6.6 refers to the run-to-run variability of the underlying stochastic output, i.e. the end-to-end delay times produced by the communication simulation[4]. Assuming a standard deviation of $\sigma = 5$ for the result variability, this yields a requirement of more than 200 simulation runs per scenario. Given that a single scenario run requires up to 300 s of execution time (as this was set as a time-out threshold), this would result in approximately 60 450 s per scenario. For all 726 scenarios, the total execution time would amount to about 43 886 700 s, corresponding to more than 508 days (≈ 1.39 years) of computation time. Consequently, achieving this confidence level is computationally infeasible within the scope of this work, and the number of simulation runs is therefore limited to five[5] per scenario. The resulting statistical uncertainty is explicitly acknowledged and must be taken into account when interpreting the results.

Across all scenarios, the mean execution time of the detailed simulations amounts to 125.17 s, while the meta-model executions required 68.32 s on average. Substitution of the detailed simulation by the meta-model occurred in 264 out of 462 applicable configurations, corresponding to a substitution success rate of 57.14%.

In the following subsections, the results of the Step 1 optimization will be presented by focusing on different aspects such as the influence of the model hyper-parameters and different test-training configurations.

[4] The accuracy metrics are derived from these delay values and therefore inherit their variability, but they are not themselves the quantities for which confidence intervals are constructed in the replication analysis.

[5] Assuming a below threshold execution time of 100 s per scenario, this would result in ≈ 4.2 days computation time for Step 1.

6.3.1 Distribution of Evaluation Metrics

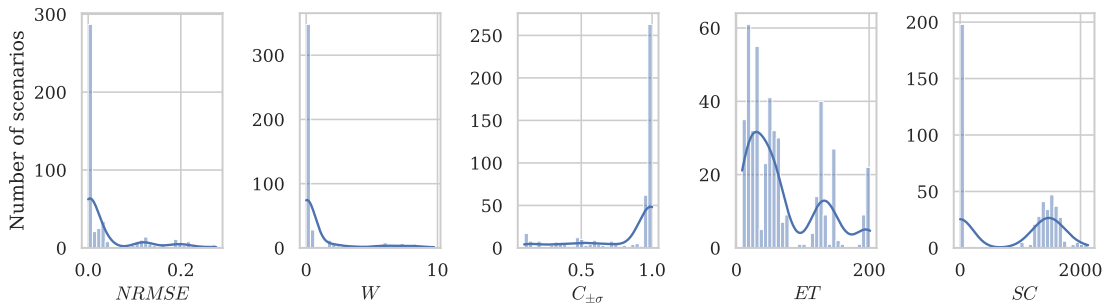


Figure 6.6: Distribution of the evaluation metrics for the meta-model (with substitution enabled) during Step 1. The plots show the variability of the Normalized RMSE ($NRMSE$), Wasserstein distance (W), empirical coverage ($C_{\pm\sigma}$), execution time (ET), and composite score (SC).

The distribution of the evaluation metrics obtained in Step 1 is shown in Figure 6.6. The distribution of the $NRMSE$ metric is strongly right-skewed, indicating that the majority of configurations achieve low deviations in mean delay times compared to the detailed simulation. However, a small number of configurations exhibit substantially higher error values, reflecting unfavorable hyper-parameter settings or insufficient generalization under certain training–testing splits. A similar pattern is observed for the Wasserstein distance W. Most scenarios result in low distributional distances, suggesting that the meta-model is generally able to reproduce the delay distributions of the detailed simulation. Nevertheless, isolated configurations yield larger Wasserstein distances, highlighting cases where higher-order distributional characteristics are not captured adequately.

The empirical coverage $C_{\pm\sigma}$ is more evenly distributed across the interval $[0, 1]$, with a clear concentration toward higher coverage values. This indicates that, in most scenarios, the variability of the delay distributions produced by the meta-model aligns well with the dispersion observed in the detailed simulation.

Execution times vary considerably across scenarios, ranging from very short run-times to values of approximately 200 s. This spread reflects differences in scenario duration, traffic load, and substitution behavior. Configurations without substitution generally exhibit higher execution times due to the reliance on the detailed simulation.

The composite score SC is zero for all scenarios in which no substitution occurred, as intended by its definition. For scenarios with successful substitution, the score spans a wide range of values, reflecting the combined influence of accuracy, performance efficiency, and distributional similarity.

6.3.2 Influence of Hyper-Parameters

As a first step, the results of Step 1 are analyzed with respect to their statistical significance in order to quantify the impact of the meta-model hyper-parameters on performance and accuracy relative to the detailed communication simulation baseline.

To disentangle structural effects from model-specific behavior, the analysis distinguishes between two complementary evaluation perspectives. An Intention-To-Treat (ITT) view includes all meta-model configurations and accounts for substitution as an explicit co-variate, thereby reflecting realistic deployment behavior. In contrast, a Per-Protocol (PP) view restricts the analysis to configurations in which substitution of the detailed simulation by the meta-model actually occurred, allowing the influence of hyper-parameters on approximation quality to be assessed independently of the substitution decision. [109]

Prior to statistical testing, the evaluation metrics are transformed to account for skewness and bounded domains. Execution time (ET), the accuracy metrics NRMSE and Wasserstein distance (W) are log-transformed, and the empirical coverage metric ($C_{\pm\sigma}$) is logit-transformed. These transformations improve the validity of linear-model assumptions.

For each metric, a joint multi-factor ordinary least squares model is fitted that includes all meta-model hyper-parameters simultaneously, namely the cluster distance threshold (C-DT), batch size i_{pupa} (C-IP), learning rate α (C-LR), butterfly threshold $\theta_{\text{butterfly}}$ (C-BT), and substitution priority (C-SP). To control for scenario-dependent effects, the model further includes communication network type, traffic configuration, and test–training split as co-variates. Statistical significance is assessed using Type-II Analysis of Variance (ANOVA), which yields order-invariant tests for the main effects of each factor [110]. Effect sizes are reported in terms of partial η^2, quantifying the proportion of explainable variance attributable to each factor beyond the residual error.

The results reveal that substitution behavior in Step 1 is fully determined by the butterfly threshold. For $\theta_{\text{butterfly}} = 0.9$, substitution does not occur in any configuration, whereas for lower threshold values substitution always takes place. Consequently, C-BT acts as a structural switch between detailed simulation and approximation rather than as a fine-grained tuning parameter.

This behavior is reflected in the results regarding the performance estimation. In the ITT view, execution time is dominated by the butterfly threshold, followed by the batch size. All remaining hyper-parameters exhibit negligible and statistically non-significant effects. In the PP view, where only configurations with successful substitution are considered, the influence of C-BT vanishes, and execution time is almost entirely determined by the batch size, reflecting the computational overhead of online retraining.

For the accuracy-related metrics, significant effects in the ITT view are again observed

Table 6.4: Statistically significant hyper-parameter effects identified in Step 1. Only effects with substantial explanatory power (partial $\eta^2 > 0.1$) are shown.

Metric	**Factor**	**View**	partial η^2
ET	C-BT	ITT	0.77
ET	C-IP	ITT	0.31
ET	C-BT	PP	0.84
$NRMSE$	C-BT	ITT	0.47
W	C-BT	ITT	0.46
$C_{\pm\sigma}$	C-BT	ITT	0.50

exclusively for the butterfly threshold. These effects do not reflect differences in approximation quality, but rather the contrast between substituted and non-substituted configurations. Once the analysis is restricted to substituted configurations (PP), no hyper-parameter exhibits a statistically significant impact on NRMSE or W, and only a moderate effect of the batch size on empirical coverage remains.

An overview of all statistically significant effects with substantial explanatory power ($\eta^2 > 0.1$) is provided in Table 6.4. Taken together, these findings indicate that the primary role of Step 1 is not the fine-tuning of approximation accuracy, but the identification of a robust hyper-parameter configuration that yields stable accuracy while enabling efficient substitution.

In order to find this robust configuration of hyper-parameters, Figure 6.7 summarizes the (partially non-significant) influence of the evaluated hyper-parameters on the standardized evaluation metrics. For this analysis, all metric values were transformed into z-scores across configurations to allow direct comparison of effects. The metrics ET and W were inverted prior to standardization such that higher values consistently indicate better performance. Darker cells therefore correspond to above-average outcomes, whereas lighter cells indicate below-average results.

Overall, the results show that the choice of hyper-parameters has an impact on both accuracy-related and performance-related metrics. In particular, the cluster distance threshold (C-DT) exhibits an influence in the scenarios investigated. A moderate threshold value (C-DT = 3) consistently achieves favorable trade-offs across metrics, whereas smaller or larger thresholds tend to either over-fragment the clustering or overly smooth scenario distinctions, leading to reduced performance or accuracy.

As mentioned earlier in the statistical analysis, the substitution threshold $\theta_{\text{butterfly}}$ (C-BT) shows the strongest individual effect. Higher threshold values result in improved accuracy metrics, as substitution is delayed or prevented and the detailed simulation dominates. However, this comes at the cost of increased execution time and reduced substitution success.

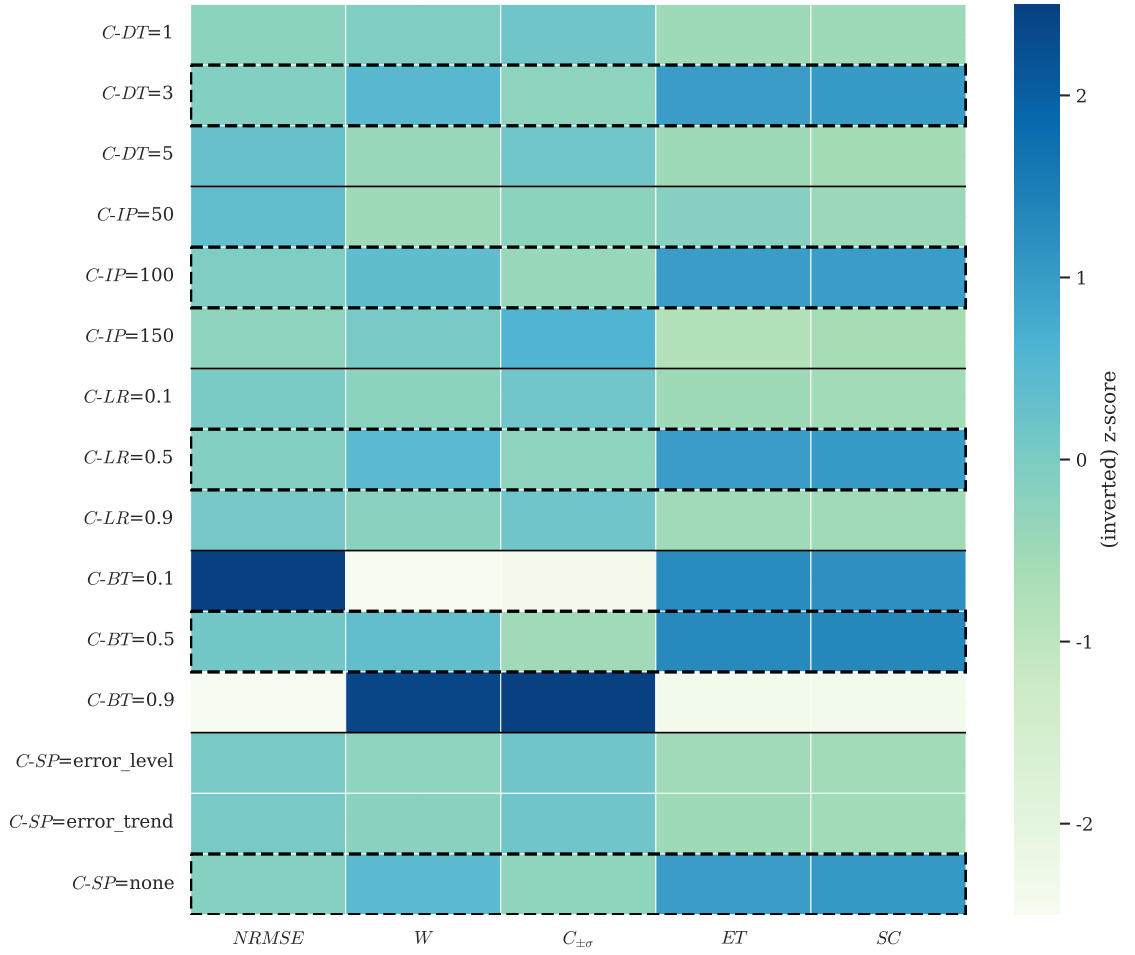


Figure 6.7: Comparison of standardized mean values across meta-model hyper-parameter configurations during Step 1 optimization. Metric values are z-score normalized; ET and W are inverted so that darker cells consistently indicate better performance. The configuration selected for Step 2 is marked with dashed boxes, reflecting the most balanced trade-off across all metrics and the highest composite score SC.

Conversely, lower threshold values promote earlier substitution, improving performance efficiency but increasing approximation error. This behavior highlights the fundamental trade-off between accuracy and computational efficiency inherent in the meta-model design.

The batch size i_{pupa} (C-IP) and learning rate α (C-LR) primarily affect stability and convergence behavior. Intermediate values (C-IP = 100, C-LR = 0.5) yield more balanced results across metrics, whereas extreme settings tend to favor either responsiveness or stability at the expense of the other. The substitution priority (C-SP) further influences

the balance between error control and performance. A neutral weighting of substitution criteria (C-SP = none) achieves the most consistent overall performance, as reflected in the composite score. Based on the composite score SC, which integrates accuracy, distributional similarity, and execution time while accounting for substitution success[6], the configuration C-DT=3, C-IP=100, C-LR=0.5, C-BT=0.5, and C-SP=none emerges as the most robust overall setting. C-BT=0.1 and C-BT=0.5 show similar results in terms of the SC metric. However, since C-BT=0.5 shows more balanced results in terms of accuracy metrics, this configuration was chosen. This configuration is then selected as the default for the comparative evaluation in Step 2.

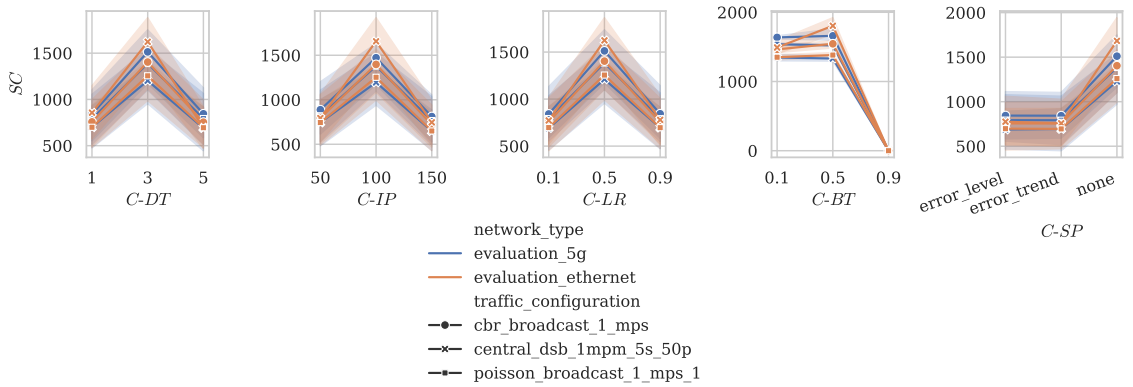


Figure 6.8: Influence of hyper-parameter configurations on the composite score SC across different network technologies (blue: 5G, orange: Ethernet) and traffic configurations (line styles). The butterfly threshold C-BT dominates the response, with C-BT=0.5 producing a sharp peak across all scenarios.

Figure 6.8 provides a condensed view of the influence of individual hyper-parameters on the composite score SC, differentiated by communication network technology and traffic configuration. The dominant finding is the strong and consistent effect of the butterfly threshold C-BT: the score peaks sharply at C-BT=0.5 across all network types and traffic configurations, while both extreme values (C-BT=0.1 and C-BT=0.9) yield substantially lower scores. This reflects the fundamental trade-off identified in the statistical analysis: too low a threshold triggers premature substitution and degrades accuracy, while too high a threshold prevents substitution entirely and eliminates the performance benefit. The remaining hyper-parameters show comparatively flat and less decisive response curves, with moderate settings consistently outperforming extremes but without producing the sharp

[6]It should be noted at this point that the ET and SC metrics show very similar results in the Figure 6.7. This is not because ET has a greater influence on SC than the other metrics, but because the other metrics usually show similar performance in the configurations and therefore do not have a strong influence on SC in the comparison.

optimum observed for C-BT. Importantly, the qualitative shape of all response curves is consistent across network technologies and traffic configurations, confirming that the selected default configuration represents a stable compromise independent of the specific deployment scenario.

6.3.3 Test-Training Configurations

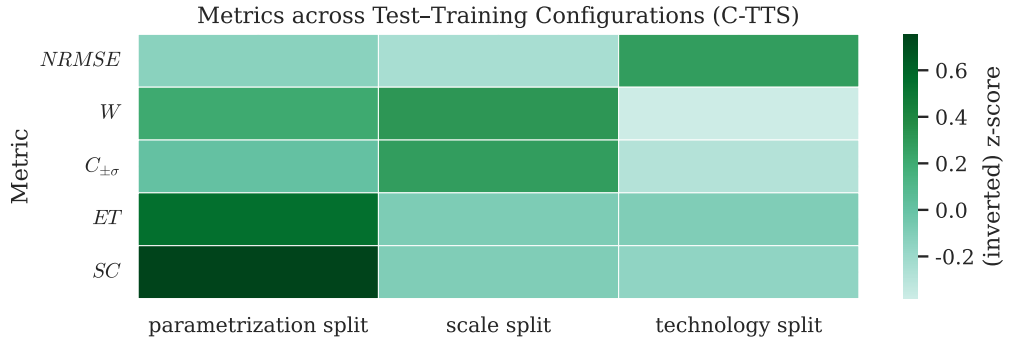


Figure 6.9: Standardized mean performance of the meta-model across different test–training configurations (C-TTS).

Figure 6.9 illustrates the influence of different test–training split strategies (C-TTS) on the standardized evaluation metrics. As in the hyper-parameter analysis, all metrics were z-score normalized across configurations, with the values for execution time ET and Wasserstein distance W inverted to ensure consistent interpretation.

The results show that the choice of test–training configuration has a noticeable impact on both accuracy-related and performance-related metrics. The parametrization split achieves comparatively high standardized values, particularly for ET and the composite score SC. This behavior can be attributed to the high similarity between training and testing data in this split, which allows the cluster-based predictions to remain accurate from the beginning of the simulation and enables earlier substitution of the detailed model.

Favorable performance across all metrics is also exhibited by the scale split. Although the number of devices differs between training and testing scenarios, the underlying communication patterns and technologies remain comparable. This indicates that the meta-model generalizes well with respect to system size, provided that traffic characteristics are sufficiently represented in the training data.

In contrast, the technology split results in the lowest standardized performance. In this configuration, the communication technology used during testing differs from that seen during training, leading to mismatches in delay characteristics and network behavior.

As a result, both accuracy and substitution performance degrade, which is reflected in lower values for ET and SC. This outcome is expected and highlights the importance of technology-specific training data when transferring the meta-model across heterogeneous communication infrastructures.

Overall, the analysis confirms that the meta-model benefits most from training data that is representative of the deployment context, while still maintaining a degree of robustness with respect to changes in scale.

6.3.4 Scenarios without Substitution

This part of the Step 1 analysis focuses on the behavior of the meta-model when substitution of the detailed simulation is disabled. The objective is to isolate the effects of online learning duration and the amount of available training data on prediction accuracy and execution time. By excluding substitution (and therefore the BUTTERFLY PHASE), the analysis allows direct observation of how the meta-model's predictive performance evolves over time as additional message observations become available.

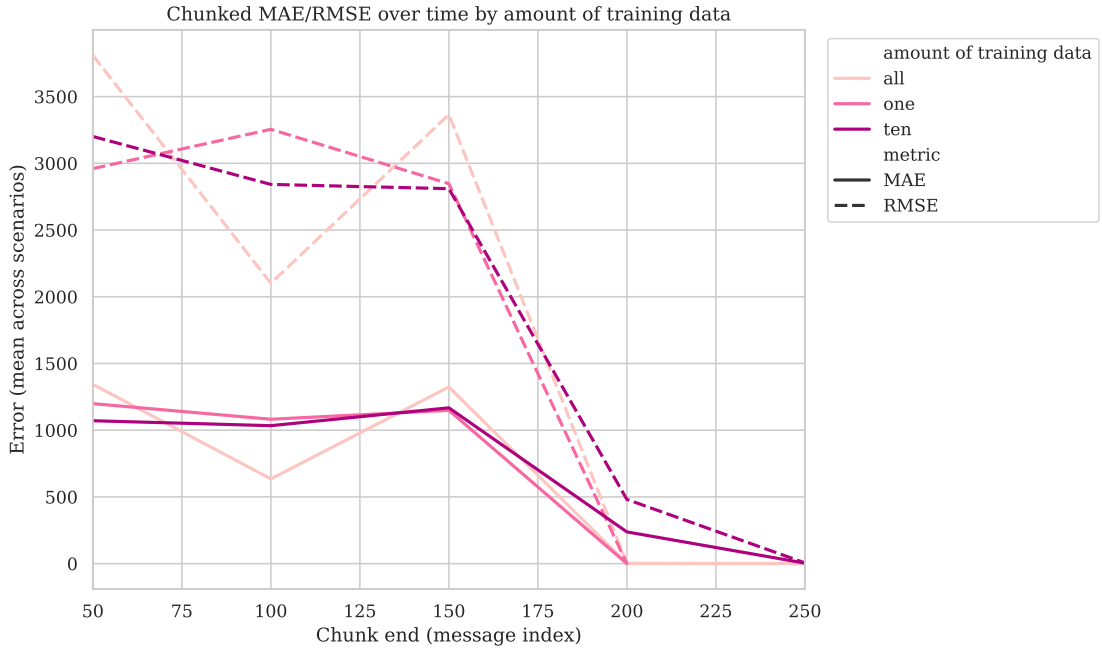


Figure 6.10: Chunked Mean Absolute Error (MAE) and RMSE over time for different amounts of training data. Each curve represents the mean weighted prediction error within consecutive message intervals (step size = 50). The lines show how the meta-model accuracy evolves over time, averaged across scenarios that share the same amount of training data used during training.

Figure 6.10 illustrates the evolution of the weighted prediction error over the course of the simulation for different amounts of initial training data under the parametrization split. Prediction errors are computed in consecutive chunks of 50 messages and averaged across all scenarios sharing the same training data configuration. Both MAE and RMSE exhibit a clear dependence on the amount of training data during the early phase of the simulation.

At the beginning of the execution, configurations initialized with larger and more diverse training datasets show higher prediction errors, whereas configurations trained on a single scenario achieve lower initial errors. This behavior reflects the closer alignment of narrowly trained models with the test scenarios, while more broadly trained models exhibit increased initial variance.

For configurations trained on the full training dataset, the error does not decrease monotonically but exhibits a temporary increase in the intermediate phase of the simulation. This behavior indicates that parts of the training data may encode communication patterns that are not fully consistent with the currently observed scenario, leading to transient negative transfer before sufficient online observations allow the model to adapt.

As the simulation progresses, prediction errors decrease across all configurations, demonstrating effective online learning. Towards the end of the plot (message indices 200 and 250), both MAE and RMSE converge to similarly low levels for all training configurations, indicating that long-term predictive performance is governed primarily by online adaptation, while the amount and diversity of initial training data mainly influence early and transient behavior.

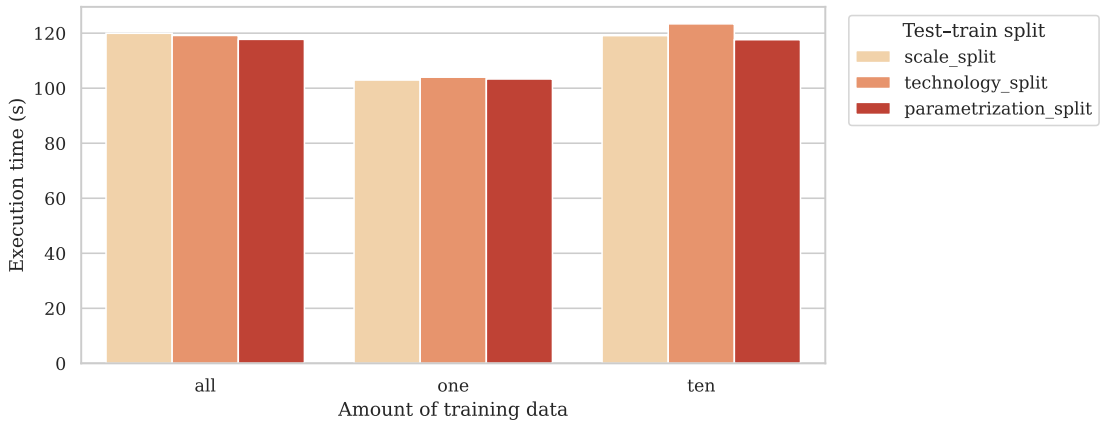


Figure 6.11: Execution time of the scenarios as a function of the amount of training data, differentiated by test–train split. Bars show mean execution time across all scenarios within each configuration.

The impact of training data size on execution time is shown in Figure 6.11. Execution time increases with the amount of training data, reflecting the higher computational overhead associated with larger clustering structures and more complex regression models. In contrast to the substitution-enabled scenarios discussed earlier, differences between test–training split strategies are comparatively small when substitution is disabled. This indicates that, in the absence of substitution, execution time is dominated by the complexity of the training data rather than by scenario similarity between training and testing.

It is important to note that even in scenarios without substitution, where delay generation is exclusively performed by the detailed communication simulation, the resulting accuracy metrics do not attain ideal values. Across all Step 1 scenarios with disabled substitution, the average empirical coverage amounts to 0.97, with a minimum observed value of 0.76. Similarly, the Wasserstein distance is non-zero, with an average value of 0.36 and maximum values exceeding 3, while the normalized root mean squared error reaches an average of 0.018 and a maximum of 0.18.

These deviations arise from the inherent non-deterministic behavior of the detailed communication simulation itself. Due to stochastic effects in scheduling, queuing, and protocol dynamics, repeated executions of identical scenarios yield distributions of delay outcomes rather than identical realizations. Consequently, even comparisons between realizations of the detailed model exhibit non-zero error and incomplete empirical coverage.

This inherent baseline uncertainty must therefore also be taken into account in the interpretation of the Step 2 results, as it defines a lower bound on achievable accuracy that applies independently of the chosen communication modeling approach.

6.3.5 Comparison to Random Forest Regressor

In addition to the Decision Tree Regressor used in the meta-model configuration discussed so far, the same evaluation was conducted using a Random Forest Regressor. This choice is motivated by the observation in Subsection 5.1.3 that inherently ensemble-based tree methods can yield strong predictive performance in static regression settings and are often considered a robust alternative to single-tree models.

Figure 6.12 compares the distribution of evaluation metrics obtained when using Decision Tree and Random Forest Regressors within the meta-model. The comparison includes accuracy-related metrics ($NRMSE$, W, $C_{\pm\sigma}$) as well as execution time (ET).

With respect to performance efficiency, the Random Forest configuration achieves slightly lower execution times on average. This effect can be attributed to differences in the substitution behavior and prediction dynamics of the regressor, which in some scenarios allow earlier or more frequent substitution of the detailed simulation. However, this performance

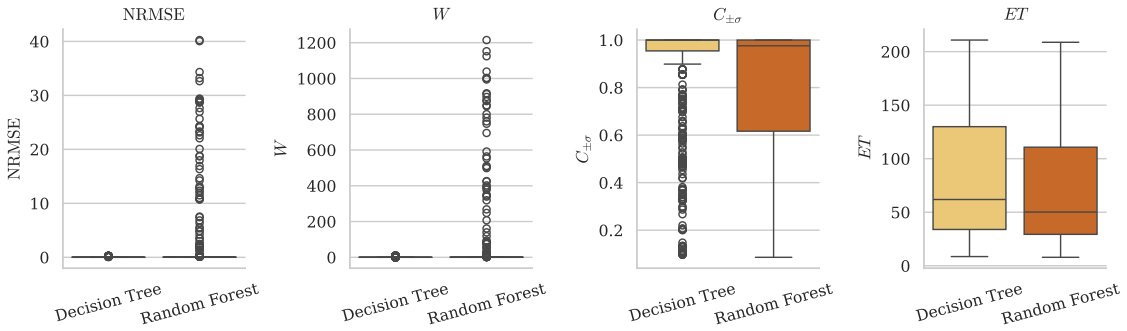


Figure 6.12: Comparison of the meta-model performance when using Decision Tree and Random Forest Regressors.

advantage is comparatively small and must be interpreted in conjunction with the observed accuracy behavior.

In contrast, the accuracy-related metrics reveal a markedly less stable behavior of the Random Forest Regressor. While median values of NRMSE and W are already higher than those obtained with the Decision Tree, the most prominent difference is the presence of numerous extreme outliers. This effect is particularly pronounced for the Wasserstein distance, where values exceed 1500 in several configurations, whereas the corresponding values for the Decision Tree Regressor remain below 10 across all evaluated scenarios. A similar pattern is visible for the NRMSE, indicating substantial deviations in predicted mean delay times in a subset of scenarios.

The empirical coverage metric $C_{\pm\sigma}$ further supports this observation. Although high coverage values are achieved in many cases, the Random Forest configuration exhibits a wider spread and a larger number of low-coverage realizations, indicating a reduced ability to consistently reproduce the variability of the delay distributions observed in the detailed simulation.

A more detailed analysis of the scenario configurations reveals that the unfavorable Random Forest results are strongly associated with the *technology split* test–training configuration. In this setting, the communication technology used during evaluation differs from that seen during training, leading to pronounced shifts in delay characteristics. The ensemble-based Random Forest Regressor appears to be less robust under such distributional changes, resulting in amplified prediction errors and heavy-tailed error distributions. This behavior suggests that, while Random Forests can effectively reduce variance in stationary settings, they may struggle to extrapolate under stronger regime changes that are common when transferring between communication technologies.

6.3.6 Summary of Step 1 Findings

The main findings of the hyper-parameter optimization and analysis in Step 1 can be summarized as follows:

- Substitution behavior and execution time are dominated by the butterfly threshold, which effectively controls whether the meta-model replaces the detailed simulation or not.
- Once substitution occurs, execution time is primarily determined by the batch size, reflecting the computational overhead of online retraining.
- Approximation accuracy is largely insensitive to most hyper-parameters in substituted configurations, indicating stable predictive behavior over a wide parameter range.
- Test–training configurations strongly influence early-phase accuracy and substitution timing, with parametrization and scale splits yielding the most favorable results.
- Online learning enables the meta-model to compensate for limited or partially mismatched training data, although transient accuracy degradation may occur.
- A single hyper-parameter configuration emerges as a robust default based on the composite score and is selected for the comparative evaluation in Step 2.
- Even in scenarios without substitution, accuracy metrics do not attain ideal values due to the inherent non-deterministic behavior of the detailed communication simulation, establishing a baseline uncertainty that must be considered when interpreting substitution-enabled results.

Based on these findings, the hyper-parameter configuration C-DT=3, C-IP=100, C-LR=0.5, C-BT=0.5, and C-SP=none is selected as the default for the comparative evaluation in Step 2. This configuration achieves the most balanced trade-off across accuracy, distributional similarity, and execution time while maintaining robust substitution behavior across heterogeneous scenarios. The following section applies this configuration to assess the meta-model against alternative communication modeling approaches across the full scenario space defined in Table 6.2.

6.4 Comparative Evaluation (Step 2)

The objective of this step is to evaluate the defined NFRs (accuracy, performance efficiency, scalability, and adaptivity) for the proposed meta-modeling approach in comparison to the

other communication modeling approaches. In contrast to Step 1, which focused on internal optimization and the identification of robust hyper-parameters, Step 2 assesses the validity and practical applicability of the meta-model across a broad range of communication scenarios.

Table 6.5: Overview of evaluated NFR categories, their influencing factors, and the corresponding metrics.

NFR (focus)	**Main Factors**	**Main Metrics**
Accuracy	all	$NRMSE$, $C_{\pm\sigma}$, W
Adaptivity		$NRMSE$, $C_{\pm\sigma}$, W
(regarding communication network technologies)	CN (Communication Network Model)	
(regarding test-training set-up)	C-TTS (Test-Training Split)	
Performance Efficiency	all	ET
Scalability		ET
(increasing complexity & message volumes)	T (Traffic Model)	
(increasing network topologies)	ND (Number of Devices)	
(increasing scenario durations)	S (Scenario Duration)	

Table 6.5 summarizes how the individual NFRs are evaluated. Accuracy is assessed across all scenarios using metrics that capture deviations in mean delay behavior, distributional similarity, and empirical coverage. Adaptivity is evaluated indirectly by analyzing the stability of these accuracy metrics across different communication network technologies (CN) and test–training configurations (C-TTS), thereby assessing the meta-model's ability to generalize beyond its training context. Performance efficiency is measured using the execution time (ET), while scalability is analyzed by observing how execution time evolves with increasing traffic complexity, network size, and scenario duration. The accuracy metrics with regard to increasing network topologies are analyzed as part of the accuracy evaluation.

Overall, 1 584 distinct scenarios were evaluated, each executed with five independent runs (cf. the discussion in Section 6.3 for details on the required number of simulation

runs) to account for non-deterministic behavior. The scenarios are distributed across the different communication modeling approaches as follows: 960 scenarios use the meta-model, 192 scenarios each use the channel model, detailed model, and static graph model, and 48 scenarios use the ideal model. The different numbers of scenarios result from different parameterization options (e.g., the ideal model does not consider different communication networks). Within the meta-model configurations, substitution of the detailed simulation occurred in 806 scenarios, while 154 scenarios completed without substitution. Detailed numerical results for all evaluated metrics are provided in the appendix (see Table A.1, Table A.2, and Table A.3).

Table 6.6: Results of nested linear model tests assessing the influence of the communication modeling approach (CM) on the evaluation metrics in Step 2 (#CM refers to the number of stages in CM considered for the metric).

Metric	**#CM**	**F**	**p-value**	**partial** η^2
ET (log)	5	1737.99	$< 10^{-15}$	0.709
$NRMSE$ (log)	3	3.96	0.0468	0.0031
W (log)	3	4.11	0.0428	0.0032
$C_{\pm\sigma}$ (logit)	3	211.25	$< 10^{-40}$	0.141

As an initial global analysis, the influence of the communication modeling approach (CM) on the evaluation metrics is assessed using nested linear models that control for scenario characteristics (cf. Table 6.6). The results show that the choice of communication model has a highly significant and dominant effect on execution time, confirming that computational performance is primarily determined by the modeling approach. For accuracy-related metrics, statistically significant but substantially smaller effects were observed for mean delay error and distributional distance. In contrast, empirical coverage exhibits a large effect size, demonstrating that the ability to reproduce delay variability depends strongly on the selected communication model.

While this global analysis establishes that the choice of communication modeling approach has a statistically significant influence on the evaluated metrics, it does not yet reveal how the individual modeling approaches compare under specific scenario conditions. To this end, the following subsections examine the behavior of the different communication models in more detail, disentangling their relative strengths and limitations with respect to Accuracy, Performance Efficiency, Scalability, and Adaptivity across the considered communication scenarios.

6.4.1 Accuracy

Accuracy is a central requirement for the proposed meta-model, as it is intended to exceed the fidelity of simple analytical communication models while avoiding the computational cost of detailed simulation. Consequently, accuracy is evaluated across the full space of scenario factors considered in Step 2, including traffic models, network sizes, communication technologies, and scenario durations.

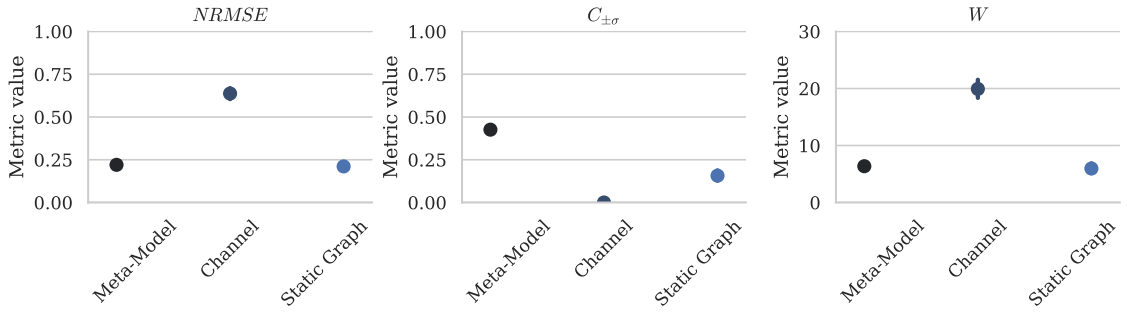


Figure 6.13: Overall comparison of accuracy metrics across communication modeling approaches aggregated over all Step 2 scenarios.

Figure 6.13 shows a clear separation between the evaluated communication modeling approaches. The channel model exhibits the weakest accuracy across all metrics, with comparatively high normalized errors, consistently low empirical coverage close to zero, and the largest Wasserstein distances. This indicates that, while computationally efficient, the channel model fails to reproduce both mean delay behavior and delay variability across heterogeneous scenarios.

The static graph model and the meta-model achieve substantially better accuracy. For $NRMSE$ and W, both approaches show comparable median values, indicating similar performance in reproducing mean delays and overall delay distributions. However, the meta-model consistently achieves higher empirical coverage $C_{\pm\sigma}$ than the static graph model. This suggests that the meta-model more accurately captures the dispersion and variability of delays, rather than only their central tendency.

Further insights are provided by Figure 6.14, which decomposes accuracy by network size and traffic model. Across all numbers of devices and traffic configurations, the channel model consistently shows the highest $NRMSE$, near-zero empirical coverage, and the largest Wasserstein distances. The static graph model improves upon these results but exhibits increasing error and variability as the number of devices grows.

In contrast, the meta-model shows a comparatively narrow spread in both $NRMSE$ and W across network sizes and traffic models, indicating stable and predictable error bounds.

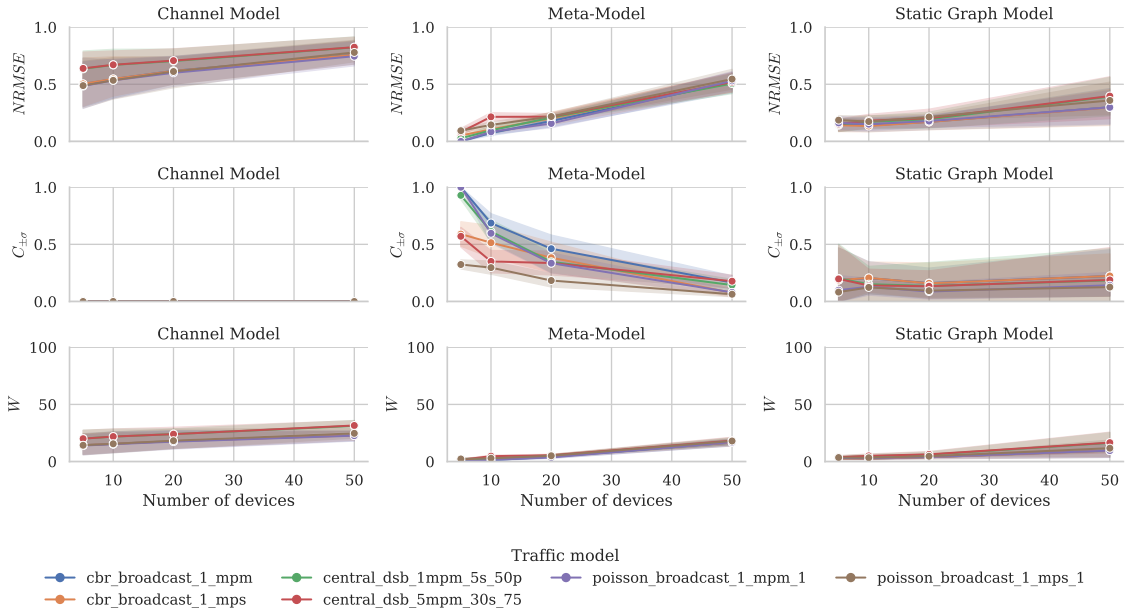


Figure 6.14: Accuracy comparison of analytical communication models and the meta-model across varying numbers of devices and traffic models for all accuracy metrics.

While empirical coverage decreases with increasing network size for all approaches, the meta-model maintains noticeably higher coverage values than the other analytical models, particularly in scenarios with smaller and medium-sized networks.

A closer inspection of the empirical coverage metric reveals that its decline with increasing network size is not specific to the meta-model but reflects a general increase in delay variability in larger systems. As the number of devices grows, contention, routing diversity, and queue interactions lead to broader and more heterogeneous delay distributions in the detailed simulation baseline. Analytical models, which rely on static or mean-based abstractions, fail to reproduce this variability and therefore exhibit near-zero coverage across all network sizes.

For the meta-model, the decrease in empirical coverage at larger scales is more gradual. This indicates that the meta-model partially adapts to the increased variance through online learning, but does not fully capture higher-order distributional characteristics under high-complexity conditions.

Figure 6.15 serves to contextualize the achievable accuracy of the meta-model by again explicitly illustrating that the lower error bound of the evaluation metrics is not zero. Scenarios without substitution correspond to continued execution of the detailed communication simulation and therefore represent comparisons between stochastic realizations of the same underlying model. Due to inherent non-deterministic effects in scheduling,

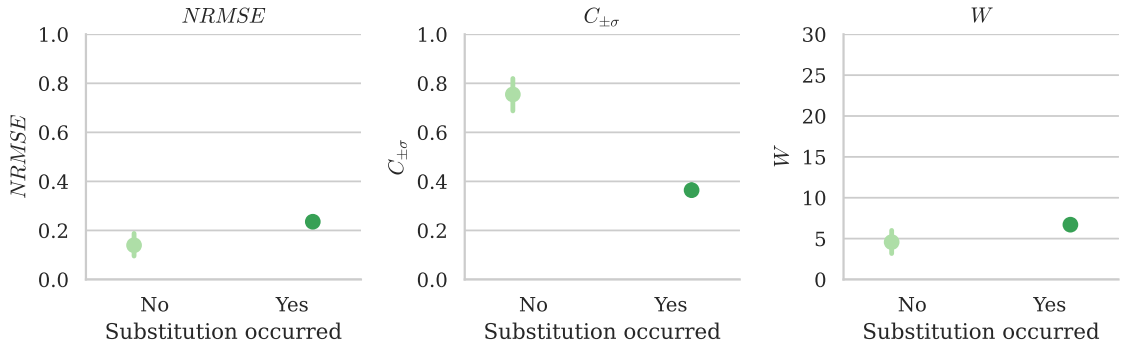


Figure 6.15: Accuracy metrics of the meta-model differentiated by substitution outcome. Scenarios without substitution correspond to continued execution of the detailed simulation.

queuing, and protocol dynamics, these configurations already exhibit non-zero values for $NRMSE$, Wasserstein distance, and empirical coverage below one (as already mentioned in Subsection 6.3.4). This demonstrates that the meta-model maintains accurate and stable predictions across the evaluated scenarios, and that observed deviations should be interpreted relative to the intrinsic uncertainty of the reference simulation rather than as absolute modeling errors.

6.4.2 Adaptivity

Adaptivity also is a key NFR of the proposed meta-model, as it is intended to be applicable across heterogeneous communication network configurations and adaptable to varying communication behaviors without requiring extensive re-engineering. While the integration of the meta-model into the simulation environment has already been discussed in Section 6.1, the focus of this subsection is on its empirical adaptivity. This is assessed by analyzing how robustly the meta-model maintains accuracy across different communication network technologies and under varying test–training configurations.

Figure 6.16 compares the accuracy metrics of the different communication modeling approaches across the evaluated communication network technologies. For wireless technologies (5G, LTE, and LTE450), the meta-model consistently achieves strong accuracy results across all metrics. In particular, it outperforms the simpler analytical approaches and maintains comparatively low $NRMSE$ and W values, while achieving higher empirical coverage $C_{\pm\sigma}$. This indicates that the meta-model adapts well to the stochastic and technology-specific characteristics of wireless communication networks.

For wired Ethernet-based communication, the static graph model slightly outperforms the meta-model in terms of accuracy. This behavior can be attributed to the highly pre-

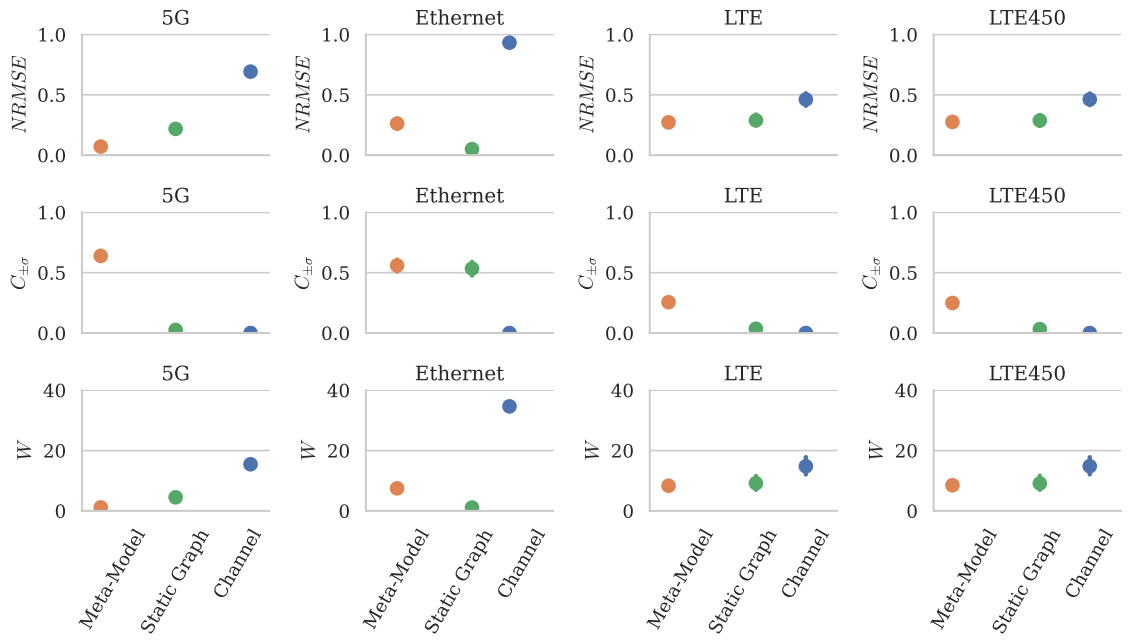


Figure 6.16: Comparison of accuracy metrics across communication network models. Each subplot shows one accuracy metric evaluated for different communication technologies and modeling approaches.

dictable and largely static delay characteristics of Ethernet networks, which align well with the assumptions underlying static delay graph representations. While the meta-model remains applicable in these scenarios, the results suggest that its default hyper-parameter configuration is not fully optimized for wired network environments. Targeted tuning could further improve its performance in this setting.

Across all communication technologies, the relative accuracy of the meta-model remains stable across different traffic models. Differences are most pronounced in the empirical coverage metric, which is more sensitive to the inherent non-determinism of the detailed simulation baseline. These variations reflect stochastic effects rather than systematic modeling errors.

The influence of different test–training configurations on meta-model accuracy is shown in Figure 6.17. Overall, the meta-model exhibits comparable performance across all evaluated split strategies, indicating robust generalization beyond the specific conditions seen during training. The best accuracy is achieved for test–training splits that preserve similar traffic characteristics between training and testing. In particular, the traffic load split, where the traffic pattern remains unchanged but message volume varies, and the parametrization split, where traffic characteristics change only slightly, yield consistently low $NRMSE$ and

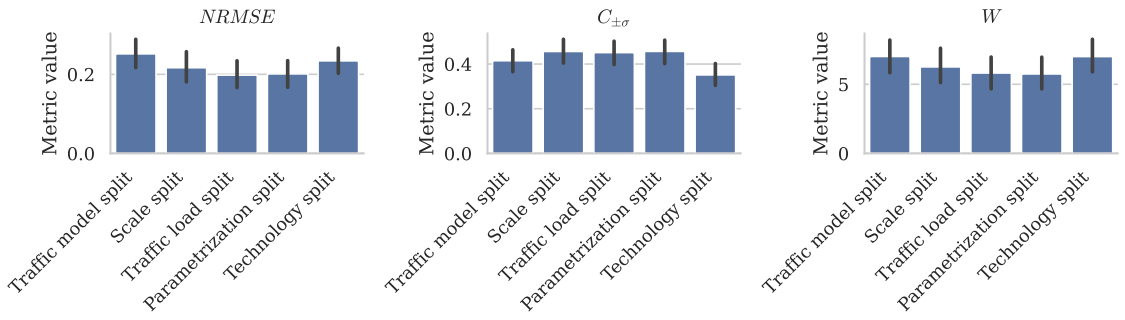


Figure 6.17: Accuracy metrics of the meta-model across different test–training split configurations. Each subplot shows one accuracy metric aggregated over all Step 2 scenarios.

W values and higher empirical coverage.

In contrast, splits that introduce stronger deviations between training and testing conditions, such as the technology split, lead to moderately reduced accuracy. This behavior is expected, as fundamental network characteristics differ between communication technologies and are only partially transferable without technology-specific training data.

Overall, the results demonstrate that the meta-model can be applied across heterogeneous communication network technologies and remains robust under different assumptions about training data availability and deployment context. The observed stability of accuracy metrics across both network technologies and test–training configurations provides strong empirical evidence for the adaptivity of the proposed meta-modeling approach.

6.4.3 Performance Efficiency

Performance efficiency is another central requirement for the proposed meta-model, as it aims to significantly reduce computational overhead compared to traditional, fully integrated communication simulations while retaining acceptable accuracy. Since this is intended to be an inherent property of the modeling approach, execution time is evaluated across all scenario factors considered in Step 2, including traffic characteristics, communication models, network size, and scenario duration.

Figure 6.18 compares the execution times of all communication modeling approaches aggregated over all evaluated scenarios. The ideal communication model serves as a lower bound for execution time, as it introduces no communication-related computation beyond message passing. Both analytical approaches (the channel model and the static graph model) exhibit execution times close to this lower bound, reflecting their reliance on simple look-up operations and lightweight delay calculations. In contrast, the detailed communication

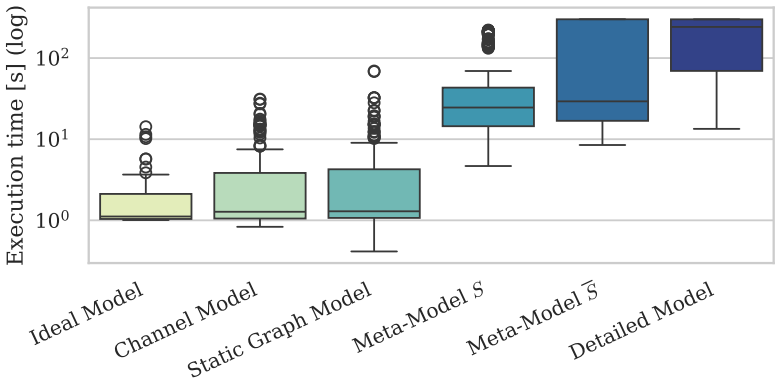


Figure 6.18: Execution time comparison across communication modeling approaches. The boxplots show the distribution of execution times over all Step 2 scenarios. For the meta-model, scenarios with substitution (S) and without substitution ($\overline{S}$) are shown separately.

model shows substantially higher execution times, with median values exceeding 100 s. This reflects the computational cost of executing a full-fledged discrete-event network simulation and maintaining synchronization between the simulation environments.

The meta-model occupies an intermediate position between these extremes. For scenarios in which substitution occurs, execution times are significantly reduced compared to the detailed model and are closer to those of the analytical approaches, while still exceeding the ideal baseline due to the overhead of online learning and prediction. Scenarios without substitution exhibit higher execution times, approaching those of the detailed model, as communication events continue to be handled by the detailed simulation throughout execution.

Overall, the results demonstrate that the meta-model substantially improves performance efficiency relative to detailed communication simulation, particularly when substitution is achieved.

6.4.4 Scalability

Scalability is a NFR for communication modeling approaches in CPESs, as real-world applications are characterized by increasing system sizes, longer operational horizons, and growing communication volumes. In this subsection, scalability is evaluated by analyzing how execution time evolves with increasing numbers of devices, longer scenario durations, and more complex traffic patterns.

Figure 6.19 shows execution times aggregated over all scenarios with respect to network size and scenario duration. The analytical communication models exhibit consistently low

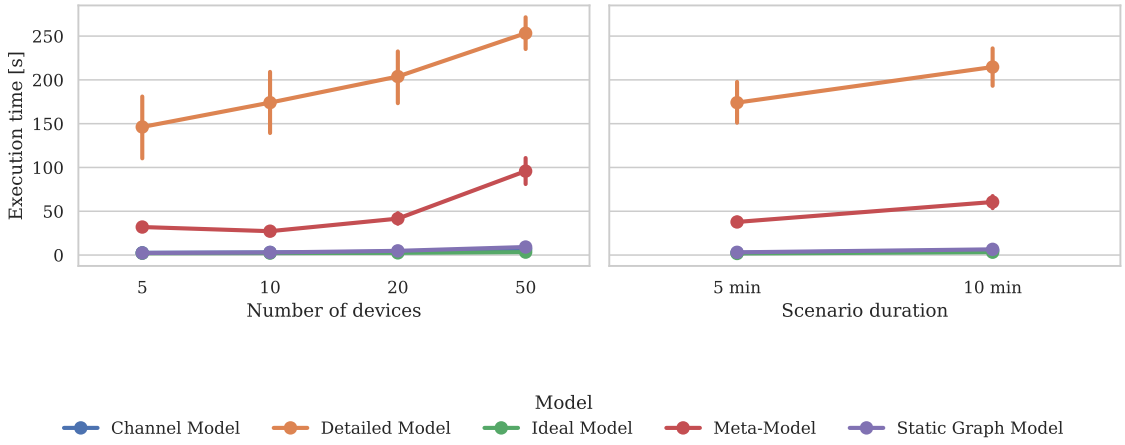


Figure 6.19: Execution time as a function of system scale. The left plot shows execution time over increasing numbers of devices, while the right plot shows execution time for different scenario durations.

execution times across all configurations. Their runtime is largely insensitive to both the number of devices and the scenario duration, reflecting their reliance on simple computations and static delay representations.

In contrast, the execution time of the detailed communication model increases steadily with both the number of devices and the scenario duration. This trend reflects the growing number of simulated communication events and the increased complexity of network state handling in the discrete-event simulation. It is worth noting that a timeout of 300 s was configured, which bounds the maximum observed execution time and further highlights the limited scalability of fully detailed communication simulations.

The meta-model again exhibits an intermediate behavior. Execution time increases with network size and scenario duration, but at a substantially lower rate than for the detailed model. Even for the largest network sizes and longest scenarios considered, the meta-model's execution times remain well below those of the detailed simulation. This indicates that the meta-model mitigates the scalability limitations of detailed communication simulation while still accounting for dynamic communication effects.

Figure 6.20 further differentiates execution time by traffic model class. For regular CBR traffic and Poisson-distributed traffic, similar execution times are observed across all modeling approaches. In contrast, scenarios with CDSB traffic lead to noticeably higher execution times for the detailed communication model, reflecting the increased burstiness and clustering of communication events.

For the meta-model, execution times under CDSB traffic are comparatively lower than those of the detailed simulation. This behavior may be attributed to the fact that bursty

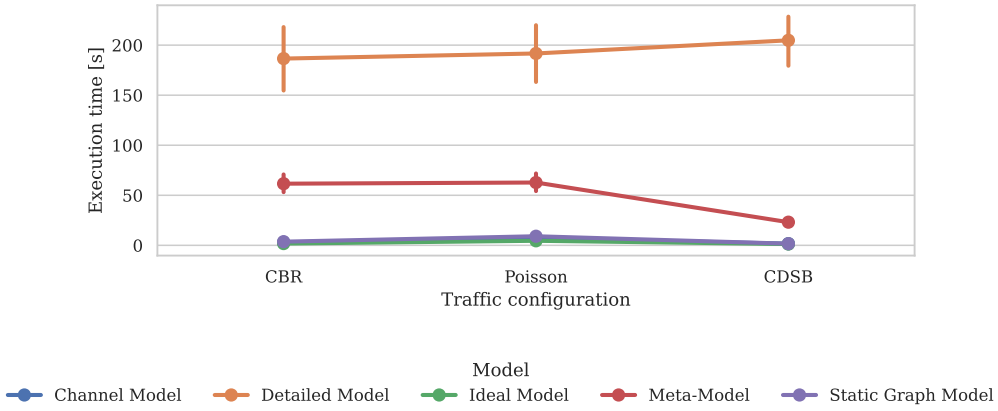


Figure 6.20: Execution time comparison across traffic model classes. Results are aggregated over all network sizes and scenario durations.

traffic accelerates the accumulation of training data and increases the likelihood of substitution, thereby reducing the duration for which the detailed simulation remains active. As a result, the meta-model benefits from complex traffic patterns in terms of performance efficiency, whereas the detailed simulation incurs additional computational overhead.

Overall, the results demonstrate that the meta-model scales more favorably than detailed communication simulation with respect to system size, scenario duration, and traffic complexity. While analytical models remain the most computationally efficient, they do so at the cost of reduced accuracy. The meta-model provides a scalable compromise that substantially improves performance efficiency over detailed simulation while retaining significantly higher fidelity than simple analytical approaches.

6.4.5 Summary of Step 2 Findings

The comparative evaluation in Step 2 yields the following key observations:

- The choice of communication modeling approach has a dominant effect on execution time and a substantial impact on the reproduction of delay variability.
- The meta-model often outperforms simple analytical models and improves upon static graph-based approximations, particularly with respect to distributional accuracy.
- With increasing network size, empirical coverage decreases for all communication modeling approaches due to rising delay variability; the meta-model exhibits a more gradual degradation, indicating partial adaptation to increased system complexity.

- The meta-model achieves significant runtime reductions compared to detailed simulation and scales more favorably with increasing system size, scenario duration, and traffic complexity.
- Accuracy remains stable across different communication technologies and test–training configurations, with predictable degradation under strong distribution shifts such as technology changes.

Taken together, these findings demonstrate that the meta-model constitutes a robust compromise between accuracy and computational efficiency, effectively bridging analytical abstractions and detailed communication simulation. The empirical results are subsequently translated into application-oriented recommendations in Section 6.6, which provide practitioners with concrete guidance on when and how to deploy the proposed approach, and into a formal trade-off formulation in Subsection 6.6.1, which defines a quantitative decision criterion for selecting between detailed simulation and meta-modeling under fixed computational budgets and accuracy requirements.

6.5 Digression: Application to Real-World Data

The main intention behind the meta-modeling approach developed in this work is to approximate simulation models. This aspect has already been evaluated in the previous sections. Another possible application, however, is to use the approach to approximate real systems. This is not the primary focus of this work but will be demonstrated in this section in the form of a case study. At the current stage of this work, no suitable real-world data are publicly available. To be suitable, the data would need to include message information in the form of sender, receiver, packet size, delay times, and timestamp of message transmission. For this reason, real communication data were collected manually. A field test from the project "DEER" was used [13, 14], in which a multi-agent system coordinates the aggregation and market-based offering of power flexibilities from small-scale distributed energy assets, including battery storage systems and heat pumps, to support redispatch operations in low-voltage grids. In this setting, individual *FlexibilityAgents* representing household-level assets communicate with a central *AggregatorAgent*, which consolidates flexibility offers and submits planning data to a market platform. The field test took place in two phases during the second half of 2025, involving ten geographically distributed agents communicating with an aggregator hosted on a virtual machine. The exact content and purpose of the messages are not relevant at this point. The idea of this demonstration is to apply the developed meta-model approach to approximate the expected message delays. This could, for instance, support future system developments involving

time-critical processes by allowing communication delays to be considered already during the design phase of the system. It is assumed that data from one day (October 6th) are available for training the meta-model, and that the model is then used to approximate subsequent messages. Accordingly, the training data include the messages from the first day, and testing is performed on a set of 1 000 additional events (i.e., 500 messages) from the following day.

The procedure can be described as follows:

1. Transformation of the data (containing transmission time, sender, receiver, delay time) into an event format (each message is converted into two events: one for sending and one for receiving)

2. Training of the meta-model using the messages from the first day

3. Testing of different meta-model configurations with respect to various hyper-parameters (as described in Subsection 6.2.4)

4. Recording of the predicted delay times for the messages in the test period and computation of error metrics (MAE and RMSE) between the predictions and the actual delays

5. Selection of the model with the lowest overall error (with respect to RMSE)

The prediction results of the meta-model with the selected best configuration[7] are depicted in Figure 6.21.

The figure illustrates that the meta-model successfully captures the temporal characteristics of communication delays between agents and the aggregator. The predicted curve follows the general trend of the measured delays, reproducing both short-term fluctuations and broader delay variations. The per-segment error annotations reveal that prediction accuracy remains relatively stable across most message intervals, with minor deviations during periods of higher latency dynamics. This demonstrates that the meta-model can approximate real-world network behavior with reasonable precision, even when trained on limited data from a single day. The observed deviations likely reflect stochastic variations in network conditions and message routing, which were not explicitly modeled but are implicitly represented through the meta-model's learned structure.

[7] C-DT=1, C-IP=50, C-LR=0.1

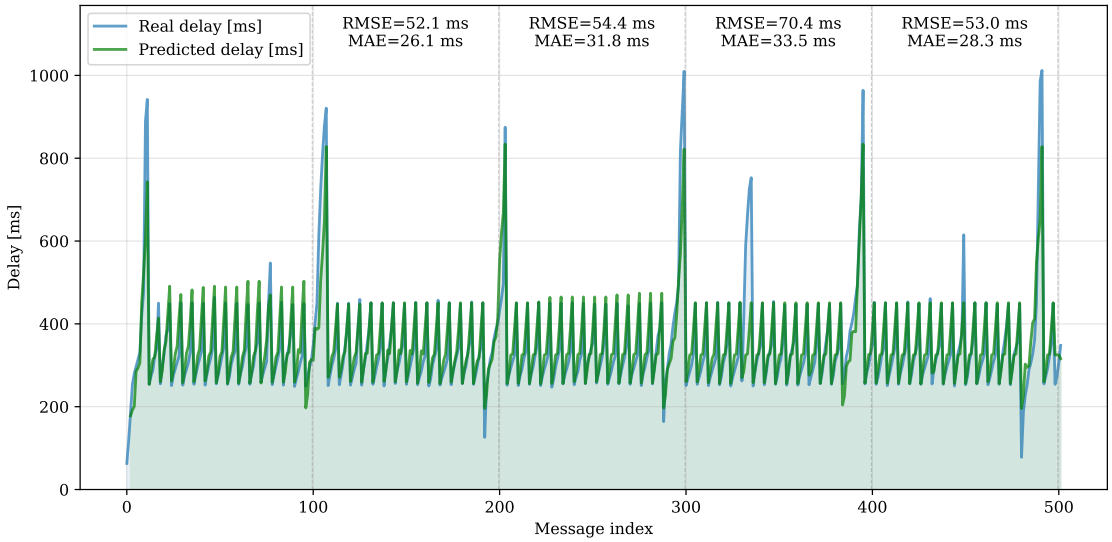


Figure 6.21: Comparison of real and predicted message delay times over sequential message indices. Each dashed vertical line separates 100-message intervals, annotated with corresponding RMSE and MAE values to quantify model performance within each segment.

6.6 Recommendations for Application

The Comparative Evaluation (Step 2) highlights that no single communication modeling approach is universally optimal across all analysis objectives. Instead, the suitability of a modeling approach depends strongly on the focus of the analysis, the dominant NFRs, and the characteristics of the investigated scenarios. To support practitioners and researchers in selecting an appropriate modeling approach, the evaluation results are synthesized into a set of application-oriented recommendations. Figure 6.22 summarizes the recommended communication modeling approaches for different analysis focuses. The figure should be read from top to bottom: starting from the primary objective of an analysis, the dominant non-functional requirements are identified, followed by a suitable modeling recommendation and its key properties and prerequisites. The recommendations are directly derived from the empirical results presented in the preceding sections.

For early-stage conceptual analyses and rapid prototyping, analytical communication models are recommended. Their near-ideal execution time and minimal implementation effort make them well suited for trend exploration and functional testing, where precise delay modeling is not critical. However, their limited accuracy restricts their applicability to exploratory analyses and a suitable abstraction level must be chosen in the implementation (which in turn requires knowledge about the underlying system).

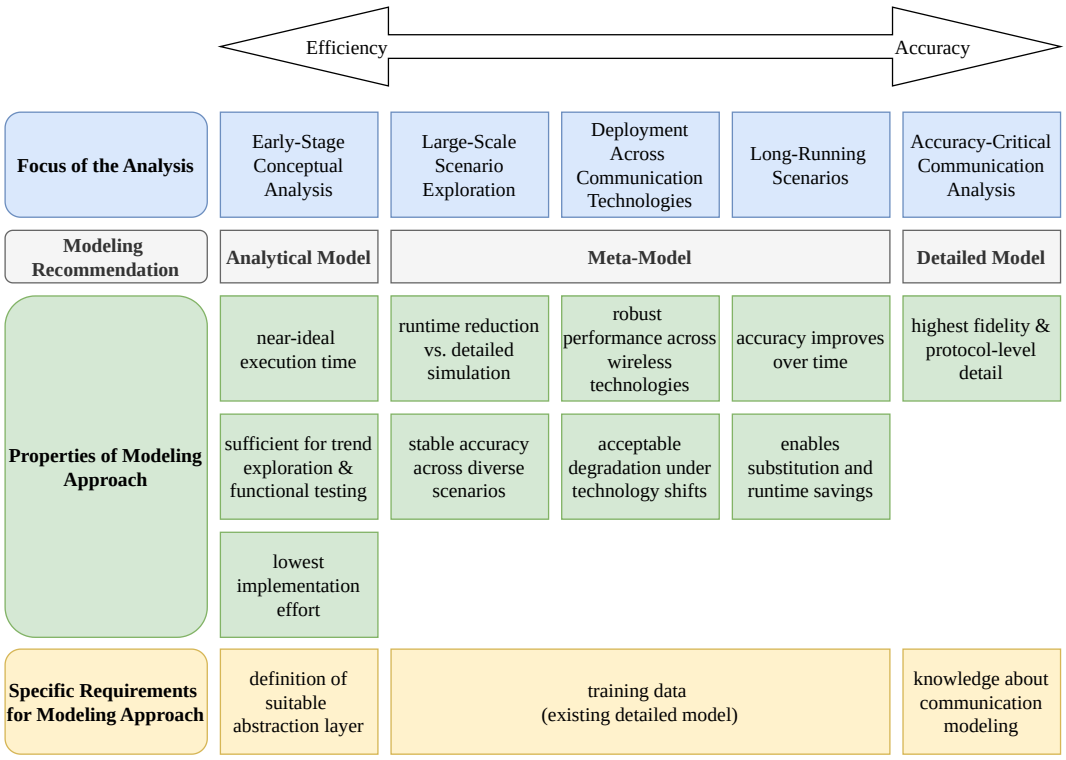


Figure 6.22: Recommended communication modeling approaches for different analysis objectives based on the evaluated non-functional requirements and observed trade-offs. The figure should be read from top to bottom: starting from the primary analysis objective, the appropriate modeling recommendation is identified, followed by the key properties of the recommended approach and the specific requirements that must be met for its application.

For large-scale scenario exploration and sensitivity studies, the meta-model represents the most suitable compromise. The evaluation shows that it provides substantial runtime reductions compared to detailed simulation while maintaining stable accuracy across diverse scenarios. This makes it particularly well suited for studies involving long scenario durations, or repeated execution under varying parameters, provided that representative training data from a detailed model is available.

In contrast, accuracy-critical communication analyses benefit most from detailed communication simulation. When protocol-level fidelity and precise delay modeling are essential, detailed simulators remain the reference approach despite their high computational cost and limited scalability.

The figure further highlights deployment-oriented considerations. For scenarios involving

heterogeneous or wireless communication technologies, the meta-model demonstrates robust accuracy and acceptable degradation under technology shifts, supporting its adaptivity across deployment contexts. In the scenarios under consideration, the online learning capability of the meta-model enables accuracy improvements over time and facilitates substitution, leading to increasing runtime savings as simulations progress.

In addition to the presented recommendations, it is important to note that the scope of the evaluation (and consequently of the meta-model in its current form) is focused on delay-related QoS metrics. For analyses that require additional QoS properties, such as packet loss, jitter correlations beyond delay distributions, or protocol-specific reliability mechanisms, detailed communication simulations remain necessary. Alternatively, the meta-model would need to be extended to explicitly incorporate these metrics into its state representation and learning objectives.

The same limitation applies to scenarios involving disturbances, faults, or malicious behavior, such as link failures, congestion collapse, denial-of-service attacks, or adversarial message injection. These phenomena often depend on low-level protocol interactions and non-stationary network dynamics that are not directly observable from delay measurements alone. In such cases, detailed simulation models provide the required explanatory power, while the meta-model could only be applied if augmented with additional features and training data that capture these effects.

Accordingly, the meta-model is best suited for performance-oriented and scalability-driven analyses under nominal operating conditions. Detailed communication models remain indispensable for investigations targeting fault tolerance, security, and advanced QoS guarantees, whereas future extensions of the meta-model could aim to bridge this gap by integrating richer network state information and disturbance-aware learning mechanisms.

Overall, the recommendations emphasize that the meta-model should be understood neither as a replacement for analytical models nor for detailed simulation, but as a complementary approach. It fills the gap between these extremes by enabling communication-aware analyses that balance accuracy and computational efficiency in scenarios where neither purely analytical nor fully detailed modeling is practical.

6.6.1 Trade-Off Formulation

The selection of an appropriate communication modeling approach can be formalized as a trade-off between achievable statistical accuracy and available computational resources. This formulation is particularly relevant for application scenarios that require repeated simulation runs, such as sensitivity analyses, uncertainty quantification, or large-scale scenario exploration.

For the detailed communication model, assume that the observed delay metric exhibits a standard deviation σ^{real}. To estimate the mean delay with a confidence level $1-\alpha$ and an admissible absolute error ϵ, the required number of simulation runs R_{det} is bounded by Equation 6.6.

For the meta-model, the achievable accuracy is limited not only by statistical variance but also by the inherent approximation error of the model, denoted by ϵ_{cocoon}. Let σ_{cocoon} represent the standard deviation of the meta-model outputs. The required number of runs R_{cocoon} to achieve an overall error bound ϵ is then given by

$$R_{\text{cocoon}} \geq \left(\frac{t_{\alpha/2,R_0} \cdot \sigma^{\text{cocoon}}}{\epsilon - \epsilon_{\text{cocoon}}} \right)^2 \tag{6.7}$$

This formulation assumes $\epsilon > \epsilon_{\text{cocoon}}$, i.e., the target accuracy must exceed the inherent approximation error of the meta-model. Otherwise, the required number of runs is undefined.

In practical applications, the number of executable simulation runs is constrained by an available execution time budget T. Let t_{det} and t_{cocoon} denote the mean execution time per run of the detailed model and the meta-model, respectively. The maximum number of realizable runs within the time budget is

$$n_{\text{det}} = \frac{T}{t_{\text{det}}}, \qquad n_{\text{cocoon}} = \frac{T}{t_{\text{cocoon}}}. \tag{6.8}$$

Under this constraint, the achievable error of the detailed simulation is

$$\epsilon_{\text{achievable,det}} = \frac{t_{\alpha/2,R_0} \cdot \sigma^{\text{real}}}{\sqrt{n_{\text{det}}}} = \frac{t_{\alpha/2,R_0} \cdot \sigma^{\text{real}} \sqrt{t_{\text{det}}}}{\sqrt{T}}, \tag{6.9}$$

whereas the achievable error of the meta-model is

$$\epsilon_{\text{achievable,cocoon}} = \epsilon_{\text{cocoon}} + \frac{t_{\alpha/2,R_0} \cdot \sigma^{\text{cocoon}}}{\sqrt{n_{\text{cocoon}}}} = \epsilon_{\text{cocoon}} + \frac{t_{\alpha/2,R_0} \cdot \sigma^{\text{cocoon}} \sqrt{t_{\text{cocoon}}}}{\sqrt{T}}. \tag{6.10}$$

From an application-oriented perspective, the meta-model is preferable whenever it yields a lower achievable error under the same execution time constraint, i.e.,

$$\epsilon_{\text{achievable,cocoon}} < \epsilon_{\text{achievable,det}}. \tag{6.11}$$

Solving the equality condition

$$\epsilon_{\text{achievable,cocoon}} = \epsilon_{\text{achievable,det}} \tag{6.12}$$

for T yields a critical execution time threshold

$$T_{\text{critical}} = \left(\frac{t_{\alpha/2,R_0} \left(\sigma^{\text{real}} \sqrt{t_{\text{det}}} - \sigma^{\text{cocoon}} \sqrt{t_{\text{cocoon}}} \right)}{\epsilon_{\text{cocoon}}} \right)^2 . \tag{6.13}$$

For execution time budgets below T_{critical}, the meta-model achieves more reliable results than the detailed simulation due to the higher number of realizable runs. Conversely, for sufficiently large time budgets, the detailed simulation becomes preferable as its statistical error can be reduced without incurring approximation bias. This formulation provides a quantitative decision criterion that complements the application-oriented recommendations derived from the evaluation.

Example Consider a confidence level of 95% ($\alpha = 0.05$). Given the number of runs used in the evaluation ($R_0 = 5$), the corresponding value for $t_{\alpha/2,R_0}$ is given as

$$t_{\alpha/2,R_0} = 2.015.$$

Based on the empirical results of Step 2, representative values for delay variability and execution times are:

$$\sigma^{\text{real}} = 25\,\text{ms}, \qquad \sigma^{\text{cocoon}} = 25\,\text{ms},$$

$$t_{\text{det}} = 120\,\text{s}, \qquad t_{\text{cocoon}} = 15\,\text{s}, \qquad \epsilon_{\text{cocoon}} = 5\,\text{ms}.$$

Substituting these values into the trade-off equation yields

$$T_{\text{critical}} \approx 5\,090\,\text{s}. \tag{6.14}$$

Figure 6.23 visualizes the resulting trade-off between computational effort and achievable error for both communication modeling approaches. The left ordinate shows the number of completed simulation runs as a function of the available execution time budget, while the right ordinate depicts the corresponding achievable estimation error. Step-wise curves indicate the discrete nature of realizable simulation runs, whereas continuous curves represent the resulting error bounds derived from the statistical formulation.

For execution time budgets below approximately 5 090 s (about 1.41 hours), the meta-model achieves a lower achievable estimation error than the detailed communication simulation. This advantage arises from its substantially shorter execution time per run, which enables a significantly larger number of realizations within the same time budget. As illustrated in Figure 6.23, this effect outweighs the inherent approximation error of the

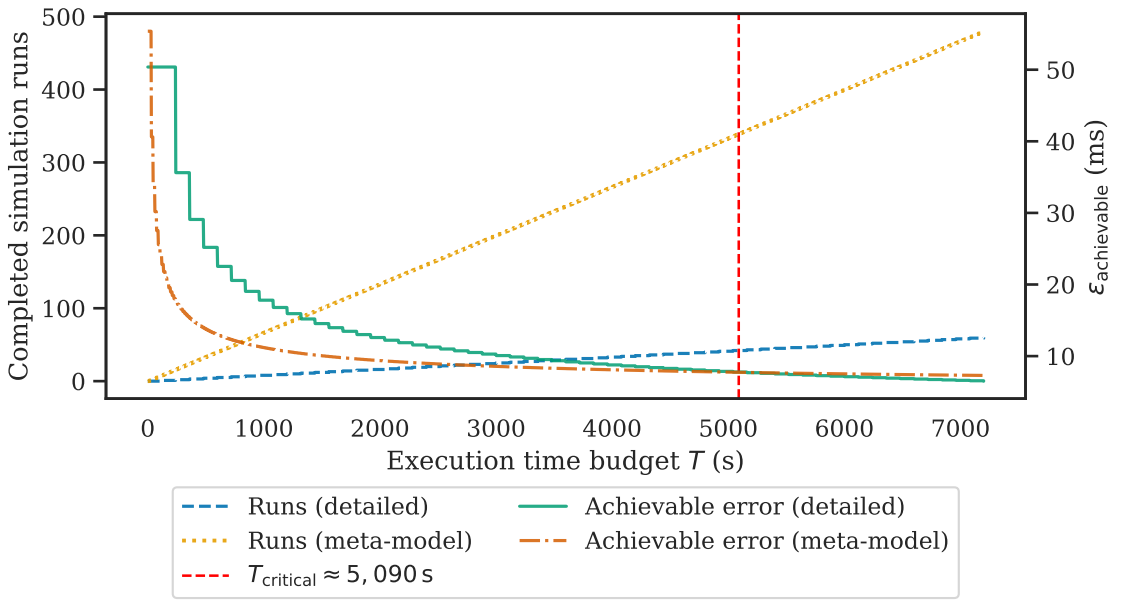


Figure 6.23: Trade-off between execution time budget and achievable estimation accuracy for the detailed communication model and the meta-model.

meta-model in the low-budget regime. Conversely, for execution time budgets exceeding T_{critical}, the detailed simulation becomes preferable, as its purely statistical error can be further reduced without incurring approximation bias.

The sensitivity of the critical execution time threshold T_{critical} to the approximation error of the meta-model and to delay variability can be illustrated by varying these parameters in the trade-off formulation. For instance, reducing the approximation error of the meta-model to $\epsilon_{\text{cocoon}} = 1\,\text{ms}$ shifts the critical threshold to approximately 127 256 s (about 35.3 hours), substantially enlarging the regime in which the meta-model is preferable. Conversely, increasing the approximation error to $\epsilon_{\text{cocoon}} = 10\,\text{ms}$ reduces T_{critical} to approximately 1 273 s (about 21 minutes), indicating that detailed simulation becomes advantageous already at comparatively small execution time budgets.

A similar effect is observed when varying the delay variability. Reducing the standard deviation of delay times to $\sigma = 10\,\text{ms}$ decreases the critical threshold to approximately 814 s, as fewer simulation runs are required to achieve a given statistical confidence. These examples illustrate that T_{critical} depends strongly on both the intrinsic approximation error of the meta-model and the variability of the underlying communication process, and that improvements in approximation accuracy or increases in stochastic variability directly expand the range of scenarios in which the meta-model yields superior results under fixed computational budgets.

6.7 Chapter Summary

This chapter addressed Sub-RQ 4 by systematically evaluating the proposed online-trained communication meta-model in comparison to established analytical and detailed communication modeling approaches. The evaluation was conducted across a broad and representative set of communication scenarios, covering different traffic patterns, network technologies, system scales, and execution horizons.

The results demonstrate that the meta-model fulfills its intended role of reducing computational complexity while maintaining substantially higher fidelity than simple analytical models. In terms of accuracy, the meta-model often outperforms channel-based and static graph models, particularly with respect to capturing delay variability and distributional characteristics, while remaining bounded by the inherent non-determinism of detailed communication simulation. The adaptivity analysis shows that the meta-model generalizes robustly across heterogeneous communication technologies and test–training configurations, confirming its applicability beyond narrowly defined deployment contexts.

With respect to performance efficiency and scalability, the meta-model achieves significant reductions in execution time compared to detailed simulation, especially when substitution occurs. Although execution time increases with growing network size and scenario duration the observed scaling behavior remains markedly more favorable than that of fully detailed communication simulation. This enables communication-aware analysis in scenarios that would otherwise be computationally infeasible.

Finally, the chapter translated these empirical findings into application-oriented recommendations and a formal trade-off formulation. This formulation provides a quantitative decision criterion for selecting between detailed simulation and meta-modeling under fixed computational budgets and accuracy requirements. Taken together, the results provide a comprehensive answer to Sub-RQ 4 by demonstrating that the proposed meta-model constitutes a viable and well-founded compromise between accuracy and computational efficiency for the evaluation of communication effects in CPESs.

In addition, the findings provide direct support for Hypotheses 3 and 4. The evaluation confirms that a principled, execution-time–dependent threshold can be defined beyond which the meta-model approximates the detailed simulation with sufficient accuracy to justify substitution. At the same time, substitution leads to substantial runtime improvements while maintaining accuracy within the variability bounds imposed by the non-deterministic reference simulation. These results demonstrate that both the feasibility and the practical benefit of online substitution can be quantified and systematically assessed in realistic communication-aware simulation settings.

7 Conclusion

> An approximate answer to the right problem is worth a good deal more than an exact answer to an approximate problem.
>
> *John Tukey*

The objective of this thesis was to enable combined energy and communication system simulation in CPESs while mitigating the computational burden imposed by detailed communication network models. As energy systems become increasingly interconnected, digitalized, and adaptive, their communication infrastructures grow in complexity and significance. Paradoxically, this raises a fundamental and seemingly counter-intuitive methodological question: *as systems become more complex, how can the models and methods used to analyze them become simpler rather than more elaborate – without losing explanatory power or validity?*

In current CPES studies, communication effects are often either neglected or modeled at a level of detail that severely limits scalability. While detailed packet-level simulations offer high fidelity, they frequently render large-scale, long-horizon, or many-run analyses computationally infeasible. This tension between increasing system complexity and the need for computationally tractable models forms a central challenge for communication-aware CPES simulation.

To address this challenge, this work proposed the approximation of computationally expensive detailed communication network simulations by means of an online-trained, discrete-event-compatible meta-model. Rather than simplifying the problem by reducing its scope, the approach simplifies the modeling effort by learning the essential input–output behavior of the communication system during execution.

The remainder of this chapter first summarizes the research contributions and how they collectively address the posed research questions. Subsequently, future directions are outlined, highlighting opportunities to extend the proposed approach toward broader metrics, domains, and operational settings.

7.1 Summary

In the introduction of this thesis in Chapter 1, the relevance of communication modeling for CPES analysis was established, and a set of RQs was formulated to structure the investigation. The overarching RQ addressed how meta-models for communication network simulation can be created automatically by approximating input–output behavior during simulation, such that they can be integrated seamlessly into co-simulation environments and applied in heterogeneous CPES scenarios.

The fundamentals and related work reviewed in the Chapter 2 provided the basis for identifying the central research gap. Combined energy and communication simulations offer the opportunity to investigate mutual inter-dependencies in CPESs, but they require DES-compatible communication models for seamless integration. At the same time, detailed packet-level simulations are often computational bottlenecks in scenario-rich or large-scale studies. Meta-models offer a promising means to reduce runtime, yet existing approaches are predominantly trained offline, rely on synthetic traffic models, and act as black-box predictors that are poorly suited for dynamic CPES scenarios. Additionally, communication models must be technology-agnostic, topology-aware, and robust to highly heterogeneous traffic patterns induced by diverse energy system applications. These aspects are further complicated by the inherent trade-off between realism and computational effort in communication modeling.

Against this background, the research gap was summarized as the lack of communication models that (i) preserve essential DES characteristics and can be integrated into CPES co-simulation frameworks, while (ii) avoiding the full computational cost of detailed simulation, and (iii) overcoming the limitations of offline-trained, non-adaptive meta-models.

Subsequent chapters addressed this gap in a structured manner. First, a CPES-specific framework for the abstract representation of communication traffic and scenarios was developed in Chapter 3. Communication traffic was derived from application-level behavior, and parameterized communication network models were generated automatically from power system reference networks. This enabled the systematic creation and simulation of a broad set of representative communication scenarios. The resulting simulation data were analyzed to characterize the diversity of CPES communication behavior and to define the operational space within which the meta-model must generalize.

Building on this foundation, in Chapter 4 the detailed communication network simulation was abstracted into an internal, technology-agnostic representation suitable for meta-modeling. Communication endpoints and their interactions were captured as a dynamically evolving directed graph, enriched with state variables at network and node level. Event semantics and time advancement were formalized to ensure DES compatibility and

executable integration into co-simulation environments.

This internal representation then served as the basis for the development of an online approximation algorithm structured into four distinct phases in Chapter 5. Historical data were used to initialize cluster-based priors, while scenario-specific behavior was learned online during simulation. Ensemble-style mechanisms enabled adaptive weighting between prior knowledge and online learning, and a multi-criteria confidence assessment governed the substitution of detailed simulation by the meta-model at runtime.

Finally, in Chapter 6 the proposed approach was evaluated comprehensively against analytical and detailed communication modeling approaches across a wide range of scenarios. The results demonstrated that the meta-model achieves a favorable compromise between accuracy and computational efficiency: it substantially outperforms simple analytical abstractions in capturing delay behavior and variability, while enabling significant reductions in execution time compared to detailed simulation. The evaluation further confirmed robustness across technologies, traffic patterns, and scales in heterogeneous CPES scenarios.

7.2 Answers to the Research Questions

The overarching RQ of this thesis asked *how online-learned meta-models can be generated and integrated into discrete-event communication network simulations such that detailed models can be substituted while preserving simulation fidelity.*

This work demonstrates that such meta-models can indeed be constructed by observing the input–output behavior of detailed communication simulations and training predictive models online during simulation execution. By embedding the meta-model implemented in *cocoon* into the overall simulation set-up and continuously evaluating its prediction quality, detailed simulations can be substituted dynamically once sufficient confidence is achieved. The results show that this approach preserves essential delay characteristics while substantially reducing computational complexity, thereby enabling communication-aware energy system studies that would otherwise be infeasible at scale.

Sub-RQ 1 addressed *how communication behavior in cyber-physical energy systems can be abstracted into simulation-relevant input–output characteristics.*

The thesis shows that heterogeneous communication behavior can be systematically described using traffic models combined with structured communication network representations[1]. By abstracting communication behavior in terms of message generation processes, payload characteristics, and network topology, realistic communication scenarios can be

[1]The generation of representative communication scenarios and simulation data is realized in *trace*: `https://github.com/OFFIS-DAI/trace`

generated and simulated. These abstractions provide a sufficient basis for detailed communication simulation while remaining independent of specific application logic.

Sub-RQ 2 investigated *how observations from discrete-event communication simulations can be transformed into representations suitable for online training of communication meta-models.*

The results demonstrate that communication events observed during simulation execution can be transformed into feature representations that capture both local message characteristics and contextual network information. Using these representations, online learning algorithms can be trained incrementally to approximate the response of the detailed simulation[2]. The proposed transformation enables continuous adaptation of the meta-model to changing communication conditions without interrupting simulation execution.

Sub-RQ 3 focused on *identifying quantitative criteria that allow assessing accuracy, temporal consistency, and robustness of online-trained meta-models to support principled substitution decisions.*

This work shows that substitution can be guided by a combination of different factors. The introduced threshold estimation methodology allows the simulation to decide when approximation quality is sufficient to replace the detailed model. The evaluation confirms that these criteria enable robust substitution decisions while avoiding uncontrolled degradation of simulation fidelity.

Sub-RQ 4 examined *under which communication scenario characteristics an online-trained meta-model maintains validity compared to established communication modeling approaches.*

The comparative evaluation demonstrates that the meta-model consistently outperforms simple analytical communication models in terms of accuracy while achieving substantial runtime reductions compared to detailed simulation. The results further show that model choice has a dominant influence on performance efficiency and delay variability, whereas mean delay accuracy is largely governed by scenario characteristics. Overall, the meta-model maintains validity across a wide range of traffic patterns, network technologies, and system scales, provided that representative training data are available.

[2]The online-trained communication meta-model is implemented in the *cocoon* framework, which integrates online learning, substitution logic, and scheduler-based coupling into discrete-event simulations: `https://github.com/OFFIS-DAI/cocoon`

7.3 Future Directions

The results presented in this thesis provide a foundation for a range of subsequent research directions. In order to establish a coherent and tractable contribution, the scope of this work was necessarily constrained. Several promising extensions and open research questions arise naturally from these limitations and are outlined in the following.

Extension to Additional QoS Metrics In its current form, the proposed meta-model approximates end-to-end message delay as the primary communication-related QoS metric. While delay constitutes a central factor for many analyses, a comprehensive assessment of communication effects in energy system operation and control may require the consideration of additional QoS properties, such as packet loss, jitter, or reliability-related metrics, as discussed for example in [6]. Future work could therefore extend the meta-modeling approach to incorporate multiple QoS metrics as joint prediction targets. This would likely require a refinement and extension of the internal state representation in order to capture a broader range of network dynamics.

Disturbance-, Fault-, and Attack-Aware Modeling The communication scenarios considered in this thesis focus on nominal operating conditions and do not explicitly include disturbances, faults, or adversarial behavior. However, such phenomena are highly relevant when analyzing the resilience and robustness of CPESs. A natural extension of this work would be to investigate how fault and attack scenarios can be integrated into the scenario construction process and whether their effects can be captured by the meta-modeling approach. In this context, classifications of system states into normal, limited, and failed operation, as proposed in [24], could serve as a conceptual basis for extending the model state space and learning objectives.

Adaptive Hyper-Parameter Tuning During Execution At present, the hyper-parameters governing clustering behavior, learning dynamics, and substitution decisions must be specified prior to simulation execution. While this design choice simplifies experimental control, it limits usability. Future research could investigate adaptive hyper-parameter tuning strategies that operate online during simulation execution, enabling the meta-model to adjust its behavior automatically in response to observed performance and accuracy. Such mechanisms could further reduce manual configuration effort and improve robustness across heterogeneous scenarios.

Uncertainty Quantification of Meta-Model Outputs The substitution mechanism introduced in the BUTTERFLY PHASE governs *when* the meta-model is sufficiently reliable to replace the detailed simulator. However, once substitution has occurred, the predicted delay d_{pred} is provided as a point estimate, leaving the residual approximation uncertainty implicit. In co-simulation environments, this limitation is non-trivial: as shown by Steinbrink and Lehnhoff [102], uncertainties of co-simulated subsystems propagate and compound across simulation components, so that unquantified output uncertainty in one simulator may distort the behavior of downstream models. A natural extension of *cocoon* within a *mosaik*-based co-simulation would therefore be to complement the substitution decision with post-substitution uncertainty quantification. Two complementary sources of uncertainty are relevant in this context. First, *hyperparameter uncertainty* arises from the sensitivity of d_{pred} to the choice of meta-model hyper-parameters; treating each hyper-parameter configuration as a sample from an uncertain parameter space and pooling the resulting delay distributions yields a probabilistic output interval. Second, *output parameter uncertainty* captures the residual stochastic variability in d_{pred} itself - the spread of delay values the meta-model produces across repeated evaluations under identical conditions. Following the ensemble-based probabilistic propagation approach of [102], *cocoon* could be wrapped by a Uncertainty Quantification (UQ) component that jointly accounts for both sources and expresses d_{pred} as a probability distribution rather than a point estimate, propagating residual approximation uncertainty into downstream system simulations. This integration is architecturally feasible because *cocoon* could itself be a black-box simulator component from *mosaik*'s perspective, satisfying the non-intrusive precondition of the approach. To keep the additional computational overhead acceptable, the ensemble would operate over multiple instances of *cocoon* rather than of the detailed simulator, exploiting the fact that *cocoon* is computationally cheap relative to *OMNeT++*.

Online Adaptation to Live Network Conditions Beyond simulation-based applications, an interesting future direction is the deployment of the meta-model alongside a real communication system. By continuously updating its parameters based on observed delays and traffic patterns, the meta-model could provide a near-real-time approximation of current network behavior. This capability could support decision-making, monitoring, or what-if analyses in operational environments, bridging the gap between simulation-based studies and live system observation.

Meta-Modeling of Multi-Domain Co-Simulations The principles developed in this thesis are not inherently limited to communication network simulation. A broader research direction is the application of online-trained meta-models to approximate entire

sub-simulators within multi-domain co-simulation environments. This could enable the selective replacement of computationally expensive components across domains such as communication, power system dynamics, or control logic, thereby improving scalability while retaining essential cross-domain interactions.

Networked Control Systems and Industrial Automation Finally, the proposed approach could be transferred to domains such as networked control systems, industrial automation, or cooperative robotic systems, where communication delays and variability directly affect control stability and performance. Applying online-trained meta-models in these contexts may support scalable analysis of timing effects and robustness in large-scale or safety-critical control applications.

A | Appendix

Own Contribution to the Publications Cited

This section summarizes the author's individual contributions to the publications cited in this thesis. For each publication, contributions are reported in a compact and standardized form following the CRediT taxonomy (Contributor Roles Taxonomy). The labels *lead*, *equal*, and *supporting* indicate whether the author primarily drove a contribution, contributed comparably to other authors, or contributed in a supporting role.

Publication [2]
F. Oest, M. Radtke, M. Blank-Babazadeh, S. Holly, and S. Lehnhoff, "Evaluation of Communication Infrastructures for Distributed Optimization of Virtual Power Plant Schedules," Energies, vol. 14, no. 5, Art. no. 5, 2021.
CRediT: Conceptualization & Methodology: supporting; Software: lead (implementation of simulation setup and execution of experiments); Writing: equal

Publication [3]
F. Oest, E. Frost, and M. Radtke, "Coupling OMNeT++ and mosaik for Integrated Co-Simulation of ICT-Reliant Smart Grids," ACM SIGEnergy Energy Informatics Review, 2023.
CRediT: Conceptualization & Methodology: equal; Software: equal; Writing: equal

Publication [4]
E. Frost, M. Radtke, M. Nebel-Wenner, F. Oest, and S. Stark, "cosima-mango: Investigating Multi-Agent System Robustness through Integrated Communication Simulation," SoftwareX, vol. 26, Art. 101667, 2024.
CRediT: Conceptualization & Methodology: equal (with E. Frost); Software: equal (with E. Frost); Writing: equal

Publication [5]
M. Radtke et al., "Integrating Agent-Based Control for Normal Operation in Interconnected Power and Communication Systems Simulation," in Proceedings of the IEEE Symposium

Series on Computational Intelligence (SSCI), 2023, pp. 228–233.
CRediT: Conceptualization & Methodology: lead; Software: equal (with most authors); Writing: equal (with most authors)

Publication [6]
M. Radtke and E. Frost, "Simulative Analysis of Multi-Agent Systems in Energy Systems: Impact of Communication Networks," in Proceedings of the 17th International Conference on Agents and Artificial Intelligence (ICAART), vol. 1, pp. 523–530, 2025.
CRediT: Conceptualization & Methodology: equal (with E. Frost); Software: –; Writing: equal (with E. Frost)

Publication [7]
M. Radtke, E. Frost, A. Nieße, and S. Lehnhoff, "Communication Modeling Approaches in Energy System Applications: A Systematic Overview," IEEE Access, vol. 13, 2025.
CRediT: Conceptualization & Methodology: equal (with E. Frost); Software: –; Writing: equal (with E. Frost)

Publication [8]
M. Radtke, "Poster Abstract: Towards Online Meta-Modeling of Communication Networks in Energy Systems," ACM SIGEnergy Energy Informatics Review, 2024.
CRediT: Conceptualization & Methodology: lead; Software: –; Writing: lead

Publication [9]
M. Radtke and S. Lehnhoff, "Enhancing Cyber-Physical Energy Systems Simulations Through Communication Behavior Classification," 2024.
CRediT: Conceptualization & Methodology: lead; Software: lead; Writing: lead

Publication [1]
M. Radtke and S. Lehnhoff, "Understanding the Diversity of Communication Scenarios in Cyber-Physical Energy Systems," Energy Informatics Review, 2025.
CRediT: Conceptualization & Methodology: lead; Software: lead; Writing: lead

Publication [10]
M. Radtke and S. Lehnhoff, "Online Meta-Modeling of Communication Network Simulations for Agent-Based Energy Systems," in Proceedings of IEEE SmartGridComm, 2025.
CRediT: Conceptualization & Methodology: lead; Software: lead; Writing: lead

Publication [11]
M. Radtke, E. Frost, S. Holly, A. Nieße, and S. Lehnhoff, "Multi-Agent System Analysis under Communication Influence: Practical Applications," Lecture Notes in Artificial

Intelligence, 2025.
CRediT: Conceptualization & Methodology: equal (with E. Frost); Software: equal (with E. Frost); Writing: equal (with E. Frost)

Publication [13]
M. Radtke, S. Stark, and S. Holly, "Agent-Based Flexibility Aggregation for a Distributed Redispatch," in Energy Informatics, Springer, 2026, pp. 354–370.
CRediT: Conceptualization & Methodology: equal (with S. Holly); Software: equal (with S. Holly); Writing: equal (with S. Holly)

Publication [14]
"A Multi-Agent Approach with Verifiable and Data-Sovereign Information Flows for Decentralizing Redispatch in Distributed Energy Systems," Energy Informatics, vol. 24, 2025.
CRediT: Conceptualization & Methodology: equal (with most authors); Software: –; Writing: equal (with most authors)

Publication [15]
M. Radtke, S. Holly, and A. Nieße, "Distributed Flexibility Fitness Landscape Analysis for Parameterization of Algorithms in Multi-Agent Energy Systems," IDC 2023.
CRediT: Conceptualization & Methodology: lead; Software: lead; Writing: lead

Use of Generative AI

Generative AI tools were used in this dissertation solely in a supportive manner. *ChatGPT* was employed for brainstorming, conceptual discussions, and assistance with Python-based plotting and visualization. In addition, tools such as *DeepL Write* were used for linguistic refinement and formulation support. All scientific content, analyses, and conclusions are the result of the author's own work. All sources are based on and cited from established scientific literature.

Historical and Current Use of the Term "Online"

Early work in theoretical computer science and optimization often used the hyphenated form *on-line* to describe algorithms that process inputs sequentially as they arrive, without knowledge of future requests, in contrast to *off-line* algorithms that have access to the entire input in advance [111, 112]. Over time, the field has converged toward the closed form *online*, as seen in modern textbooks and survey articles on online algorithms and competitive analysis, which define an *online algorithm* as one that must react to a sequence of inputs without seeing the future and compare its performance to an optimal *offline algorithm* [113]. In parallel, machine learning and signal-processing communities have adopted the term *online learning* or *online machine learning* to denote methods that update model parameters incrementally as new data arrive, contrasting them with batch or offline learning that trains on a fixed dataset [114, 115].

Throughout this thesis, the term *online* (without a hyphen) will be used for both online algorithms and online learning procedures, while noting that older literature may use the synonymous spelling *on-line* in the same technical sense.

Result Tables for Comparative Evaluation

The following tables summarize the detailed numerical results of the Comparative Evaluation (Step 2). They report aggregated metric values for all evaluated communication modeling approaches across the considered scenario factors. Results are grouped by factor and stage, and values are reported as mean ± standard deviation over all corresponding scenarios and simulation runs.

The tables provide a comprehensive quantitative reference for the comparative analysis presented in Chapter 6. They are intended to support transparency and reproducibility, while the main body of the thesis focuses on the interpretation of trends and trade-offs rather than individual numeric values.

Table A.1: Overview on step 2 results for $NRMSE$ metric.

Factor	**Stages**	$NRMSE$						
CM		Channel Model	Static Graph Model	Meta-Model				
C-TTS				pa	tl	tm	sc	te
T	CBR	0.61 ± 0.25	0.19 ± 0.14	0.18 ± 0.22	0.20 ± 0.25	0.21 ± 0.28	0.21 ± 0.28	0.23 ± 0.24
	Poisson	0.60 ± 0.26	0.22 ± 0.14	0.21 ± 0.25	0.19 ± 0.20	0.24 ± 0.28	0.22 ± 0.27	0.24 ± 0.24
	CDSB	0.71 ± 0.18	0.23 ± 0.18	0.20 ± 0.25	0.21 ± 0.25	0.31 ± 0.22	0.22 ± 0.25	0.23 ± 0.19
S	5 min	0.64 ± 0.24	0.21 ± 0.16	0.20 ± 0.24	0.19 ± 0.24	0.25 ± 0.26	0.21 ± 0.27	0.23 ± 0.22
	10 min	0.64 ± 0.24	0.21 ± 0.16	0.20 ± 0.24	0.20 ± 0.23	0.26 ± 0.26	0.22 ± 0.27	0.24 ± 0.22
ND	5	0.54 ± 0.28	0.16 ± 0.08	0.03 ± 0.04	0.03 ± 0.04	0.07 ± 0.10	0.03 ± 0.05	0.05 ± 0.06
	10	0.58 ± 0.24	0.16 ± 0.08	0.10 ± 0.09	0.09 ± 0.08	0.17 ± 0.14	0.09 ± 0.08	0.15 ± 0.09
	20	0.64 ± 0.19	0.19 ± 0.09	0.18 ± 0.13	0.18 ± 0.14	0.21 ± 0.13	0.18 ± 0.13	0.23 ± 0.10
	50	0.78 ± 0.13	0.34 ± 0.24	0.50 ± 0.28	0.48 ± 0.26	0.55 ± 0.31	0.56 ± 0.31	0.52 ± 0.23
CN	LTE	0.46 ± 0.18	0.29 ± 0.17	0.26 ± 0.20	0.27 ± 0.20	0.30 ± 0.20	0.28 ± 0.21	0.27 ± 0.21
	LTE450	0.46 ± 0.18	0.29 ± 0.17	0.26 ± 0.19	0.27 ± 0.20	0.30 ± 0.20	0.28 ± 0.21	0.28 ± 0.22
	Ethernet	0.93 ± 0.04	0.05 ± 0.04	0.22 ± 0.36	0.21 ± 0.32	0.30 ± 0.39	0.26 ± 0.40	0.32 ± 0.26
	5G	0.69 ± 0.06	0.22 ± 0.05	0.06 ± 0.07	0.05 ± 0.06	0.10 ± 0.12	0.06 ± 0.06	0.08 ± 0.09

Table A.2: Overview on step 2 results for $C_{\pm\sigma}$ (sigma-interval coverage).

Factor	**Stages**	$C_{\pm\sigma}$						
CM		Channel Model	Static Graph Model	Meta-Model				
C-TTS				pa	tl	tm	sc	te
T	CBR	0.00 ± 0.00	0.19 ± 0.27	0.50 ± 0.43	0.50 ± 0.42	0.51 ± 0.41	0.52 ± 0.41	0.39 ± 0.40
	Poisson	0.00 ± 0.00	0.11 ± 0.13	0.37 ± 0.33	0.37 ± 0.35	0.38 ± 0.34	0.37 ± 0.33	0.31 ± 0.35
	CDSB	0.00 ± 0.00	0.17 ± 0.30	0.50 ± 0.37	0.48 ± 0.37	0.35 ± 0.33	0.48 ± 0.38	0.35 ± 0.32
S	5 min	0.00 ± 0.00	0.16 ± 0.24	0.46 ± 0.38	0.47 ± 0.39	0.43 ± 0.37	0.47 ± 0.38	0.37 ± 0.36
	10 min	0.00 ± 0.00	0.15 ± 0.24	0.45 ± 0.38	0.44 ± 0.39	0.40 ± 0.36	0.44 ± 0.38	0.33 ± 0.35
ND	5	0.00 ± 0.00	0.16 ± 0.30	0.77 ± 0.30	0.76 ± 0.31	0.72 ± 0.33	0.75 ± 0.32	0.68 ± 0.35
	10	0.00 ± 0.00	0.16 ± 0.20	0.55 ± 0.33	0.56 ± 0.34	0.49 ± 0.34	0.55 ± 0.33	0.40 ± 0.30
	20	0.00 ± 0.00	0.13 ± 0.20	0.40 ± 0.38	0.37 ± 0.38	0.31 ± 0.33	0.40 ± 0.38	0.24 ± 0.28
	50	0.00 ± 0.00	0.18 ± 0.27	0.12 ± 0.15	0.12 ± 0.15	0.14 ± 0.15	0.13 ± 0.14	0.08 ± 0.15
CN	LTE	0.00 ± 0.00	0.03 ± 0.05	0.26 ± 0.31	0.25 ± 0.31	0.24 ± 0.30	0.25 ± 0.30	0.26 ± 0.31
	LTE450	0.00 ± 0.00	0.03 ± 0.05	0.26 ± 0.31	0.25 ± 0.31	0.24 ± 0.31	0.25 ± 0.30	0.25 ± 0.31
	Ethernet	0.00 ± 0.00	0.53 ± 0.19	0.65 ± 0.36	0.63 ± 0.39	0.63 ± 0.32	0.66 ± 0.34	0.22 ± 0.32
	5G	0.00 ± 0.00	0.03 ± 0.06	0.66 ± 0.32	0.67 ± 0.32	0.55 ± 0.36	0.66 ± 0.33	0.66 ± 0.30

Table A.3: Overview on step 2 results for Wasserstein distance W.

Factor	**Stages**	W						
CM		Channel Model	Static Graph Model	Meta-Model				
C-TTS				pa	tl	tm	sc	te
T	CBR	17.68 ± 11.39	4.70 ± 5.26	5.24 ± 7.45	5.67 ± 8.33	5.79 ± 8.77	5.91 ± 8.88	6.74 ± 7.88
	Poisson	17.76 ± 11.96	5.36 ± 5.51	5.78 ± 8.12	5.44 ± 6.94	6.69 ± 9.22	6.28 ± 9.05	7.28 ± 8.60
	CDSB	24.33 ± 9.40	7.84 ± 8.58	6.22 ± 8.86	6.33 ± 8.88	8.55 ± 7.65	6.58 ± 8.97	7.00 ± 7.62
S	5 min	19.94 ± 11.38	5.96 ± 6.75	5.73 ± 8.19	5.74 ± 8.08	6.85 ± 8.64	6.20 ± 8.99	6.88 ± 8.04
	10 min	19.90 ± 11.38	5.97 ± 6.74	5.76 ± 8.11	5.89 ± 8.08	7.18 ± 8.61	6.32 ± 8.91	7.13 ± 8.01
ND	5	16.19 ± 12.74	3.33 ± 1.70	0.44 ± 0.64	0.63 ± 1.35	1.48 ± 2.61	0.65 ± 1.06	1.15 ± 2.20
	10	17.58 ± 11.91	3.53 ± 2.05	1.83 ± 1.69	1.73 ± 1.61	3.36 ± 3.32	1.71 ± 1.59	3.54 ± 2.98
	20	19.91 ± 10.42	4.75 ± 2.67	4.17 ± 3.44	4.50 ± 3.69	4.99 ± 3.07	4.16 ± 3.42	6.03 ± 3.09
	50	26.01 ± 7.31	12.26 ± 10.70	16.55 ± 9.35	16.39 ± 9.24	18.22 ± 9.81	18.51 ± 9.86	17.29 ± 9.00
CN	LTE	14.80 ± 9.94	9.13 ± 8.17	8.01 ± 8.03	8.25 ± 8.35	9.00 ± 8.42	8.62 ± 8.86	8.23 ± 8.74
	LTE450	14.81 ± 9.99	9.11 ± 8.17	7.82 ± 7.64	8.31 ± 8.50	9.09 ± 8.76	8.67 ± 8.89	8.61 ± 9.08
	Ethernet	34.64 ± 3.46	1.11 ± 0.82	6.10 ± 10.11	5.94 ± 9.18	8.09 ± 10.75	6.87 ± 11.08	10.36 ± 7.42
	5G	15.46 ± 4.13	4.49 ± 1.22	0.86 ± 1.05	0.82 ± 1.09	1.95 ± 2.57	0.93 ± 1.17	1.20 ± 1.45

List of Figures

List of Tables

Own publications

[1] Radtke, M., Lehnhoff, S.: Understanding the diversity of communication scenarios in cyber-physical energy systems. SIGENERGY Energy Inform. Rev. **5**(3), 40–51 (2025). doi:10.1145/3777518.3777522

[2] Oest, F., Radtke, M., Blank-Babazadeh, M., Holly, S., Lehnhoff, S.: Evaluation of communication infrastructures for distributed optimization of virtual power plant schedules. Energies (2021). doi:10.3390/en14051226

[3] Oest, F., Frost, E., Radtke, M., Lehnhoff, S.: Coupling OMNeT++ and mosaik for integrated co-simulation of ICT-reliant smart grids. SIGENERGY Energy Inform. Rev. **3**(1), 14–25 (2023). doi:10.1145/3607120.3607123

[4] Frost, E., Radtke, M., Nebel-Wenner, M., Oest, F., Stark, S.: cosima-mango: Investigating Multi-Agent System robustness through integrated communication simulation. SoftwareX **26**, 101667 (2024). doi:10.1016/j.softx.2024.101667

[5] Radtke, M., Stucke, C., Trauernicht, M., Montag, C., Oest, F., Frost, E., Bremer, J., Lehnhoff, S.: Integrating Agent-Based Control for Normal Operation in Interconnected Power and Communication Systems Simulation. In: 2023 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 228–233 (2023). doi:10.1109/SSCI52147.2023.10371932

[6] Radtke, M., Frost, E.: Simulative analysis of multi-agent systems in energy systems: Impact of communication networks. In: Proceedings of the 17th International Conference on Agents and Artificial Intelligence (ICAART 2025) - Volume 1, pp. 523–530. SCITEPRESS – Science and Technology Publications, Lda, (2025). doi:10.5220/0013241400003890

[7] Radtke, M., Frost, E., Nieße, A., Lehnhoff, S.: Communication modeling approaches in energy system applications: A systematic overview. IEEE Access **13**, 80329–80339 (2025). doi:10.1109/ACCESS.2025.3566594

[8] Radtke, M.: Poster abstract: Towards online meta-modeling of communication networks in energy systems. ACM SIGEnergy Energy Informatics Re-

view (2024). https://energy.acm.org/eir/poster-abstract-towards-online-meta-modeling-of-communication-networks-in-energy-systems/

[9] Radtke, M., Lehnhoff, S.: Enhancing Cyber-Physical Energy Systems simulations through communication behavior classification. European Simulation and Modelling Multiconference (ESM '24), 5 (2024)

[10] Radtke, M., Lehnhoff, S.: Online meta-modeling of communication network simulations for agent-based energy systems. In: 2025 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), pp. 1–6 (2025). doi:10.1109/SmartGridComm65349.2025.11204595

[11] Radtke, M., Frost, E., Holly, S., Nieße, A., Lehnhoff, S.: Multi-agent system analysis under communication influence: Practical applications. In: Lecture Notes in Artificial Intelligence (2025). accepted for publication

[12] Radtke, M., Frost, E.: Communication Modeling Approaches in Energy System Applications: A Systematic Overview. Open Research Knowledge Graph (2024). doi:10.48366/R727659. https://orkg.org/comparisons/R727659

[13] Radtke, M., Stark, S., Holly, S.: Agent-Based flexibility aggregation for a distributed redispatch. In: Martinac, I., Jørgensen, B.N., Ma, Z.G., Unnþórsson, R., Bordin, C. (eds.) Energy Informatics, pp. 354–370. Springer, Cham (2026). doi:10.1007/978-3-032-03101-3_25

[14] Heess, P., Holly, S., Körner, M.-F., Nieße, A., Radtke, M., Schick, L., Stark, S., Strüker, J., Zwede, T.: A multi-agent approach with verifiable and data-sovereign information flows for decentralizing redispatch in distributed energy systems. Energy Informatics, 24 (2025). doi:10.1186/s42162-024-00464-7

[15] Radtke, M., Holly, S., Nieße, A.: Distributed flexibility fitness landscape analysis for parameterization of algorithms in multi-agent energy systems. IDC 2023 (2023). doi:10.1007/978-3-031-60023-4_17

References

[16] NSF: NSF Workshop on Cyber-Physical Systems. https://cps-vo.org/node/179. [Online; last access 2025-09-25] (2006)

[17] Parizad, A., Baghaee, H.R., Rahman, S.: Overview of smart cyber-physical power systems: Fundamentals, challenges, and solutions. In: Smart Cyber-Physical Power Systems, pp. 1–69. John Wiley & Sons, Ltd, (2025). doi:10.1002/9781394191529.ch1

[18] Zografopoulos, I., Ospina, J., Liu, X., Konstantinou, C.: Cyber-physical energy systems security: Threat modeling, risk assessment, resources, metrics, and case studies. IEEE Access **9**, 29775–29818 (2021). doi:10.1109/ACCESS.2021.3058403

[19] Abdelmalak, M., Venkataramanan, V., Macwan, R.: A survey of cyber-physical power system modeling methods for future energy systems. IEEE Access **10**, 99875–99896 (2022). doi:10.1109/ACCESS.2022.3206830

[20] Cao, Y., Li, Y., Liu, X., Rehtanz, C.: Cyber-Physical Energy and Power Systems: Modeling, Analysis and Application. Springer, Singapore (2020). doi:10.1007/978-981-15-0062-6

[21] Zhao, A.P., Li, S., Gu, C., Yan, X., Hu, P.J.-H., Wang, Z., Xie, D., Cao, Z., Chen, X., Wu, C., Luo, T., Wang, Z., Hernando-Gil, I.: Cyber vulnerabilities of energy systems. IEEE Journal of Emerging and Selected Topics in Industrial Electronics **5**(4), 1455–1469 (2024). doi:10.1109/JESTIE.2024.3434350

[22] Palensky, P., van der Meer, A., Lopez, C., Joseph, A., Pan, K.: Applied cosimulation of intelligent power systems: Implementing hybrid simulators for complex power systems. IEEE Industrial Electronics Magazine **11**(2), 6–21 (2017). doi:10.1109/MIE.2017.2671198

[23] Palensky, P., Van Der Meer, A.A., Lopez, C.D., Joseph, A., Pan, K.: Cosimulation of intelligent power systems: Fundamentals, software architecture, numerics, and coupling. IEEE Industrial Electronics Magazine **11**(1), 34–50 (2017). doi:10.1109/MIE.2016.2639825

[24] Narayan, A., Brand, M., Lehnhoff, S.: Quantifying the resilience of ICT-enabled grid services in cyber-physical energy system. Energy Informatics **6**(1), 23 (2023). doi:10.1186/s42162-023-00287-y

[25] Priyadarshana, H.V.V., Sandaru, M.K., Hemapala, K., Wijayapala, W.: A review on multi-agent system based energy management systems for micro grids. AIMS Energy **7**(6), 924–943 (2019). doi:10.3934/energy.2019.6.924

[26] Coelho, V.N., Cohen, M.W., Coelho, I.M., Liu, N., Guimarães, F.G.: Multi-agent systems applied for energy systems integration: State-of-the-art applications and trends in microgrids. Applied Energy **187**, 820–832 (2017). doi:10.1016/j.apenergy.2016.10.056

[27] Roche, R., Blunier, B., Miraoui, A., Hilaire, V., Koukam, A.: Multi-agent systems for grid energy management: A short review. In: IECON 2010 - 36th Annual Conference on IEEE Industrial Electronics Society, pp. 3341–3346 (2010). doi:10.1109/IECON.2010.5675295

[28] Wooldridge, M.: An Introduction to Multi-Agent Systems. John Wiley & Sons, (2009). ISBN: 978-0-470-51946-2

[29] Russell, S., Norvig, P.: Artificial Intelligence, Global Edition. Pearson Education, (2021). ISBN: 978-1-292-40117-1

[30] Klaes, M., Zwartscholten, J., Narayan, A., Lehnhoff, S., Rehtanz, C.: Impact of ICT latency, data loss and data corruption on active distribution network control. IEEE Access **11**, 14693–14701 (2023). doi:10.1109/ACCESS.2023.3243255

[31] Alfalouji, Q., Schranz, T., Falay, B., Wilfling, S., Exenberger, J., Mattausch, T., Gomes, C., Schweiger, G.: Co-simulation for buildings and smart energy systems — a taxonomic review. Simulation Modelling Practice and Theory **126**, 102770 (2023). doi:10.1016/j.simpat.2023.102770

[32] Jiang, P., Zhou, Q., Shao, X.: Surrogate model-based engineering design and optimization. In: Surrogate Model-Based Engineering Design and Optimization, pp. 1–5. Springer, Singapore (2020). doi:10.1007/978-981-15-0731-1_1

[33] Mestres, A., Alarcón, E., Ji, Y., Cabellos-Aparicio, A.: Understanding the modeling of computer network delays using neural networks. In: Proceedings of the 2018 Workshop on Big Data Analytics and Machine Learning for Data Communication Networks, pp. 46–52. ACM, Budapest Hungary (2018). doi:10.1145/3229607.3229613

[34] Ferriol-Galmés, M., Paillisse, J., Suárez-Varela, J., Rusek, K., Xiao, S., Shi, X., Cheng, X., Barlet-Ros, P., Cabellos-Aparicio, A.: Routenet-fermi: Network modeling with graph neural networks (2022). doi:10.48550/arXiv.2212.12070

[35] Ziazet, J.M., Boudreau, C., Jaumard, B., Duong, H.: Addressing routenet scalability through input and output design. ITU Journal on Future and Evolving Technologies (2022). doi:10.52953/GIOD4389

[36] Veerakumar, N., Cremer, J.L., Popov, M.: Dynamic incremental learning for real-time disturbance event classification. International Journal of Electrical Power & Energy Systems **148**, 108988 (2023). doi:10.1016/j.ijepes.2023.108988

[37] Varga, A., Hornig, R.: An overview of the OMNeT++ simulation environment. In: Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops. Simutools '08. ICST, Brussels, BEL (2008). ISBN: 9789639799202

[38] Ofenloch, A., Schwarz, J.S., Tolk, D., Brandt, T., Eilers, R., Ramirez, R., Raub, T., Lehnhoff, S.: Mosaik 3.0: Combining time-stepped and discrete event simulation. In: 2022 Open Source Modelling and Simulation of Energy Systems (OSMSES), pp. 1–5 (2022). doi:10.1109/OSMSES54027.2022.9769116

[39] Schrage, R., Sager, J., Hörding, J.P., Holly, S.: mango: A modular python-based agent simulation framework. SoftwareX **27**, 101791 (2024). doi:10.1016/j.softx.2024.101791

[40] DIN: DIN IEC 60050-351: International Electrotechnical Vocabulary – Part 351: Control technology. Deutsches Institut für Normung, Berlin. S. 21 (2014)

[41] VDI: VDI-Richtlinie 3633 Blatt 1: Simulation von Logistik-, Materialfluss- und Produktionssystemen – Grundlagen. Verein Deutscher Ingenieure, Düsseldorf (2014)

[42] Banks, J., Carson, J., Nelson, B., Nicol, D.: Discrete-Event System Simulation, 5th edn. Prentice-Hall, (2010). ISBN: 978-1-292-03726-4

[43] Law, A.M.: Simulation Modeling and Analysis. McGraw-Hill Education, (2015). ISBN: 0073401323

[44] Gutenschwager, K., Rabe, M., Spieckermann, S., Wenzel, S.: Grundlagen der ereignisdiskreten Simulation. In: Simulation in Produktion und Logistik: Grundlagen und Anwendungen, pp. 51–84. Springer, Berlin, Heidelberg (2017). doi:10.1007/978-3-662-55745-7_3

[45] Iversen, V.: Teletraffic Engineering Handbook, Technical University of Denmark (2001). https://www.itu.int/dms_pub/itu-d/opb/stg/D-STG-SG02.16.1-2001-PDF-E.pdf

[46] Fishman, G.: Discrete-event Simulation: Modeling, Programming, and Analysis vol. 5, (2002). doi:10.1007/978-1-4757-3552-9

[47] Vogt, M., Marten, F., Braun, M.: A survey and statistical analysis of smart grid co-simulations. Applied Energy **222**, 67–78 (2018). doi:10.1016/j.apenergy.2018.03.123

[48] Mets, K., Ojea, J.A., Develder, C.: Combining power and communication network simulation for cost-effective smart grid analysis. IEEE Communications Surveys & Tutorials **16**(3), 1771–1796 (2014). doi:10.1109/SURV.2014.021414.00116

[49] Alizadeh, R., Allen, J.K., Mistree, F.: Managing computational complexity using surrogate models: a critical review. Research in Engineering Design **31**(3), 275–298 (2020). doi:10.1007/s00163-020-00336-7

[50] Qian, Z., Seepersad, C.C., Joseph, V.R., Allen, J.K., Jeff Wu, C.F.: Building surrogate models based on detailed and approximate simulations. Journal of Mechanical Design **128**(4), 668–677 (2005). doi:10.1115/1.2179459

[51] Wehenkel, L., Pavella, M.: Decision trees and transient stability of electric power systems. Automatica **27**(1), 115–134 (1991). doi:10.1016/0005-1098(91)90010-Y

[52] Balduin, S., Westermann, T., Puiutta, E.: Evaluating different machine learning techniques as surrogate for low voltage grids. Energy Inform 3 (Suppl 1) (2020). doi:10.1186/s42162-020-00127-3

[53] Zorn, M., Catunda, N., Claus, L., Kobylinska, N., Frey, M., Wortmann, T.: Replacing energy simulations with surrogate models for design space exploration. Bauphysik **44**(6), 311–316 (2022)

[54] Zhou, Z.-H.: Machine Learning. Springer eBook Collection. Springer, (2021). ISBN: 978-981-15-1967-3

[55] Bartz-Beielstein, T., Bartz, E. (eds.): Online Machine Learning: Eine Praxisorientierte Einführung. Springer, Wiesbaden (2024). doi:10.1007/978-3-658-42505-0

[56] Sagi, O., Rokach, L.: Ensemble learning: A survey. WIREs Data Mining and Knowledge Discovery **8**(4), 1249 (2018). doi:10.1002/widm.1249

[57] Pintelas, P.E., Livieris, I.E. (eds.): Ensemble Algorithms and Their Applications. MDPI - Multidisciplinary Digital Publishing Institute, Basel, Switzerland (2020). ISBN: 978-3-03936-959-1

[58] Che, D., Liu, Q., Rasheed, K., Tao, X.: Decision tree and ensemble learning algorithms with their applications in bioinformatics. In: Arabnia, H.R., Tran, Q.-N. (eds.) Software Tools and Algorithms for Biological Systems, pp. 191–199. Springer, New York, NY (2011). doi:10.1007/978-1-4419-7046-6_19

[59] Suhaimy, N., Radzi, N.A.M., Ahmad, W.S.H.M.W., Azmi, K.H.M., Hannan, M.A.: Current and future communication solutions for smart grids: A review. IEEE Access **10**, 43639–43668 (2022). doi:10.1109/ACCESS.2022.3168740

[60] Kabalci, E., Kabalci, Y. (eds.): Smart Grids and Their Communication Systems. Energy Systems in Electrical Engineering. Springer, Singapore (2019). doi:10.1007/978-981-13-1768-2. ISBN: 978-981-13-1768-2

[61] Sharma, D.K., Rapaka, G.K., Pasupulla, A.P., Jaiswal, S., Abadar, K., Kaur, H.: A review on smart grid telecommunication system. Materials Today: Proceedings **51**, 470–474 (2022). doi:10.1016/j.matpr.2021.05.581

[62] Cali, U., Kuzlu, M., Pipattanasomporn, M., Kempf, J., Bai, L.: Smart Grid Applications and Communication Technologies, pp. 17–38. Springer, Cham (2021). doi:10.1007/978-3-030-83301-5_2. ISBN: 978-3-030-83301-5

[63] Abrahamsen, F.E., Ai, Y., Cheffena, M.: Communication technologies for smart grid: A comprehensive survey. Sensors **21**(23) (2021). doi:10.3390/s21238087

[64] IEEE: IEEE Standard for Low-Rate Wireless Networks. IEEE Standards Association. https://standards.ieee.org/ieee/802.15.4/7029/

[65] IEEE: IEEE Standard for Information Technology-Telecommunications and Information Exchange Between Systems-Local and Metropolitan Area Networks-Specific Requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Institute of Electrical and Electronics Engineers. https://standards.ieee.org/ieee/802.11/7028/

[66] 3rd Generation Partnership Project (3GPP): Evolved universal terrestrial radio access (e-utra) and evolved universal terrestrial radio access network (e-utran); overall description. Technical Report 3GPP TS 36.300 (2024). Release 17. https://www.3gpp.org/DynaReport/36300.htm

[67] Caldeira, R.T., Melgarejo, D.C., Coutinho, R.: Application of LTE 450 MHz in the electric energy sector. In: 2017 European Conference on Networks and Communications (EuCNC), pp. 1–5 (2017). doi:10.1109/EuCNC.2017.7980770

[68] IEEE: IEEE Recommended Practice for Supervisory Control and Data Acquisition (SCADA) and Automation Systems. doi:10.1109/IEEESTD.2014.6840. Institute of Electrical and Electronics Engineers. https://standards.ieee.org/ieee/C93.5/6840/

[69] IEEE: IEEE Standard for Ethernet. Institute of Electrical and Electronics Engineers. https://standards.ieee.org/ieee/802.3/10422/

[70] Tanenbaum, A.S., Wetherall, D.J., Feamster, N.: Computer Networks, 6th edn. Pearson Education, Boston (2021). ISBN: 978-1-292-37401-7

[71] Kurose, J.F., Ross, K.W.: Computer Networking: A Top-Down Approach, 8th edn. Pearson, Boston (2021). ISBN: 978-1-292-40551-3

[72] Happ, M., Herlich, M., Maier, C., Du, J.L., Dorfinger, P.: Graph-neural-network-based delay estimation for communication networks with heterogeneous scheduling policies. ITU Journal on Future and Evolving Technologies **2**(4), 1–8 (2021). doi:10.52953/TEJX5530

[73] Wu, L., Cui, P., Pei, J., Zhao, L. (eds.): Graph Neural Networks: Foundations, Frontiers, and Applications. Springer, Singapore (2022). doi:10.1007/978-981-16-6054-2

[74] Kang, B.G., Seo, K.-M., Kim, T.G.: Machine learning-based discrete event dynamic surrogate model of communication systems for simulating the command, control, and communication system of systems. SIMULATION **95**(8), 673–691 (2019). doi:10.1177/0037549718809890

[75] Stadler, C., Flamm, X., Gruber, T., Djanatliev, A., German, R., Eckhoff, D.: A stochastic V2V LOS/NLOS model using neural networks for hardware-in-the-loop testing. In: 2017 IEEE Vehicular Networking Conference (VNC), pp. 195–202 (2017). doi:10.1109/VNC.2017.8275597

[76] Sujil, A., Verma, J., Kumar, R.: Multi agent system: concepts, platforms and applications in power systems. Artificial Intelligence Review **49**(2), 153–182 (2018). doi:10.1007/s10462-016-9520-8

[77] Ma, Z., Schultz, M.J., Christensen, K., Værbak, M., Demazeau, Y., Jørgensen, B.N.: The application of ontologies in multi-agent systems in the energy sector: A scoping review. Energies **12**(16) (2019). doi:10.3390/en12163200

[78] Yao, R., Hu, Y., Varga, L.: Applications of agent-based methods in multi-energy systems-a systematic literature review. Energies **16**(5), 2456 (2023). doi:10.3390/en16052456

[79] Wehinger, L.A., Hug-Glanzmann, G., Galus, M.D., Andersson, G.: Modeling electricity wholesale markets with model predictive and profit maximizing agents. IEEE Transactions on Power Systems **28**(2), 868–876 (2013). doi:10.1109/TPWRS.2012.2213277

[80] Cintuglu, M.H., Martin, H., Mohammed, O.A.: Real-time implementation of multiagent-based game theory reverse auction model for microgrid market operation. IEEE Transactions on Smart Grid **6**(2), 1064–1072 (2015). doi:10.1109/TSG.2014.2387215

[81] Nguyen, D.H.: Optimal solution analysis and decentralized mechanisms for peer-to-peer energy markets. IEEE Transactions on Power Systems **36**(2), 1470–1481 (2021). doi:10.1109/TPWRS.2020.3021474

[82] Ren, F., Zhang, M., Sutanto, D.: A multi-agent solution to distribution system management by considering distributed generators. IEEE Transactions on Power Systems **28**(2), 1442–1451 (2013). doi:10.1109/TPWRS.2012.2223490

[83] Hinrichs, C., Sonnenschein, M.: A distributed combinatorial optimisation heuristic for the scheduling of energy resources represented by self-interested agents. International Journal of Bio-Inspired Computation **10**(2), 69 (2017). doi:10.1504/IJBIC.2017.085895

[84] Latifi, M., Khalili, A., Rastegarnia, A., Bazzi, W.M., Sanei, S.: Demand-side management for smart grid via diffusion adaptation. IET Smart Grid **3**(1), 69–82 (2020). doi:10.1049/iet-stg.2018.0271

[85] Nimalsiri, N.I., Mediwaththe, C.P., Ratnam, E.L., Shaw, M., Smith, D.B., Halgamuge, S.K.: A survey of algorithms for distributed charging control of electric vehicles in smart grid. IEEE Transactions on Intelligent Transportation Systems **21**(11), 4497–4515 (2020). doi:10.1109/TITS.2019.2943620

[86] Nizami, M.S.H., Hossain, M.J., Mahmud, K.: A coordinated electric vehicle management system for grid-support services in residential networks. IEEE Systems Journal **15**(2), 2066–2077 (2021). doi:10.1109/JSYST.2020.3006848

[87] Garlapati, S., Lin, H., Veda, S., Shukla, S., Thorp, J.s.: Agent based supervision of zone 3 relays to prevent hidden failure based tripping, pp. 256–261 (2010). doi:10.1109/SMARTGRID.2010.5622051

[88] Shadi, M.R., Ameli, M.-T., Azad, S.: A real-time hierarchical framework for fault detection, classification, and location in power systems using pmus data and deep learning. International Journal of Electrical Power & Energy Systems **134**, 107399 (2022). doi:10.1016/j.ijepes.2021.107399

[89] Nareshkumar, K., Choudhry, M.A., Lai, J., Feliachi, A.: Application of multi-agents for fault detection and reconfiguration of power distribution systems. In: 2009 IEEE Power & Energy Society General Meeting, pp. 1–8 (2009). doi:10.1109/PES.2009.5276005

[90] Dimeas, A., Hatziargyriou, N.: Operation of a multiagent system for micro-grid control. Power Systems, IEEE Transactions on **20**, 1447–1455 (2005). doi:10.1109/TPWRS.2005.852060

[91] Bülo, T., Geibel, D., Sutter, S., Boldt, K.: Aktives, intelligentes niederspannungsnetz. Technical report (2013). `https://www.uni-kassel.de/eecs/index.php?eID=dumpFile&t=f&f=153&token=a6c775110f52a5ae241703896b3bf9bba1994ef5`

[92] Vaccaro, A., Velotto, G., Zobaa, A.F.: A decentralized and cooperative architecture for optimal voltage regulation in smart grids. IEEE Transactions on Industrial Electronics **58**(10), 4593–4602 (2011). doi:10.1109/TIE.2011.2143374

[93] Holly, S., Nieße, A.: Dynamic communication topologies for distributed heuristics in energy system optimization algorithms. In: 2021 16th Conference on Computer Science and Intelligence Systems (FedCSIS), pp. 191–200 (2021). doi:10.15439/2021F60

[94] Meinecke, S., Sarajlić, D., Drauz, S.R., Klettke, A., Lauven, L.-P., Rehtanz, C., Moser, A., Braun, M.: Simbench-a benchmark dataset of electric power systems to compare innovative solutions based on power flow analysis. Energies **13**(12), 3290 (2020). doi:10.3390/en13123290

[95] Nardini, G., Sabella, D., Stea, G., Thakkar, P., Virdis, A.: Simu5g–an OMNeT++ library for end-to-end performance evaluation of 5G networks. IEEE Access **8**, 181176–181191 (2020). doi:10.1109/ACCESS.2020.3028550

[96] Ye, F., Bose, A.: Multiple communication topologies for pmu-based applications: Introduction, analysis and simulation. IEEE Transactions on Smart Grid **11**(6), 5051–5061 (2020). doi:10.1109/TSG.2020.2999066

[97] Rajdl, K., Lansky, P., Kostal, L.: Fano factor: A potentially useful information. Frontiers in Computational Neuroscience **14** (2020). doi:10.3389/fncom.2020.569049

[98] Bacher, J., Pöge, A., Wenzig, K.: Clusteranalyse: Anwendungsorientierte Einführung in Klassifikationsverfahren. Oldenbourg Wissenschaftsverlag, (2011). doi:10.1524/9783486710236

[99] Von Der Hude, M.: Predictive Analytics und Data Mining: Eine Einführung Mit R. Springer, Wiesbaden (2020). doi:10.1007/978-3-658-30153-8

[100] Mahajan, P.S., Ingalls, R.: Evaluation of Methods Used to Detect Warm-Up Period in Steady State Simulation, (2005). doi:10.1109/WSC.2004.1371374

[101] Hansun, S.: A new approach of moving average method in time series analysis, pp. 1–4 (2013). doi:10.1109/CoNMedia.2013.6708545

[102] Steinbrink, C., Lehnhoff, S.: Quantifying probabilistic uncertainty in smart grid co-simulation. In: 2016 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES), pp. 1–6 (2016). doi:10.1109/MSCPES.2016.7480222

[103] Bayram, F., Ahmed, B.S., Kassler, A.: From concept drift to model degradation: An overview on performance-aware drift detectors. Knowledge-Based Systems **245**, 108632 (2022). doi:10.1016/j.knosys.2022.108632

[104] Razak, A., Nirmala, C.R., Aljohani, M., Sreenivasa, B.R.: A novel technique for detecting sudden concept drift in healthcare data using multi-linear artificial intelligence techniques. Frontiers in Artificial Intelligence **5**, 950659 (2022). doi:10.3389/frai.2022.950659. Accessed 2026-03-19

[105] Mémoli, F.: Gromov–wasserstein distances and the metric approach to object matching. Foundations of Computational Mathematics **11**(4), 417–487 (2011). doi:10.1007/s10208-011-9093-5

[106] Villani, C.: Topics in Optimal Transportation. Graduate Studies in Mathematics, vol. 58. American Mathematical Society, (2003). ISBN: 978-0-8218-3312-4

[107] Peyré, G., Cuturi, M.: Computational Optimal Transport: With Applications to Data Science vol. 11, pp. 355–607. Foundations and Trends in Machine Learning, (2019). doi:10.1561/2200000073

[108] Siebertz, K., Van Bebber, D., Hochkirchen, T.: Statistische Versuchsplanung. Springer, Berlin, Heidelberg (2017). doi:10.1007/978-3-662-55743-3

[109] McCoy, C.E.: Understanding the intention-to-treat principle in randomized controlled trials. Western Journal of Emergency Medicine **18**(6), 1075–1078 (2017). doi:10.5811/westjem.2017.8.35985

[110] Lindman, H.R.: Analysis of Variance in Experimental Design. Springer Texts in Statistics. Springer, New York, NY (1992). doi:10.1007/978-1-4613-9722-9. ISBN: 978-1-4613-9722-9

[111] Jeuring, J.: The derivation of on-line algorithms, with an application to finding palindromes. Algorithmica **11**(2), 146–184 (1994). doi:10.1007/BF01182773

[112] Albers, S., Charikar, M., Mitzenmacher, M.: Delayed Information and Action in On-Line Algorithms. Information and Computation **170**(2), 135–152 (2001). doi:10.1006/inco.2001.3057

[113] Albers, S.: Online algorithms: A survey. Math. Program. **97**, 3–26 (2003). doi:10.1007/s10107-003-0436-0

[114] Niu, H., Sharma, S., Qiu, Y., Li, M., Zhou, G., Hu, J., Zhan, X.: When to Trust Your Simulator: Dynamics-Aware Hybrid Offline-and-Online Reinforcement Learning. Advances in Neural Information Processing Systems **35**, 36599–36612 (2022). Accessed 2026-03-19

[115] Raviv, T., Park, S., Simeone, O., Eldar, Y.C., Shlezinger, N.: Online Meta-Learning for Hybrid Model-Based Deep Receivers. IEEE Transactions on Wireless Communications **22**(10), 6415–6431 (2023). doi:10.1109/TWC.2023.3241841

Erklärung / Declaration

Ich versichere an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die allgemeinen Prinzipien wissenschaftlicher Arbeit und Veröffentlichungen, wie sie in den Leitlinien guter wissenschaftlicher Praxis der Carl von Ossietzky Universität Oldenburg festgelegt sind, befolgt habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

I hereby confirm in lieu of oath, that I have written the accompanying thesis by myself, without contributions from any sources other than those cited in the text and acknowledgments. I certify, that I have followed the general principles of scientific work and publications as written in the guidelines of good research of the Carl von Ossietzky University of Oldenburg. This work has not yet been submitted to any examination office in the same or similar form.

Oldenburg, March 31

Malin Radtke