UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Towards a Robust B5G/6G Transport Network With Self-adaptive Network Digital Twin

Cláudio Matheus Modesto Barata

DM: 17/2025

UFPA / ITEC / PPGEE
Campus Universitário do Guamá
Belém-Pará-Brasil
2025

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Cláudio Matheus Modesto Barata

Towards a Robust B5G/6G Transport Network With Self-adaptive Network Digital Twin

DM: 17/2025

UFPA / ITEC / PPGEE
Campus Universitário do Guamá
Belém-Pará-Brasil
2025

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Cláudio Matheus Modesto Barata

Towards a Robust B5G/6G Transport Network With Self-adaptive Network Digital Twin

A dissertation submitted to the examination committee in the graduate department of Electrical Engineering at the Federal University of Pará in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering with emphasis in Telecommunications.

UFPA / ITEC / PPGEE
Campus Universitário do Guamá
Belém-Pará-Brasil
2025

Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD
Sistema de Bibliotecas da Universidade Federal do Pará
Gerada automaticamente pelo módulo Ficat, mediante os dados fornecidos pelo(a) autor(a)

M691t Modesto Barata, Cláudio Matheus.
Towards a Robust B5G/6G Transport Network With Self-adaptive Network Digital Twin / Cláudio Matheus Modesto Barata, . — 2025.
82 f. : il. color.

Orientador(a): Prof. Dr. Aldebaro Barreto da Rocha Klautau Júnior
Dissertação (Mestrado) - Universidade Federal do Pará, Instituto de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica, Belém, 2025.

1. Concept drift. 2. Graph neural network. 3. Lifecycle management. 4. Twinning. I. Título.

CDD 621.3821

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

"TOWARDS A ROBUST B5G/6G TRANSPORT NETWORK WITH SELF-ADAPTIVE NETWORK DIGITAL TWIN"

AUTOR: **CLÁUDIO MATHEUS MODESTO BARATA**

DISSERTAÇÃO DE MESTRADO SUBMETIDA À BANCA EXAMINADORA APROVADA PELO COLEGIADO DO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA, SENDO JULGADA ADEQUADA PARA A OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA ELÉTRICA NA ÁREA DE TELECOMUNICAÇÕES.

APROVADA EM: 04/07/2025

BANCA EXAMINADORA:

Prof. Dr. Aldebaro Barreto da Rocha Klautau Júnior
(Orientador – PPGEE/ITEC/UFPA)

Prof. Dr. Glauco Estácio Gonçalves
(Avaliador Interno - PPGEE/ITEC/UFPA)

Prof. Dr. Bruno Souza Lyra Castro
(Avaliador Externo ao Programa – CAMPUS CASTANHAL/UFPA)

Prof. Dr. Cristiano Bonato Both
(Avaliador Externo – UNISINOS)

VISTO:

Prof. Dr. Diego Lisboa Cardoso

(Coordenador do PPGEE/ITEC/UFPA)

Acknowledgments

Firstly, I would like to express my heartfelt gratitude to my family, especially to my grandparents, Raimundo Campos and Raimunda Modesto, for their kindness and unconditional love throughout my entire journey of this master's degree.

Secondly, I would like to sincerely thank my advisor Prof. Aldebaro Klautau for the support since my undergraduate studies and now my master's degree. I am really grateful for his guidance, stories, and advice during these years. The synergy of our collaboration not only provided me with the confidence and peace of mind to pursue this master's degree but also made the journey enjoyable and rewarding.

Furthermore, I would like to extend my thanks to the excellent team of professors and researchers with which I have the privilege to collaborate in the NeTwins Project. Especially, thanks are given to Rebecca Aben-Athar, João Borges, and Cleverson Nahum, for all contributions to the development of the digital twin related topic, which were essential to this work and the success of the NeTwins. Moreover, I am grateful to Kelly Pereira and the LASSE team for all the pleasant talks that are always an enjoyable part of my day and for being a place that I consider home.

I also would like to express my sincere gratitude to the committee members, Prof. Cristiano Both, Prof. Glauco Gonçalves, and Prof. Bruno Castro, for their invaluable insights, thoughtful comments, and expert guidance, all of which greatly contributed to the final version of this work.

Finally, I would like to thank the sponsors of this work, the Innovation Center and Ericsson Telecomunicações.

Cláudio Modesto

July 2025

It is by logic that we prove,
but by intuition that we discover.

Henri Poincaré

Abstract

The ability of the Network Digital Twin (NDT) to remain aware of changes in its physical counterpart, known as the Physical Twin (PTwin), is a fundamental condition to enable timely adaptation and synchronization, also referred to as *twinning*. In this way, considering a transport network, a key requirement is to handle unexpected traffic variability and dynamically adapt to maintain optimal performance in the associated virtual model, known as the Virtual Twin (VTwin). In this context, we propose a robust and self-adaptive implementation of a novel NDT architecture designed to provide accurate delay prediction for network flows, even under fluctuating traffic conditions. This architecture addresses an essential challenge, underexplored in the literature: improving the resilience of data-driven NDT platforms against traffic variability and improving synchronization between the VTwin, based on neural networks, and its physical counterpart. Therefore, the contributions of this dissertation rely on a relatively underexplored stage of the NDT lifecycle by focusing on the operational phase, where telemetry modules are used to monitor incoming traffic and concept drift detection techniques guide retraining decisions aimed at updating and redeploying the VTwin when necessary. We validate our architecture across various emulated network topologies and diverse traffic patterns to demonstrate its effectiveness in preserving acceptable performance and predicting total per-flow delay under unexpected traffic variation, useful for maintaining reliable performance in applications such as service-level agreement monitoring considered as a use case in this work. The results in all tested topologies, using the normalized mean square error as the evaluation metric, demonstrate that our proposed architecture, after a traffic concept drift, achieves a performance improvement in prediction of at least 56.7% compared to a configuration without digital twin synchronization.

Keywords — Concept drift, Graph Neural Network (GNN), Lifecycle Management (LCM), *twinning*.

Resumo

A capacidade de um gêmeo digital para redes (NDT) de se manter ciente das mudanças em seu equivalente físico, conhecido como gêmeo físico (PTwin), é uma condição fundamental para viabilizar a adaptação e a sincronização em tempo hábil, também chamadas de *twinning*. Dessa forma, considerando uma rede de transporte, um requisito chave é lidar com a variabilidade inesperada do tráfego e se adaptar dinamicamente para manter um desempenho ideal no modelo virtual associado, conhecido como gêmeo virtual (VTwin). Nesse contexto, esta dissertação propõe uma implementação robusta e auto-adaptativa de uma nova arquitetura de gêmeos digitais para redes, projetada para fornecer predição precisa de atraso dos fluxos de rede, mesmo sob condições de tráfego flutuantes. Essa arquitetura aborda um desafio essencial, ainda pouco explorado na literatura: melhorar a resiliência de plataformas de gêmeos digitais baseadas em dados frente à variabilidade do tráfego, além de aprimorar a sincronização entre o VTwin, baseado em redes neurais, e seu equivalente físico. Portanto, as contribuições desta dissertação se concentram em uma etapa relativamente pouco explorada do ciclo de vida do NDT, ao focar na fase operacional, na qual módulos de telemetria são utilizados para monitorar o tráfego de entrada, e técnicas de detecção de desvio de conceito que orientam as decisões de re-treinamento, com o objetivo de atualizar e reimplantar o gêmeo virtual quando necessário. A arquitetura proposta foi avaliada em diversas topologias de rede emuladas e sob diferentes padrões de tráfego, a fim de demonstrar sua eficácia na preservação de um desempenho aceitável e na predição do atraso total por fluxo, mesmo sob variações inesperadas de tráfego, aspecto essencial para manter um desempenho confiável em aplicações como o monitoramento de *service-level agreement* considerado como caso de uso neste trabalho. Os resultados obtidos em todas as topologias testadas, utilizando o erro quadrático médio normalizado como métrica de avaliação, demonstram que com a arquitetura proposta, após a ocorrência de um desvio de conceito no tráfego, é alcançada uma melhoria de desempenho na predição de pelo menos $56,7\%$ em comparação com uma configuração sem sincronização do gêmeo digital.

Palavras-chave — Desvio de conceito, gerenciamento de ciclo de vida, redes grafos neurais, *twinning*.

List of Figures

List of Tables

List of Acronyms

3GPP 3rd Generation Partnership Project

LLDP Link Layer Discovery Protocol

AI Artificial Intelligence

5G 5th Generation

B5G/6G Beyond 5th and 6th Generation

QoS Quality of Service

SDN Software-Defined Networking

PDB Packet Delay Budget

ETSI European Telecommunications Standards Institute

SLA Service-Level Agreement

API Application Programming Interface

VTwin Virtual Twin

GRU Gated Recurrent Unit

NMSE Normalized Mean Square Error

PTwin Physical Twin

MPNN Message Passing Neural Network

IETF Internet Engineering Task Force

DT Digital Twin

NDT Network Digital Twin

KSWIN Kolmogorov-Smirnov Windowing

GNN Graph Neural Network

MLP Multilayer Perceptron

IP Internet Protocol

ITU-T International Telecommunication Union Standardization Sector

UDP User Datagram Protocol

D-ITG Distributed Internet Traffic Generator

ML Machine Learning

GCN Graph Convolutional Network

GAT Graph Attention Network

NOS Network Operating System

MAPE Mean Absolute Percentage Error

CDF Cumulative Distribution Function

LSTM Long Short-Term Memory

DGAT Dynamic Graph Attention Network

ONOS Open Network Operating System

LCM Lifecycle Management

TCP Transmission Control Protocol

REST Representational State Transfer

DDM Drift Detection Method

EDDM Early Drift Detection Method

ADWIN Adaptive Windowing

nGRG next Generation Research Group

List of Symbols

$\bigoplus$	Aggregation function
$\hat{\mathcal{D}}$	Predicted total per-flow delay
$\mathbf{A}$	Adjancency matrix
$\mathbf{D}$	Degree matrix
$\mathbf{g}_{e_j}$	Link embedding vector
$\mathbf{g}_{f_t}$	Flow embedding vector
$\mathbf{H}$	Matrix output layer
$\mathbf{h}$	Vector output layer
$\mathbf{i}$	Vector of flow indices
$\mathbf{j}$	Vector of link indices
$\mathbf{X}$	Input feature matrix
$\mathbf{x}$	Input feature vector
$\mathcal{D}$	ground truth total per-flow delay
$\mathcal{E}$	Set of links
$\mathcal{E}'$	Set of links in a flow path
$\mathcal{F}$	Set of flows
$\mathcal{G}$	Transport network
$\mathcal{H}$	2-tuple graph

$\mathcal{N}_u$	Neighborhood of the node u
$\mathcal{O}$	Occupancy rate
$\mathcal{V}$	Set of switches
$\overline{\mathcal{D}}$	Average ground truth total per-flow delay
ϕ	Update function
ψ	Message function
θ	MLP learnable parameters
a	Attention score
b	Flow embedding size
c_{uv}	Importance of node u to node v's representation
D	Destination switch
E	Set of edges in the graph $\mathcal{H}$
e_j	j-th network link
f_t	t-th flow traffic
h	Mapping rule
K	Number of discrete time steps in the message phase
m	Number of edge features
n	Number of node features
O	Source switch
P_t	Probability distribution at time t
q_j	j-th edge in the set E
r	Link embedding size
V	Set of nodes in the graph $\mathcal{H}$

v_i	i-th node in the set V
x	Input feature
y	Target label

Contents

Chapter 1

Introduction

The concept of a Digital Twin (DT) was first introduced by Michael Grieves and John Vickers (GRIEVES; VICKERS, 2017), referring to an auxiliary platform that generates a virtual replica of a physical system. This replica captures the essential characteristics of environments such as manufacturing plants, communication networks, buildings, or supply chains, typically using simulation or emulation tools (BARRICELLI; CASIRAGHI; FOGLI, 2019). Such a digital counterpart allows decision-makers to test and evaluate alternative configurations and organizational strategies without interfering with real-world operations. When applied specifically to network communication systems, this concept gives rise to a specialized case known as a Network Digital Twin (NDT).

NDT is a specialized form of Digital Twin (DT) designed to serve as a platform for managing and automating physical network communication tasks. It operates through interactions with its virtual counterpart, the Virtual Twin (VTwin), following a defined periodicity and fidelity (BARRICELLI; CASIRAGHI; FOGLI, 2019; ÖHLÉN et al., 2022). This relation allows the network operator to emulate different scenarios in network planning to perform a *what-if* analysis (SUÁREZ-VARELA et al., 2023; ABEN-ATHAR et al., 2025; ZALAT et al., 2024). This analysis aims to evaluate the impact of management changes in the network infrastructure when these tests are impossible due to logistical issues or costs that make this unfeasible, such as in production environments (ALMASAN et al., 2022a; ÖHLÉN et al., 2022). For that reason, this platform is considered an enabling technology to optimize Beyond 5th and 6th Generation (B5G/6G) network communication (ÖHLÉN et al., 2022), to provide improved solutions to problems such as slicing management (WANG et al., 2022; FARRERAS et al., 2025), routing optimization (RUSEK et al., 2020; ALMASAN et al., 2022b), resource power alloca-

tion, and flow admission control (MESSAOUDI et al., 2023). These appointments align with their need to establish reliable environments for developing and testing its critical parts, such as wired environments (MESSAOUDI et al., 2023), where DTs can provide excellent support. However, during this wave of adoption, the intersection of DT and networking—still an emerging area of study—led to a diverse range of approaches. Many of these did not fully align with the definitions of an NDT as systematized by standards organizations (Recommendation ITU-T Y.3090, 2022; ZHOU et al., 2025) and industry stakeholders (ÖHLÉN et al., 2022; O-RAN ALLIANCE, 2024). Consequently, some of the related works found in the literature presented limited analyses of practical aspects. For example, regarding the system lifecycle or the twinning rates, the latter of which refers to the rate of synchronization between the PTwin current state and its corresponding representation in the VTwin (JONES et al., 2020).

To address these challenges, we propose an enhanced architecture grounded in the core lifecycle principles of the NDT, with particular emphasis on maintenance and resilience. The architecture is designed to support a robust and adaptive platform capable of handling input traffic variability, commonly referred to as *covariate drift*, a type of concept drift (BAIER et al., 2023; AMEUR; BRIKAUTHORREFMARK; KSENTINI, 2025). The concept drift is a realistic and ubiquitous phenomenon that should be addressed when employing AI-based VTwin models and reflects the non-stationary behavior of the physical environment. In communication systems, such drift can result from disruptions including congestion, unreachable switches, link failures, or shifts in traffic patterns over time. Therefore, the proposed approach emphasizes synchronization between the VTwin and the Physical Twin (PTwin), with a focus on maintaining this synchronization. So, we adopted a data-driven synchronization pattern (ALGHAMDI; ALBASSAM, 2024) managed through a concept drift detector, which monitors network traffic for changes. When significant deviations are detected, a retraining process is triggered to realign the VTwin with the current state of the PTwin. Since the simulations employ a supervised learning setup, where the "ground truth" total delay is known, it is evaluated and validated the generalization performance of the VTwin in the proposed architecture as the PTwin experiences dynamic changes in traffic intensity and pattern. The results from these experiments demonstrate a performance improvement across all tested emulated topologies, with different network traffic patterns, and using the Normalized Mean Square Error (NMSE) as the evaluation metric, exhibiting that the proposed architecture achieves an improvement in the delay prediction of at least 56.7% after a traffic concept drift, and in some other topologies, this improvement is

greater than 99%. Finally, with these improvements, it is also proposed a last evaluation in a use case application based on Service-Level Agreement (SLA) monitoring, to demonstrate the importance of the NDT synchronization considering high-level metrics, such as the accuracy in the SLA violations.

1.1 Related Work

The literature regarding NDTs of the transport network domain focused on using Machine Learning (ML)/surrogate model-based methods to generate the VTwin. The increase in popularity of this DT approach, which can also be called data-driven (BARIAH; DEBBAH, 2024), can be traced back to the work of Rusek et al. (RUSEK et al., 2020), which established the usage of a GNN model called RouteNet for predicting path-level Quality of Service (QoS) metrics in routing optimization and network upgrade use cases. The usage of this model was further investigated by several works, which were built on the previously established RouteNet (HAPP et al., 2021; SARAVANAN; KUMAR; KUMAR, 2022; FARRERAS et al., 2023; LI et al., 2023; FERRIOL-GALMÉS et al., 2023; MESSAOUDI et al., 2023; MODESTO et al., 2024).

For example, more recently, the work of Ferriol-Galmés et al. (FERRIOL-GALMÉS et al., 2023) used a modified version of RouteNet, called RouteNet-Fermi, to model network communication with different topologies, traffic distributions, and queuing configurations, allowing the prediction of packet loss, delay, and jitter. In this work, since all data is provided from a packet-level simulator, there are no two-way interactions between PTwin and VTwin. Therefore, the case presented in the paper does not constitute a closed-loop NDT. Also, the investigations do not address the possible performance degradation of the VTwin in cases of concept drift (LU et al., 2018). In this context, the cases presented in this work, and several others built upon RouteNet (HAPP et al., 2021; SARAVANAN; KUMAR; KUMAR, 2022; FARRERAS et al., 2023; LI et al., 2023; FERRIOL-GALMÉS et al., 2023), did not address the robustness aspect of an NDT, as they did not explore its lifecycle either.

The exception to this lack of closed-loop implementation in works based on RouteNet is related to Messaoudi et al. (MESSAOUDI et al., 2023), Zalat et al. (ZALAT et al., 2024), and Aben-Athar et al. (ABEN-ATHAR et al., 2025), which are, to the best of our knowledge, the first closed-loop environments considering physical layers interacting with an Artificial Intelligence (AI) model. The works in (MESSAOUDI et al., 2023; ABEN-ATHAR et al., 2025)

feed an application module, tailored to operate over the physical counterpart, defined by virtual switches on Mininet (Mininet Project Contributors, 2024). The work in (ZALAT et al., 2024) took the same approach, but considered a small-scale physical testbed as PTwin. However, a noteworthy aspect of these works is that there are in (ZALAT et al., 2024) elements for twin synchronization, which are based on a trigger-based online learning approach. For the other works, despite closed-loop interactions between the physical and virtual domains, there is no DT Lifecycle Management (LCM) controlling the rate at which both parts synchronize and update their respective states, for instance, through a retraining process. In this context, the authors ran the traffic only once, and after completing it, they performed no further operations to detect whether the VTwin performance would remain acceptable.

In addition to these, other works in the literature either proposed purely GNN-based approaches, such as FlowDT (FERRIOL-GALMÉS et al., 2022) or GNNetSlice (FARRERAS et al., 2025), or adopted different ML models as their VTwins. For example, Lai et al. (LAI et al., 2023) adopted a model named eConvLSTM, Shin et al. (SHIN et al., 2023) used GNN along with Long Short-Term Memory (LSTM) and a data extraction approach, (YU et al., 2023) which uses Dynamic Graph Attention Network (DGAT). However, from these other works, only Shin et al. (SHIN et al., 2023) and Yu et al. (YU et al., 2023) presented investigations on the NDT LCM, while none presented concept drift implementations.

Finally, in terms of the datasets used in these works, a noticeable concentration of studies (WANG et al., 2022; SARAVANAN; KUMAR; KUMAR, 2022; FARRERAS et al., 2025; SHIN et al., 2023; FARRERAS et al., 2023) is based on specific datasets derived from the OMNeT++ simulator (VARGA, 2001). This trend presents advantages and limitations. On the one hand, using datasets from a common source provides a stable baseline, enabling fair comparisons across different approaches in terms of VTwin generalization performance. On the other hand, relying solely on pre-built datasets limits the exploration of more realistic aspects of NDT, such as the mentioned closed-loop operations or NDT synchronization, as there is no interaction with a real or emulated environment.

Given the current state of the literature on twin synchronization in the transport network domain, we can summarize the works mentioned in this section by Table 1.1, considering four primary characteristics: closed-loop operation, that is, an interaction between VTwin and PTwin. Support for the concept drift detector for traffic variability. Any form of twin synchronization that does not necessarily adopt a method using concept drift. Finally, whether to

consider or discuss NDT LCM elements, such as the planning, training, or operational phase.

Table 1.1 – Comparison between NDT characteristics of the related works and the proposed architecture with support to concept drift

Authors	**Closed-loop**	**Concept drift detection support**	**Twin synchronization**	**NDT LCM elements**
(RUSEK et al., 2020)	✗	✗	✗	✗
(HAPP et al., 2021)	✗	✗	✗	✗
(WANG et al., 2022)	✗	✗	✗	✗
(GÜEMES-PALAU et al., 2022)	✗	✗	✗	✗
(FERRIOL-GALMÉS et al., 2022)	✗	✗	✗	✗
(SARAVANAN; KUMAR; KUMAR, 2022)	✗	✗	✗	✗
(LAI et al., 2023)	✗	✗	✗	✗
(FARRERAS et al., 2023)	✗	✗	✗	✗
(MESSAOUDI et al., 2023)	✓	✗	✗	✗
(SHIN et al., 2023)	✗	✗	✗	✓
(YU et al., 2023)	✗	✗	✗	✓
(FERRIOL-GALMÉS et al., 2023)	✗	✗	✗	✗
(MODESTO et al., 2024)	✗	✗	✗	✗
(ZALAT et al., 2024)	✓	✗	✓	✗
(FARRERAS et al., 2025)	✗	✗	✗	✗
(ABEN-ATHAR et al., 2025)	✓	✗	✗	✗
This dissertation	✓	✓	✓	✓

1.2 Contributions

The contributions of this dissertation are as follows:

- Discussion about data-driven NDT synchronization, presenting the main challenges related to this problem.

- The first data-driven NDT architecture that incorporates concept drift mechanisms to monitoring the network traffic states of the physical counterpart, intended to updating the states of the virtual counterpart accordingly.

- Relevant NDT experiments that account for environments with varying traffic conditions, including changes in traffic patterns and levels of congestion.

- The NDT implementation (Mininet + *Tensorflow* + ONOS + *River*) and all evaluation scripts are available on GitHub, fostering reproducibility and accelerating follow-up research.

1.3 Outline

This dissertation is organized as follows:

- Chapter 2 presents the theoretical foundations to understand the main elements of this dissertation, such as the main theoretical concepts regarding the Software-Defined Networking (SDN) architecture, the definition of an NDT platform considering the standardization bodies, the main concepts about GNN models, and the theoretical definitions regarding concept drift and their sources.
- Chapter 3 presents the proposed enhanced NDT architecture, taking into account the main functional requirements of NDT and its lifecycle, presenting the essential building blocks of each, such as the VTwin architecture, the synchronization methods based on concept drift detection techniques, and complementary modules.
- Chapter 4 details the experiments conducted to evaluate the proposed enhanced architecture with twin synchronization across different topologies and considering an use case related to SLA monitoring. The results are compared against an environment without the proposed architecture, highlighting key findings across various network topologies and traffic patterns.
- Lastly, Chapter 5 summarizes the main contributions of this dissertation and outlines directions for future research.

1.4 Outcomes

The following list summarizes the papers and awards produced directly as part of this dissertation's efforts and complementary to it.

Journal Papers:

1. **Cláudio Modesto**, Rebecca Aben-Athar, Andrey Silva, Silvia Lins, Glauco Gonçalves and Aldebaro Klautau. *Delay Estimation Based on Multiple Stage Message Passing with Attention Mechanism using a Real Network Communication Dataset*. ITU Journal on Future and Evolving Technology, vol. 5 (4), pp. 465-477, Dec/2024, doi:10.52953/RBNE4256.
2. **Cláudio Modesto**, Lucas Mozart, Pedro Batista, André Cavalcante, and Aldebaro Klautau. *Accelerating Ray Tracing-Based Wireless Channels Generation for Real-Time Network Digital Twins*. IEEE Open Journal of the Communications Society, Jun/2025, doi: 10.1109/OJCOMS.2025.3583202.
3. **Cláudio Modesto**, João Borges, Cleverson Nahum, Lucas Matni, Cristiano Bonato Both, Kleber Cardoso, Glauco Gonçalves, Ilan Correa, Silvia Lins, Andrey Silva, and Aldebaro Klautau. *Towards a Robust Transport Network With Self-adaptive Network Digital Twin*. Submitted on Computer Networks.

Awards:

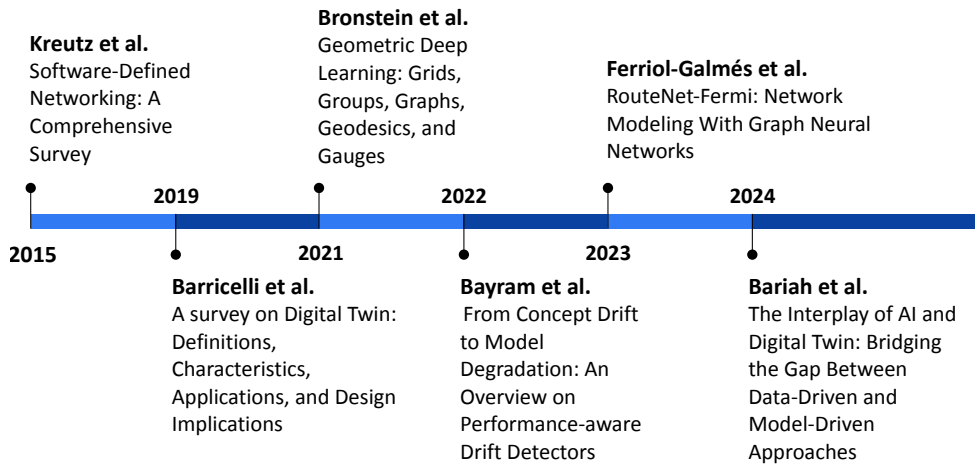
1. **Cláudio Modesto**, Rebecca Aben-Athar, Andrey Silva, and Silvia Lins. 1st place on ITU AI/ML in 5G - Graph Neural Networking Challenge 2023 - *Creating a Network Digital Twin with Real Network Data*, organized by Barcelona Neural Networking Center (BNNC) from Universitat Politècnica de Catalunya (UPC).

Chapter 2

Theoretical Foundations

This chapter presents a concise overview of the fundamental concepts necessary to understand the proposed architecture in this dissertation. It covers topics such as the SDN architectural layers, the definitions related to NDTs, the concept drift detection in AI models, and the mathematical foundations of the main GNN models. Although these concepts are discussed in various works, this chapter is primarily based on six key references, as illustrated in Figure 2.1.

Figure 2.1 – Main works used to construct the theoretical foundations of this dissertation



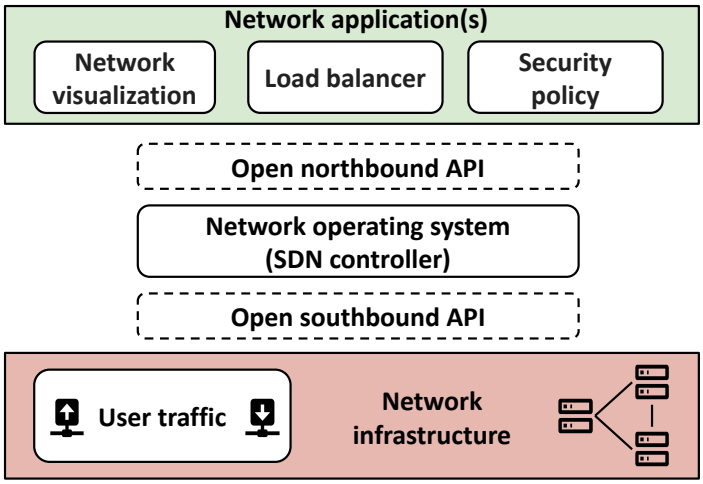
Source: The Author (2025).

The timeline presented in Figure 2.1 also highlights the interdisciplinary nature of this dissertation, which integrates theoretical and practical concepts from SDN architecture emulation, mathematical and statistical foundations from GNN model use to predict QoS metrics, and concept drift models used to detect changes in time series. These elements are combined to support the development and application of an enabling technology such as the proposed NDT architecture.

2.1 Software-Defined Networking

The SDN architecture is a widely adopted network paradigm that decouples the control plane from the data plane of forwarding devices (e.g., switches) (KREUTZ et al., 2015), which allows flexibility and a more simplified management of the network devices through programmable software applications. This separation centralizes the network's "intelligence" in the control plane, while the data plane—represented by the forwarding devices—focuses solely on transmitting user traffic from source to destination. In addition to this plane-based abstraction, the SDN architecture can also be represented in terms of functional layers. As illustrated in Figure 2.2, it comprises five main layers: network infrastructure, open southbound Application Programming Interface (API), the Network Operating System (NOS) (or SDN controller), open northbound API, and network applications.

Figure 2.2 – SDN architecture considering five main layers. Here it has been omitted layers related to the network hypervisor or programming language



Source: The Author (2025).

Adopting a bottom-up perspective, the network infrastructure layer consists of traditional networking components, such as switches and routers, which correspond to the data plane. In an SDN environment, this layer relies on protocols like OpenFlow (MCKEOWN et al., 2008) to enable communication between the forwarding devices and the SDN controller. This interaction occurs through the southbound API, which allows the controller to dynamically modify flow entries[1] within the switches. These entries are stored in flow tables on the devices and define

[1]It is noteworthy that the term *flow* in this context differs from its usage in later sections. Here, a flow entry refers to a set of rules managed by the SDN controller to control how traffic is handled by the network devices.

the actions and time of existence to be taken for matching packets, such as forwarding them to a specific port or dropping them entirely for ten seconds; after that, the same flow entry can be automatically deleted.

Serving as a bridge between the top layer and the bottom layer of the SDN architecture, the NOS provides the necessary abstraction to enable easier network management. The controller can be made in a centralized or distributed way. In the former case, a single controller manages all the network devices, allowing a strong consistency semantics (KREUTZ et al., 2015). In the latter case, this management is made by different controller nodes, allowing, for instance, the creation of a more resilient scenario against controller failures (BANNOUR; SOUIHI; MELLOUK, 2018). Furthermore, in terms of abstraction, the NOS can provide support for configuration or information of the physical network, similar to a traditional operating system. For the former case, it is crucial that the NOS provides support to the network operator so that they can change the network infrastructure based on their own policies. Examples of this support are the routing scheme used to forward a packet through the network, the Link Layer Discovery Protocol (LLDP) used for link discovery and the topology manager. On the other hand, in terms of information availability, it is also a key characteristic of the NOS to provide the necessary current states of the network, such as low-level traffic information that is passing through each device (for instance, the number of bytes in a port), as well as the network topology information.

At the top of the SDN architecture, various network applications operate using the northbound interface to control and optimize the behavior of the network infrastructure. Examples include routing configuration applications that dynamically adjust the packet forwarding rules in network devices to achieve objectives such as load balancing, network visualization for management tasks, and applications related to the security policies adopted in the infrastructure. Moreover, unlike the southbound interface, where OpenFlow is widely adopted as a standard API for communication with the data plane, the northbound interface does not have a universally standardized protocol. Instead, the choice of API depends on the specific SDN controller used in the control plane. Common implementations include ad hoc APIs or Representational State Transfer (REST)-based APIs, which enable applications to interact with the controller and influence network behavior.

Given this brief theoretical concept regarding the SDN architecture elements, the next subsection moves the discussion to a practical perspective, considering ways to use this paradigm

with the entire protocol stack to emulate scenarios and further experiments.

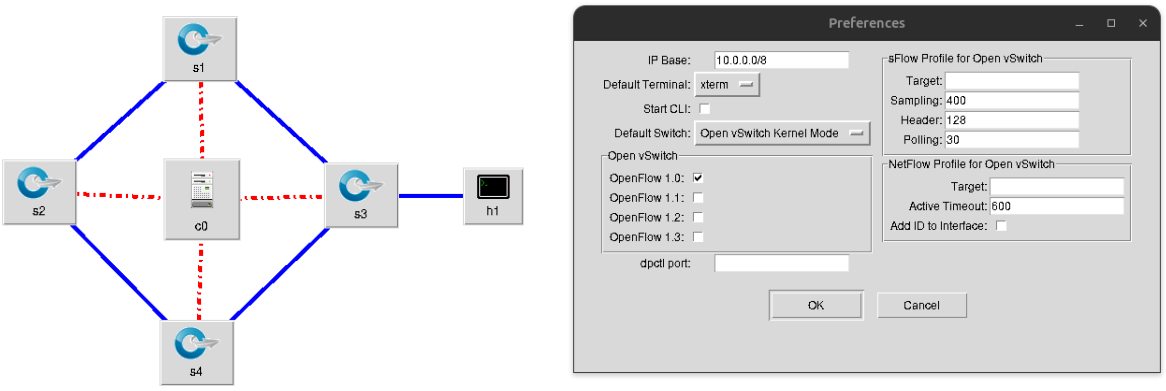
2.1.1 Practical elements and emulation

To use the elements of a SDN architecture, without a real physical environment, for research purposes, it is possible to use different open-source tools for each layer. For instance, for the network infrastructure, tools like Mininet (Mininet Project Contributors, 2024), MaxiNet (WETTE et al., 2024), or ContainerNet (PEUSTER; KARL; ROSSEM, 2016) are a few examples for SDN emulation with support for the OpenFlow protocol. For the southbound, northbound API, and the SDN controller, there are also several open-source tools that group these layers in a single package. Examples of them are the Open Network Operating System (ONOS) (Open Networking Foundation, 2025), RYU NOS (Ryu SDN Framework Community, 2017), and OpenDaylight (FARHADY; LEE; NAKAO, 2015). Among these possibilities, this section will demonstrate the use of Mininet for SDN emulation, with ONOS serving as the SDN controller.

The Mininet tool can be operated with or without a graphical interface. For graphical usage, a complementary tool called MiniEdit is included in the same package, offering an intuitive interface, illustrated in Figure 2.3a, to design transport network topologies. Through this interface, users can configure parameters such as the OpenFlow protocol version, the type of virtual switch, and the Internet Protocol (IP) address assigned to each switch, as depicted in Figure 2.3b. For non-graphical use, Mininet provides a Python API that offers methods to programmatically instantiate SDN topologies, providing more flexibility than MiniEdit while enabling automation and customization through scripting.

From the controller side, it is also possible to consult underlying information from the network topology by a REST API. In Figure 2.4 there is an example with more switches in a non-commercial transport network topology called NSFNet (ZHU et al., 2021), with 14 nodes. In this sense, the ONOS controller concentrates the main information aforementioned, such as the topology organization with the number of devices, links, and the associated host to each device, in this case, respectively, 14, 42, and 14. Other useful information is the version of OpenFlow, in this case, 1.0, the switch identification, and the type of virtual switches, in this case, Open vSwitch.

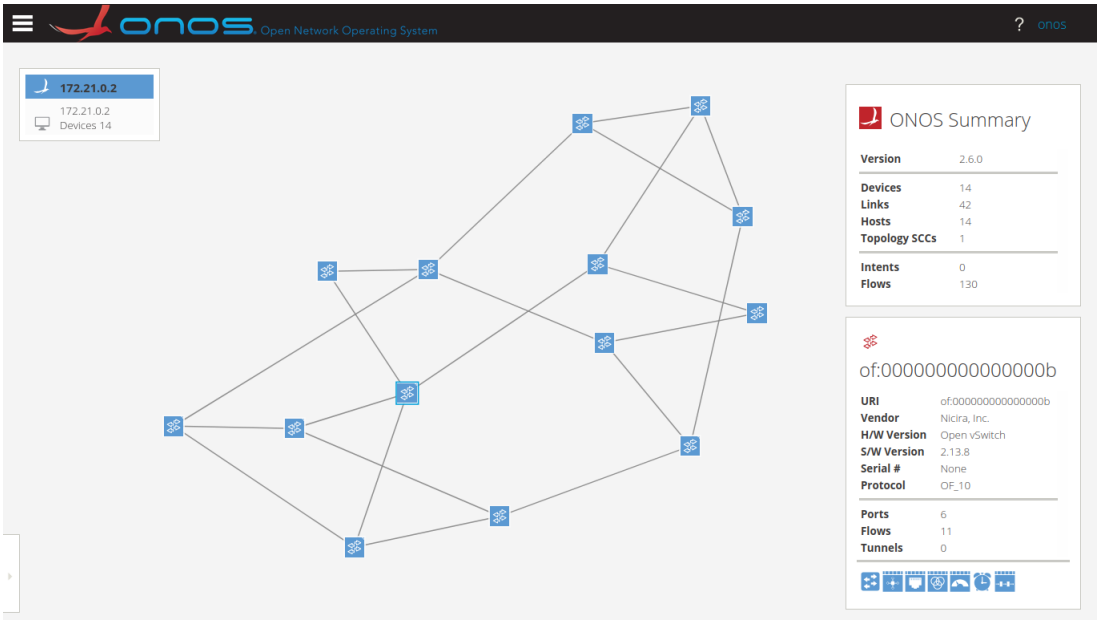
Figure 2.3 – Mininet graphical interface, called MiniEdit, used to construct SDN topologies



(a) SDN topology construct with MiniEdit. **(b)** Network configuration parameter on MiniEdit.

Source: The Author (2025).

Figure 2.4 – NSFNet topology from the controller perspective using ONOS graphical interface



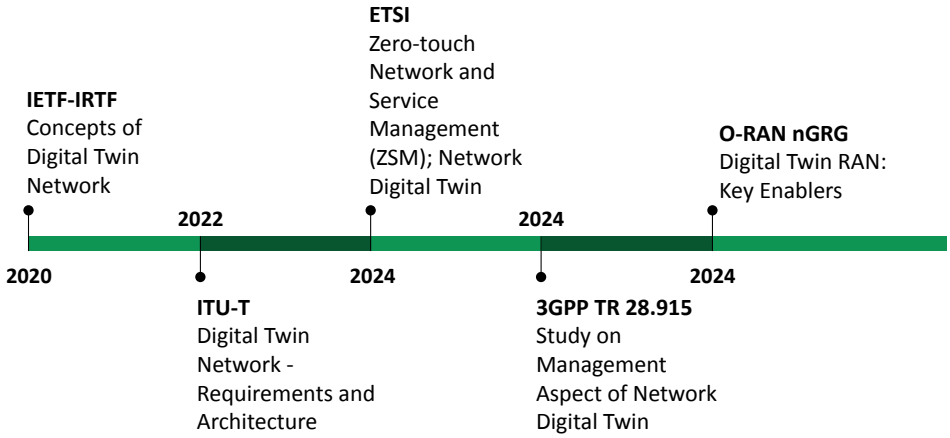
Source : The Author (2025).

2.2 Network Digital Twin

To rigorously define the core components of an NDT, various standardization bodies have proposed frameworks that specify how these elements should operate and be structured within coherent architectures. Notable examples include the efforts of the International Telecommunication Union Standardization Sector (ITU-T) Recommendation Y.3090 (Recommendation ITU-

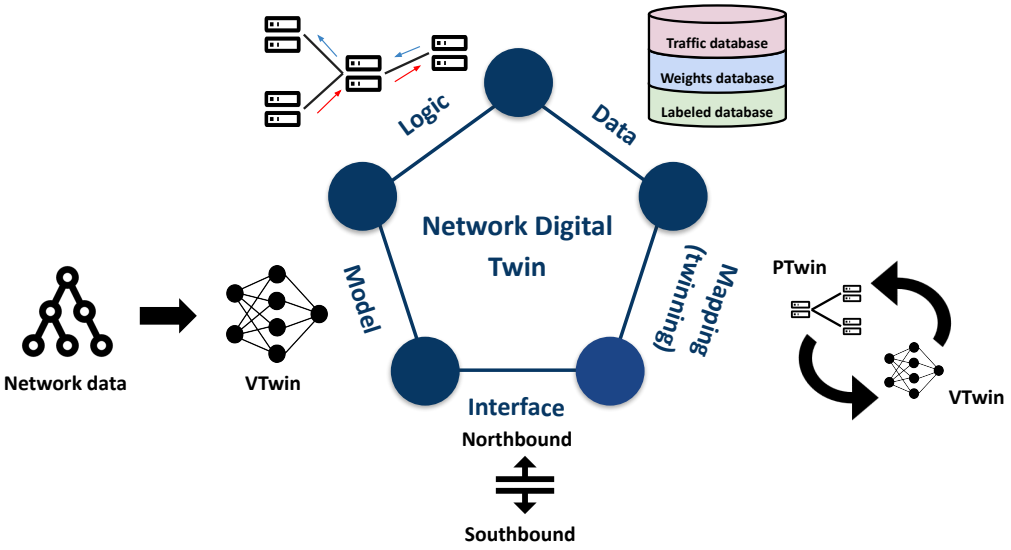
T Y.3090, 2022) and Internet Engineering Task Force (IETF) (ZHOU et al., 2025), which created the first documents regarding the technical specifications of an NDT, as depicted by the timeline in Figure 2.5. Thereby, from standardized architectures defined in the ITU-T and IETF documents, it is possible to define the core NDT elements as depicted by Figure 2.6. Considering two different organizations, it is natural to have some differences and similarities. In this sense, it is possible to group the similarities, considering five main modules: Mapping, Interface, Models, Logic, and Data.

Figure 2.5 – Timeline of the main documents related to the technical specifications of an NDT from the standardization bodies



Source: The Author (2025).

Figure 2.6 – NDT architecture core elements based on IETF and ITU standardization



Source: The Author (2025).

The Mapping element defines the synchronization modules of a NDT, managing the pe-

riodicity that the states of the PTwin will be updated in the VTwin and vice versa. This synchronization frequency is called twinning rate (JONES et al., 2020). Moreover, the ways to "equalize" both twins can be categorized into two types: synchronization between the PTwin and VTwin, and synchronization between multiple VTwins. These interactions can be characterized by one-to-one relationships—enabling bidirectional communication between physical and virtual counterparts—or one-to-many relationships, which apply specifically to synchronization among VTwins. A direct effect of twin synchronization is to ensure that the Logic module, the "brain" of the NDT architecture, maintains high accuracy in supporting various network applications, such as recommendation systems, policy evaluation, and network management, since this module has the ability to influence and modify network behavior.

In terms of Interface modules, the IETF and ITU-T define two primary types of interfaces: southbound and northbound. The first facilitates indirect communication between the PTwin and VTwin, while the second enables indirect communication between the VTwin and the network applications, within the Logic module. These interfaces closely resemble the southbound and northbound APIs of the SDN architecture, which, in the context of this work, are conceptually aligned.[2] Thus, the former interface is responsible for supporting the collection of data from the PTwin, enabling the VTwin to replicate its current state. This includes information such as network topology and performance metrics. The output generated by the VTwin is stored in a dedicated database and made available to high-level applications for tasks such as network optimization, analysis, and diagnostics. These applications, in turn, use the latter interface to send control commands to the network controller, influencing the behavior of the physical network. All data involved in these processes is managed by the Data module, which organizes and stores both historical and real-time data from the PTwin—including the aforementioned topology, traffic, and performance metrics—as well as data produced by the VTwin, such as predicted metrics or ML model parameters.

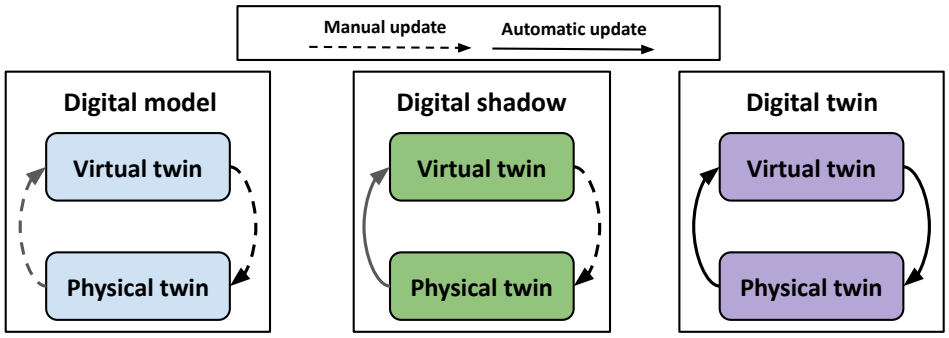
The Models module, in turn, is defined by the tool that will be used by the VTwin to abstract some aspect of the PTwin, by using emulated, simulated (BORGES et al., 2024; MODESTO et al., 2025), or AI-based approaches (FARRERAS et al., 2025; MESSAOUDI et al., 2023). These models can be constructed considering two main approaches: data-driven or model-driven (BARIAH; DEBBAH, 2024). The former approach is oriented to the data collected from the PTwin, to be used as data for training AI models, such as generative models (LI

[2]The standardization bodies also allows the possibility that these interfaces do not necessarily correspond to the SDN northbound and southbound interfaces.

et al., 2025) or GNN (ABEN-ATHAR et al., 2025; MESSAOUDI et al., 2023). The latter approach is related to the use of simulation models to abstract the target key characteristics of the PTwin, such as using discrete-event simulators like ns-3 (HENDERSON et al., 2008) to simulate different parts of the network (BORGES et al., 2024) or ray tracing tools to simulate and extract low-level channel characteristics of a wireless communication (MODESTO et al., 2025) for future management tasks.

Having established an understanding of the five core elements of an NDT, it is possible to shift to focus on its practical applicability and explore more realistic aspects of the platform and its potential use cases. So, unlike the scope taken by ITU-T and IETF, the primary focus of other standardization bodies—such as 3rd Generation Partnership Project (3GPP), O-RAN next Generation Research Group (nGRG), and European Telecommunications Standards Institute (ETSI)—has been on application-driven purposes, built upon the ITU-T and IETF architectures. This is particularly evident in 3GPP's TR 28.915 (Release 19) (3GPP, 2024) and ETSI's ZSM 015 (ETSI, 2024), where the focus of both specifications is to define the potential requirements for the use cases such as network failure and risk prediction, data generation for NDT, emergency preparedness, and many others. Beyond these requirements, the specifications also organize possible solutions, defining in high-level description how the NDT will provide for each use case.

Figure 2.7 – Different level of integration induced by the level of automation in the synchronization between the PTwin and VTwin



Source: The Author (2025), adapted from (BORGES et al., 2022).

Once the core elements of the NDT are defined and the position of the standardization body regarding this platform is well-defined, it is relevant to focus the discussion on its two main components: the PTwin and the VTwin. In particular, this discussion considers the level of integration determined by the degree of data flow automation between them—that is, whether

the states of the VTwin and the PTwin are updated manually or automatically. Dependent on this level of automation, different classifications can be used to define an interaction between the PTwin and the VTwin. In this sense, using a classification from a generic DT, at given different levels of synchronization, it can have the following nomenclature defined by Figure 2.7, where the DT is the last level of integration: digital model, digital shadow, and DT (KRITZINGER et al., 2018).

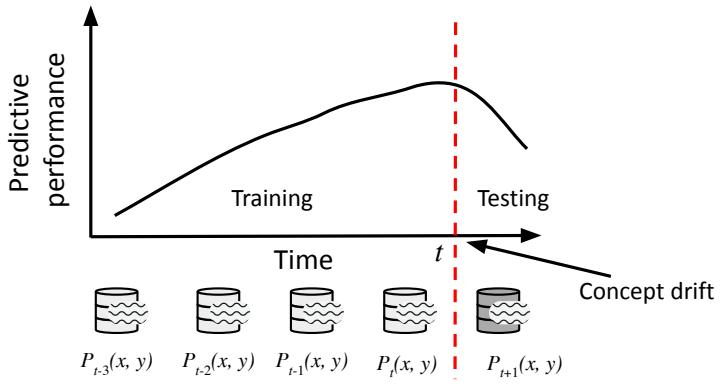
Before reaching the fully automated relationship that characterizes a DT, the simplest form of this paradigm is referred to as a digital model. This classification represents the least automated relationship between the PTwin and the VTwin, where data exchange between them is entirely manual. Examples of this type of implementation can be found in some related works discussed in the previous chapter, such as (WANG et al., 2022; HAPP et al., 2021; MODESTO et al., 2024). From a data-driven perspective, this twin relationship is analogous to training a ML model (in this case represented by the VTwin) to generalize unseen data from observations of the PTwin. Despite being the simplest form of NDT, this approach represents the early stages of research in this area and serves as the foundation for more advanced developments. The next level of evolution is the digital shadow, which introduces the first step toward automated synchronization between the twins. In this stage, the state of the PTwin is automatically reflected in the VTwin via the previously mentioned southbound interfaces. However, a key limitation of the digital shadow is that it does not yet support the automatic use of VTwin predictions to influence or modify the state of the PTwin. At the highest level, where a DT is fully implemented, the synchronization between the physical and VTwins occurs automatically. This automation enables network operators to emulate various scenarios for network planning and optimization, supporting what-if analyses (SUÁREZ-VARELA et al., 2023). Such analyses allow testing and evaluating network configurations without impacting the production environment. Consequently, this approach assesses the potential effects of management decisions on the network infrastructure, particularly in cases where direct testing is impractical due to operational or logistical constraints (ALMASAN et al., 2022a; ÖHLÉN et al., 2022).

2.3 Concept Drift

Concept drift is a well-recognized challenge in production environments where AI models are deployed (AMEUR; BRIKAUTHORREFMARK; KSENTINI, 2025; BAYRAM; AHMED;

KASSLER, 2022). It refers to changes in the statistical properties of the input data, the label data, or their joint distribution over time in a data stream. Such changes lead to a progressive degradation in model performance or may eventually render the model ineffective for its intended task due to the mismatch between the training data distribution and the data encountered during operation, as depicted in Figure 2.8.

Figure 2.8 – ML model predictive performance in a scenario with concept drift, generating a degradation performance after that



Source: The Author (2025), adapted from (BAYRAM; AHMED; KASSLER, 2022).

Traditional ML models assume a stationary data distribution—meaning that the distribution of test data matches that of the training data. However, as illustrated in Figure 2.8, this assumption fails in the presence of concept drift, where shifts in the input space, label space, or both introduce a distribution mismatch between the training and deployment environments (ŽLIOBAITĖ; PECHENIZKIY; GAMA, 2016; AMEUR; BRIKAUTHORREFMARK; KSENTINI, 2025). So, to understand the mathematical details behind this phenomenon, it is useful to start from the Bayesian decision theory (MA, 2019), considering a joint distribution $P_t(x_t, y_t)$, where x_t and y_t are the input feature and target label from an ML model at time t. In this sense, from the Bayes theorem, it is possible to state the following equalities

$$P_t(x, y) = P_t(y|x)P_t(x) = P_t(x|y)P_t(y). \tag{2.1}$$

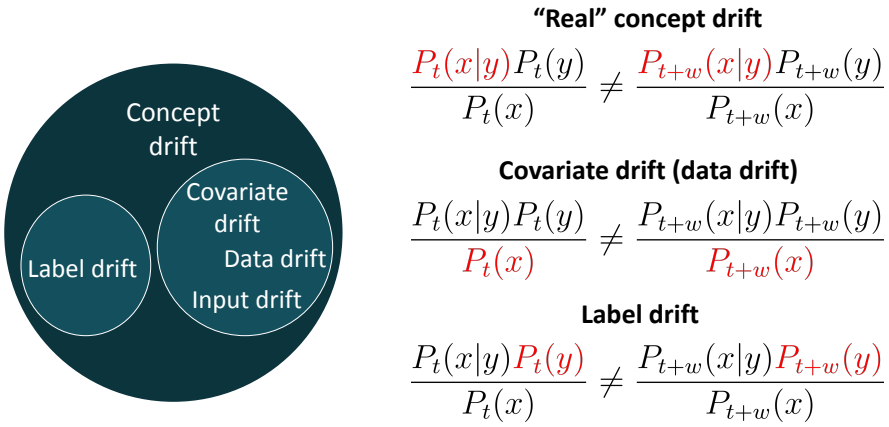
From Eq. (2.1), a concept drift would be characterized if in a time instant $t + w$, where w is a pre-defined window size, $P_t(x, y) \neq P_{t+w}(x, y)$. Which means that $P_t(y|x) \neq P_{t+w}(y|x)$. Thereby

$$\frac{P_t(x|y)P_t(y)}{P_t(x)} \neq \frac{P_{t+w}(x|y)P_{t+w}(y)}{P_{t+w}(x)}. \tag{2.2}$$

The inequality in Eq. (2.2) holds if at least one of the probability distributions changes

over time. Consequently, concept drift in an ML model can originate from three different sources, as depicted in Figure 2.9.

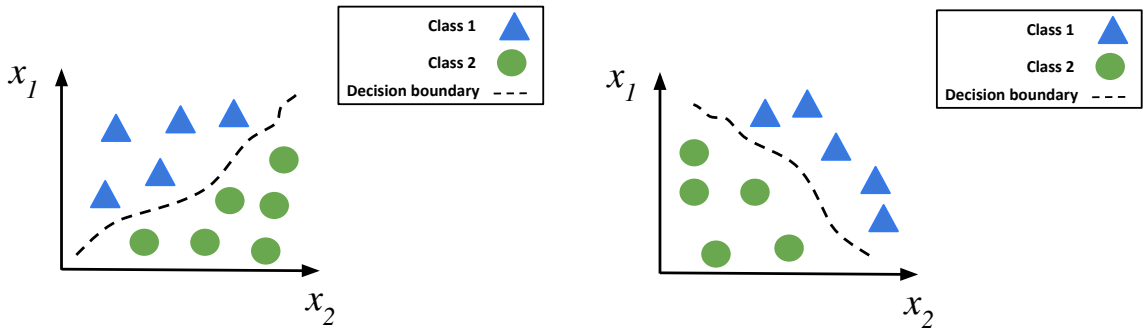
Figure 2.9 – Different sources of concept drift and some nomenclatures adopted



Source: The Author (2025).

The first occurs when the posterior probabilities change, that is, when $P_t(x|y) \neq P_{t+w}(x|y)$. This situation is known as "real" concept drift, as it reflects a change in the underlying relationship between inputs and labels of the dataset. To better illustrate the impact of this phenomenon, let's assume a classification task, considering the toy dataset at time t with two features (x_1 and x_2) shown in Figure 2.10a, which contains two classes. When the posterior distribution changes, the model's decision boundaries are inevitably altered—as depicted in Figure 2.10b—since the input-output relationship itself has shifted.

Figure 2.10 – AI model considering an environment with and without concept drift



(a) Original data classification at time t **(b)** Data classification with concept drift at time $t + w$

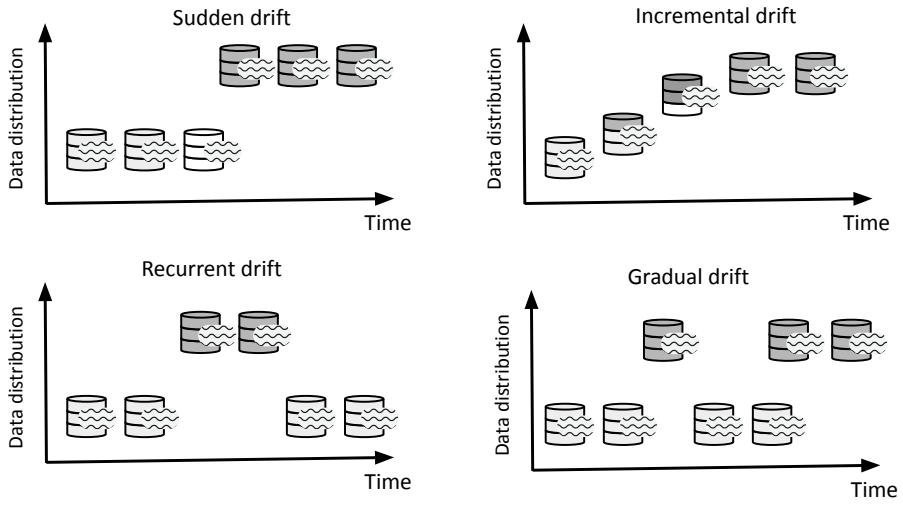
Source: The Author (2025).

Similarly, when the input data distribution $P_t(x) \neq P_{t+w}(x)$ changes, there is the case of concept drift, also called *covariate shift*. In this case, not necessarily do the decision boundaries

have to change, since these changes cannot affect the target label. Finally, when the prior probabilities $P_t(y) \neq P_{t+w}(y)$ change, there is the case of label drift, which directly affects the model's performance. In this case, the prior probabilities can be affected by the emergence of new classes (in a classification task) or the change in the frequencies of each class.

Furthermore, when dealing with data streams, changes in probability distributions can exhibit different patterns over time. Consequently, the terminology used to describe concept drift depends on the nature of these patterns. For instance, when an abrupt change occurs in the probability distributions, it is referred to as sudden drift. Figure 2.11 illustrates three additional types of drift: incremental, recurrent, and gradual.

Figure 2.11 – Different concept drift pattern considering a data stream: sudden, incremental, recurrent and gradual drift



Source: The Author (2025).

Using a network communication as an environment for example, the sudden drift can be linked to events like traffic congestion, where network elements may behave unexpectedly due to resource depletion. The incremental drift represents a smooth and continuous change in the input-output characteristics of an AI system, which is common in production environments. For example, the gradual increase in network traffic load throughout the day as more users connect. The recurrent drift is characterized by cyclic patterns, such as the daily variation in the number of users accessing a website, and the traffic load generated from this. Lastly, the gradual drift resembles incremental drift but occurs through intermittent, less continuous changes over time until the data distribution fully transitions to new characteristics.

Finally, with this variability in a production AI system, it is useful to have methods that

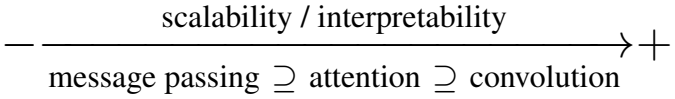
can estimate these changes, and possible solutions can be used to mitigate this problem. Thus, in the related literature (IWASHITA; PAPA, 2019), there are several ways to classify these methods, but in this dissertation, two types will be considered: the input distribution-based and performance-based methods. The former method, as the name suggests, aims to identify changes in the distribution of input features. Examples of proposed methods in the literature are the Kolmogorov-Smirnov Windowing (KSWIN) (RAAB; HEUSINGER; SCHLEIF, 2020) and the Page-Hinkley (SEBASTIÃO; FERNANDES, 2017). Performance-based methods take a different approach, since this type of method aims to mitigate the drift generated from the prior distribution probability difference over time. Examples of performance-based methods are the Drift Detection Method (DDM), Early Drift Detection Method (EDDM), Adaptive Windowing (ADWIN), and many others (LU et al., 2018).

2.4 Graph Neural Networks

GNN models are a subset of deep learning architectures adapted to processing graph-structured data (SCARSELLI et al., 2009), also referred to as non-Euclidean data. In this sense, an example of this non-Euclidean data is a 2-tuple graph $\mathcal{H} = (V, E)$, where $V = \{v_1, v_2, \ldots, v_i, \ldots, v_{|V|}\}$ is the set of nodes and $E = \{q_1, q_2, \ldots, q_j, \ldots, q_{|E|}\}$ is the set of edges. Moreover, each node v_i is characterized by a node feature $\mathbf{x}_{v_i}$, such that $\mathbf{x}_{v_i} \in \mathbb{R}^n$, where n is the number of features. Based on this definition, several ML models have been developed to support predictive and classification tasks within this type of structure, leveraging supervised or semi-supervised learning approaches. Among these, three models are particularly noteworthy: the Graph Convolutional Network (GCN) (KIPF; WELLING, 2016), the Graph Attention Network (GAT) (VELIČKOVIĆ et al., 2018), and the Message Passing Neural Network (MPNN) (GILMER et al., 2017). Figure 2.12 illustrates the relationship between these models in terms of scalability and interpretability, representing each as an element within a formal set (BRONSTEIN et al., 2021).

Regarding the interpretability of these models, it refers to the ability to understand the reasons that the GNN model takes a decision to generate a given output. Considering these three GNN variants, interpretability tends to decrease as the models become more general, as illustrated in Figure 2.12. For instance, the attention mechanism in the GAT, while offering greater flexibility in the message passing phase, introduces non-linear components that contribute to

Figure 2.12 – Relation between the scalability and interpretability in the set of GNN models



Source: The Author (2025).

a black-box nature, making the model less interpretable. In terms of scalability, this refers to the computational cost required to train each model. In this way, the relative relationship between the models remains unchanged: scalability tends to decrease as the models become more general. This is mainly due to the dimensionality of the message passing operations. While GAT and GCN typically perform scalar-valued message passing, the MPNN model requires more complex vector-valued computations, which are more computationally intensive (BRONSTEIN et al., 2021).

Focusing on the interpretability aspect highlighted in Figure 2.12, it becomes more evident when we consider the definition of the simplest form of a GCN, commonly referred to as the vanilla GCN, where the output layer $\mathbf{H}$ is defined as follows:

$$\mathbf{H} = \mathbf{D}^{-1}\mathbf{A}\mathbf{X}, \tag{2.3}$$

where $\mathbf{A} \in \mathbb{R}^{|V|\times|V|}$ represents the adjacency matrix of the graph $\mathcal{H}$, $\mathbf{D} \in \mathbb{R}^{|V|\times|V|}$ is the corresponding degree matrix, and $\mathbf{X} \in \mathbb{R}^{|V|\times n}$ is the input feature matrix, which contains the features of each node in the graph. The intuition behind Eq. (2.3) is to propagate information, known as message passing, between nodes based on the graph's adjacency matrix. This process allows each node to aggregate information from its neighbors. Furthermore, this message passing can be refined by employing a normalized adjacency matrix, as proposed by (KIPF; WELLING, 2016), which stabilizes the learning process and mitigates issues related to scale and node degree.

Based on this GCN formulation, it is evident that interpretability is inherently preserved, since the forward propagation of this model, in its simplest form, consists of a straightforward multiplication between the adjacency matrix and the input feature matrix. However, this level of interpretability diminishes as more complex, black-box operations are introduced into the message passing process—such as the attention mechanism employed in GAT. Consequently,

to better capture and analyze these nuances across different GNN models, it becomes advantageous to adopt a vector-based and more general notation, rather than the matrix-based representation used in Eq. (2.3). In this sense, the output $\mathbf{h}_u$ of the GCN model (and its variations) can also be defined by

$$\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{u \in \mathcal{N}_u} c_{uv}\psi(\mathbf{x}_v)\right), \tag{2.4}$$

where c_{uv} is a constant obtained from the graph adjacency matrix $\mathbf{A}$ and defines the importance of node v to node u's representation, $\bigoplus$ is an aggregation function, $\mathcal{N}_u$ is the neighborhood of the node u, $\phi(\cdot)$ is the update function that adjusts the features of node u after the message passing, and $\psi(\cdot)$ is the message function that propagates the information from a node u to its neighbors. In contrast, the GAT definition is a generalized form of the GCN and can be described by

$$\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{u \in \mathcal{N}_u} a(\mathbf{x}_u, \mathbf{x}_v)\psi(\mathbf{x}_v)\right), \tag{2.5}$$

and the key difference lies in the inclusion of the attention score $a(\cdot)$, which serves a similar purpose as the constant c_{uv} in GCN. However, unlike c_{uv}, the attention score a is a learnable function with its own parameters and is not entirely derived from the graph adjacency matrix. Finally, both GCN and GAT can be understood as specific instances of the MPNN framework, which represents the most general form of these GNN models, which is defined by

$$\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{u \in \mathcal{N}_u} \psi(\mathbf{x}_u, \mathbf{x}_v)\right). \tag{2.6}$$

This general MPNN formulation is useful for processing data related to communication networks, such as from a transport network abstracted into heterogeneous graphs (MODESTO et al., 2024), since this data can represent the wired transport network, in which the edges can be seen as an abstraction of links from several nodes' interconnections, in this case defined as network switches/routers, in a way that through this network can pass a set of flow[3] traffic from several applications (MODESTO et al., 2024). This data representation is the basis of the multiple stage message passing used in the RouteNet model and its flavors (FERRIOL-GALMÉS et al., 2023; FERRIOL-GALMÉS et al., 2022), widely explored to predict QoS metrics, as shown in the Related Work section. Therefore, given these characteristics, the GNN model can serve as a virtual counterpart to an NDT, the VTwin, by abstracting key aspects of the

[3]Here, it is assumed the flow terminology considering a flow traffic from a source switch, to a destination one.

transport network, the PTwin. This enables effective performance, with a certain extrapolation capability, on high-level tasks such as QoS prediction—including metrics like delay, jitter, and packet loss (RUSEK et al., 2020; FERRIOL-GALMÉS et al., 2023).

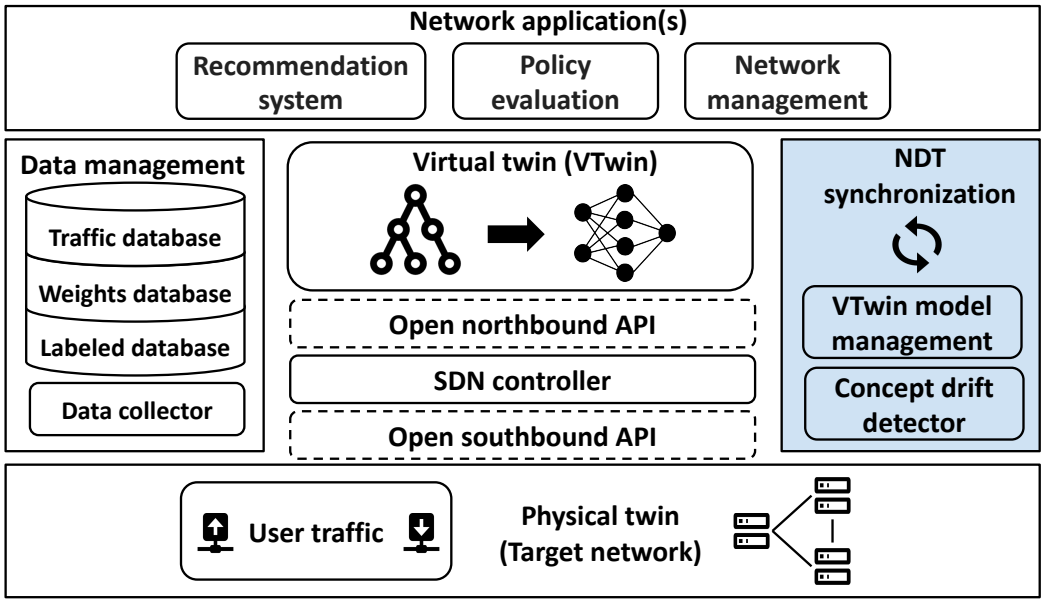
This extrapolation capability highlights a remarkable feature of RouteNet: its ability to generalize to previously unseen network topologies beyond those encountered during training. For instance, RouteNet can be trained on a relatively small dataset and still achieve acceptable performance, in QoS prediction, when evaluated on significantly larger topologies, in terms of both nodes and links (AFONSO; BERTON, 2022). This property is particularly valuable in the context of NDT synchronization. When such a model is employed as a VTwin, changes in the topology structure do not necessarily compromise its generalization ability, thereby eliminating the need for retraining. Nonetheless, this capability operates within certain limits, and the following chapter investigates the range within which this model remains effective, as well as the boundaries where its limitations emerge.

Chapter 3

Enhanced Digital Twin-Based Transport Network

The proposed enhanced architecture is divided into the modules illustrated in Figure 3.1, and shares some inheritance from standardization bodies (Recommendation ITU-T Y.3090, 2022; ZHOU et al., 2025) (which in turn share some elements from the SDN architecture) and industry stakeholders (ÖHLÉN et al., 2022).

Figure 3.1 – Enhanced NDT architecture to construct a reactive approach against traffic variability. This self-adaptive is reached with the proposed implementation of the NDT synchronization functions (in blue) such as concept drift detector and VTwin model management



Source: The Author (2025).

Adopting a bottom-up approach, the first module corresponds to the PTwin defined by the switches of the transport network, the same as the SDN architecture. Above the PTwin, we have the SDN controller and its related interfaces north and southbound. Next to it, the VTwin, which in the presented solution is implemented using a GNN model. It provides a digital representation of the relationship between network traffic and topology links. This model plays a crucial role in estimating the total per-flow delay in the PTwin. The following module is the data management, which stores useful traffic database information, such as average traffic rate, packet loss, flow length, and so on. This module maintains a unified repository of transport network data, which is essential for feeding the VTwin with relevant features in the retraining process. Additionally, this module includes the necessary functions to collect and store traffic data, ensuring seamless integration with the overall system.

Operating in the background to enable communication between these NDT modules, we have the interfaces defined by the SDN controller, which allow indirect data exchange between the VTwin and the controller, intended to apply any necessary modifications in the PTwin, and indirect data exchange between the database collector and the PTwin to fulfill this database with updated information from the physical environment. With this closed loop of control and traffic data between both twins, the VTwin predictions can be used in high-level applications, either with the involvement of a network operator, as defined by Aben-Athar et al. (ABEN-ATHAR et al., 2025), or autonomously, without human supervision, as demonstrated by Messaoudi et al. (MESSAOUDI et al., 2023). In this dissertation, we assume an SLA monitoring application. Complementary to this closed loop, the main contribution relies on the modules related to the synchronization of the NDT modules, which are responsible for synchronizing the status of the transport network of both domains in a reactive way. Along with the concept drift method, this module monitors if the incoming traffic that reaches the VTwin remains similar. Once changed and detected by the drift detector, this module has the capability to trigger a retraining process, since this change can also mean that the VTwin model is no longer predicting the total per-flow delay with an acceptable error performance.

Finally, in terms of implementation, the proposed architecture source code, with its integration with concept drift techniques, the datasets used, and the evaluation methodology, are available in a Git repository.[1]

[1] ⟨https://github.com/lasseufpa/robust-ndt⟩

3.1 Physical Twin

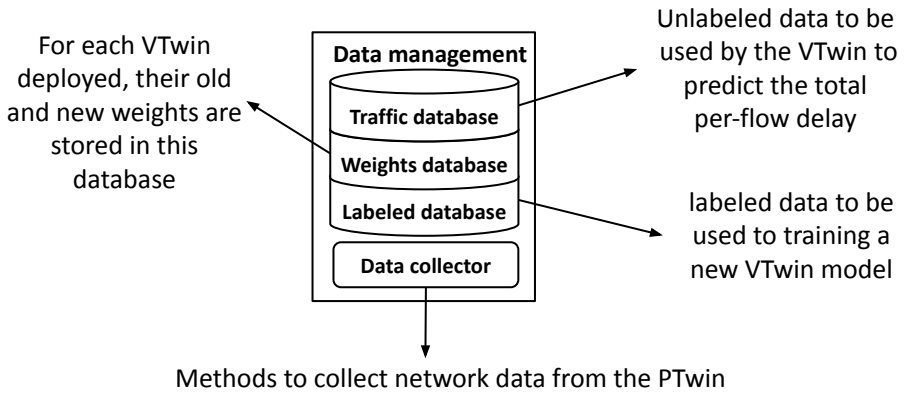
As the PTwin, it is considered a transport network topology modeled as a 2-tuple graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. In this graph, $\mathcal{V}$ denotes the set of nodes, each representing a network switch, while $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ defines the set of links that connect pairs of nodes. Within this network, a traffic generator can produce a set of flow traffic with a total of T instances denoted as $\mathcal{F} = \{f_1, f_2, \ldots, f_t, \ldots, f_T\}$, in a discrete time t. Each flow instance consumes network resources, such as link bandwidth, by applying variable load intensities along a specific path composed of links $\mathcal{E}' = \{e_1, e_2, \ldots, e_j, \ldots, e_J\}$, where J denotes the flow length. Each flow instance is defined as a tuple $f_t = (\mathbf{x}_{f_t}, O, D, \mathbf{j})$, where $\mathbf{x}_{f_t} \in \mathbb{R}^n$ is a feature vector that characterizes the flow's behavior. In this sense, each feature vector $\mathbf{x}_{f_t}$ is associated, by a mapping rule h, to a real number $\mathcal{D}_t$ defined as the total per-flow delay, such that $h : \mathbf{x}_{f_t} \rightarrow \mathcal{D}_t$. Furthermore, O and D represent the source and destination nodes ($\{O, D\} \in \mathbb{Z}^+$); and $\mathbf{j}$ is a vector of link indices that the t-th flow traverses. Similarly, each link instance is defined as $e_j = (\mathbf{x}_{e_j}, \mathbf{i})$, where $\mathbf{x}_{e_j} \in \mathbb{R}^m$ is a feature vector describing the link's characteristics, where m is the number of features; and $\mathbf{i}$ is a vector of flow indices representing the flows traversing that j-th link. Furthermore, some flow features, such that $x_t \in \mathbf{x}_t$ at time t can also be described by a probability distribution P_t, such that $x_{f_t} \sim P_t$.

3.2 Database Organization and SDN Controller

The database utilized to perform the decision-making process is divided into three storages. The first one, called *traffic database*, is related to the stream data that was filtered after it came from the traffic generator and will be the input for the VTwin model to perform the necessary delay prediction. The second type of database is related to the examples that should be used to retrain the VTwin model, and it is called *labeled database* as it manipulates the ground truth total delay directly to update the VTwin model weights when necessary. The last storage, called *weights database*, aims to store the history of all VTwin model weights during the NDT operation. In summary, the primary roles of each database are depicted in Figure 3.2.

To store the raw data related to the labeled and traffic database, it is also considered data collection functions to discard unnecessary packet-level traffic data, such as the IP addresses of the server and client or transmission time timestamps. Therefore, this data collection aims to store only QoS related features that are possible to extract from the PTwin environment and

Figure 3.2 – Primary roles of each database in the NDT data management module



Source: The Author (2025).

that are useful for the VTwin prediction task. It is important to note that it is assumed the data required for the retraining process has already been collected, labeled, and stored in the database. Furthermore, issues related to synthetic data generation or real-time data collection for retraining are considered outside the scope of this dissertation.

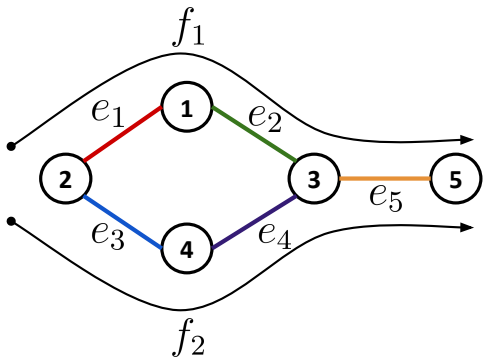
The interfaces employed in the proposed architecture enable the indirect exchange of control data between both twins, with the SDN controller serving as the NOS. To enable this interaction, the SDN controller is utilized as the central bridging tool, leveraging its available APIs. Specifically, another feature stored in the traffic and labeled database is the vector of links $\mathbf{i}$ traversed by a given flow. Consequently, one of the critical functions of the interface module is to work with the path collector to provide this information, which identifies and records the routes taken between each source–destination node pair in a flow traffic. In this sense, the path collector works considering a specific node identifier, based on a 16-hexadecimal character that comes from the PTwin. So, given that flow traffic was initialized between two node pairs, it is possible to consult, by the controller REST API, the path of each flow. In this case, it is assumed that the SDN controller operates in a reactive forwarding way, that is, considering a forwarding decision and route establishment on demand, working only when a packet should be sent. Moreover, the shortest path defines the routing table of each switch in the network.

3.3 Virtual Twin

Considering the adopted NDT architecture, the VTwin assumes a role in mimicking some key aspects of its physical counterpart. In this sense, this VTwin is based on a RouteNet-Fermi

GNN model. Accordingly, the VTwin utilized will be responsible for abstracting the most complex organization of the transport network and synthesizing the key aspects of it to evaluate the interest QoS parameter, keeping a certain level of fidelity in the physical-virtual relationship. In this case, for the objective of estimating the total per-flow delay, it is important to model the relation between the flow traffic applied to propagate information through the wired medium and the links used for it, as shown in the generic graph in Figure 3.3. This modeling step is important, as the VTwin input is not the original graph $\mathcal{G}$ representing the transport network, but rather the hypergraph derived from it.

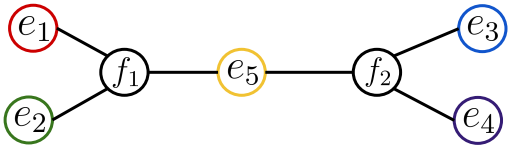
Figure 3.3 – Relation between links and flows in a generic network topology



Source: The Author (2025).

In this sense, the relation between the flows and links of Figure 3.3 creates an essential circular dependency (the state of flows depends on the state of link and vice versa) that is explored to generate the heterogeneous graph used as input of the VTwin model. In the case of Figure 3.3, it is possible to see that flow f_1 depends on links e_1, e_2, and e_5. Similarly, the flow f_2 has a dependency on the links e_3, e_4, and e_5. Therefore, these relationships generate the graph depicted in Figure 3.4.

Figure 3.4 – Heterogeneous graph generated from the transport network of Figure 3.3

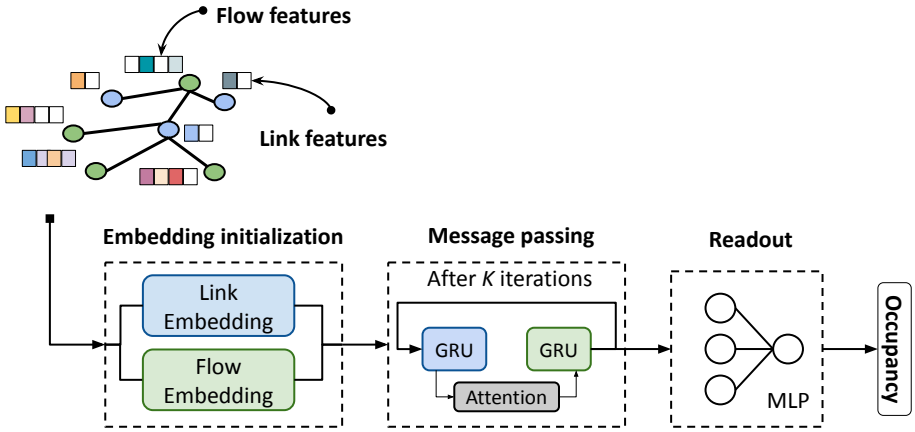


Source: The Author (2025).

Regarding the architecture of the VTwin, it is considered a GNN based on modified RouteNet-Fermi with a self-attention mechanism (MODESTO et al., 2024), as illustrated

in Figure 3.5. This model processes flow and link characteristics through message passing across two layers of Gated Recurrent Unit (GRU). The aggregated messages are then used by a Multilayer Perceptron (MLP) to compute occupancy and estimate the total per-flow delay.

Figure 3.5 – Flow and link features from generic topology processed by the VTwin model



Source: The Author (2025).

Specifically, the embedding initialization has the purpose of processing the node features $\mathbf{x}_{f_t}$ and $\mathbf{x}_{e_j}$ from a heterogeneous graph obtained from the relationship between the links and the flows, that is, the f_t flows that passed by several links and the e_j links that received each flow (MODESTO et al., 2024). Since in this graph there are two types of nodes, with different feature vectors, specific embedding initialization functions are used to convert the features of each node type from their original dimension to high-dimensional space $\mathbf{g}_{f_t} \in \mathbb{R}^b, \mathbf{g}_{e_j} \in \mathbb{R}^r$, where b and r are model hyperparameters, respectively, the size of the flow and link embedding vectors.

In practical terms, to obtain these high-dimensional vectors for each node type, two MLP layers are used as an initialization function, both with one hidden layer and having the output dimension defining the embedding size, $\|\mathbf{g}_{f_t}\| = b$ and $\|\mathbf{g}_{e_j}\| = r$. Thereby, this embedding initialization is defined by equations

$$\mathbf{g}_{f_t} = \text{MLP}(\mathbf{x}_{f_t}, \theta_{f_t}), \tag{3.1}$$

and

$$\mathbf{g}_{e_j} = \text{MLP}(\mathbf{x}_{e_j}, \theta_{e_j}), \tag{3.2}$$

where θ_{f_t}, θ_{e_j} are the learnable parameters of each initialization function.

In the message-passing phase, these high-dimensional vectors are then processed iteratively during $k \leq K$ (which K is another model hyperparameter) time steps by the message function $\psi(\cdot)$ and updated by functions $\phi_{\mathrm{f}}(\cdot)$, $\phi_{\mathrm{l}}(\cdot)$, knowing that $\mathbf{h}_{f_t}^0 = \phi_{\mathrm{f}}\left(\mathbf{g}_{f_t}, \psi(\mathbf{g}_{f_t}, \mathbf{g}_{e_j})\right)$, as defined as follows, (BRONSTEIN et al., 2021)

$$\mathbf{h}_{f_t}^{k+1} = \phi_{\mathrm{f}}\left(\mathbf{h}_{f_t}^{k}, \psi(\mathbf{h}_{f_t}^{k}, \mathbf{h}_{e_j}^{k})\right), \tag{3.3}$$

and

$$\mathbf{h}_{e_j}^{k+1} = \phi_{\mathrm{l}}\left(\mathbf{h}_{e_j}^{k}, \bigoplus_{p \in \mathcal{N}_{e_j}} a(\mathbf{h}_{e_j}^{k}, \mathbf{h}_{p}^{k+1})\psi(\mathbf{h}_{e_j}^{k}, \mathbf{h}_{p}^{k+1})\right), \tag{3.4}$$

where $\bigoplus$ is an aggregation function based on summation, $a(\cdot)$ the attention score, and $\mathcal{N}_e$ is the neighborhood of j-th link node. Moreover, this model treats the message function as a GRU layer and the auxiliary assignment operation as an update function.

The output of the message-passing module is used to compute the occupancy $\mathcal{O}$ through a readout function, implemented using an MLP with two hidden layers. The MLP produces a single-neuron output that yields the occupancy rate for each link in the network, as defined by the following equation

$$\mathcal{O} = \mathrm{MLP}(\mathbf{h}_{f_t}^{k+1}, \theta_{\mathrm{o}}), \tag{3.5}$$

where θ_{o} are the learnable parameters of the readout MLP layer. So, considering that a flow set passes through N links, generating the occupancy mentioned above $\mathcal{O}$, the predicted total per-flow delay $\hat{\mathcal{D}}$ is evaluated by

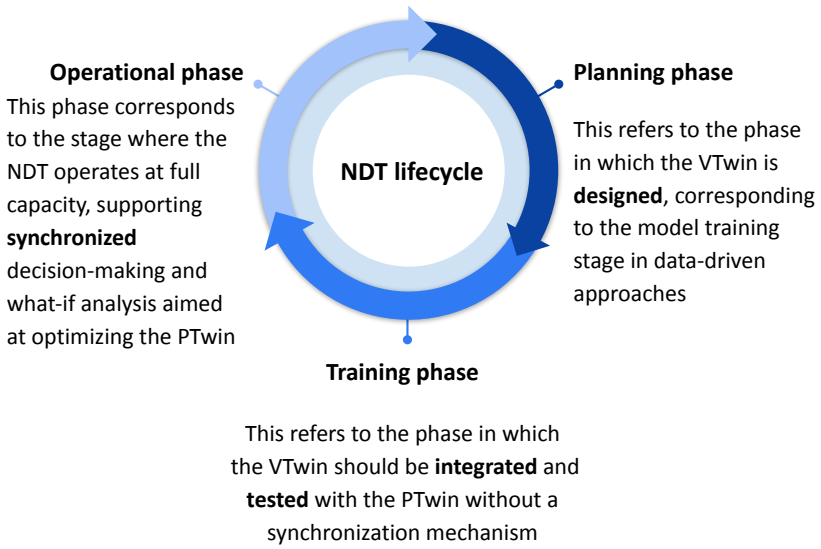
$$\hat{\mathcal{D}} = \sum_{j=0}^{J}\left(\frac{\mathcal{O}_j}{\mathcal{E}_j'}\right). \tag{3.6}$$

3.4 NDT Synchronization

A well-defined LCM provides the NDT platform with the essential capabilities to function effectively in a production environment. It defines how the NDT should behave under various conditions, especially during the operational phase (KHERBACHE; MAIMOUR; RONDEAU, 2021), whether those conditions are routine or exceptional. In this sense, we can separate the lifecycle of a NDT considering three phases: planning, training, and operational phase, as depicted in Figure 3.6.

All LCM phases can encompass different modules of the architecture presented in Figure

Figure 3.6 – NDT lifecycle comprising the planning, training, and operational phase

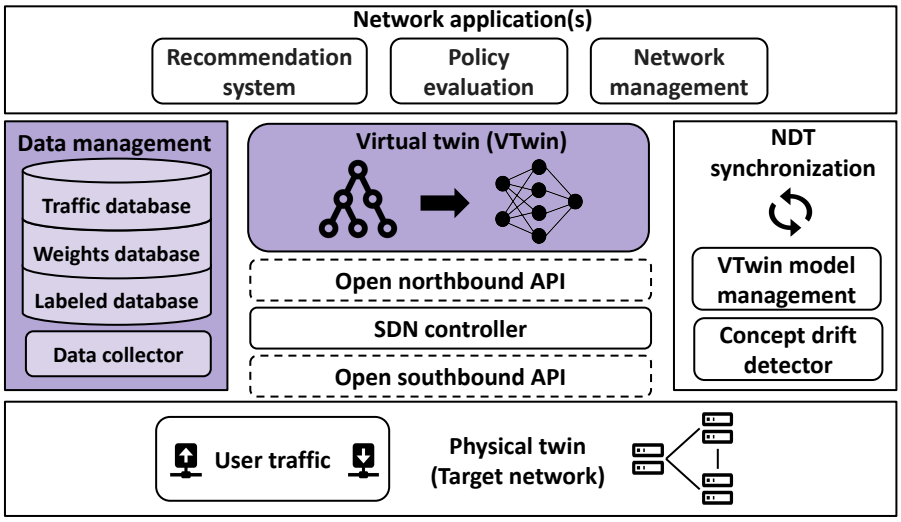


Source: The Author (2025).

3.1, and each of them has a specific role and lifespan to ensure the correct operation of the NDT tasks (KHERBACHE; MAIMOUR; RONDEAU, 2021). Since each phase has a different lifespan, some modules may not exist in particular phases. For instance, during the planning stage of the VTwin, the PTwin is not yet effectively required. At this point, the VTwin can be trained using pre-existing datasets (BARIAH; DEBBAH, 2024), considering related features from the target network environment, eliminating the need for real-time data integration. As a result, only the VTwin and selected components of the data management module are active, as depicted in Figure 3.7. This phase closely resembles the process of training an AI model for predictive tasks using offline data, where the focus is on optimizing model parameters and hyperparameters based on generalization performance. Thereby, in this phase, we have a digital model.

As the NDT lifecycle progresses, the training phase begins to incorporate physical components of the SDN infrastructure, including the interfaces connected to the SDN controller. In this context, training refers to the process of evaluating the VTwin prior to its deployment in a production environment. This phase enables the identification and mitigation of potential critical issues that could impact the proper functioning of the NDT platform. As illustrated in Figure 3.8, the VTwin starts to operate in a closed loop with the PTwin, delivering insights and outcomes to support network applications in performing analyses and providing recommendations to network operators. However, despite the inclusion of more NDT modules at this stage,

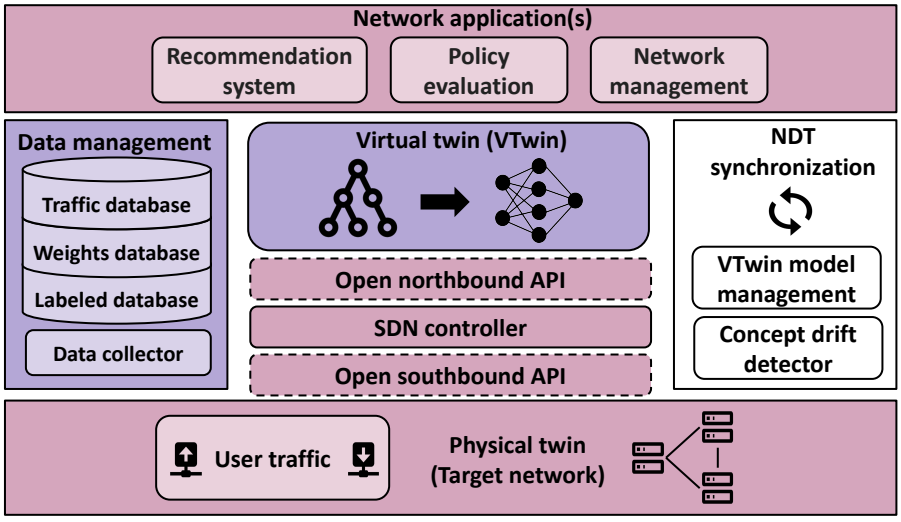
Figure 3.7 – NDT during the planning phase using just the VTwin and related tools of the database (in purple)



Source: The Author (2025).

the system still resembles a digital model rather than a fully functional twin. This is because there are no automation mechanisms to synchronize the VTwin state automatically in response to changes in the PTwin state.

Figure 3.8 – NDT during the training phase using the related modules to the PTwin, SDN controller and the network application (in red)

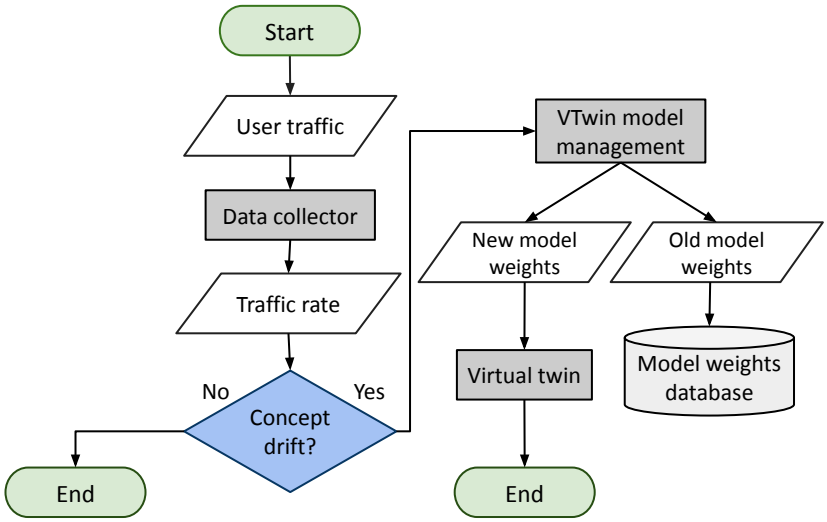


Source: The Author (2025).

Finally, in the operational phase, it was employed two submodules that monitor the synchronization states of the NDT: concept drift detector and VTwin model management. The

relationship in terms of input and output during the NDT operational phase is defined by Figure 3.9. These submodules are essential as they ensure the NDT robustness against traffic variability after NDT planning and training, which is intimately associated with the degree of synchronization between the VTwin and PTwin (twinning rate) (ALGHAMDI; ALBASSAM, 2024; LIU et al., 2024). This twinning rate will dictate the frequency at which the NDT will start the process of synchronizing the network states of the VTwin (JONES et al., 2020).

Figure 3.9 – Sequence of steps in the operational phase to synchronize the PTwin states in the VTwin



Source: The Author (2025).

3.4.1 Concept drift detector

In a data-driven supervised learning approach, the synchronization process is associated with a retraining operation and can be triggered based on either the input or the output of the neural network model. This choice is essential, as the synchronization between the VTwin and PTwin will differ significantly depending on the target variable considered in the concept drift detector process. In the first case, synchronization is driven by the distribution pattern of each input feature, such as the flow of traffic passing through a set of links. In this sense, when decisions regarding concept drift are made solely based on the input data, we have the mentioned *distribution-based* method (BAYRAM; AHMED; KASSLER, 2022). In contrast, when the correct labels are also considered, the method is classified as *performance-based* (BAYRAM; AHMED; KASSLER, 2022). In the latter approach, in the context of this problem, synchronization depends on the ground truth label, specifically the total per-flow delay. However, using

this approach, under realistic conditions, detecting the *aging* of a neural network model is not straightforward, as the ground truth total delay is not readily available for direct comparison with the VTwin's predictions. Therefore, the proposed synchronization approach employs a univariate distribution-based method that monitors the flow average traffic rate (the average throughput of a flow) to detect concept drift during the VTwin operation, thereby eliminating the need to rely on the ground truth total delay to trigger the synchronization process through VTwin retraining. This feature is selected because it effectively captures the primary characteristic of user traffic, its average throughput. In operational terms, to reach these synchronization steps, it is necessary to pass through different NDT modules, as defined by the flowchart in Figure 3.9. Therefore, the synchronization process begins with the user traffic from the transport network. This raw traffic data is collected and filtered by the corresponding functions and stored in the traffic database. A telemetry agent for the concept drift detector continuously monitors this database, extracting a single feature: the average traffic rate of the flow. When a concept drift is identified using this feature, the concept drift detector triggers the VTwin model management to initiate the retraining of the VTwin model, using data already available in the labeled database.

3.4.2 VTwin model management

When concept drift is detected, the concept drift detector notifies the VTwin model management submodule with a warning to start the retraining process using data already available in the labeled database. Upon the start of the retraining process, the VTwin model management locks the trigger to prevent the drift detector from issuing duplicate warnings for the same drift event previously addressed, until training is finished. Upon completion of the retraining process, the VTwin model management sends a control message to the VTwin, archives the previously trained model weights in a dedicated database, unlocks the trigger for possible new concept drifts, and updates the VTwin model with the newly learned weights.

3.5 Network application

The network application is organized to use the predictions provided to the VTwin to perform adjustments in the PTwin considering applications that possibly will change the state of the PTwin, such as routing optimization or flow admission control or related to the management

of the network, such as SLA monitoring for what-if analysis (WANG et al., 2022) or emergency preparedness (3GPP, 2024). To achieve this, various submodules can be employed within these applications, including, but not limited to, a recommendation system to propose different configurations in the PTwin, a policy evaluation module to evaluate the configurations suggested by the recommendation system, and a network management submodule designed to provide application-level metrics from the PTwin.

In scenarios where the network application aims to optimize certain states of the PTwin, this module incorporates auxiliary tools, such as a recommendation system, that help the network operator by leveraging outputs from the VTwin. These tools can generate new policies that can be potentially applied to the PTwin. For example, in routing optimization, the recommendation system may propose new routing policies based on total per-flow delay metrics provided by the VTwin (ABEN-ATHAR et al., 2025). Then, this recommendation is sent to the policy evaluation submodule, to be accepted or not by a network operator (ÖHLÉN et al., 2022). In a man-in-the-loop architecture, a policy evaluation submodule is essential to prevent network modifications without human oversight—an important safeguard in production environments, particularly in architectures built on 5th Generation (5G) networks (MESSAOUDI et al., 2023). This is especially critical given the interdependence of network components, such as the reliance of the radio access network on the transport network. The role of this submodule is to consolidate all relevant information and assess the potential impacts on the network, thereby enabling informed decision-making by the operator. Once a policy is evaluated, the final decision on its implementation rests with the operator. If approved, the corresponding control data is transmitted to the SDN controller via the northbound API and subsequently applied to the PTwin, thus completing the NDT loop.

When the application serves as a network management tool for what-if analysis, the task becomes conceptually simpler, as it does not require direct interaction with the PTwin to implement changes. Instead, its main objective is to provide the network operator with insight into the possible outcomes of specific scenarios or conditions (3GPP, 2024), which can be useful in network planning tasks. This dissertation explores a particular use case in the Experimental Results section, focusing exclusively on the network management submodule to provide SLA violations in the network (KHAN et al., 2022; SOMMERS et al., 2010). The selected application assesses compliance with flow SLAs, specifically detecting violations related to the Packet Delay Budget (PDB) assigned to each packet. For example, if the total per-flow delay exceeds

the PDB assigned for this flow. So, for a given set of flows that traverse the transport network, this application can estimate the number of SLA violations, relying on the per-flow delay predictions provided by the VTwin.

Chapter 4

Experimental Results

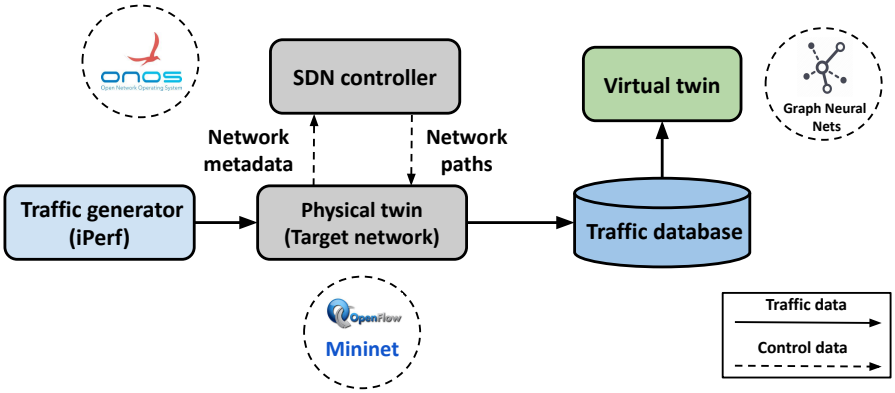
The validation of the enhanced architecture is divided into two types of environments. The first environment considers a VTwin operation without the proposed NDT synchronization modules and uses a simple deterministic variation of the traffic pattern passing through the transport network, with and without traffic congestion. In contrast, the second environment adopts the synchronization modules and faces a more challenging traffic pattern, with a distinct packet size and average traffic rate distribution throughout the NDT operation. Furthermore, also using this second environment, additional experiments are proposed aimed at evaluating a use case application's performance considering high-level metrics, in this case related to a SLA monitoring application.

4.1 Simplest Environment

For this environment without synchronization elements, the PTwin is emulated with a certain realistic aspect using Mininet, considering a computer equipped with Intel® Xeon® Silver 4514Y and that 512 GB of RAM has been used. The characteristics of the emulation setup can be summarized, including complementary utilities, in Table 4.1, and the relationship between these tools can be seen in Figure 4.1.

Regarding the PTwin topologies, it was considered: 14-nodes NSFNet, 17-nodes GBN (Unviersitat Politècnica da Catalunya, 2025) and 24-nodes GEANT2 (Unviersitat Politècnica da Catalunya, 2025), taking into account different propagation delays and link capacities, in seconds and $\mathrm{Mbits/s}$, respectively. Then, for each topology, different datasets were created that include features influenced by congestion, using a traffic generator based on iPerf (GUEANT,

Figure 4.1 – The tools used to construct the first environment for the experiments



Source: The Author (2025).

Table 4.1 – Emulation setup used to build the PTwin

Tools	Version
Operating system	Ubuntu 20.04.6
ONOS	2.6.0
Mininet	2.2.2
Open vSwitch	2.13.8
OpenFlow	1.0
iPerf	2.2.0

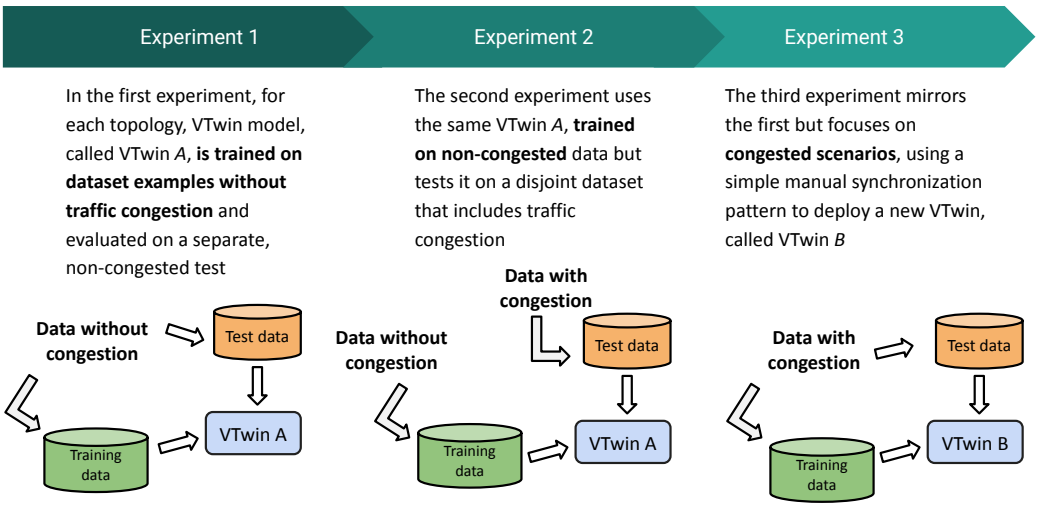
2025), which was used to simulate varying traffic loads from multiple connections. Specifically, for each topology, two distinct datasets were created, each with different characteristics in terms of the number of examples (input-output instances). These datasets are grouped and described in Table 4.2, considering the presence of congestion, the number of training and testing examples, and the number of flows per connection, for each topology. Congestion in these datasets is induced by increasing the number of flows, similar to simulating a rise in the number of applications competing for the same network resources, such as the bandwidth of the link along a path. Each flow is configured to generate deterministic UDP traffic. Specifically, flows in the NSFNet and GBN topologies are assigned a throughput of $15\ \mathrm{Mbits/s}$, while those in the GEANT2 topology use $25\ \mathrm{Mbits/s}$. The server-client pairs are randomly selected to emulate diverse application traffic with uniform bandwidth demands.

Considering these six datasets detailed in Table 4.2, three experiments were proposed to evaluate the generalization capabilities of the VTwin model under both congested and non-congested traffic scenarios, as summarized by Figure 4.2.

Table 4.2 – Traffic datasets used to evaluate the VTwin performance without and under unstable network conditions

#	Traffic congestion	# Examples train/test	Flows per connection	Topology
1	✗	360 000/60 000	5	NSFNet
2	✓	788 000/131 400	12	NSFNet
3	✗	360 000/60 000	4	GBN
4	✓	503 000/84 000	7	GBN
5	✗	156 000/26 000	2	GEANT2
6	✓	311 000/52 000	4	GEANT2

Figure 4.2 – Set of experiments used to evaluate the performance of different VTwin models in a scenario with and without traffic congestion



Source: The Author (2025).

In the first experiment, for each topology, the VTwin model, called VTwin *A*, is trained on dataset examples without traffic congestion and evaluated on a separate, non-congested test set to measure its generalization performance. The second experiment uses the same VTwin *A*, trained on non-congested data but tests it on a disjoint dataset that includes traffic congestion, allowing us to assess how well the model generalizes to unseen congestion conditions. The third experiment mirrors the first but focuses on congested scenarios, using a simple manual synchronization pattern (ALGHAMDI; ALBASSAM, 2024). To update the VTwin a new model is trained, called VTwin *B*, on data with traffic congestion and evaluated on a disjoint test set that also includes congestion to analyze generalization within congested environments.

To evaluate the performance of the VTwin prediction over the proposed experiments in

this environment, we used the NMSE and R^2 (coefficient of determination), which are expressed by equations

$$\text{NMSE} = 10\log_{10}\left\{\frac{\sum_{t=1}^{T}\left(\mathcal{D}_t - \hat{\mathcal{D}}_t\right)^2}{\sum_{t=1}^{T}\mathcal{D}_t^2}\right\}, \tag{4.1}$$

and

$$R^2 = 1 - \frac{\sum_{t=1}^{T}\left(\mathcal{D}_t - \hat{\mathcal{D}}_t\right)^2}{\sum_{t=1}^{T}\left(\mathcal{D}_t - \overline{\mathcal{D}}\right)^2}, \tag{4.2}$$

where $\hat{\mathcal{D}}_t$, $\mathcal{D}_t$ and $\overline{\mathcal{D}}_t$ represent the t-th estimated total per-flow delay, the corresponding ground truth total per-flow delay, and the average ground truth total delay, respectively, across a total of T flows in the test dataset. To compute the average, a window size of 100 consecutive examples over time was used. While other window sizes could be adopted, using 100 samples strikes a balance: it yields a sufficiently narrow confidence interval for the mean and enhances sensitivity in terms of NMSE to short-term variations, particularly important in the presence of a sudden concept drift.

For the VTwin training, TensorFlow 2.14.0 was used and consistency was maintained in all training parameters across all experiments. Specifically, a learning rate of 0.001 was used, trained with 50 epochs, a batch size of 200, applying Z-score normalization for feature scaling, adopting the *Adam* optimizer, and using Mean Absolute Percentage Error (MAPE) as the loss function. Moreover, the model hyperparameters used were: $T = 12$ and $||\mathbf{g}_{f_t}|| = ||\mathbf{g}_{e_j}|| = 16$. This uniform configuration was chosen to ensure that any observed differences in model performance could be attributed solely to variations in the input data. In this sense, the input data used in all experiments for this environment is defined by Table 4.3.

4.1.1 Main results

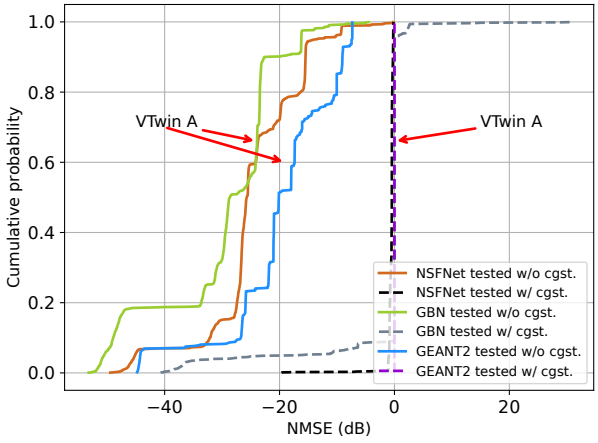
In the first experiment, when traffic congestion is not considered, an expected result was obtained in all topologies, with the best NMSE and R^2, close to -17.5 dB and 0.95, respectively, reached in the NSFNet topology. The VTwin *A* performance worsens in the second experiment when this model is used to generalize datasets that include traffic congestion. The model yields a NMSE, in the worst case, of approximately 3.6 dB and a negative R^2. This worsening in performance is explained by the change in feature distribution (and consequently

Table 4.3 – Flow and link features that are used as input feature for training and testing the VTwin model

Feature	Unit	Type
Average traffic rate	bits/s	Flow
Path propagation delay	s	Flow
Flow length	—	Flow
Average packet rate	packets/s	Flow
Average packet sent	—	Flow
Average packet loss	packets/s	Flow
Capacity	bits/s	Link
Link load	%	Link

in the delay distribution) of the testing dataset, since the congestion situation leads to an overflow of the switches' buffers, causing an increase in the delay metrics' magnitude. Therefore, as the VTwin *A* is not ready to deal with a delay of that magnitude (100 times greater), retraining with new data is necessary. In numerical terms, this performance difference is shown in Table 4.4. Moreover, in terms of Cumulative Distribution Function (CDF), Figure 4.3 describes the VTwin *A* performance considering both test scenarios.

Figure 4.3 – CDF comparison of VTwin *A* performance in experiments 1 and 2. Both experiments used a model trained without traffic congestion. Then, to evaluate the generalization performance, it was considered traffic with and without traffic congestion

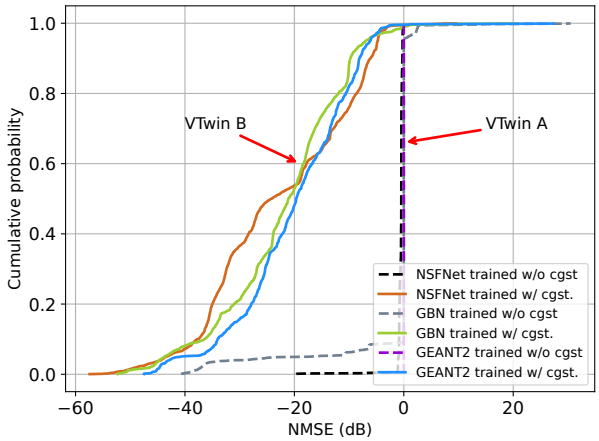


Source: The Author (2025).

Considering the results from experiment 2 as a baseline, the last experiment converges to the expected performance when the training process uses data with traffic congestion, after the manual synchronization. The VTwin *B* reaches the best NMSE and R^2, also in the NSFNet topology, with values close to -10.7 dB and 0.323, respectively. This result can be attributed to

the dataset used for model retraining, which represented a congested environment. As a result, the model adjusted its weights accordingly, yielding a performance, in terms of the CDF, that resembles the first experiment, as illustrated in Figure 4.4 and numerically described in Table 4.4.

Figure 4.4 – CDF comparison of VTwin performance in experiments 2 and 3. Considering the experiment 2 as a baseline, it was trained the VTwin *B* on the same topologies, but this time under traffic congestion



Source: The Author (2025).

Table 4.4 – NMSE and R^2 metrics obtained in the testing dataset for each experiment

#	Trained with congest.	Tested with congest.	Topology	NMSE (dB)	R^2
1	✗	✗	NSFNet	−17.532	0.956
			GBN	−22.031	0.927
			GEANT2	−14.120	0.886
2	✗	✓	NSFNet	−0.551	−4.351
			GBN	3.610	−11.924
			GEANT2	0.001	−4.052
3	✓	✓	NSFNet	−10.775	0.323
			GBN	−7.857	0.804
			GEANT2	−2.001	0.698

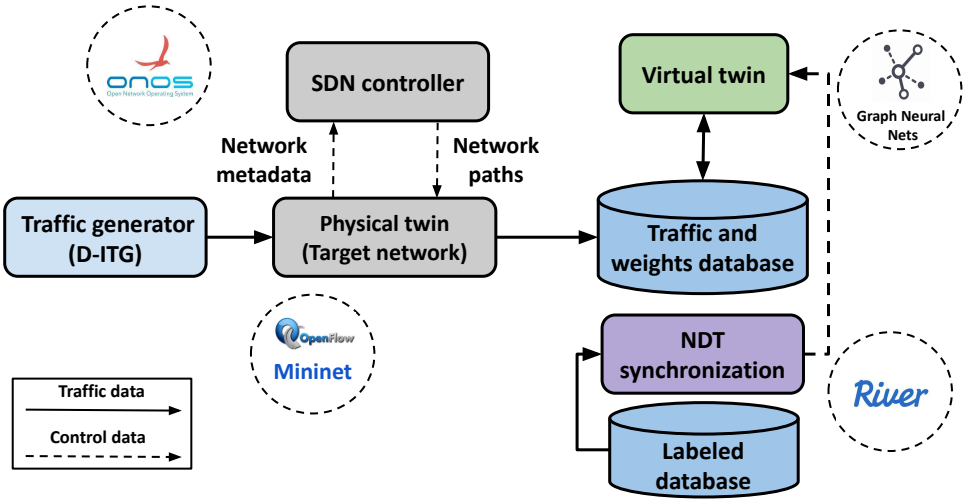
The results obtained in this initial environment introduced key terminologies—some even related to manual processes—that are consistently used in the experiments of this environment, such as model training and versioning. Moreover, the results highlighted the significant impact of concept drift on the performance of the VTwin model, even under a simple traffic pattern with limited variability generated by iPerf. Consequently, the subsequent experiments adopt a more

challenging environment, featuring a more robust traffic generation setup, to further validate the proposed architecture.

4.2 Robust Environment

This more robust environment was created with the objective of being more challenging than the previous one, in the validation of the proposed architecture. The main elements of the environment remain the same; the PTwin framework integrates both emulation and simulation components using Mininet, also with the ONOS controller, and the simulation component handles more robust traffic generation, using different probability distributions. The main difference is the use of the NDT synchronization module, as depicted in Figure 4.5.

Figure 4.5 – The tools employed to build a more robust experimental environment



Source: The Author (2025).

Therefore, for the emulated network topologies, three transport networks were used: the PASSION project topology (MORAIS et al., 2023), the 5G-Crosshaul topology (MORAIS et al., 2023), and the *Germany*-50 topology (Unviersitat Politècnica da Catalunya, 2025), comprising networks with 128, 51, and 50 nodes, respectively, and each with 224, 88, and 63 links. Moreover, it was employed Distributed Internet Traffic Generator (D-ITG) using a client-server model over the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) protocols to generate realistic traffic patterns. To do so, the computer used was equipped with Intel® Core™ i7-10700F CPU @ 2.90 GHz, 128 GB of RAM, and a storage capacity of 9 TB.

This setup enables the incorporation of various traffic distributions within an emulation environment that is orchestrated using a suite of tools, summarized in Table 4.5, to ensure accurate and reproducible testing conditions.

Table 4.5 – Setup configuration used to build the PTwin

Tools	Version
Operating system	Ubuntu 20.04
ONOS	2.6.0
Mininet	2.3.0
Open vSwitch	2.13.8
OpenFlow	1.0
D-ITG	2.8.1

In terms of traffic generation, without loss of generality, different packet sizes (in bytes) and average traffic rate (in $\mathrm{mbits/s}$) distributions were used to model internet traffic (JURKIEWICZ; RZYM; BORYŁO, 2021; LUO; MARIN, 2005), such as *Poisson*, *Exponential*, *Uniform*, and Deterministic, with the presence of congestion. The parameters for each one are defined in Table 4.6, considering the minimum and maximum values, $\mathcal{U}(512, 1024)$, for Uniform distribution; the mean for Exponential, $\mathcal{E}(1024)$ and $\mathcal{P}(2048)$ for a Poisson distribution; and a deterministic pattern with traffic congestion defined by a constant value $K = 512$ for a constant packet size in traffic generation.

Table 4.6 – Packet size distributions used to simulate user traffic

Traffic pattern	Protocol	Packet size parameters (bytes)
Uniform	TCP	$\mathcal{U}(512, 1024)$
Exponential	TCP	$\mathcal{E}(1024)$
Poisson	TCP	$\mathcal{P}(2048)$
Deterministic	UDP	$K = 512$

The datasets created from this traffic generation, for a possible retraining process, are featured in Table 4.7, characterized by the four distribution patterns of packet size and traffic rate, the type of topology, and the number of training examples. The training examples in this case refer to the number of samples that are to be used as input-output in the model retraining.

Considering these emulated environments into account, we propose an experiment using topologies that experience a sudden concept drift in traffic characteristics. We introduced an

Table 4.7 – Type of labeled datasets stored used to retraining the VTwin

#	Traffic pattern	Topology	Training examples
1	Exponential	5G-Crosshaul	121 400
		Germany	129 600
		PASSION	129 600
2	Poisson	5G-Crosshaul	234 501
		Germany	257 401
		PASSION	243 001
3	Uniform	5G-Crosshaul	347 002
		Germany	378 802
		PASSION	355 002
4	Deterministic	5G-Crosshaul	447 803
		Germany	440 003
		PASSION	478 203

artificial drift at specific time intervals by altering the packet size distribution. As a result, the traffic follows the sequence of packet size and rate patterns as described below:

$$\text{Start} \rightarrow \boxed{\mathcal{E}} \rightarrow \boxed{\mathcal{P}} \rightarrow \boxed{\mathcal{U}} \rightarrow \boxed{K} \rightarrow \text{End}$$

For each topology, we analyzed different time intervals of network traffic. Specifically, the network traffic duration in seconds for 5G-Crosshaul, Germany, and PASSION are 25,103 s, 24,603 s, and 26,703 s, respectively. Moreover, we conducted the experiment considering that the VTwin model was trained using traffic flows characterized by an exponential packet size and rate distribution. The first sudden drift is introduced by switching this exponential traffic pattern distribution to a Poisson model. This changing distribution behavior persists until the final phase, where the traffic pattern transitions to a constant packet size distribution in the presence of traffic congestion. Finally, to evaluate performance throughout these concept drifts, the NMSE was also considered as the evaluation metric, and a window of 100 examples was taken into account to compute this metric.

Using this environment, an experiment with a distribution-based detector was presented, aimed at detecting the input changes. The concept drift detector uses the KSWIN method, as implemented by the *river* library (MONTIEL et al., 2021). To work properly, this method takes three hyperparameters: the *significance* level, a sensitive parameter that controls the probability of false positives, represented by a value of 0.001, a window size, and a buffer for the most

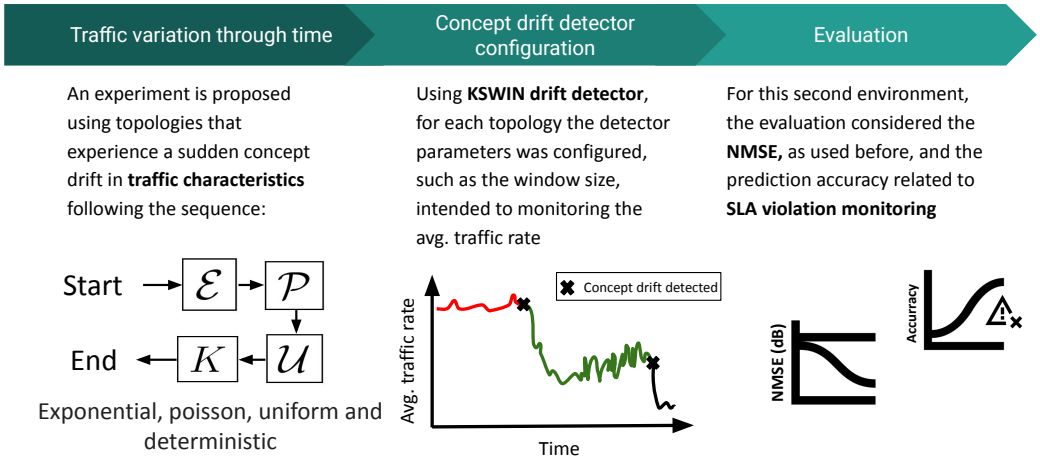
recent samples. The sizes of these components vary across different topologies and were determined empirically in this dissertation. Furthermore, for the VTwin training, similar parameters were used as in the first environment. The only difference is the batch size equal to 100. Moreover, the model hyperparameters remain the same, using: $T = 12$ and $||\mathbf{g}_{f_t}|| = ||\mathbf{g}_{e_j}|| = 16$. Regarding the data used, since a different traffic generator was assumed, some features are not available as provided by iPerf in the first environment. Thus, the input features used are described by Table 4.8.

Table 4.8 – Flow and link features that are extracted to be used to training and testing the VTwin model

Feature	Unit	Type
Average traffic rate	bits/s	Flow
Path propagation delay	s	Flow
Flow length	—	Flow
Average packet sent	—	Flow
Average packet loss	packets/s	Flow
Capacity	bits/s	Link
Link load	%	Link

In summary, the main process to construct the proposed experiment for this environment is depicted in Figure 4.6.

Figure 4.6 – Experiment organization to evaluate the VTwin performance in a scenario with concept drift inducted by different traffic pattern

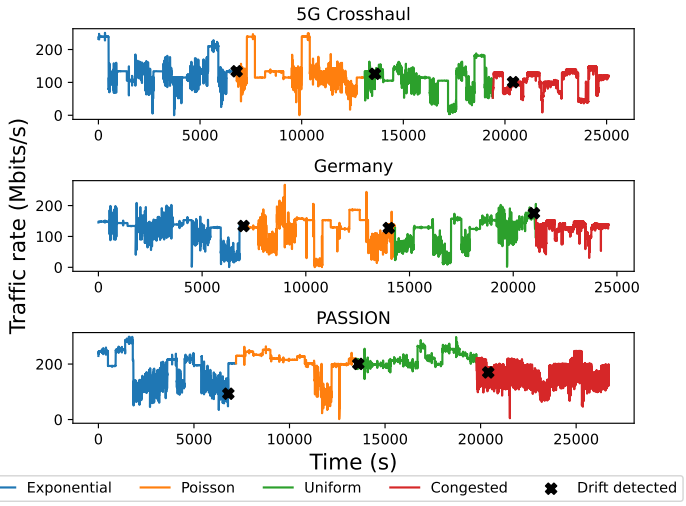


Source: The Author (2025).

4.2.1 Main results

From the proposed experimental organization, depicted in Figure 4.6, it was proposed an experiment to evaluate the proposed synchronization elements with a distribution-based concept drift detector, considering the three emulated topologies and the average traffic rate in $\mathrm{Mbits/s}$ as the feature to be tracked. Figure 4.7 illustrates the operation of the KSWIN detector and the location of each detected drift. For each network topology, a specific window size (in seconds) was selected: 6800 for 5G-Crosshaul, 7000 for Germany, and 6800 for the PASSION topology. Although alternative window sizes could be explored, determining the optimal size to minimize false positives falls outside the scope of this dissertation. Nevertheless, in the evaluated environments, smaller window sizes generally result in a higher likelihood of false positives, as illustrated in Figure 4.8, which analyzes various window sizes in all topologies. The curves in this figure exhibit a consistent pattern and converge in the number of detected concept drifts as the window size increases. This behavior can be attributed to the timing of the concept drift events. For example, the first concept drift, shown in Figure 4.7, occurs at approximately 7000 s across all topologies, coincident with the point at which the curves in Figure 4.8 end converging.

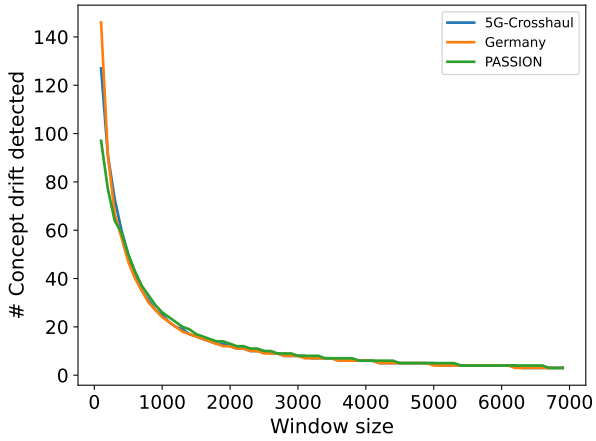
Figure 4.7 – Drift detection pattern considering 5G-Crosshaul, Germany and PASSION topologies and four type of traffic patterns. In this case, we considered the average traffic rate (in $\mathrm{Mbits/s}$ for better readability) of one flow per second



Source: The Author (2025).

In the first experiment, it is possible to note that the concept drift was effectively detected in all existing deviations due to the change of the traffic pattern. However, there were different

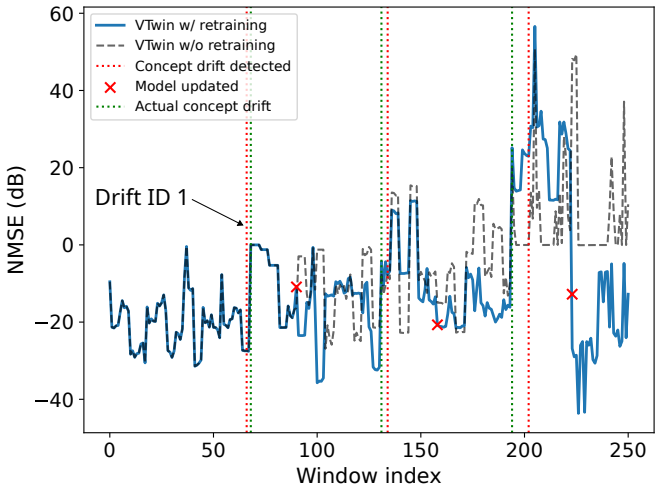
Figure 4.8 – Relationship between the window size and the number of concept drift detection on the three topologies



Source: The Author (2025).

delays considering the time when the concept drift occurred and when the detection occurred. Now, considering the effects of these delayed drift detections, Figure 4.9 depicts the VTwin performance over time considering the proposed synchronization elements in comparison with the same VTwin without them, in terms of NMSE.

Figure 4.9 – Validating proposed architecture on 5G-Crosshaul topology. The blue line represents the VTwin performance with the proposed twinning elements, while the gray dashed line represents the performance without it



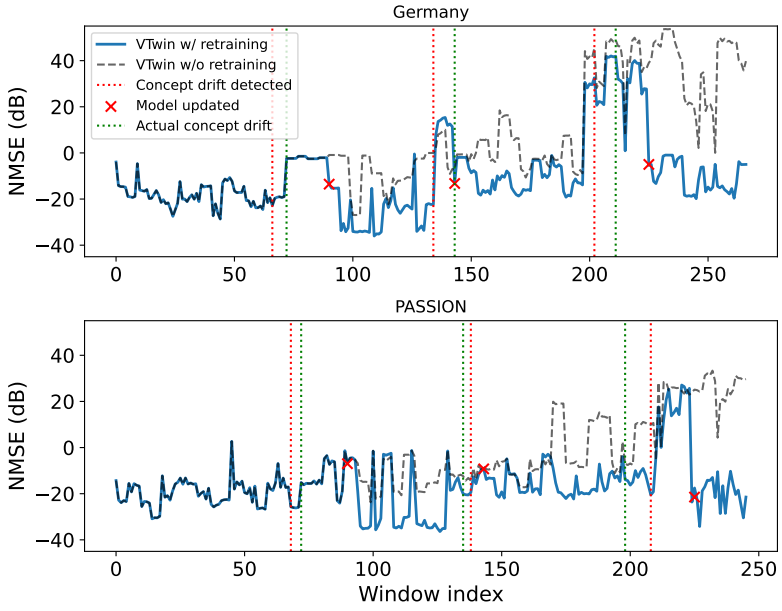
Source: The Author (2025).

Before the first concept drift is detected, the VTwin operates in a typical manner, as seen in the related work: a GNN model already trained to generalize to unseen datasets and, based on its predictions, performs the required analysis within the proposed application. However, after

the first concept drift occurs, the scenario changes due to traffic variability, leading to model degradation, evidenced by some sample windows exhibiting NMSE values near 0 dB. Despite this decline, the VTwin model remains stable, even without retraining, as the underlying data distributions, such as Poisson and Exponential, are still related. This behavior changes after the second concept drift, as the input data distribution changes to a packet size with a distinct pattern, requiring retraining, which is evident in the accentuated NMSE difference between the model with and without retraining. In the last concept drift, the VTwin becomes obsolete without a retraining process as the input data distribution shifts to a UDP connection characterized by constant packet size and the absence of congestion control. At this point, retraining becomes extremely necessary, as demonstrated by the pronounced performance gap between the models with and without retraining.

The results for the other two topologies aim to validate the purpose synchronization method considering topologies with different characteristics, for instance, in the PASSION topology that has 128 nodes and three times more links than the 5G-crosshaul. In this way, Figure 4.10 also shows consistent results in both topologies as those observed in the 5G-Crosshaul topology, specifically in the last drift detected by the concept drift detector, highlighting the same observations related to mandatory retraining due to poor VTwin prediction performance.

Figure 4.10 – Evaluation performance of proposed architecture considering Germany (upper) and PASSION topologies (bottom)



Source: The Author (2025).

Finally, Table 4.9 summarizes the average NMSE across all window examples, highlighting VTwin's performance before and after each concept drift. Incorporating the NDT synchronization module results in a performance improvement of at least 56.7% in predicting compared to configurations without it. Furthermore, given an empirical goal of achieving an average NMSE below -10 dB, architectures without NDT synchronization meet this target in only two of the nine concept drifts, across all evaluated topologies. In contrast, the proposed approach, which incorporates NDT synchronization, achieves this performance in five out of nine cases following concept drift. Thus, from a general perspective, for all topologies and after all concept drifts, the accuracy in the prediction was improved, thus highlighting the importance of twin synchronization in production environments where concept drift is pervasive.

Table 4.9 – NMSE performance in dB before and after concept drift, considering the NDT operation with and without the proposed synchronization module implementation

Topology	Drift ID	Pre-drift avg. NMSE w/o sync.	Post-drift avg. NMSE w/o sync.	Pre-drift avg. NMSE w/ sync.	Post-drift avg. NMSE w/ sync.
5G-Crosshaul	1	-20.72	-10.84	-20.72	$\mathbf{-14.47}$
	2	-10.84	-5.42	-14.47	$\mathbf{-11.22}$
	3	-5.42	7.78	-11.22	$\mathbf{1.43}$
Germany	1	-19.98	-12.11	-19.98	$\mathbf{-20.10}$
	2	-12.11	-1.50	-20.10	$\mathbf{-14.85}$
	3	-1.50	25.10	-14.85	$\mathbf{-5.73}$
PASSION	1	-18.44	-6.62	-18.44	$\mathbf{-19.17}$
	2	-6.62	0.47	-19.17	$\mathbf{-8.04}$
	3	0.47	38.70	-8.04	$\mathbf{5.83}$

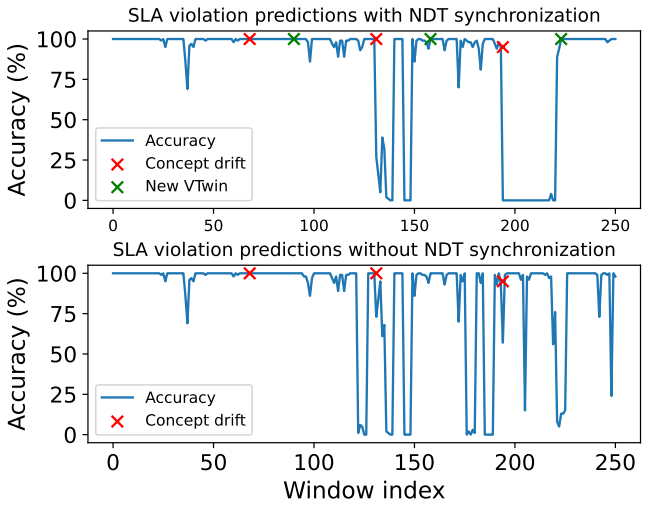
4.2.2 SLA monitoring use case

The use case scenario used to evaluate the proposed NDT synchronization, considering a high-level metric related to SLA monitoring tasks, was carried out using only data coming from 5G-crosshaul topology. In this sense, we evaluated the same scenario presented in Figure 4.9, but considering high-level metrics such as the accuracy in the SLA violation predictions. Thus, we assume that each flow has a specific PDB value that was assigned empirically, based on the characteristics of the 5G-crosshaul topology, such as the propagation delay of each link.

Regarding the experiments, we consider a total of $25\,100$ flows traversing the network, each of which must be classified by the SLA monitoring application as compliant or in vio-

lation with the PDB, based on the delay predicted by the VTwin. The SLA violation in this case is characterized when a per-flow delay exceeds the PDB for a given path. From the total of flows considered in this experiment, 536 flows are violating with their SLA (ground truth). Hence, the accuracy in the prediction of SLA violations is shown in Figure 4.11, which considers the VTwin with the NDT synchronization module and without it, and each window index corresponds to the classification of 100 flows, for a better visualization of the results in this figure.

Figure 4.11 – Accuracy of SLA violation predictions in scenario without (bottom) and with twin synchronization (upper)



Source: The Author (2025).

As shown in Figure 4.11, the application exhibits the expected degradation in classification performance when the NDT synchronization module is disabled, due to the VTwin states not being updated with the PTwin states after a concept drift. Quantitatively, this leads to the misclassification of $7,744$ flows that violate their respective SLAs. In contrast, enabling the NDT synchronization module significantly reduces the number of misclassified flows to $3,333$. It is important to note that part of these SLA violations also occurs during periods when the VTwin is undergoing retraining. Overall, this experiment highlights that the practical usability of an NDT strongly depends on the presence of an effective synchronization mechanism between the twins, operating at an acceptable rate, which is essential for transitioning the solution from a prototype to a production-ready platform or framework.

Chapter 5

Conclusions

This dissertation proposed an enhanced NDT architecture incorporating data-driven synchronization modules, taking initial steps toward addressing practical and production-level aspects of NDT deployment in transport networks. The proposed modules within this architecture specifically tackle the challenge posed by the non-stationary nature of network traffic over time. In this sense, this architecture aims to ensure synchronization between two twins of the NDT platform. To achieve this, the NDT synchronization modules, such as the concept drift detector, aim to track and detect possible changes in the traffic pattern, triggering a retraining process and ultimately updating the VTwin weights through the VTwin model management.

In Chapter 2, we presented the fundamental theoretical concepts of the SDN architecture, discussing its main layers and functionalities, which are essential for understanding the key components of an NDT architecture. Building on this foundation, we conducted an in-depth analysis of the NDT specifications, addressing its core elements, the role and contributions of standardization bodies such as the IETF and ITU-T, and concluding with a classification of the relationships between the physical and virtual counterparts, which depend on the level of integration between them. Finally, we conclude this chapter by detailing the mathematical foundations of GNN models and concept drift detection techniques, which were crucial elements of this dissertation.

In Chapter 3, we described the main components of the proposed architecture from a bottom-up perspective, starting with the PTwin and concluding with the high-level application layer and its submodules. Between these layers, we introduced the NDT synchronization module, responsible for updating the state of the VTwin based on changes observed in the PTwin. To achieve this, the architecture incorporates a concept drift detection submodule, which signals

when the input feature distribution undergoes significant changes. This mechanism is designed to trigger the retraining process, ensuring the VTwin remains properly aligned with its physical counterpart, by the VTwin model management.

Finally, we presented a set of experiments to evaluate the architecture's operation considering two network environments. In the first environment, used as an exploratory analysis, we proposed a setup with a simple traffic generator that only had a constant bitrate traffic pattern and only one traffic variability, induced by traffic congestion. The results from that showed poor performance in all tested topologies when we tried to generalize a test dataset with traffic congestion without properly training the VTwin for that. In possession of these preliminary results, a more challenging environment was used to evaluate the proposed architecture, using three different topologies and traffic flows with four packet size distributions, generating different traffic patterns. Finally, to detect and start the retraining process, a distribution-based approach was considered, which was based on the KSWIN method. Our results demonstrated the importance of a synchronization process between the VTwin and PTwin, achieving a performance improvement of at least 56.7% in prediction compared to a configuration without a concept drift detector. After this achievement, aimed to obtain insight into the impact of the proposed NDT architecture at a high level, we also evaluated the NDT synchronization considering an application related to SLA monitoring.

5.1 Future Works

The results of the proposed experiments offer valuable insights into the challenges of twin synchronization, positioning this dissertation as a foundational step toward future developments. A key direction for future work involves progressively incorporating more realistic elements into the experimental environments. One major limitation is the gap between the current setup and real-world production environments. Although the proposed module presents a promising approach to addressing specific challenges faced by NDT systems, it does not fully resolve all the issues required for the complete feasibility of the NDT platform. For instance, while the data management module includes essential mechanisms for collecting information from the PTwin, the experiments rely on traffic generated by a traffic generator, with metrics directly obtained from simulator logs. In contrast, data collection in a production environment is significantly more complex and may require integration with auxiliary tools, such as those

defined by protocols such as the Simple Network Management Protocol (SNMP) also used to collect network metrics. Furthermore, the feasibility of the proposed synchronization method also depends on the ability to generate synthetic data to retrain the VTwin model when necessary, which is not addressed in this dissertation. In this context, the use of generative AI emerges as a promising solution and represents a potential avenue for future research.

In addition to the data management tasks associated with an NDT, future research will also focus on improving the synchronization method itself, which can be pursued along two main directions. In the short term, the goal is to evaluate synchronization performance in variable transport network environments using a more robust multivariate concept drift detection approach, one that simultaneously considers multiple input features such as average traffic rate, per-link load, and link capacity. In the long term, we aim to develop a more proactive synchronization strategy by integrating online learning techniques. This would reduce the latency in adapting the VTwin model parameters without waiting for an explicit concept drift detection, thereby minimizing the duration during which outdated model weights are in effect.

Bibliography

3GPP. *3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Management and orchestration; Study on Management Aspect of Network Digital Twin (Release 19)*. [S.l.], 2024.

ABEN-ATHAR, R. et al. *Network Digital Twin for Route Optimization in 5G/B5G Transport Slicing With What-If Analysis*. [S.l.]: IEEE International Conference on Communications, 2025.

AFONSO, B. K. d. A.; BERTON, L. Qt-routenet: Improved gnn generalization to larger 5g networks by fine-tuning predictions from queueing theory. *ITU Journal on Future and Evolving Technologies*, v. 3, p. 49–56, 2022.

ALGHAMDI, W.; ALBASSAM, E. Synchronization Patterns for Digital Twin Systems. *Journal of Applied Data Sciences*, v. 5, n. 3, p. 1026–1037, 2024.

ALMASAN, P. et al. Network Digital Twin: Context, Enabling Technologies, and Opportunities. *IEEE Communications Magazine*, v. 60, n. 11, p. 22–27, 2022.

ALMASAN, P. et al. Deep Reinforcement Learning Meets Graph Neural Networks: Exploring a Routing Optimization Use Case. *Comput. Commun.*, Elsevier Science Publishers B. V., NLD, v. 196, n. C, p. 184–194, dez. 2022. ISSN 0140-3664.

AMEUR, M.; BRIKAUTHORREFMARK, B.; KSENTINI, A. Dual Self-Attention is what You Need for Model Drift Detection in 6G Networks. *IEEE Transactions on Machine Learning in Communications and Networking*, p. 1–1, 2025.

BAIER, L. et al. Detecting Concept Drift with Neural Network Model Uncertainty. In: BUI, T. (Ed.). *Proceedings of the 56th Hawaii International Conference on System Sciences*. [S.l.: s.n.], 2023. p. 835 – 844. ISBN 978-099813316-4.

BANNOUR, F.; SOUIHI, S.; MELLOUK, A. Distributed SDN Control: Survey, Taxonomy, and Challenges. *IEEE Communications Surveys & Tutorials*, v. 20, n. 1, p. 333–354, 2018.

BARIAH, L.; DEBBAH, M. The Interplay of AI and Digital Twin: Bridging the Gap Between Data-Driven and Model-Driven Approaches. *IEEE Wireless Communications*, IEEE, 2024.

BARRICELLI, B. R.; CASIRAGHI, E.; FOGLI, D. A Survey on Digital Twin: Definitions, Characteristics, Applications, and Design Implications. *IEEE Access*, v. 7, p. 167653–167671, 2019.

BAYRAM, F.; AHMED, B. S.; KASSLER, A. From concept drift to model degradation: An overview on performance-aware drift detectors. *Knowledge-Based Systems*, v. 245, p. 108632, 2022. ISSN 0950-7051.

BORGES, J. et al. CAVIAR: Co-Simulation of 6G Communications, 3-D Scenarios, and AI for Digital Twins. *IEEE Internet of Things Journal*, v. 11, n. 19, p. 31287–31300, 2024.

BORGES, J. P. T. et al. Hybrid CAVIAR Simulations and Reinforcement Learning Applied to 5G Systems: Experiments with Scheduling and Beam Selection. Universidade Federal do Pará, 2022.

BRONSTEIN, M. M. et al. Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges. *ArXiv*, abs/2104.13478, 2021.

ETSI. *ETSI GR ZSM 015 V1.1.1: Zero-touch network and Service Management (ZSM); Network Digital Twin*. 2024.

FARHADY, H.; LEE, H.; NAKAO, A. Software-Defined Networking: A Survey. *Computer Networks*, v. 81, p. 79–95, 2015. ISSN 1389-1286.

FARRERAS, M. et al. GNNetSlice: A GNN-based Performance Model to Support Network Slicing in B5G Networks. *Comput. Commun.*, Elsevier Science Publishers B. V., v. 232, n. C, mar. 2025. ISSN 0140-3664.

FARRERAS, M. et al. Improving Network Delay Predictions Using GNNs. *Journal of Network and System Management*, Springer Science and Business Media LLC, v. 31, n. 4, out. 2023.

FERRIOL-GALMÉS, M. et al. FlowDT: A Flow-Aware Digital Twin for Computer Networks. In: IEEE. *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. [S.l.], 2022. p. 8907–8911.

FERRIOL-GALMÉS, M. et al. RouteNet-Fermi: Network Modeling With Graph Neural Networks. *IEEE/ACM Transactions on Networking*, v. 31, n. 6, p. 3080–3095, 2023.

FERRIOL-GALMÉS, M. et al. Routenet-erlang: A graph neural network for network performance evaluation. In: *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*. [S.l.: s.n.], 2022. p. 2018–2027.

GILMER, J. et al. Neural message passing for Quantum chemistry. In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. [S.l.]: JMLR.org, 2017. (ICML'17), p. 1263–1272.

GRIEVES, M.; VICKERS, J. Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems. In: ____. *Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches*. Cham: Springer International Publishing, 2017. p. 85–113. ISBN 978-3-319-38756-7.

GUEANT, V. *iPerf - iPerf3 and iPerf2 user documentation — iperf.fr*. 2025. ⟨https://iperf.fr/iperf-doc.php⟩. [Accessed 10-04-2025].

GÜEMES-PALAU, C. et al. Accelerating Deep Reinforcement Learning for Digital Twin Network Optimization with Evolutionary Strategies. In: *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*. [S.l.: s.n.], 2022. p. 1–5.

HAPP, M. et al. Graph-neural-network-based Delay Estimation for Communication Networks with Heterogeneous Scheduling Policies. *ITU Journal on Future and Evolving Technologies*, v. 2, n. 4, p. 1–8, 2021.

HENDERSON, T. R. et al. Network Simulations With the ns-3 Simulator. *SIGCOMM demonstration*, v. 14, n. 14, p. 527, 2008.

IWASHITA, A. S.; PAPA, J. P. An overview on concept drift learning. *IEEE Access*, v. 7, p. 1532–1547, 2019.

JONES, D. et al. Characterising the Digital Twin: A Systematic Literature Review. *CIRP Journal of Manufacturing Science and Technology*, v. 29, p. 36–52, 2020. ISSN 1755-5817.

JURKIEWICZ, P.; RZYM, G.; BORYŁO, P. Flow Length and Size Distributions in Campus Internet Traffic. *Computer Communications*, v. 167, p. 15–30, 2021. ISSN 0140-3664.

KHAN, S. U. et al. Adaptive Runtime Monitoring of Service Level Agreement Violations in Cloud Computing. *Computers, Materials and Continua*, v. 71, n. 3, p. 4199–4220, 2022. ISSN 1546-2218.

KHERBACHE, M.; MAIMOUR, M.; RONDEAU, E. When Digital Twin Meets Network Softwarization in the Industrial IoT: Real-Time Requirements Case Study. *Sensors*, v. 21, n. 24, 2021. ISSN 1424-8220.

KIPF, T. N.; WELLING, M. Semi-supervised Classification With Graph Convolutional Networks. *arXiv preprint arXiv:1609.02907*, 2016.

KREUTZ, D. et al. Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE*, v. 103, n. 1, p. 14–76, 2015.

KRITZINGER, W. et al. Digital Twin in Manufacturing: A Categorical Literature Review and Classification. *IFAC-PapersOnLine*, v. 51, n. 11, p. 1016–1022, 2018. ISSN 2405-8963. 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018.

LAI, J. et al. Deep Learning Based Traffic Prediction Method for Digital Twin Network. *Cognitive Computation*, Springer, v. 15, n. 5, p. 1748–1766, 2023.

LI, B. et al. Learnable Digital Twin for Efficient Wireless Network Evaluation. In: IEEE. *MILCOM 2023-2023 IEEE Military Communications Conference (MILCOM)*. [S.l.], 2023. p. 661–666.

LI, T. et al. Generative AI Empowered Network Digital Twins: Architecture, Technologies, and Applications. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 57, n. 6, fev. 2025. ISSN 0360-0300.

LIU, W. et al. When Digital Twin Meets 6G: Concepts, Obstacles, and Research Prospects. *IEEE Communications Magazine*, IEEE, 2024.

LU, J. et al. Learning Under Concept Drift: A review. *IEEE transactions on knowledge and data engineering*, IEEE, v. 31, n. 12, p. 2346–2363, 2018.

LUO, S.; MARIN, G. Realistic Internet traffic simulation through mixture modeling and a case study. In: *Proceedings of the Winter Simulation Conference, 2005.* [S.l.: s.n.], 2005. p. 9 pp.–.

MA, W. J. Bayesian decision models: A primer. *Neuron*, v. 104, n. 1, p. 164–175, 2019. ISSN 0896-6273.

MCKEOWN, N. et al. Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, Association for Computing Machinery, New York, NY, USA, v. 38, n. 2, p. 69–74, mar. 2008. ISSN 0146-4833.

MESSAOUDI, S. et al. GNN-Based SDN Admission Control in Beyond 5G Networks. In: IEEE. *GLOBECOM 2023-2023 IEEE Global Communications Conference*. [S.l.], 2023. p. 6103–6108.

Mininet Project Contributors. *Mininet: An Instant Virtual Network on Your Laptop (or Other PC) - Mininet — mininet.org*. 2024. ⟨http://mininet.org/⟩. [Accessed 04-12-2024].

MODESTO, C. et al. Delay Estimation Based on Multiple Stage Message Passing With Attention Mechanism Using a Real Network communication dataset. *ITU J. (Geneva)*, International Telecommunication Union, v. 5, n. 4, p. 465–477, dez. 2024.

MODESTO, C. et al. Accelerating Ray Tracing-Based Wireless Channels Generation for Real-Time Network Digital Twins. *IEEE Open Journal of the Communications Society*, p. 1–1, 2025.

MONTIEL, J. et al. River: Machine Learning for Streaming Data in Python. *Journal of Machine Learning Research*, v. 22, n. 110, p. 1–8, 2021.

MORAIS, F. Z. et al. PlaceRAN: Optimal Placement of Virtualized Network Functions in Beyond 5G Radio Access Networks. *IEEE Transactions on Mobile Computing*, v. 22, n. 9, p. 5434–5448, 2023.

O-RAN ALLIANCE. *Research reports — o-ran.org*. 2024. ⟨https://www.o-ran.org/research-reports⟩. [Accessed 12-12-2024].

Open Networking Foundation. *Open Network Operating System (ONOS) SDN Controller for SDN/NFV Solutions — opennetworking.org*. 2025. ⟨https://opennetworking.org/onos/⟩. [Accessed 12-01-2025].

PEUSTER, M.; KARL, H.; ROSSEM, S. van. Medicine: Rapid prototyping of production-ready network services in multi-pop environments. In: *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. [S.l.: s.n.], 2016. p. 148–153.

RAAB, C.; HEUSINGER, M.; SCHLEIF, F.-M. Reactive soft prototype computing for concept drift streams. *Neurocomputing*, v. 416, p. 340–351, 2020. ISSN 0925-2312.

Recommendation ITU-T Y.3090. *Digital twin network – Requirements and architecture*. [S.l.], 2022.

RUSEK, K. et al. RouteNet: Leveraging Graph Neural Networks for Network Modeling and Optimization in SDN. *IEEE Journal on Selected Areas in Communications*, IEEE, v. 38, n. 10, p. 2260–2270, 2020.

Ryu SDN Framework Community. *Ryu SDN Framework — ryu-sdn.org*. 2017. ⟨https://ryu-sdn.org/⟩. [Accessed 09-07-2025].

SARAVANAN, M.; KUMAR, P. S.; KUMAR, A. R. Enabling Network Digital Twin to Improve QoS Performance in Communication Networks. In: IEEE. *2022 IEEE Smartworld, Ubiquitous Intelligence & Computing, Scalable Computing & Communications, Digital Twin, Privacy Computing, Metaverse, Autonomous & Trusted Vehicles (SmartWorld/UIC/ScalCom/DigitalTwin/PriComp/Meta)*. [S.l.], 2022. p. 2151–2160.

SCARSELLI, F. et al. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, v. 20, n. 1, p. 61–80, 2009.

SEBASTIÃO, R.; FERNANDES, J. M. Supporting the page-hinkley test with empirical mode decomposition for change detection. In: SPRINGER. *International Symposium on Methodologies for Intelligent Systems*. [S.l.], 2017. p. 492–498.

SHIN, H. et al. Network Traffic Prediction Model in a Data-Driven Digital Twin Network Architecture. *Electronics*, MDPI, v. 12, n. 18, p. 3957, 2023.

SOMMERS, J. et al. Multiobjective Monitoring for SLA Compliance. *IEEE/ACM Transactions on Networking*, v. 18, n. 2, p. 652–665, 2010.

SUÁREZ-VARELA, J. et al. Graph Neural Networks for Communication Networks: Context, Use Cases and Opportunities. *IEEE Network*, v. 37, n. 3, p. 146–153, 2023.

Unviersitat Politècnica da Catalunya. *Knowledge-Defined Networking Training Datasets*. 2025. ⟨https://www.knowledgedefinednetworking.org/⟩. [Accessed 09-01-2025].

VARGA, A. The OMNET++ Discrete Event Simulation System. *Proc. ESM'2001*, v. 9, 01 2001.

VELIČKOVIĆ, P. et al. *Graph Attention Networks*. 2018.

WANG, H. et al. A Graph Neural Network-Based Digital Twin for Network Slicing Management. *IEEE Transactions on Industrial Informatics*, v. 18, n. 2, p. 1367–1376, 2022.

WETTE, P. et al. *MaxiNet: Distributed Emulation of Software-Defined Networks*. 2024. ⟨https://maxinet.github.io/⟩. [Accessed 04-12-2024].

YU, P. et al. Digital Twin Driven Service Self-healing With Graph Neural Networks in 6G Edge Networks. *IEEE Journal on Selected Areas in Communications*, IEEE, 2023.

ZALAT, M. et al. A Practical Network Digital Twin for IGP Weight Optimization. In: *2024 20th International Conference on Network and Service Management (CNSM)*. [S.l.: s.n.], 2024. p. 1–7.

ZHOU, C. et al. *Network Digital Twin: Concepts and Reference Architecture*. [S.l.], 2025. Work in Progress. Disponível em: ⟨https://datatracker.ietf.org/doc/draft-irtf-nmrg-network-digital-twin-arch/10/⟩.

ZHU, Q. et al. Auxiliary-Graph-Based Energy-Efficient Traffic Grooming in IP-Over-Fixed/Flex-Grid Optical Networks. *Journal of Lightwave Technology*, v. 39, n. 10, p. 3011–3024, 2021.

ŽLIOBAITĖ, I.; PECHENIZKIY, M.; GAMA, J. An Overview of Concept Drift Applications. *Big data analysis: new algorithms for a new society*, Springer, p. 91–114, 2016.

ÖHLÉN, P. et al. Network Digital Twins – Outlook and Opportunities. *Ericsson Technology Review*, v. 2022, n. 12, p. 2–11, 2022.