6G TWIN

Integrating Network Digital Twinning into Future AI-based 6G Systems

D1.2

Secured, scalable, and distributed data exposure and collection framework (initial)

Document Information	
Grant Agreement N°	101136314
Authors and institutions	Ayşe Sayın (EBY, main editor); All partners in T1.2
Date	24 December 2024
Related WP	WP1 | 6G Architecture Design
Dissemination level	PU | Public, fully open

Co-funded by
the European Union

6GSNS

Document Change History

Version	Date	Author	Description
V0.1	09/07/2024	Ayşe Sayın (EBY)	Document creation and initial ToC.
V0.2	16/11/2024	All partners in T1.2	Initial contributions to the document and first evaluation of the document.
V0.3	03/12/2024	All partners in T1.2	Contributions from all the partners to complete deliverable for first internal revision.
V0.4	16/12/2024	Paola Soto (IMEC), Burkhard Hensel (TUD), Julien Baudouin, Rajarshi Sanyal (PX), Ayat Zaki Hindi (LIST)	Revision and corrections from deliverable partners.
V0.5	18/12/2024	All partners in T1.2	Contributions from all the partners to complete deliverable for first internal revision.
V0.6	20/12/2024	Ayşe Sayın, Ramin Fuladi (EBY), Paola Soto (IMEC)	Editorial revision and merging contributions from all partners.
V1	24/12/2024	Miguel Camelo (IMEC), Sébastien Faye (LIST)	Final submitted version.

Quality Control

	Name	Organisation	Date
Editor:	Ayşe Sayın	EBY	24/12/2024
Peer review 1:	Burkhard Hensel	TUD	09/12/2024
Peer review 2:	Paola Soto	IMEC	06/12/2024
Peer review 3:	Julien Baudouin & Rajarshi Sanyal	PX	16/12/2024
Peer review 4:	Ayat Zaki Hindi	LIST	18/12/2024
Authorised and submitted by (Project Coordinator):	Sébastien Faye	LIST	24/12/2024

Legal Disclaimer

This document is issued within the framework of and for the purpose of the 6G-TWIN project. This project has received funding from the European Union’s Horizon Europe Framework Programme, through the Smart Networks and Services Joint Undertaking under the powers delegated by the European Commission and under Grant Agreement No. 101136314. Opinions expressed and arguments employed herein do not necessarily reflect the official views of the European Commission. Neither the European Commission nor the 6G-TWIN partners bear any responsibility for any use that may be made of the information contained herein. This document and its content are the property of the 6G-TWIN Consortium. All rights relevant to this document are determined by the applicable laws. Access to this document does not grant any right or license on the document or its contents. 6G-TWIN partners may use this document in conformity with the 6G-TWIN Consortium Grant Agreement provisions.

Executive Summary

The rapid evolution of wireless networks is driven by the growing demand for low latency, high-speed, and ultra-reliable connectivity across industries. As 6G networks begin to take shape, they are poised to revolutionize communication, enabling advancements in ultra-reliable communication, augmented reality (AR), Artificial Intelligence (AI)-driven applications, and interconnected ecosystems. However, the increasing complexity of networks and services presents significant challenges in orchestration and management. Network Digital Twins (NDTs) are proposed as a solution to bridge the physical and virtual worlds, facilitating streamlined network orchestration and automated management. This deliverable focuses on creating an architecture capable of real-time data collection, harmonization, and analysis to realize an NDT-enabled 6G network that can optimize network performance, ensure security, and enhance scalability. It should be noted that the present document is an initial version, which will be updated at a later stage of 6G-TWIN through a new deliverable (D1.5).

The framework is developed with scalability and efficiency in mind and is set to be implemented through a TRL 3 prototype. Specifically, this document provides a comprehensive approach to identifying data processing pipelines and efficient collection mechanisms tailored to diverse traffic patterns, protocols, and data formats in heterogeneous environments. It also explores the use of distributed computing and parallel processing techniques, specifying data structures and algorithms to manage an increasing number of devices and data flows without compromising performance. Additionally, the document defines algorithms that ensure availability and dependable performance while maintaining balance and load management across multiple nodes. It also outlines principles to safeguard data security throughout its lifecycle in the NDT-enabled 6G framework. Finally, it offers a foundation for implementing a TRL 3 prototype to facilitate the integration of data-driven mechanisms across multiple domains.

This deliverable is organized into several sections as described below. In Section 2, the 6G-TWIN architecture is enhanced by bringing a functional view focusing on data collection and exposure. The proposed architecture is structured into three primary layers: the Physical Network Layer, which gathers data from network devices and elements; the Network Digital Twin (NDT) Layer, which harmonizes and processes this data for simulation and decision-making; and a Data Communication Bus, which facilitates secure and efficient data transfer across the system. Updates to functional requirements ensure that the framework supports target-driven data collection, efficient multi-destination delivery, lightweight protocols, and distributed data stream processing (DDSP) integration for scalable performance. The Physical Network Layer employs telemetry systems to collect real-time data using protocols such as SNMP, NETCONF, and RESTCONF, ensuring comprehensive visibility into network operations. This data is then processed through a Telemetry Data Layer (TDL), presented in Section 3, where filtering, aggregation, and storage mechanisms prepare it for further analysis and integration into the NDT.

Section 4 introduces the Data Harmonization Layer (DHL), which is critical in transforming diverse telemetry inputs into standardized formats suitable for advanced modeling and simulation. Through robust preprocessing tools and harmonization pipelines, this layer ensures data consistency and readiness for integration. Complementing these efforts is the Data Communication Bus, introduced in Section 5, which supports dynamic routing, prioritization, and error-checking to guarantee seamless interaction between architectural components. Distributed computing frameworks like Apache Kafka and Zenoh are studied regarding their scalability and reliability.

Data management is presented in Section 6 which addresses the key challenges of scalability, privacy, and federated control. The framework ensures secure data handling through encryption, access controls, and privacy-preserving mechanisms, making it resilient to the demands of high-frequency and large-scale data streams. Additionally, the deliverable introduces in Section 7 an initial implementation of the 6G-TWIN framework, showcasing its realization and integration in Open Radio Access Networks (O-RAN) and its support of cross-domain data sharing through advanced North-South and East-West interfaces with Transport Networks (TN). Finally, the deliverable highlights the role of simulations in validating the framework in Section 8. These simulations use real-world data to test the capabilities of NDTs, ensuring accurate performance modeling and optimization under various network conditions. The report concludes with an assessment of the proposed framework, emphasizing its ability to address the complexities of future 6G networks and its potential to serve as a foundation for ongoing innovation in scalable, secure, and distributed network solutions. An Annex in Section 10 complements the data inventory and interface definition of D2.1 with the data sources and interfaces for the edge/cloud computing and the transport domain.

Abbreviations and Acronyms

Abbreviations and acronyms	
ACL	Access Control List
OAM	Administration, And Management
ADT	AI-Driven Telemetry
AM	Alternate-Marking
API	Application Programming Interface
AI	Artificial Intelligence
AR	Augmented Reality
BMP	BGP Monitoring Protocol
BSS	Business Support System
CapEx	Capital Expenditures
CU	Centralized Unit
CCM	CIS Cluster Management
CIR	Container Image Registry
CIS	Container Infrastructure Service
CISM	Container Infrastructure Service Management
CP	Connection Points
CDN	Content Delivery Network
CUPS	Control And User Plane Separation
CIM	Core Information Model
CN	Core Network
CRUD	Create, Read, Update, Delete
DCI	Data Center Interconnect
DME	Data Management and Exposure
DT	Digital Twin
DT-C	Digital Twin Connector
DDPS	Distributed Data Processing Systems
DU	Distributed Unit
DNP	Dynamic Network Probe
EWI	East-West Interface
EON	Elastic Optical Network
EM	Element Management
eMBB	Enhanced Mobile Broadband
E2E	End-To-End
EI	External Enrichment Information
ETL	Extract, Transform and Load
GCP	Google Cloud Platform
gNMI	GRPC Network Management Interface
GTP	GPRS Tunnelling Protocol
HDL	Harmonization Data Layer
IOAM	In Situ OAM
INT	In-Band Network Telemetry
IETF	Internet Engineering Task Force

IoT	Internet Of Thing
ISR	In-Sync Replica
JCAS	Joint Communication and Sensing
KPI	Key Performance Indicator
ML	Machine Learning
MANO	Management And Orchestration
MIB	Management Information Base
MaxFro	Maximum Of Frobenius Norm
MTTR	Mean Time to Recovery
MAC	Medium Access Control
MinPinv	Minimum Of the Pseudoinverse Frobenius Norm
MEC	Multi-Access Edge Computing
MSTP	Multi-Service Transmission Platform
MSCS	Multi-Site Connectivity Service
NDT	Network Digital Twin
NE	Network Elements
NFV	Network Function Virtualization
NFVI	Network Functions Virtualization Infrastructure
NFVO	NFV Orchestrator
NFVI-PoP	NFVI-Point of Presence
Near-RT	Near-Real-Time
NDE	Netflow Data Export
NMS	Network Management System
NR	New Radio
NSI	North-South Interface
NBI	Northbound Interface
O-RAN	Open Radio Access Network
OID	Object Identifier
OM	Object Mapping
OpEx	Operational Expense
OTN	Optical Transport Network
OSS	Operations Support System
PSAMP	Packet Sampling
PTN	Packet Transport Network
PII	Personally Identifiable Information
PNF	Physical Network Function
PDH	Plesiochronous Digital Hierarchy
PBT	Postcard-Based Telemetry
Protobuf	Protocol Buffer
QoS	Quality Of Service
RAN	Radio Access Network
RU	Radio Unit
RIC	RAN Intelligent Controller
RIS	Reconfigurable Intelligent Surface
RPC	Remote Procedure Call
SSH	Secure Shell
SAP	Service Access Points
SMO	Service Management and Orchestration

SLA	Service-Level Agreement
SNMP	Simple Network Management Protocol
SDN	Software-Defined Network
SBI	Southbound Interface
SPAN	Spatial Sampling In-Band Network Telemetry
SO	Specific Objective
SC	Status Counter
SCTP	Stream Control Transmission Protocol
SDH	Synchronous Digital Hierarchy
SONET	Synchronous Optical Network
TRL	Technology Readiness Level
TC	Telemetry Collector
TDL	Telemetry Data Layer
TGW	Telemetry Gateway
TOSCA	Topology And Orchestration Specification for Cloud Application
TAPI	Transport API
TN	Transport Network
T-SDN	Transport Software-Defined Networking
TF	Transparent Forwarding
TCP	Transmission Control Protocol
URLLC	Ultra-Reliable Low-Latency Communications
UDP	User Datagram Protocol
UE	User Equipment
VNFs	Virtual Network Functions
VIM	Virtualized Infrastructure Manager
VoIP	Voice Over IP
WIM	WAN Infrastructure Manager
WDM	Wavelength Division Multiplexing
YANG	Yet Another Next Generation

Table of Contents

Table of Figures

Table of Tables

1 Introduction

The rapid digitization of industries necessitates advancements in network technologies, particularly as we transition towards 6G systems. The 6G-TWIN project addresses this need by proposing an AI-native reference architecture that incorporates Network Digital Twins (NDTs). This chapter outlines the aims and objectives of the 6G-TWIN project, focusing on its mission to develop an advanced network framework. We present the specific objectives of the project, aligning them with those of Deliverable 1.2 (D1.2). This deliverable proposes a secure, scalable, and distributed framework for data exposure and collection within the 6G-TWIN system. Additionally, we highlight how this deliverable connects with other project activities, ensuring a comprehensive and coordinated approach to shaping the future of network technologies.

1.1 Aims and Objectives

The 6G-TWIN project seeks to pioneer the integration of NDT into 6G networks by establishing the foundation for designing, implementing, and validating an AI-native 6G NDT architecture. This deliverable outlines a scalable data framework for monitoring physical network infrastructure, enabling real-time optimization and orchestration in complex 6G scenarios.

1.1.16G-TWIN Objectives

Wireless networks are evolving to serve low latency, high speed, and ultra-reliable communication expected by various industries. 6G networks aim to enable these rapid advancements and take connectivity further through ultra-reliable communication, augmented reality (AR), Artificial Intelligence (AI)-driven applications, and interconnected ecosystems. Meanwhile, it is getting harder to orchestrate, manage, and enable beyond technologies with the fast-expanding networks and fast-increasing services to meet the demands. At this point, NDTs are proposed [1] to integrate the physical and virtual worlds to facilitate smooth network orchestration and automated management solutions.

The 6G-TWIN project aims to shape the future 6G network by integrating NDT. For this purpose, 6G-TWIN plans to provision the required base study for designing, implementing, and validating an AI-native 6G NDT architecture. Additionally, 6G-TWIN aims to be a core mechanism for complex and dynamic 6G network scenarios that need end-to-end, real-time optimization and orchestration.

To achieve its ambition, the 6G-TWIN has been built around several specific objectives:

- Specific Objective 1 (SO1) is central to the project's ambition, promising to design an open, federated, and AI-native network architecture for the imminent 6G landscape. This architectural blueprint is designed to leverage NDTs, empowering intelligent data analytics and real-time decision-making, laying the groundwork for unprecedented network efficiency and performance.
- Moreover, Specific Objective 2 (SO2) underscores the project's commitment to constructing a federated, graph-based NDT capable of accurately representing the

intricate dynamics of highly dynamic and complex network scenarios. By establishing this digital sandbox for network planning, management, and control, 6G-TWIN will enhance operational agility and adaptability.

- Simultaneously, Specific Objective 3 (SO3) drives the project's efforts towards implementing a robust modelling and simulation framework. This framework serves as a cornerstone for accurately portraying networked environments and rigorously testing the functionalities of the envisioned 6G architecture.
- Ultimately, as the culmination of its efforts, 6G-TWIN aims to materialize Specific Objective 4 (SO4) by testing, validating, and demonstrating the transferability of its solutions. Through the development of dynamic demonstrators catering to tele driving and energy efficiency use cases, the project aims to showcase the practical impact of its architectural foundation on real-world network scenarios, heralding a new era of connectivity and innovation.

Embedded within the core of the 6G-TWIN project lies a foundational framework driven by specific objectives aimed at revolutionizing the architecture of future 6G systems.

1.1.2Deliverable Objectives

This deliverable document defines an initial data framework for the collection and exposure processes to enable monitoring of the physical network infrastructure through an NDT-enabled 6G network. Additionally, a TRL 3 prototype is planned to be implemented. This data framework definition is aimed to be scalable and effective. More specifically, this deliverable document and task provides:

- Identify the data processing pipelines and efficient data collection mechanisms to cope with various traffic patterns, protocols, and data formats across the heterogeneous environment.
- Leverage distributed computing and parallel processing techniques besides specifying data structures and related algorithms to handle an increased number of devices, users, and data flow while ensuring reliability without a performance loss.
- Define a basis for the algorithms that ensure availability and dependable performance while balancing and managing the load across multiple nodes.
- List principles to ensure security of the data through its life cycle through NDT-enabled 6G framework.
- Provide a basis to implement a TRL 3 prototype that provides data related mechanisms across multiple domains.

This document will be updated through a new deliverable, D1.5, provided at the end of the project.

1.2 Relation to Other Activities in the Project

Task 1.2 aims to enable successful data processing through network monitoring by providing a secure, scalable, and distributed data exposure/collection framework. The findings of this task are inputs for decision-making processes and closed-loop analytics conducted in Task 1.3 and management and orchestration process in Task 1.4. Furthermore, the data harmonization processes required for Task 2.2 are enabled in this task with the described framework. Additionally, this document extends the deliverable document D2.1, which is `Data governance, privacy and harmonization` in terms of new data types for different domains such as Transport Networks (TN) and edge/cloud computing infrastructure.

1.3 Report Structure

This document includes various attitudes associated with the data exposure and collection for network monitoring in the context of the 6G-TWIN project. Task 1.2 and this deliverable enable secure data management processes for the proposed NDT architecture. The first section introduces the 6G-TWIN project, outlines its goals and objectives, specifies the objectives of this deliverable, connects this document within the overall project, and provides contributions of the related partners.

Section 1 introduces document D1.2, which includes the project objectives and the specific task T1.2. It also provides the document structure and the partners' contributions.

Section 2 introduces the data collection architecture for a scalable and distributed data exposure and collection framework. This section provides key concepts for the final architecture definition considering the different data sources and interfaces. It indicates the proposed data space with the data flow that binds the physical network to the NDTs.

Section 3 defines the telemetry data layer within its data sources and interfaces, including the generic definitions of data types and timescales. It also presents the digital twin connector (DTC) within its functions and interfaces. The standard RAN architectures of 3GPP and O-RAN are examined to create a foundation for the related subsections, including the data collection framework and architecture. For this purpose, cross-domain data sources and interfaces are investigated and integrated within a broader system in this section.

Section 4 explains the data harmonization processes and layers. In this section, we explain and detail how the output of the telemetry system is processed and structured in the required data models through the 6G-TWIN data harmonization layer.

Section 5 describes the communication paradigms and their supportive protocols through the data communication buses to enable data transfer in 6G-TWIN. This section explains data communication by considering data management processes such as filtering, prioritization, and serialization. The interoperability of the data communication protocols, and the performance metrics are also discussed in this section.

In Section 6, the data management system and its functions inside the 6G-TWIN management and orchestration layer are discussed and analysed from the security and privacy aspects. This section first discusses the data ingestion layer that connects all subsystems within the NDT. Then, the threat surfaces of this system are analysed to mitigate the risk exposed.

Section 7 provides the telemetry framework as an NDT connector and lists technical needs for data processing and harmonization through the proposed framework. The section emphasizes the prerequisites for a consistent and logical integration of NDT across the 6G network. For this purpose, a discussion about heterogeneous data source handling and harmonization is conducted in this section.

Section 8 describes the simulation framework on top of these data collection, exposure, and harmonization processes since one of the aims of the data gathering is conducting realistic simulations of the system. The section explains how to set up a simulation with the gathered data that is close to the real world.

Section 9 summarizes and concludes the contributions and findings in this deliverable. It includes the discussion results of the proposed secure, scalable, and distributed data exposure and collection framework to monitor the network. Additionally, this section outlines the objectives of this deliverable while assessing it with the results and findings.

Section 10 (Annex) complements the data inventory and interface definitions of D2.1 with the data sources and interfaces for the edge/cloud computing and the transport domain.

1.4 Contribution of Partners

Table 1 presents the contributions from all the partners to the deliverable.

Table 1 Partner's contributions to the D1.2 deliverable.

Partner	Section(s)	Contributions
EBY	**1**, 2.1, **6**, **9**	Lead editor on the document; responsible for the content in Section 1 (Introduction), 2.1 (Requirements from D1.1 and updates), 6 (data management and exposure), and 9 (conclusions).
IMEC	**2, 3,** 6.1, 7.2, **10 (Annex)**	IMEC led the development and coordination of Section 2 (data collection architecture), Section 3 (telemetry data layer), 6.1 data management (scalability, reliability, and parallelization), Section 7.2 (cross-domain data collection and optimization) and Section 10 (Annex focused on updates on data types and interfaces).
ACC	**7**	Section 7 led ACC (Implementing the 6G-TWIN data collection framework) and author of 7.1.
POLIBA	10	Contributing to Section 10 (updates on data types and interfaces)
LIST	-	Continuous feedback and document review.
UBOU	**5**	Coordination of the Section 4 (Data buses).
PX	-	Continuous feedback and document review.
UBI	**4**	UBI led the Section 4 (harmonization data layer)
TUD	**8**	TUD led and wrote the Section 8 (Data collection from the viewpoint of view simulations)
VIAVI	-	Continuous feedback and document review.

Bold *numbers represent section technical leaders*

2 6G-TWIN Data Exposure and Collection Architecture

Integrating advanced data exposure and collection frameworks is crucial for successfully implementing NDTs in the rapidly evolving telecommunications landscape. In this context, the 6G-TWIN project proposed a unified AI-native architecture with support for NDTs [2], consisting of three distinct layers, as shown in Figure 1.

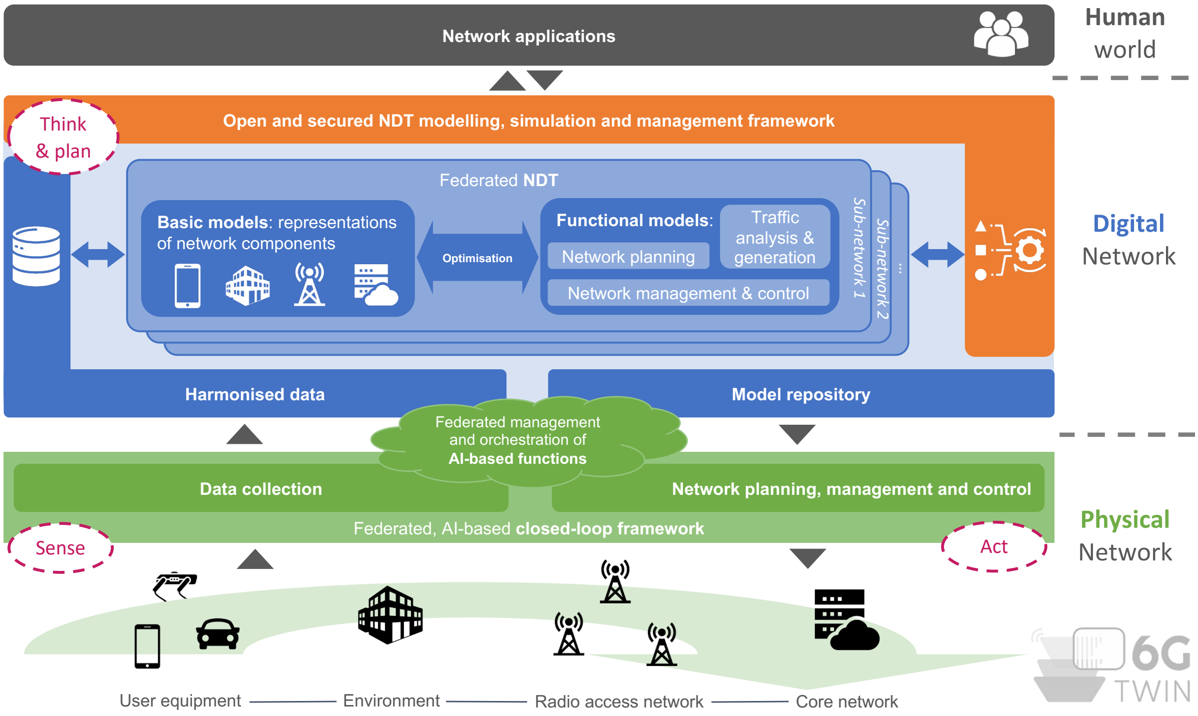

Network applications
Human world
Think & plan
Open and secured NDT modelling, simulation and management framework
Federated NDT
Basic models: representations of network components
Optimisation
Functional models:
Traffic analysis & generation
Network planning
Network management & control
Sub-network 1
Sub-network 2
...
Digital Network
Harmonised data
Model repository
Federated management and orchestration of AI-based functions
Data collection
Network planning, management and control
Federated, AI-based closed-loop framework
Sense
Act
Physical Network
User equipment
Environment
Radio access network
Core network
6G TWIN

Figure 1 High-level architecture of 6G-TWIN

- **Network Applications:** This layer represents the interaction between network applications (e.g., teleoperated driving, energy savings in dense deployments) and the underlying network systems. It connects user-facing services and applications with the network infrastructure, enabling enhanced decision-making, automation, and user-driven actions.
- **Physical Network**: This layer represents the real-world infrastructure, such as user equipment, the radio access network, and the core network. This layer has sensing (collecting data) and acting (executing decisions) capabilities on the physical network.
    - **Data Collection:** Provides the mechanisms for data collection and processing pipelines for monitoring information, including handling diverse data formats, protocols, and traffic patterns; as well as cryptographic methods and privileged access management protocols for enhanced security and privacy. These mechanisms should harness:
        - Parallel processing and distributed computing techniques for scalability, accommodating more users, devices, and data streams without sacrificing performance or reliability.
        - Workload distribution across multiple nodes will also be exploited to ensure optimal performance and availability.

  - **Network planning, management, and control:** Physical devices that can act over the network (e.g., controllers, orchestrators). These elements implement decisions derived from the NDT. It interacts with the physical network for:
    - Executing configurations and optimization strategies generated by the NDT.
    - Feedback loops are provided to the NDT for continuous improvement by controlling the telemetry data layer.
  - **Network elements/devices:** Any element in the network.
- **Network Digital Twin:** This layer contains the Federated NDT, which maintains semantic data and ensures model harmonization. It enables consistent operation of models and data across different domains, manages uncertainty, and provides a unified digital twin graphical representation. Additionally, it includes the federated simulation engine, which manages multiple simulators, both homogeneous and heterogeneous, and delivers a unified view of simulation results via an open API.
  - **Basic Models:** Represent network components such as devices, base stations, and other network entities.
  - **Functional Models:** Models representing key optimization processes, including network planning, traffic analysis, and network management/control.
  - **Data harmonization:** A harmonized data and model repository will serve as the core source of information to populate basic models and will be used as the training set for functional models. In addition, the harmonized repository should be properly governed, such as in a digital space, and include data privacy and sovereignty.

In this section, we provide an update of the architecture designed for the 6G-TWIN project with a focus on the **data collection** (Physical Network layer) and **data harmonization** (NDT layer), which will provide the data collection and exposure framework. We aim to provide the functional enablers to address the complexities of data collection from diverse network sources and to process and harmonize data towards the NDT. The architecture outlined in this section is built upon a foundation of robust (and updated) functional requirements and state-of-the-art reference frameworks while emphasizing the importance of scalability, security, and interoperability, enabling the integration of various data types and formats from both legacy and modern network components.

2.1 Updates on Requirements

During the first stage of the 6G-TWIN architecture definition, several functional requirements regarding data collection were described in D1.1 Section 4.1 [3]. NDTs serve as virtual representations of network elements, processes, or entire systems, enabling simulation, analysis, and optimization across different operational scenarios. The scope of NDTs primarily involves modeling and management of network states, performance metrics, and predictive maintenance strategies. However, in the context of NDT management, the necessity for comprehensive and accurate data collection arises as a fundamental requirement to ensure the fidelity and reliability of the twin models. Recently, standardization bodies such as the Internet Engineering Task Force (IETF) have also started to define such requirements for building NDTs, as outlined in [4]. These efforts are highly important since defining and implementing robust data collection and exposure frameworks for NDTs is crucial for their development and adoption. To complement our initial set of functional requirements, let us introduce additional requirements from the IETF related to the data collection frameworks for NDTs:

- **Target-Driven and On-Demand Collection**: Data collection should be focused on specific operational needs rather than collecting all available data indiscriminately. This approach minimizes resource usage and enhances relevance. To implement it, operators can define parameters for data collection based on current network conditions or specific events, ensuring that only pertinent data is gathered. This is particularly important in large-scale networks where bandwidth and storage are limited.
- **Diverse Tools for Various Data Collection:** The monitoring information necessary for maintaining an NDT varies significantly in its characteristics. Some data, like hardware status and environmental conditions, can be collected less frequently, while other data, such as flow status and link faults, require real-time monitoring. Certain data types, such as device status and port statistics, can be easily gathered using standard tools. In contrast, more complex metrics, like per-flow latency and traffic matrices, necessitate advanced measurement technologies. Consequently, a single uniform data collection method is inadequate; instead, multiple tools and techniques are essential to gather the diverse data needed to construct the NDT effectively.
- **Lightweight and Efficient Collection:** Data collection tools and methods for NDTs should be as lightweight as possible to minimize the impact on network resources and ensure that normal operations are not disrupted.
- **Open and Standardized Interfaces:** Data collection interfaces for building NDTs should be open and standardized to prevent vendor lock-in and ensure interoperability.
- **Naming for Caching:** Effective naming conventions are crucial for managing data efficiently, particularly in caching scenarios. This involves establishing clear and consistent naming conventions for data points to facilitate easy identification and retrieval and implementing caching mechanisms that utilize these naming conventions to improve data access speeds and reduce redundant data collection.
- **Efficient Multi-Destination Delivery**: Data should be capable of being sent to multiple destinations efficiently, ensuring that all relevant systems receive the necessary information without unnecessary duplication. This can be achieved by utilizing multicast communication methods to send data to multiple recipients and implementing systems that intelligently route data to various endpoints based on their specific needs and capabilities.

With these six new requirements, the resulting set of requirements for the data collection framework for NDTs is described in Table 2.

Table 2 Updated Functional requirements regarding data collection.

Requirement ID	Requirement	Description	Status
FR.DC.01	Seamless Integration	The data collection framework shall allow seamless integration of capabilities and data sources across multiple data domains, extending the service-based architecture model to the Cloud-to-Far-Edge continuum. This ensures that data from various sources can be effectively utilized, enhancing the overall functionality of the NDT.	Unchanged
FR.DC.02	Distributed Data Sharing	The framework shall provide a distributed model for sharing and storing data, regulated by access policies. Data will be stored across multiple locations to enhance fault tolerance, improve access speeds, and ensure scalability. This supports efficient data retrieval and resilience against failures.	Unchanged
FR.DC.03	Robust Access Policies	The framework shall enforce robust access policies to control who can access the data, ensuring privacy and security. This is critical for protecting sensitive information and maintaining user trust.	Unchanged
FR.DC.04	Support for Multiple Protocols	The framework shall support multiple communication protocols to ensure compatibility with legacy systems. This allows integration with existing devices and systems that operate on older protocols, facilitating a smoother transition to modern architectures.	Moved as child requirement of FR.DC.09
FR.DC.05	Harmonized Data Formats	The framework shall provide mechanisms to harmonize data formats and standards, maintaining consistency and accuracy. This reduces the likelihood of errors and misinterpretations when data is exchanged across different parts of the network, ensuring reliable communication.	Unchanged
FR.DC.06	Data Integration Mechanism	The framework shall provide mechanisms to ensure the integration of data from various sources. This is essential for creating a comprehensive view of the network and enabling effective analysis and decision-making.	Unchanged
FR.DC.07	Efficient Data Processing	The framework shall provide mechanisms to ensure more efficient data processing, analysis, and utilization across different	Unchanged

		applications and services. This enhances the overall performance of the NDT and supports diverse operational needs.	
FR.DC.07.01	Effective Naming Conventions	Establish clear and consistent naming conventions for data points to facilitate easy identification and retrieval. Implement caching mechanisms that utilize these conventions to improve access speeds and reduce redundant data collection.	Added
FR.DC.08	Target-Driven and On-Demand Collection	Data collection should focus on specific operational needs rather than indiscriminately gathering all available data. Operators can define parameters based on current network conditions or specific events, ensuring only pertinent data is collected. Key data includes: • Provisional and operational status of devices • Configuration data • Running status of ports and links • Logs and events • Flow and port statistics • User and service data • Life-cycle operation data • Time series data.	Added
FR.DC.09	Varied Data Collection Methods	Monitoring information varies significantly; some data (e.g., hardware status) can be collected less frequently, while others (e.g., flow status) require real-time monitoring. Multiple tools and methods are essential for effectively gathering diverse data needed to construct the NDT.	Added
FR.DC.09.01	Support for Multiple Protocols	The framework shall support multiple communication protocols to ensure compatibility with legacy systems. This allows integration with existing devices and systems that operate on older protocols, facilitating a smoother transition to modern architectures.	Moved (previously FR.DC.04)
FR.DC.10	Resource-Efficient Data Collection	Data collection tools must be lightweight to minimize impact on network resources. Key aspects include: • **Efficiency Improvement:** Enhance execution efficiency and lower costs for computing, storage, and bandwidth. • **Minimization of Redundant Data:** Avoid or reduce redundant data collection.	Added

		• **Utilization of Data Compression**: Implement both lossy and lossless compression methods to lower resource costs during collection.	
FR.DC.11	Interoperable Data Collection Interfaces	Interfaces should be open and standardized to prevent vendor lock-in and ensure interoperability. Key requirements include: • **Configuration Management Support:** Manage data collection protocols and settings. • **Multiple Rate Options:** Support various collection rates (minute-level, 10-second, near real-time, real-time). • **Extensibility:** Allow for new features with minimal changes and backward compatibility. • **Secure and Reliable Information Exchange:** Ensure data integrity and confidentiality. • **Federation Policy Enforcement:** Enable controlled information exchange across domains with proper authorization.	Added
FR.DC.12	Optimized Data Distribution	Data should be sent to multiple destinations efficiently, ensuring all relevant systems receive necessary information without duplication. Utilize multicast communication methods and intelligent routing systems to achieve this.	Added

These functional requirements will guide the definition of functionalities for the target architecture of the data exposure collection framework in 6G-TWIN in the following sub-sections.

2.2 State of The Art on Reference Frameworks for Data Exposure and Collection in Networks

Network telemetry is crucial to enable real-time monitoring, simulation, and optimization using NDTs. These NDTs rely on comprehensive telemetry data to provide insights into network performance, detect anomalies, and facilitate automated management. While various standardization bodies, such as the O-RAN Alliance [5], ETSI [6], [7], and 3GPP [8], [9], [10] are developing telemetry frameworks tailored to specific network domains, this section will focus on the IETF's telemetry framework as outlined in RFC9232 [11] and Distributed Data Processing Systems (DDPS) like the ISO Reference Model for Open Distributed Processing

(RM-ODP) [12] since these frameworks are domain-agnostic, covering both networking and data requirements, making them versatile references for various network environments.

2.2.1 Standardized Architectures

According to the IETF RFC9232 [13], network telemetry extends beyond traditional Operations, Administration, and Management (OAM) techniques by incorporating other methods and techniques for data generation, collection, and analysis, aiming to enhance network visibility and operational efficiency. The key characteristics of a telemetry framework include:

- **Flexibility:** The framework supports various telemetry techniques, allowing for adaptation to different network environments and requirements.
- **Scalability:** It is designed to handle large volumes of data, making it suitable for modern, high-capacity networks.
- **Accuracy and Coverage:** The framework aims to provide precise data collection across all network layers, ensuring comprehensive monitoring.

The architectural framework for network telemetry, as described in RFC 9232, is designed to address the complexities and challenges of modern network management. It provides a structured approach to data generation, collection, processing, and consumption, facilitating enhanced visibility and operational efficiency. The framework is modular, allowing for flexibility and scalability in various network environments. RFC 9232 identifies four primary modules within the network telemetry framework, each serving distinct functions and associated with different categories of telemetry data:

- **Data Generation Module:** This module is responsible for producing telemetry data from various network sources, and it is crucial to ensure that the telemetry data reflects the current state of the network, including device performance, traffic patterns, and error rates. Key components include:
    - **Instrumentation Components:** These are embedded within network devices to capture relevant metrics and events. They can be configured to generate data based on specific triggers or conditions, ensuring that the telemetry data is both relevant and timely.
    - **Configuration Interfaces:** Operators can define what data to generate and how it should be formatted. This flexibility allows for customization based on operational needs and specific use cases.

- **Data Collection Module:** This module focuses on gathering telemetry data from the network. The efficiency of this module is vital for minimizing latency and ensuring that data is available for real-time analysis. Its components include:
    - **Data Aggregators**: These collect data from multiple sources, reducing the volume of data that needs to be processed and transmitted. Aggregators can filter and summarize data, ensuring that only the most pertinent information is forwarded for further analysis.
    - **Transport Mechanisms**: This includes protocols and methods for transmitting collected data to processing systems. The framework supports various transport

methods, including push and pull mechanisms, to accommodate different network architectures and operational requirements.

- **Data Processing Module:** Once data is collected, it must be processed to extract actionable insights. This module is essential for converting telemetry data into meaningful insights that can inform network management decisions. The **Data Processing Module** includes:
    - **Data Rendering and Encoding**: This involves transforming raw telemetry data into a format suitable for analysis. Techniques such as compression and encoding are employed to optimize data for storage and transmission.
    - **Analytics Engines**: These engines analyze the processed data to identify trends, anomalies, and performance issues. They can utilize ML algorithms to enhance predictive capabilities and automate responses to detected issues.

- **Data Consumption Module:** This module is concerned with how management applications access and utilize telemetry data. This module ensures that the insights derived from telemetry data are readily available to network operators and automated systems, facilitating timely decision-making. Key components include:
    - **Application Programming Interfaces (APIs) and Interfaces:** These provide standardized access to telemetry data for various applications, enabling integration with existing network management tools. The framework supports RESTful APIs, allowing third-party applications to easily consume data.
    - **Visualization Tools:** Dashboards and reporting tools that present telemetry data in an easily digestible format. These tools help operators to assess network health and performance quickly.

The framework categorizes telemetry data into several types, each serving different operational needs:

- **Management Plane Data**: Includes information related to network management functions, such as configuration changes and performance metrics. This data is crucial for maintaining operational oversight.
- **Control Plane Data**: Provides insights into the control mechanisms of the network, including routing updates and signaling information. Understanding control plane data is essential for optimizing routing and ensuring efficient data flow.
- **Forwarding Plane Data**: Focuses on the transmitted data packets, including flow statistics and packet loss rates. This data is vital for assessing the performance of data transmission and identifying bottlenecks.
- **External Data**: Involves data collected from external sources, such as user behavior analytics and environmental conditions impacting network performance. This type of data can enhance the understanding of network usage patterns and inform capacity planning.

Regarding security, the RFC 9232 emphasizes the importance of security in network telemetry. It outlines potential risks associated with large-scale data collection, particularly regarding user privacy. The framework explicitly states that it should not be applied to generating or retaining individual user data without consent. This principle is crucial for maintaining trust and compliance with privacy regulations.

2.2.2 Distributed Data Stream Processing

Distributed Data Stream Processing (DDSP) frameworks are essential for managing and processing real-time data at scale [14]. These frameworks, such as Apache Flink, Apache Spark Streaming, and Apache Kafka, enable organizations to handle vast amounts of data continuously flowing in from various sources, such as IoT devices, social media feeds, and financial transactions. These frameworks offer scalability, fault tolerance, and low-latency processing, making them highly suitable for real-time analytics, anomaly detection, and event-driven architectures.

Unlike traditional batch processing, which processes data in large chunks at scheduled intervals, DDSP frameworks handle data continuously as it arrives in real-time. These frameworks are designed to process data streams without delay, enabling immediate analytics and decision-making. As illustrated in Figure 2, the core components of DDSP systems, including data ingestion, processing engines, and output handling, work together to support high throughput, low latency, and fault tolerance. Let us describe them in more detail.

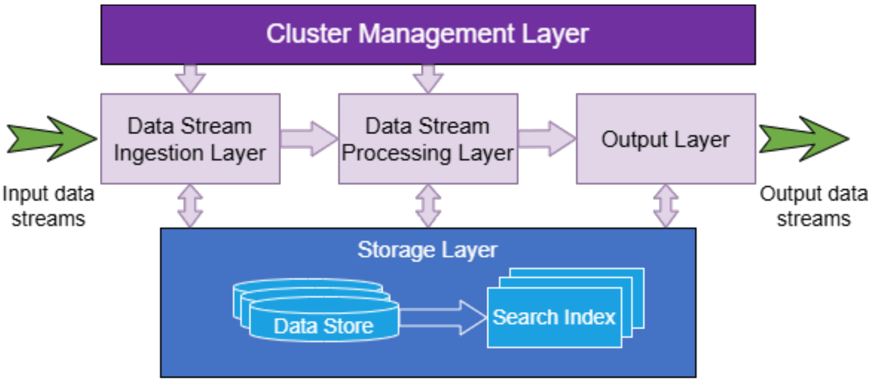


Figure 2 A layer-wise view of DDSP systems from [14].

- **Ingestion Layer:** This initial stage involves efficiently transferring data streams from sources to processing or storage systems. It ensures scalable, resilient, and fault-tolerant data distribution across the DSPS architecture, decoupling input data sources from other components. Input data streams can come from various network sources and other stream processing systems. These streams may include text, graphs, JSON, events, or time-series data. Queueing systems, such as MQTT, RabbitMQ, ActiveMQ, NSQ, and ZeroMQ, handle messaging services. Apache Kafka supports publishing and subscribing to record streams, and Zenoh is a pub/sub/query protocol unifying data in motion, data at rest, and computations.
- **Stream Processing Layer:** This is the core component where data processing occurs. It involves a set of operators performing transformations (e.g., filtering, aggregating) on the data as it flows. DDSP frameworks like Apache Storm, Flink, Zenoh Flow, and Spark Streaming provide rich APIs for defining such data processing pipelines.
- **Storage layer:** This component stores analyzed data, patterns, and extracted knowledge that can be used later for further processing (e.g., testing a new mechanism for feature extraction) or directly for deployment and execution (e.g., ML model retrieval and deployment). This data must be organized, indexed, and linked with metadata for

subsequent analytics. Data storage solutions for DSPSs range from traditional file systems like HDFS and BFS to distributed relational databases (e.g., PostgreSQL), key-value stores (e.g., Redis), in-memory databases (e.g., VoltDB), document stores (e.g., MongoDB), graph storage systems (e.g., Neo4j), NoSQL databases (e.g., Cassandra), and NewSQL databases (e.g., CockroachDB).

- **Resource Management:** This layer oversees the coordination of computing and storage nodes, handling resource allocation and scheduling for the parallel processing of high-volume data streams in distributed systems. A typical data stream cluster includes computing nodes and a cluster manager coordinating communication between them. Processes can exchange messages via a network or read/write from shared storage. Coordination tasks like master node election, group management, and metadata handling are crucial in multi-cluster systems. Advanced resource management tools include ZooKeeper, YARN, Mesos, Kubernetes, and managed services like Borg. Notice that while data stream ingestion frameworks like Kafka do not provide a direct resource management feature, they can interact with other resource management systems like Kubernetes, YARN, and Mesos to schedule Kafka brokers, manage scaling, and ensure fault tolerance in distributed computing infrastructure.
- **Output Layer:** After processing, data can be directed to other systems, visualization tools, or dashboards. This output stage ensures that processed insights are available for real-time decision-making and analysis.

This architecture allows DDSP frameworks to support modern applications that require real-time data ingestion, processing, and analysis at scale. They offer critical properties like scalability, low-latency processing, and fault tolerance, making them well-suited for applications in sectors like e-commerce, finance, and telecommunications.

2.3 6G-TWIN Data Collection and Exposure Architecture

In the previous sub-sections, we outlined the functional requirements for designing a telemetry framework for NDTs. In addition, we discussed in detail how the IETF has defined a generic framework for network telemetry. While the IETF's telemetry framework defines the functionalities to deliver the necessary data or knowledge items to requesters, it needs to be enhanced or adapted to meet the specific requirements of the NDT architecture. This is where data-oriented frameworks, such as DDSP, become crucial. These frameworks enable real-time, distributed, reliable, and highly scalable data processing. Despite their robust capabilities, DDSP frameworks must still be tailored to the unique needs of each domain. For example, in the telecommunications sector, these systems need to accommodate network-specific data types, integrate seamlessly with existing network management protocols, and meet stringent latency and reliability demands. Thus, while DDSP frameworks provide a solid foundation, their successful deployment relies on domain-specific customization.

6G-TWIN proposes a more robust framework to not only support traditional telemetry capacity but also integrates the capabilities of DDSP frameworks into consideration. As a result, Figure 3 introduces the high-level functional architecture of the data collection and exposure framework ambitioned by 6G-TWIN. The proposed architecture aims to facilitate the seamless integration and communication among various components, ensuring efficient data management, pre-processing, and synchronization across digital and physical network elements. Moreover, scalability, security, and federated distribution are included as part of the architecture to enable flexible, parallel data processing and updates, which are essential for real-time network analysis and decision-making.

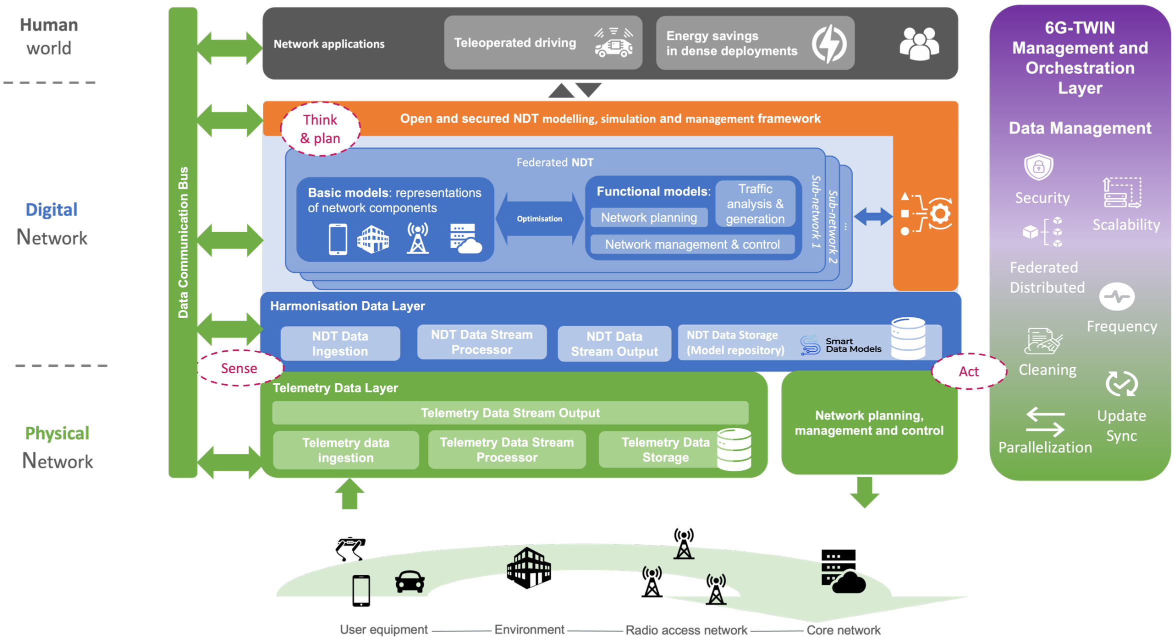


Figure 3. 6G-TWIN data collection and exposure framework.

The updated components of the framework are the following:

- **Network Digital Twin**
    - **Harmonization Data Layer:** Provides the necessary data processing capabilities, including:
        - **NDT Data Stream Ingestion and Processing:** Collects normalized data from telemetry layer and processes incoming data streams in real-time.
        - **NDT Data Storage:** Stores aggregated data from varied sources such as network infrastructure, sensors, and contextual information. Additionally, it stores the basic and functional models for reuse.
        - **NDT Data Output:** Exposes harmonized data from the NDT to other components (e.g., network applications using the NDT).
- **Physical Network**
    - **Telemetry Data Layer:** Responsible for collecting and processing real-time data from the physical network. It includes:

    - **Telemetry Data Ingestion and Processing:** Captures and analyzes network data streams.
    - **Telemetry Data Storage:** Stores processed telemetry information for further use.
    - **Telemetry Data Output:** Expose normalized telemetry data stream outputs to other components.
- **Data communication bus:** the bus acts as the "high-way" to interconnect the different components of the 6G-TWIN architecture. At the physical network interface, it captures raw telemetry data and forwards it to the telemetry layer for initial processing. The telemetry layer refines (e.g., cleaning, filtering, normalization) and enriches this data, forwarding it to the harmonization data layer in the NDT, where it is transformed to be used for simulation and modeling. Insights generated from the NDT are propagated to the application layer for decision-making, control, and user interaction. Advanced features such as dynamic routing, data prioritization, and error-checking within the bus enhance system reliability and responsiveness and are controlled by the 6G-TWIN Management and Orchestration layer. Notice that the design principles of the data communication bus apply to both the physical and the NDT layers.
- **6G-TWIN Management and Orchestration Layer (6G-TWIN MANO):** Implements the Management and Orchestration (MANO) functionalities for the 6G-TWIN system. For the data collection and exposure functionality, it provides key features that are transversal to all data operations in the system such as:
    - **Data Management & Security:** Ensures that data flowing across layers is secure, federated, and synchronized in real-time.
    - **Scalability & Parallelization:** The architecture is designed to handle high-frequency, large-scale data streams of telemetry data.

Table 3 provides a high-level comparison between the 6G-TWIN data collection and exposure architecture and the IETF network telemetry framework. In general, we can see that the 6G-TWIN architecture already covers not only the basic functionalities of a network telemetry framework but also the required components for stream data processing as provided in DDSP frameworks at multiple layers.

Table 3 Comparing 6G-TWIN architecture and other reference frameworks that include network data collection.

Component	6G-TWIN	IETF RFC9232	ITU Y.3090
Telemetry Data Layer	Yes	Yes	Yes (Data collection in the Physical Network)
Harmonization Data Layer	Yes	No	Yes (Data Collection in the NDT)
Federated NDT	Yes	No	No
Simulation Framework	Yes	N/A	Part of the NDT
Management and Orchestration layer	Yes, including data management	Yes	Yes, including data management
DDSP framework	Yes, explicitly integrated in the telemetry and harmonization layers.	Partially defined for the telemetry framework.	No defined.

The 6G-TWIN architecture introduces novel advancements over IETF RFC9232 and ITU Y.3090 by introducing a Harmonization Data Layer, a unique Federated NDT, and a dedicated Simulation Framework, which are absent in the other standards. It enhances data and model management through explicit integration of a Digital Data Stream Processing (DDSP) framework across its telemetry and harmonization layers, unlike the partial or undefined implementations in the alternatives. Furthermore, the Management and Orchestration layer in 6G-TWIN includes robust data management capabilities, building on the baseline features of IETF RFC9232 and ITU Y.3090 [15]. In the following Sections 3, 5, and 4, we will explore the telemetry and harmonization data layers, along with the data communication buses, focusing on the protocols, standards, and technologies that are required to implement them in modern communication networks.

3 Telemetry Data Layer

Telemetry in networks is essential for ensuring efficient operation, monitoring, and optimization of modern communication systems. Moreover, it will act as the data extraction layer for NDTs. It provides real-time visibility into network performance by collecting and streaming data from various sources, such as user devices, radio access networks, and core infrastructure. This continuous flow of telemetry data enables a) at management level, proactive identification of performance bottlenecks, fault detection, and traffic optimization, ensuring high service quality and reliability, and b) creation and update of NDTs in real-time.

To achieve this, 6G-TWIN introduces the TDL, showcasing its integration within a broader system for sensing and acting on data. Figure 4 outlines the following key functional elements:

- **Telemetry Data Ingestion:** Collects raw telemetry data from various sources in real-time, forming the initial entry point into the system.
- **Telemetry Data Stream Processor:** Processes and analyzes the ingested data streams, performing tasks such as filtering, aggregation, and enrichment for actionable insights.
- **Telemetry Data Storage:** Stores processed telemetry data for historical analysis, reporting, and further decision-making processes.
- **Telemetry Data Stream Output:** Facilitates the delivery of processed telemetry data to the next stage for network planning, management, and control.

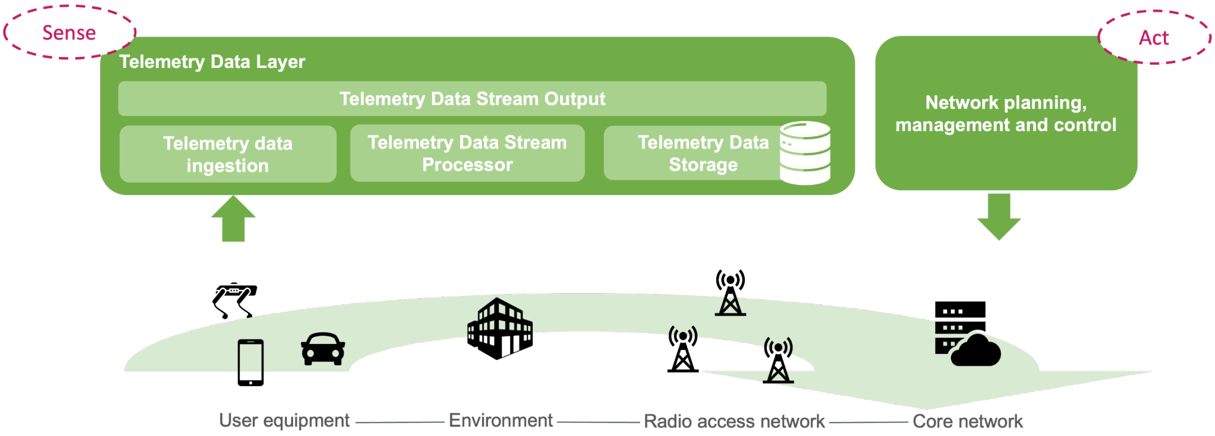


Figure 4 The Telemetry Data Layer and their functional blocks

In general, 6G-TWIN TDL supports a Sense-Act loop, where telemetry sensing feeds upper layers, such as the management or NDT layers, to drive intelligent decision-making and control actions within the network. This ensures optimized network performance, efficient management, and continuous adaptation to dynamic environmental conditions. In the following subsections, we will discuss each of these blocks in more detail. Frameworks to implement the TDL are described in more detail in the section about the Harmonization Data Layer (Section 4) since both layers are based on DDSP.

3.1 Data Ingestion Protocols

Protocols to gather telemetry data from the network devices are crucial as they establish the methods and rules for data communication between network devices (producers/generators) and management systems (consumers). They facilitate the monitoring, configuration, and control of network elements by providing standardized ways to exchange information. For instance, Command Line Interfaces (CLIs) are used for configurational (configuration commands) and operational data (show commands) directly on the network devices. In addition, protocols like Simple Network Management Protocol (SNMP) allow network administrators to send requests to devices, retrieve performance data, and receive alerts about potential issues. This structured approach ensures that data can be communicated efficiently and reliably, enabling effective network management and optimization. While CLIs and SNMP are widely utilized, they come with certain limitations. CLIs tend to be highly proprietary, requiring human intervention to interpret their text-based commands effectively. On the other hand, SNMP lacks the ability to differentiate between configuration data and operational data, which can complicate management tasks. To better understand the landscape protocols used in network telemetry, the following is a list of the most well-known for network management.

SNMP (Simple Network Management Protocol) [16]: This protocol is widely used as an application layer protocol that facilitates the monitoring and management of network devices. Originating in the 1980s, SNMP allows network administrators to collect data on device performance, such as traffic metrics, error rates, and connection speeds. It sends messages known as Protocol Data Units (PDUs) to devices that support SNMP, enabling real-time queries about their status. This capability is crucial for maintaining network health, as it allows for proactive monitoring and alerts when performance thresholds are exceeded. SNMP's architecture includes managed devices, agents that gather data, and a network management station that processes this information. One of the main limitations of SNMP is that the protocol does not distinguish between configurational and operational data.

To illustrate how SNMP packets appear in a PCAP file, consider a scenario where an SNMP GET request is made to retrieve the system uptime from a network device. In a PCAP capture, you would see UDP packets typically sent to port 161 (the standard SNMP port).

This is an example of the data obtained from an SNMP message:

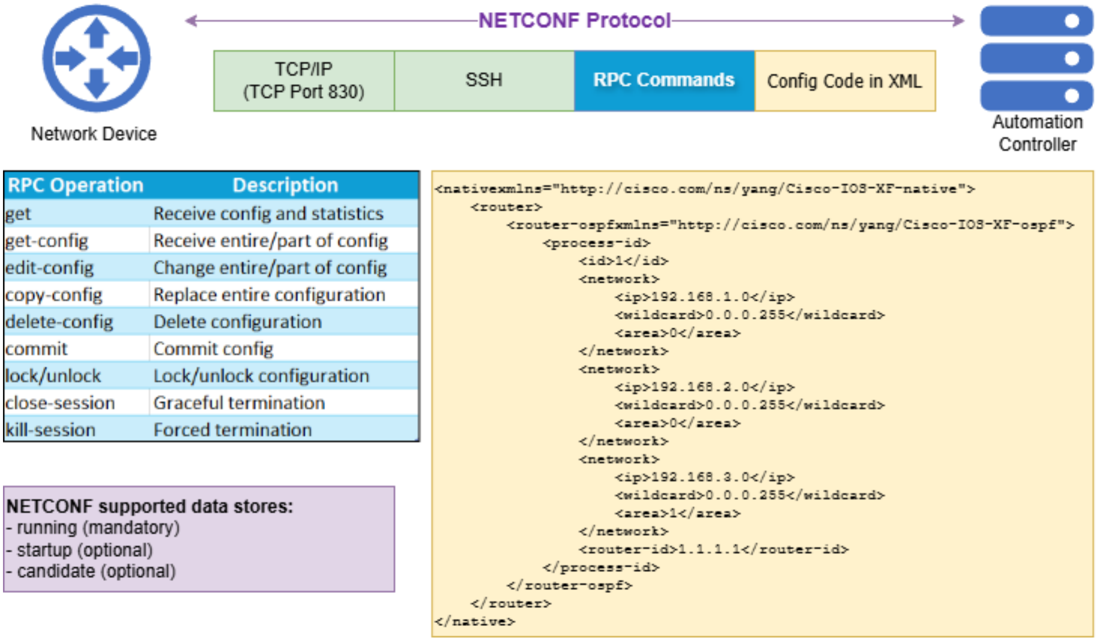
- **SNMP GET Request**: This packet would contain the following elements:
    - **Source Port**: 12345 (the port used by the SNMP manager)
    - **Destination Port**: 161 (the SNMP agent's port)
    - **PDU Type**: GET
    - **Object Identifier (OID)**: For example, 1.3.6.1.2.1.1.3.0 (which corresponds to the system uptime).
- **SNMP Response**: Following the request, the device will send back a response:
    - **Source Port**: 161 (the SNMP agent's port)
    - **Destination Port**: 12345 (the port used by the SNMP manager)
    - **PDU Type**: RESPONSE

- **Value**: The system uptime value, such as 12345678 (in seconds).

SNMP manager needs the **Management Information Base (MIB)** to process messages from devices. MIB contains definitions and info about the properties of managed resources and the services that the agents (devices) support.

NETCONF (Network Configuration Protocol) [17]: NETCONF provides a framework for installing, modifying, and deleting the configurations of network devices. It uses an XML-based data encoding for both configuration data and protocol messages as shown in Figure 5. Communication between the client and server is facilitated through a straightforward Remote Procedure Call (RPC) mechanism. The client can be a script or application integrated into a network management system, while the server is typically a network device such as a switch or router. NETCONF utilizes Secure SHell (SSH) as its transport layer, defaulting to port 830, which can be configured as needed.

Additionally, NETCONF supports capability discovery and model downloads. The IETF-NETCONF-monitoring model is used to identify supported models, with revision dates provided in the capabilities response. Data models can be downloaded from devices using the get-schema RPC, allowing users to utilize YANG models to understand or export the data model. In recent versions of network operating systems, operational data functions seamlessly on platforms running NETCONF, similar to configuration data, and is typically enabled by default [18].



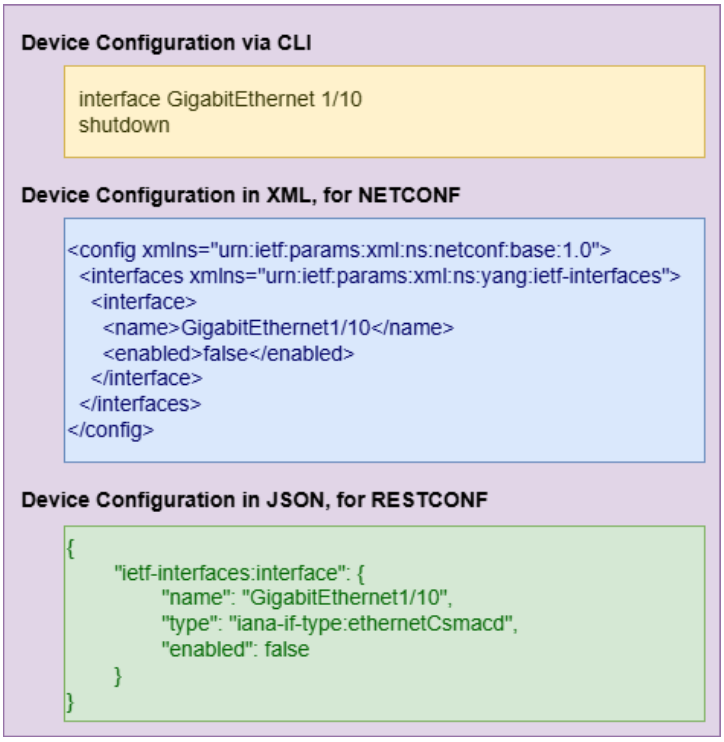
RPC Operation	Description
get	Receive config and statistics
get-config	Receive entire/part of config
edit-config	Change entire/part of config
copy-config	Replace entire configuration
delete-config	Delete configuration
commit	Commit config
lock/unlock	Lock/unlock configuration
close-session	Graceful termination
kill-session	Forced termination

NETCONF supported data stores:
- running (mandatory)
- startup (optional)
- candidate (optional)

```
<nativexmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
    <router>
        <router-ospfxmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-ospf">
            <process-id>
                <id>1</id>
                <network>
                    <ip>192.168.1.0</ip>
                    <wildcard>0.0.0.255</wildcard>
                    <area>0</area>
                </network>
                <network>
                    <ip>192.168.2.0</ip>
                    <wildcard>0.0.0.255</wildcard>
                    <area>0</area>
                </network>
                <network>
                    <ip>192.168.3.0</ip>
                    <wildcard>0.0.0.255</wildcard>
                    <area>1</area>
                </network>
                <router-id>1.1.1.1</router-id>
            </process-id>
        </router-ospf>
    </router>
</native>
```

Figure 5 Netconf architecture and an example of message structure (modified from [19])

RESTCONF [20]: RESTCONF is a protocol that provides a programmatic interface for accessing and manipulating configuration and operational state data on network devices. It leverages standard HTTP methods to perform CRUD (Create, Read, Update, Delete) operations on data defined by YANG models. RESTCONF allows users to interact with network devices using structured data formats like XML or JSON, as shown in Figure 6, making managing configurations and retrieving operational data easier. The protocol is designed to be stateless and secure, utilizing HTTPS for communication. Additionally, RESTCONF supports various features, such as data-model-specific RPC operations and event notifications, enhancing the overall programmability of network management.



Device Configuration via CLI

```
interface GigabitEthernet 1/10
shutdown
```

Device Configuration in XML, for NETCONF

```
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
    <interface>
      <name>GigabitEthernet1/10</name>
      <enabled>false</enabled>
    </interface>
  </interfaces>
</config>
```

Device Configuration in JSON, for RESTCONF

```
{
        "ietf-interfaces:interface": {
                "name": "GigabitEthernet1/10",
                "type": "iana-if-type:ethernetCsmacd",
                "enabled": false
        }
}
```

Figure 6 Interface configuration - different formats to do the same task.

gRPC Telemetry [21]: A high-performance communication protocol that uses gRPC, based on HTTP/2 [22], to transmit telemetry data in real time. It allows efficient and scalable transmission of structured data, using Protobuf serialization to reduce overhead, as shown in Figure 7. gRPC telemetry enhances observability by providing a robust framework for monitoring and managing metrics within gRPC applications. This integration allows developers to collect valuable performance data, such as latency and throughput, which are essential for troubleshooting and optimizing system performance. The gRPC OpenTelemetry [23] plugin utilizes a MeterProvider to create various instruments that track metrics at different levels, including per-call and per-attempt metrics for both clients and servers. This structured approach enables continuous monitoring and alerting, helping teams to iterate on improvements effectively. By transitioning from OpenCensus to OpenTelemetry, gRPC ensures a more standardized and comprehensive method for managing telemetry data, ultimately contributing to better system reliability and performance.

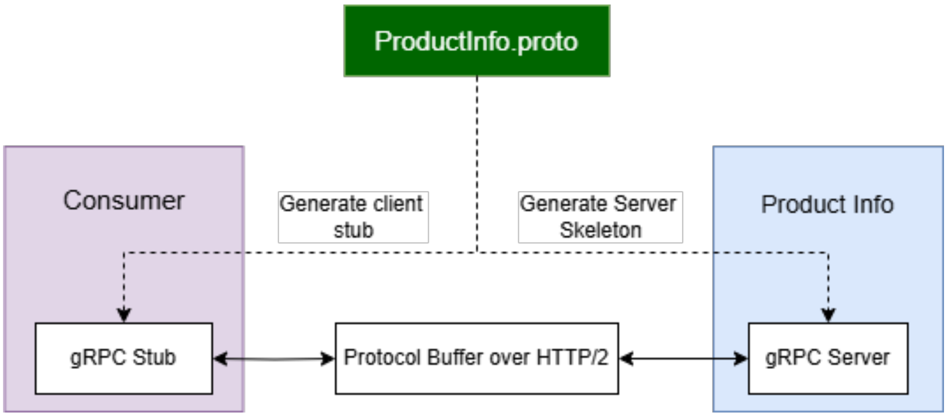


Figure 7 Microservice and a consumer based on gRPC

IPFIX (IP Flow Information Export) [24]: The IPFIX is an IETF protocol designed to provide network administrators with access to IP flow information from various devices, such as routers and probes. It was developed to establish a universal standard for exporting flow data, essential for tasks like measurement, accounting, and network management. IPFIX operates by collecting data packets at an observation point through a metering process as shown in Figure 8, which can filter and aggregate this data before sending it to a collector. The protocol is characterized as a push mechanism, meaning that data is periodically sent to configured receivers without requiring interaction. IPFIX messages are structured using templates, allowing for flexibility and extensibility in transmitting data. The IPFIX message format consists of a header followed by one or more sets, including data sets, template sets, or options template sets. While IPFIX prefers the Stream Control Transmission Protocol (SCTP) for transport, it also supports Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). This versatility makes IPFIX suitable for various network environments. By providing detailed flow information, IPFIX enhances the ability to monitor network performance and troubleshoot issues effectively, making it a valuable tool for network management and analysis.

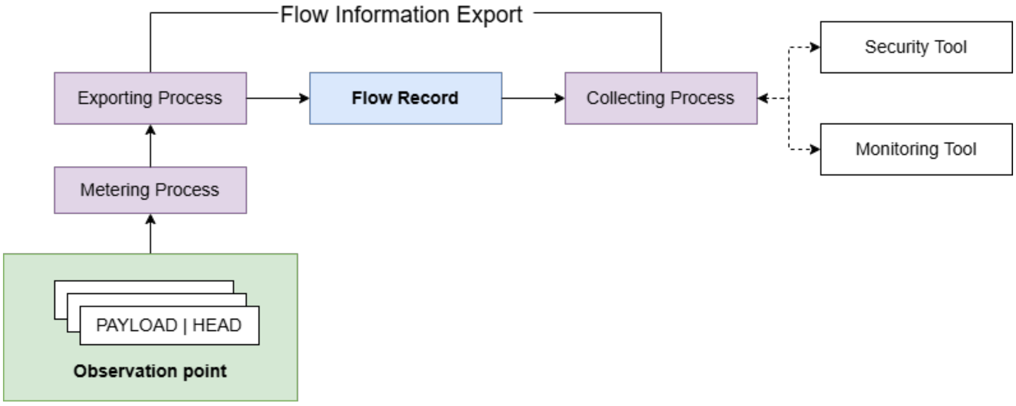


Figure 8 IPFIX Architecture

sFlow [25], [26]: sFlow is a standardized protocol designed for monitoring network traffic by exporting information about packets traversing routers and switches. It operates through a sampling method, where packets are randomly selected and analyzed, allowing for efficient data collection without capturing every packet. This approach provides a statistical overview of network usage, enabling administrators to detect issues, manage congestion, and optimize performance. sFlow is particularly effective in high-speed environments, supporting monitoring at speeds of 10Gbps and beyond, while minimizing the impact on network performance. The architecture of sFlow consists of agents embedded in network devices that continuously sample traffic and send this data to a central collector for analysis as shown in Figure 9. This system allows for real-time visibility into Layer 2 to Layer 7 traffic flows, providing insights into application usage, security threats, and performance trends. By leveraging sFlow, organizations can enhance their network management capabilities, ensuring efficient operation and facilitating proactive troubleshooting.

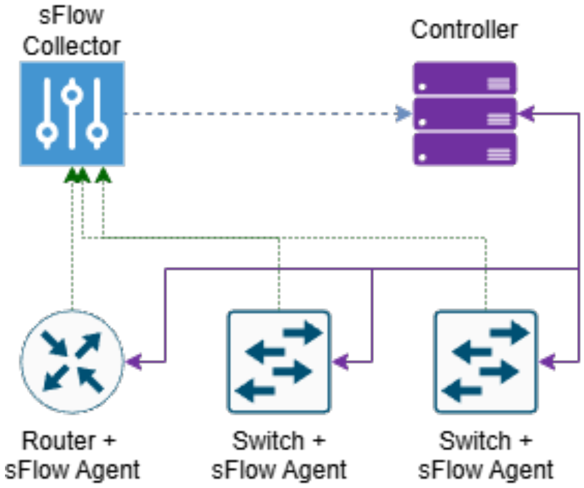


Figure 9 sFlow System Architecture

NetFlow [27] [28]: NetFlow is a network protocol developed by Cisco Systems that is widely used to collect metadata about IP traffic flowing through network devices such as routers and switches. It provides valuable insights into network performance by monitoring traffic flows, which helps IT professionals understand traffic patterns, identify bottlenecks, and optimize application performance. NetFlow captures various metrics, including the source and destination IP addresses, protocols used, and the amount of data transferred, allowing for a comprehensive analysis of network activity, as shown in Figure 10.

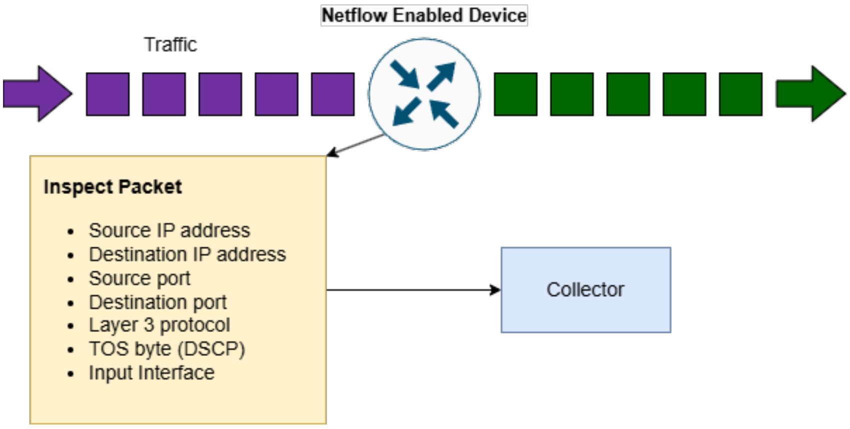


Figure 10 NetFlow Architecture

The protocol aggregates data packets into flows and exports this information to a NetFlow collector using UDP. This process, known as NetFlow Data Export (NDE), enables the collection of flow records that can be analyzed for trends, security monitoring, and usage-based billing. NetFlow supports multiple versions, with version 9 being template-based for greater flexibility. Overall, NetFlow enhances visibility into network operations, aiding in effective management and planning for future growth.

Table 4 compares key protocols used in network management and telemetry, focusing on their transport mechanisms and data encoding formats. Protocols like SNMP, IPFIX, sFlow, and NetFlow predominantly use UDP for lightweight, connectionless transport, with encoding formats tailored to their specific use cases, such as ASN.1/BER for SNMP or binary formats for flow data. In contrast, NETCONF and RESTCONF leverage SSH and HTTP/HTTPS, prioritizing secure and versatile transport layers while employing XML and JSON for human-readable, structured data exchange. Modern telemetry approaches like gRPC Telemetry use HTTP/2 for efficient, multiplexed transport, paired with Protocol Buffers (Protobuf) for compact and high-performance encoding. In general, we can see that each reflects trade-offs between efficiency, security, and flexibility tailored to specific network monitoring needs.

Table 4 Comparison of network protocols to extract telemetry data from network devices

Protocol	Transport	Encoding/decoding
SNMP	UDP	ASN.1 with BER (Basic Encoding Rules)
NETCONF	SSH	XML (default), can support JSON
RESTCONF	HTTP/HTTPS	XML or JSON
gRPC Telemetry	HTTP/2	Protocol Buffers (Protobuf)
IPFIX	UDP, SCTP, TCP	Binary (Template-based format)
sFlow	UDP	Binary (sFlow data format)
NetFlow	UDP (v5/v9), SCTP (v9)	Binary (NetFlow data format)

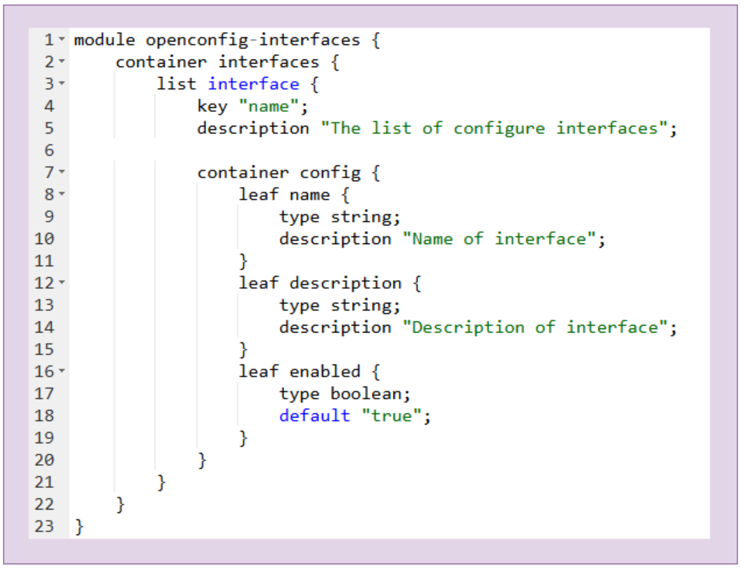
3.2 Network Data Modelling Languages

Languages in network telemetry for NDT serve as the blueprint for modeling and structuring data related to network configurations and operational states. They provide a common vocabulary and syntax that allow different devices and systems to understand and interpret the data consistently. Using standardized modeling languages greatly enhances interoperability and efficiency in network management. Below are the primary languages that define and organize telemetry data in network environments.

Yet Another Next Generation (YANG) [29]: A data modeling language used to define the structure and data elements used in the configuration and management of network devices. It was developed by the IETF and is used in conjunction with protocols such as NETCONF and RESTCONF to manage network devices. Figure 11 shows a YANG template example.
The main characteristics of YANG are:

- **Data Modeling:** Allows the creation of data models that describe the configurations and states of network devices in a standardized way.

- **Interoperability:** Different devices and systems can understand and process data consistently.
- **Extensibility:** Allows to define custom models that fit specific needs while maintaining compatibility with existing standards.



```
module openconfig-interfaces {
    container interfaces {
        list interface {
            key "name";
            description "The list of configure interfaces";

            container config {
                leaf name {
                    type string;
                    description "Name of interface";
                }
                leaf description {
                    type string;
                    description "Description of interface";
                }
                leaf enabled {
                    type boolean;
                    default "true";
                }
            }
        }
    }
}
```

Figure 11 YANG Template for interface configuration

In general, YANG is a versatile data modeling language that underpins telecom network configuration, management, and telemetry, playing a critical role in RAN, transport, edge/cloud, and core network domains. In RAN, YANG models enable automation and dynamic configuration of radio parameters, which are essential for 5G features like network slicing and scalable deployments. For Transport Networks (TN), YANG facilitates the management of MPLS, optical, and IP transport layers through protocols such as NETCONF and RESTCONF, promoting interoperability in multi-vendor environments. At the edge and in cloud environments, YANG integrates with orchestration platforms like Kubernetes and OpenStack, aiding in the management of virtualized or containerized network functions, Virtual Network Functions (VNFs), and Cloud-native Network Functions (CNFs) and enabling advanced telemetry collection. In the core network, YANG is instrumental in modeling critical protocols like BGP and OSPF, as well as powering real-time telemetry systems using frameworks like gRPC Network Management Interface (gNMI, see Section 3.3). Across these domains, YANG provides a consistent, vendor-neutral framework for network programmability and operational insights.

TOSCA (Topology and Orchestration Specification for Cloud Applications) [30]: This is a specification language designed to provide a standardized way to model cloud applications, services, and their orchestrations. By defining the structure, components, relationships, and behaviors of cloud-based workloads, as shown in Figure 12, TOSCA enables developers and operators to represent complex applications in a portable, interoperable format. This approach ensures that applications and services can be deployed, scaled, and managed consistently across diverse cloud environments, eliminating vendor lock-in and simplifying multi-cloud operations. TOSCA's emphasis on abstraction allows users to focus on high-level application design without being tied to the specifics of the underlying infrastructure.

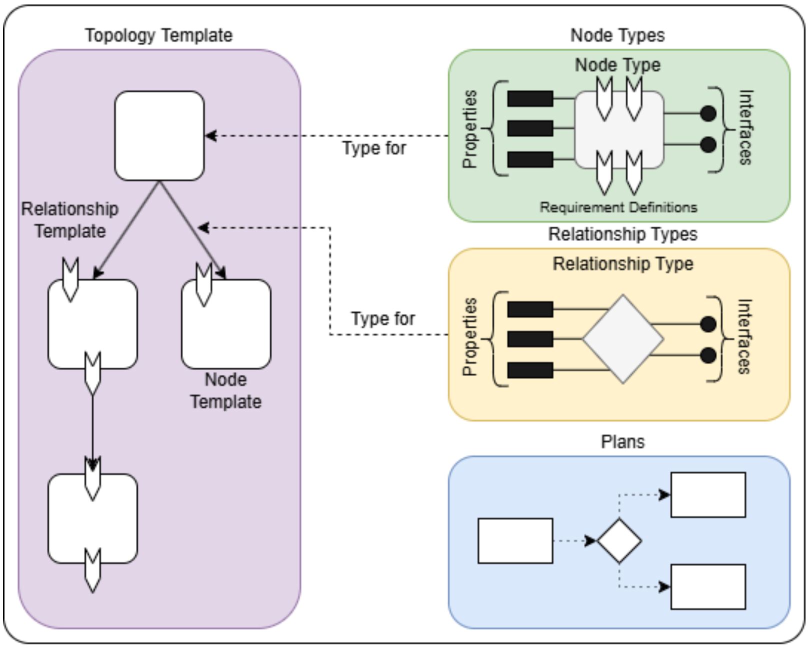


Figure 12 Structural Elements of a Service Template and their Relations. Taken from [53]

One of the key features of TOSCA is its ability to describe application components and their relationships. It uses templates to define nodes representing application components such as databases, web servers, or virtual machines and their interconnections, such as data flows or dependencies. Additionally, TOSCA supports lifecycle management processes, including deployment, scaling, and termination, through workflows that can be automated. This comprehensive modeling capability ensures that all aspects of a cloud application, from its resources to its operational requirements, are captured in a unified and machine-readable format. By leveraging TOSCA models, organizations can achieve efficient orchestration, enabling seamless integration of application components across heterogeneous cloud platforms.

The main components of TOSCA are:

- **Modeling Complex Services:** Allows defining services and applications, including all their components, configurations, and dependencies.
- **Portability and Interoperability:** TOSCA models can be interpreted by different orchestration systems, such as Tacker and Cloudify [31], facilitating the portability of services between environments.
- **Deployment and Management Automation:** Provides mechanisms to automate service deployment, configuration, and management.

Notice that while YANG is a data modeling language, TOSCA is a service automation language.

3.3 Techniques to Collect Data from The Network

In practice, several techniques exist to collect, transmit, and process telemetry data from network devices. These techniques determine how data is gathered, whether it is polled, pushed, streamed, or event-driven, and how effectively it can be analyzed for actionable insights. The choice of technique can significantly impact the performance, scalability, and real-time visibility of the network monitoring solution. The following sections outline the prevalent techniques that underpin modern network telemetry practices as presented in the appendix A of [13].

3.3.1Management Plane Telemetry

The management plane interfaces with Network Management Systems (NMS) to deliver performance metrics, logs, warnings, and state data using protocols like SNMP and syslog. Telemetry requirements include flexible data subscription (customized and periodic/on-change delivery), structured data (YANG models for automation), high-speed transport (efficient encoding and compression for rapid data flow), and mechanisms to prevent network congestion to ensure efficient, scalable, and automated network management while minimizing client-server interactions and preventing system overload. Examples of techniques and protocols for this plane are the following.

- **Push Extensions for NETCONF:** Extends NETCONF and enables subscriber applications to request a continuous, customized stream of updates from a YANG datastore.
- **gRPC Network Management Interface (gNMI):** is a network management protocol based on the RPC framework. With a single gRPC service definition, both configuration and telemetry can be covered. gRPC is an open-source micro-service communication framework based on HTTP/2. It provides several capabilities that are well-suited for network telemetry, including a full-duplex streaming transport model, a higher-level feature consistency across platforms, and a built-in load-balancing and failover mechanism.

3.3.2Control Plane Telemetry

Control plane telemetry monitors the health of network control protocols across protocol stack layers, aiding in real-time issue detection, localization, and network optimization. Key challenges include correlating End-to-End (E2E) KPIs with specific protocol layers, as issues may span multiple layers or devices, and conventional OAM methods often fail to reflect the operational status of these protocols. While tools like BGP Monitoring Protocol (BMP) offer rich analysis capabilities for routing, telemetry for other protocols and cross-layer KPI correlations are still under research.

BMP is used to monitor BGP sessions and is intended to provide a convenient interface for obtaining route views. BGP routing information is collected from the monitored device(s) to the BMP monitoring station by setting up the BMP TCP session. The BMP Peer Up and Peer Down notifications monitor the BGP peers. The BGP routes are encapsulated in the BMP Route

Monitoring Message and the BMP Route Mirroring Message, providing both an initial table dump and real-time route updates.

3.3.3Data Plane Telemetry

Data plane telemetry involves extracting and analyzing data from network devices to ensure high-quality, efficient network operations. Challenges include balancing data collection with traffic processing, managing vast data volumes without overloading bandwidth, and minimizing delays to maintain timely feedback. Techniques must provide structured, flexible data for varied applications and support incremental deployment. Methods are categorized by dimensions such as a) passive, active, or hybrid instrumentation, b) in-band or out-of-band data transport, and c) end-to-end versus in-network collection.

Alternate-Marking (AM) Technology [32]: This methodology measures and monitors packet loss and delay variations in a network. This technique involves marking packets alternately, either with a “mark” or “no mark,” to provide an efficient and simple way to assess the quality of a network path. The method enables operators to compute the packet loss ratio and delay variation by analyzing the marked packets. AM is designed to be easy to implement, as it can be applied to any traffic path without requiring significant changes to the network infrastructure.

Dynamic Network Probe (DNP) [33]: This is a framework for measuring and enhancing the performance of data networks, focusing on improving network performance and quality. It defines a methodology for assessing network health by combining traditional performance metrics with real-time data collection and analytics. This framework aims to provide accurate, actionable insights into network behavior, essential for optimizing operations, detecting anomalies, and ensuring quality across diverse network environments.

In Situ OAM (IOAM) [34]: This network telemetry framework embeds operational and measurement data within live user packets during transit. This allows real-time tracking of network conditions like latency, jitter, and path tracing. IOAM supports modes such as direct insertion, hop-by-hop updates, and data export for external analysis. It offers flexibility for path-level, node-level, and flow-based data collection while minimizing impacts on network performance.

Postcard-Based Telemetry (PBT) [35]: This method provides the mechanism for network telemetry by generating “postcards” for specific packets. These postcards, carrying metadata and telemetry data, are sent directly to a collector without altering user packets. PBT supports two modes: direct export of telemetry data and encoding of partial telemetry data into user packets. This approach enables flexible, scalable telemetry for flow, path, and node-level insights while avoiding impacts on the forwarding behavior of user packets.

Packet Sampling (PSAMP) [36]: This is a standardized methodology for network packet sampling and export. Its primary goal is to enable the efficient monitoring and analysis of large-scale networks by collecting a representative subset of packets rather than capturing all traffic. This is particularly useful in high-speed environments where processing and storing every packet is impractical. PSAMP specifies mechanisms for selecting packets, formatting the exported data, and transferring the information to monitoring systems.

Table 5 highlights a selection of existing mechanisms, primarily standardized by the IETF, emphasizing recent technological advancements. While the listed mechanisms are representative examples within the framework, they do not encompass the full breadth of available solutions in the domain. It is important to acknowledge that many protocols and techniques are multifaceted, addressing multiple components or modules of the framework. Therefore, categorizing a mechanism in a specific context emphasizes one prominent characteristic rather than its entire spectrum of functionalities.

Table 5 Comparison of techniques to collect telemetry data from network devices

	Management Plane	Control Plane	Data Plane
Data Configuration and Subscribe	gNMI, NETCONF, RESTCONF, SNMP, YANG-Push	gNMI, NETCONF, RESTCONF, YANG-Push, sFlow	NETCONF, RESTCONF, YANG-Push, NetFlow
Data Generation and Process	MIB, YANG	YANG	IOAM, PSAMP, PBT, AM
Data Encoding and Export	gRPC, HTTP, TCP	BMP, TCP	IPFIX, UDP, SNMP

3.4 Telemetry Data Processing

Network devices serve as the terminal point for several key functions: they generate telemetry data, receive configurations, and initiate data streams for consuming applications (e.g., the NDT or the MANO layer). In the context of NDTs, the software component that performs the data extraction, normalization, and exposure is called a ***Digital Twin Connector (DT-C)*** [37], [38]. This connector provides the essential first layer of cleaning and normalization needed to standardize data exchanged between physical and digital pairs, including physical devices of the same type that may use different data formats and interfaces. A benefit of allowing pre-processing in this step is that it can reduce communication overhead by eliminating the need to transmit raw data, therefore facilitating efficient data management.

From the architectural point of view, these connectors are a composition of protocols and interfaces to extract telemetry data, communication buses to move the extracted data, and a specific set of instances of the telemetry data ingestion, processing, and output. In general, this connector is responsible for:

- **Collecting Data:** Extracts information from physical devices with different data formats and communication protocols.

- **Data Preprocessing:** Performs data cleansing, normalization, and transformation tasks to ensure that the information is consistent and usable by external applications (e.g., MANO or NDT).
    - **Data Reduction:** Processing and filtering data at the source reduces the need to send raw data over the network, thus optimizing bandwidth usage and improving efficiency.

The following sub-section provides an overview of various data processing techniques from recent research on how telemetry data is handled efficiently in network environments.

3.4.1 Compact Data Structures

Compact data structures [39] are essential tools for efficiently managing and analyzing large volumes of network telemetry data, enabling accurate and scalable network monitoring within the constraints of high-speed programmable devices.

- **Count-Min Sketch:** This data structure is widely used for estimating the count of items in a data stream, such as packets or bytes. It employs multiple hash functions to map items to a matrix of counters, providing approximate counts that may be overestimated but never underestimated. This structure is space-efficient and fits well within the constraints of programmable data planes.
- **Bloom Filter:** Used for setting membership queries. Bloom filters employ multiple hash functions to map items to a bit array. They efficiently determine if an item is in a set, with a possibility of false positives but no false negatives. Implementing Bloom filters in data planes requires adaptations to handle multiple memory accesses.
- **Space Saving:** This algorithm tracks the most frequent items (heavy hitters) by maintaining a fixed-size table. It replaces the item with the smallest count when a new item is inserted, ensuring the table always contains the most significant items. Variants like HashPipe and Precision adapt this algorithm for data planes by dividing the table across multiple stages and using probabilistic recirculation.

3.4.2 Advanced Data Structures

Advanced data structures provide sophisticated methods for handling complex telemetry queries, enabling detailed and accurate network performance and security metrics analysis in high-speed environments.

- **Distinct Counting:** Techniques like Linear Counting and BeauCoup estimate the number of distinct items in a stream using hash functions and probabilistic approaches. These methods provide accurate estimates with limited memory, which is essential for high-speed network environments.
- **Entropy Estimation:** Algorithms like AMS sketch and UnivMon estimate the entropy of a data stream, which is useful for detecting anomalies and understanding traffic distributions. These sketches track a random set of flows and maintain their sizes to estimate entropy values.
- **Multi-Metric Sketches:** Solutions like UnivMon and CocoSketch support multiple queries over different keys and metrics, enabling comprehensive network monitoring

with a single data structure. These sketches leverage theoretical advances to estimate a wide range of traffic statistics efficiently.

3.4.3 Relevant Backtracking

The Relevant Backtracking method [40] enhances telemetry data collection by retracing event-related data hop-by-hop along the real path to the edge switch without involving the controller. This method allows upstream switches to obtain remote potential anomalies information and decide what information to report, reducing redundant data uploads.

- **Fast-Slow Packet Mechanism:** Fast packets quickly return to the edge switch as alarm packets, providing immediate alerts, while slow packets collect detailed information hop-by-hop and notify upstream switches about remote anomalies.
- **Custom Cuckoo Filter:** This filter handles high-concurrency data access by allowing easy insertion, update, and deletion of flow data, thus releasing storage pressure and maintaining real-time information.
- **Variable-Length Backtrack Packet Format:** Includes special fields to match collection patterns and collect only relevant data, reducing redundant data collection and extra overhead. This format provides flexibility and scalability for different troubleshooting algorithms.
- **Data Storage and Management:** Detailed flow data is stored in the control plane for long-term backup, and the Cuckoo filter is used for quick lookup, insertion, and deletion of flow data, ensuring efficient data management even in large-scale traffic scenarios.
- **Telemetry Data Collection Optimization:** Selectively uploads only relevant data based on identified potential anomalies, reducing bandwidth consumption. Aggregates telemetry data hop-by-hop back to the edge switch before uploading to the controller, minimizing communication overhead.

3.4.4 Spatial Sampling for In-band Network Telemetry Overhead Reduction

The **Spatial Sampling In-band Network Telemetry (SPAN)** [41] method reduces the overhead of In-band Network Telemetry (INT) by exploiting spatial correlations among telemetry data across different network nodes at the control layer. This approach involves collecting telemetry data from a subset of nodes and reconstructing the missing data using spatial correlations.

- **SPAN Algorithm**: This algorithm includes two main primitives: a sampling strategy and a recovering method. It determines the smallest set of INT data, allowing for accurate network status reconstruction.
  - **Sampling Strategy**: This component determines the smallest set of INT data that needs to be collected to allow for accurate reconstruction of the network status. By identifying the most relevant telemetry data, the sampling strategy minimizes the amount of data that needs to be collected, thereby reducing overhead.
  - **Recovering Method**: Once the relevant telemetry data has been collected, the recovering method reconstructs the complete network status from the sampled

data. This involves using the spatial correlations among the collected data points to estimate the values of the missing data points, ensuring that the reconstructed network status is accurate.

- **Sampling Strategies**: Four strategies are introduced: Maximum of Frobenius Norm, Minimum of the Pseudoinverse Frobenius Norm, Maximization of the Volume, and Data-aided Counters Subset Selection. These strategies aim to select the most relevant telemetry data to minimize reconstruction error.
    - **Maximum of Frobenius Norm (MaxFro)**: This strategy aims to select the rows of the matrix that maximize its Frobenius norm. The Frobenius norm is the square root of the sum of the absolute squares of the matrix elements. By selecting the rows with the largest (L2)-norm, this strategy ensures that the most significant data points are collected. The implementation is straightforward, involving the selection of rows with the largest (L2)-norm, which provides good performance with minimal computational complexity.
    - **Minimum of the Pseudoinverse Frobenius Norm (MinPinv)**: This strategy selects rows that make the sampled matrix spectrally similar to the original matrix. It minimizes the Frobenius norm of the pseudoinverse of the sampled matrix, ensuring that the sampled matrix retains the spectral properties of the original matrix. The implementation involves a greedy algorithm that iteratively selects rows to minimize the Frobenius norm of the pseudoinverse, which requires calculating the singular values of the matrix and selecting rows that maximize these values.
    - **Maximization of the Volume (MaxVol)**: This strategy selects rows that maximize the volume of the parallelepiped formed by the selected rows. The volume is a measure of the linear independence of the rows, and maximizing it ensures that the selected rows are as linearly independent as possible, improving the accuracy of the reconstruction. The implementation uses the determinant of the matrix formed by the selected rows to measure the volume, with a greedy algorithm iteratively selecting rows to maximize this determinant.
    - **Data-aided Counters Subset Selection**: This strategy enhances the performance of the sampling method by leveraging the second-order statistics of the traffic flows. It uses the sample correlation matrix of the traffic flows to inform the selection of rows, incorporating additional information to more accurately identify the most relevant data points. The implementation adjusts the objective function of the MaxVol strategy using the correlation matrix, calculating the determinant of the volume with the correlation matrix and selecting rows to maximize this adjusted determinant.

3.4.5 AI-Driven Telemetry Processing (ADT)

The AI-driven telemetry (ADT) [42] method enhances network monitoring by leveraging Artificial Intelligence (AI) to process a broader set of operational counters, not limited to handpicked KPIs. This method uses an unsupervised approach to detect complex state changes in network devices.

- **DESTIN (Detection Component)**: This component explores otherwise ignored counters and analyzes their dependencies in an unsupervised manner, sending notifications when a state change is detected.
- **Principal Angles**: Measuring the similarity between subspaces formed by high-dimensional telemetry data, helping to detect state changes.
- **Data Pre-processing**: Includes scaling and grouping data based on operational types or patterns to improve detection accuracy.
- **Event Detection and Explanation**: ADT includes components for detecting state changes and explaining the most representative counters associated with detected events, aiding operators in defining new rules and KPIs.

In general, these data processing techniques enable the telemetry data layer to effectively manage and optimize network performance, ensuring accurate and real-time visibility into network conditions. By adapting traditional data structures to the constraints of high-speed programmable network devices, these methods help realize an efficient network telemetry data collection framework aiming to not only enhance the accuracy and reliability of network monitoring but also support proactive network management, leading to more robust and resilient network operations.

4 Data Harmonization Layer

The HDL plays a central role in the architecture, bridging the TDL and the Federated NDT. It ensures that the output from the telemetry systems, which already includes normalized networking data in the form of standardized formats (e.g., aligned with 3GPP or ETSI specifications), is further processed and structured into Smart Data Models. These Smart Data Models form the backbone of the Federated NDT, enabling accurate network modeling, simulation, and management. In general, the HDL has the following functionalities as presented in:

- **Refined Data Ingestion:** Building on the TDL's efforts to standardize raw telemetry data from diverse sources (e.g., different types of routers or network devices), the HDL ingests data that has already undergone an initial harmonization phase.
- **Advanced Data Processing:** The layer transforms the pre-harmonized telemetry data into Smart Data Models. This involves organizing data into well-defined representations that encapsulate the behavior, structure, and interactions of network components in a way that supports simulation, planning, and decision-making processes.
- **Smart Data Model Output:** These outputs serve as the foundation for the Federated NDT, allowing for tasks such as network planning, traffic analysis, optimization, and control. By providing unified, detailed Smart Data Models, this layer ensures that the NDT framework can accurately simulate and optimize network operations.
- **Model Repository Support:** The HDL facilitates the maintenance of a repository of Smart Data Models, which are essential for the NDT's ability to perform continuous learning, prediction, and optimization of network performance.

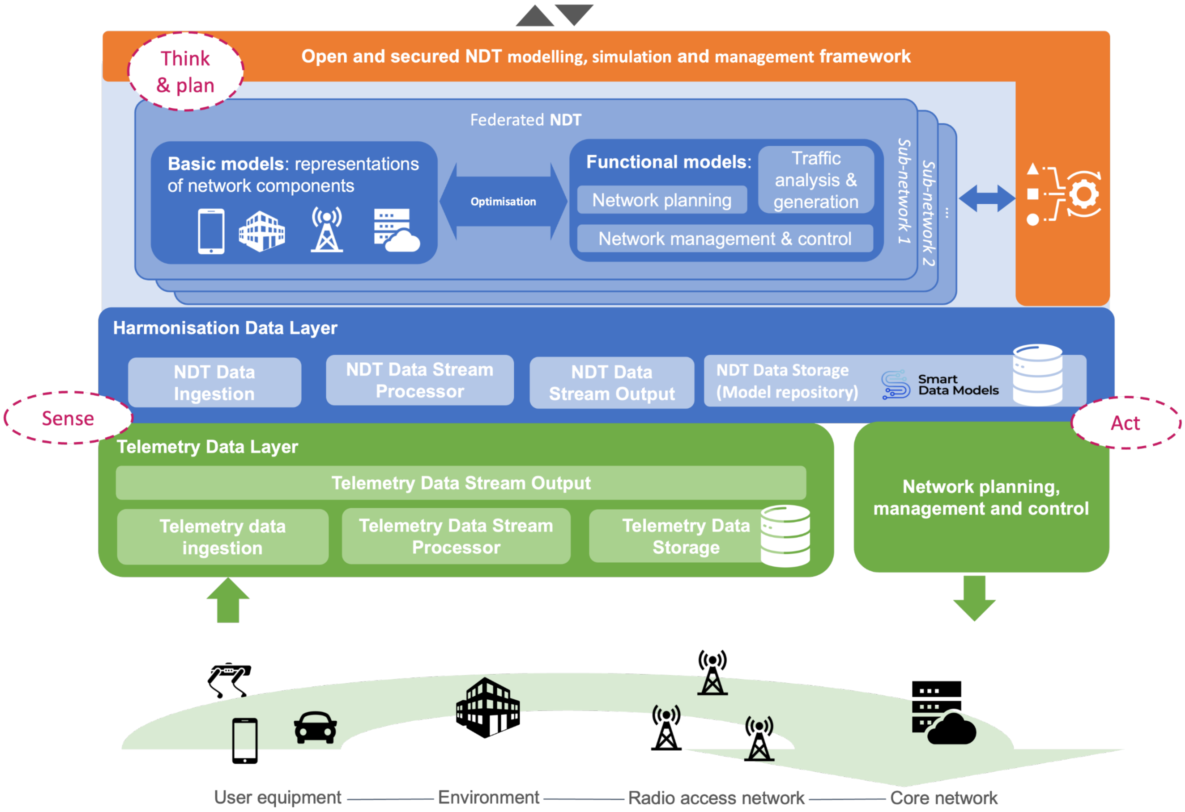

Open and secured NDT modelling, simulation and management framework
Think & plan
Federated NDT
Basic models: representations of network components
Optimisation
Functional models:
Traffic analysis & generation
Network planning
Network management & control
Sub-network 1
Sub-network 2
…
Harmonisation Data Layer
NDT Data Ingestion
NDT Data Stream Processor
NDT Data Stream Output
NDT Data Storage (Model repository)
Smart Data Models
Sense
Act
Telemetry Data Layer
Telemetry Data Stream Output
Telemetry data ingestion
Telemetry Data Stream Processor
Telemetry Data Storage
Network planning, management and control
User equipment
Environment
Radio access network
Core network

Figure 13 6G-TWIN data harmonization layer.

Notice that while the TDL (Section 3) performs the initial step of preprocessing raw telemetry data, e.g., standardizing it across devices, protocols, and formats, the HDL, illustrated by Figure 13, takes this process further, refining and structuring the data into actionable Smart Data Models. These models are critical for the NDT to represent and simulate complex network environments effectively. The layer's ability to standardize, enrich, and organize data is supported by robust data harmonization processes, which ensure consistency and reliability across diverse data inputs. This is achieved through advanced data harmonization tools that facilitate the transformation of telemetry data into unified formats and models. Furthermore, data pipelines are established to manage the continuous flow of harmonized data from telemetry inputs to the NDT, ensuring real-time availability and scalability. The following subsections will present several mechanisms and technologies enabling the HDL to effectively bridge raw telemetry data and network modeling required for high-accurate NDT.

5.1 Data Harmonization and Data Preprocessing

Data harmonization and Data Preprocessing are critical steps in enabling these necessary data pipelines. When dealing with complex systems that require integrating data from diverse sources, these processes are essential for ensuring that data is accurate, clean, and in a format suitable for analysis or modeling. They are particularly important in scenarios where data originates from different systems, sensors, or organizations and must be brought into a unified form to be actionable.

Co-funded by the European Union

Our Data harmonization processes will focus on standardizing and integrating the collected data into a common format/structure. This goal is quite relevant for the case of the NDT, as a single NDT instance can monitor various systems, and this data must be harmonized to allow for interoperability, meaningful analysis, and decision-making.

In the case of our HDL in Figure 13, harmonization processes act upon the data collected by the TDL and the data used for NDT MANO applications. These mechanisms further refine this data towards enabling the Smart Data Model structure, which reduces the complexity required for the NDT to create, store, and access these models. This is achieved through the implementation of techniques and algorithms that deal with:

- **Data Heterogeneity:** Our NDT collects data from different sources, and these sources can have diverse forms, structures, and semantics within this data ecosystem;
- **Data Quality:** Data may come with errors, missing values, or inconsistencies, and NDTs require high-quality data that is accurate, complete, consistent, timely, and relevant to be used for decision-making, analysis, and operational processes;
- **Data Silos:** Isolated or disconnected repositories of data can arise when different business units or applications store and manage their data independently, requiring the NDT to properly integrate or centralize access;
- **Data Granularity:** Some systems overseen by the NDT can produce high-frequency, granular data, while others may only offer periodic, aggregate data. The details included in the collected data can be applied to a variety of data types, including numerical values, time intervals, spatial dimensions, and more.

To achieve data harmonization, different mechanisms have to be included in our NDT framework. **Standardization** mechanisms on our TDL transform the data into a uniform format, such as converting all timestamps to a common time zone, or converting units of measurement to a standard to deal with the heterogeneity of data. Careful **Data Mapping** must happen on our HDL to align disparate datasets, where variables may mean different things in different contexts, handling potential cases of Data Silos. **Data Cleansing** techniques that effectively remove the noise and outliers from collected data and handling missing data (like using interpolation to fill in gaps), ensuring the HDL outputs quality data; and finally, **Data Integration** instruments on our Extract, Transform and Load (ETL) processes, to merge data from diverse systems into a unified platform, responding towards the data granularity issues that might occur at the TDL level.

After deploying the data harmonization mechanisms, the next step is to start data preprocessing, which encompasses transforming raw data into meaningful and usable information. It involves a set of operations to convert data into formats that can be analyzed, visualized, or acted upon to support decision-making, operational processes, or strategic objectives. The primary goal of data preprocessing in our HDL is to make the data collected by our TDL usable, accessible, and valuable. The key mechanisms behind these data processing means are:

- **Data Aggregation:** collecting, combining, and summarizing data from multiple sources or at multiple levels to generate a cohesive and insightful representation. It transforms detailed data into a more manageable and analyzable form by consolidating smaller units of data into larger, summary-based units. The goal of data aggregation is to

provide a higher-level understanding or summary of the data that can be more easily interpreted, analyzed and acted upon.
- **Data Filtering:** encompasses the processes of selecting or excluding specific data points based on certain conditions or criteria, ensuring that only relevant or useful data is kept for further analysis or processing. This process helps to clean and refine the dataset by removing irrelevant, redundant, or noisy data, making it more meaningful and suitable for analysis.
- **Real-Time Analytics:** enables the system to collect, process, and analyze data as it is generated from the actual network in real time. This allows not only operators, engineers, and data scientists to observe and manage the network's behavior instantaneously but also provide information about the zero-touch mechanisms present to automatically make optimal decisions and respond proactively to potential issues, improving performance, reliability, and security.
- **Predictive Analytics:** involves using historical and real-time data collected from the network, alongside statistical algorithms, ML models, and AI functions, to forecast future network behaviors and potential issues, leading to optimal resource allocation and more efficient and reliable network management.
- **Simulation:** Once the data is processed, our NDT simulation tool can be used to test different "what-if" scenarios, such as simulating the effect of a new network configuration or predicting how traffic will flow after deploying new infrastructure.

The separation between the TDL and HDL is necessary, as it allows the 6G-TWIN architecture to process telemetry data independently of the harmonization and preprocessing mechanisms for model building, making this data useful for network performance monitoring without activating the HDL functionalities of the NDT.

4.1.1 Data Harmonization Tools

The first point of contact with the data happens on our TDL: raw telemetry data is ingested by our NDT framework, and the heterogeneity of the multiple domains that the NDT oversees must be dealt with so that the HDL can take this harmonized data in order to preprocess it further for storage or build Smart Data Models to be used by the NDT. For the data harmonization mechanisms, Table 6 below lists some recommendations of tools currently able to be implemented on the TDL, outlining their specific purpose and relevant use cases for the NDT.

Table 6 Data harmonization tools.

Tool	Purpose	Description and use case
Apache NiFi [43]	Data ingestion, routing, transformation, and integration	**Apache NiFi** is a powerful tool for automating the movement and transformation of data between systems. It allows for data ingestion, transformation, filtering, and routing from a variety of data sources (e.g., IoT sensors, network devices, cloud services), allow for the

		normalization and aggregation of this data into a common format. Can be used for real-time data streaming and processing from IoT devices or network equipment to standardize and integrate into our NDT.
Talend Data Integration [44]	Data extraction, transformation, and loading (ETL)	**Talend Data Integration** framework provides comprehensive ETL capabilities for standardizing, cleaning, and transforming data across various sources. It helps harmonize diverse datasets by mapping them into a unified schema, with built-in connectors for cloud, relational databases and network systems. Can be used for the integration between different network performance metrics, sensor data, and device logs into a common format for processing and analysis.
Apache Camel [45]	Data routing and transformation	**Apache Camel** provides a lightweight integration framework that enables data flows between systems, supporting a variety of data formats and protocols (e.g., JMS, HTTP, REST, SQL, etc.), message routing, transformation and aggregation Can be used to route data from different network devices (e.g., routers, switches) and apply transformations to harmonise the data before sending it to the NDT system.
MuleSoft Anypoint Platform [46]	Data integration, API management, and connectivity	The **MuleSoft Anypoint Platform** offers an enterprise-level integration platform that connects various systems, including IoT devices and cloud services, allowing for data to be harmonized via APIs. It also provides data transformation capabilities, schema validation, and mapping for standardizing different formats into consistent, usable data. Can be used towards unifying network device data and sensors by transforming and aggregating them into the desired format for further processing and analysis.
Data Build Tool [47]	Data transformation and modeling for analytics	**dbt** focuses on transforming and preparing data for analytics by applying SQL-based transformations in a version-controlled environment. It's particularly useful for structuring and cleaning large datasets in cloud data warehouses.

		It can handle complex data transformation tasks like normalisation, aggregation, and ensuring that data conforms to a consistent schema. Can be used to transform raw data into an analytical form, aligning different sources of network data into consistent, searchable tables for the NDT.

4.1.2 Data Preprocessing Tools

After harmonizing the data, the HDL further processes the data. These mechanisms enable the transformation of the collected data into Smart Data Models that can be used to build basic models (describing the properties of the physical network, its elements/topology, and other relevant measured data related to its operational status) or functional models (using network information and basic models to predict the behavior of the network within the digital layer). For the implementation of data preprocessing mechanisms, Table 7 below lists some recommendations of tools currently able to be adopted by our NDT architecture.

Table 7 Data preprocessing tools.

Tool	Purpose	Description and use case
Apache Kafka [48]	Distributed event streaming and real-time data processing	Kafka, as outlined in Protocols Supporting Communication Paradigms5.2, is a distributed messaging system used to process real-time data streams. It allows the ingestion of large-scale network telemetry data and supports the real-time processing of data feeds. It also acts as a data pipeline that can process high-throughput events (e.g., network traffic, device status changes) and feed them into real-time analytics or a NDT model. Kafka also has five core APIs (Admin, Producer, Consumer, Kafka Streams, and Kafka Connect) that integrate specific functions on top of the baseline Kafka deployment, to augment its existing capabilities. Can be used to stream network telemetry data from routers, switches, or IoT devices into a centralised system for real-time monitoring and updates, as well as supporting the necessary data pipelines of our NDT.
TensorFlow [49]	Machine learning and AI model development	TensorFlow is one of the leading libraries for ML, enabling predictive analytics, anomaly detection, and classification. It provides tools for training models to predict network failures, capacity planning, and traffic

		forecasting based on real-time and historical data from the NDT. Its machine learning algorithms (e.g., anomaly detection, clustering) can be used in order to analyse network data and predict performance issues, such as potential hardware failures or congested network links.
InfluxDB [50]	Time-series database optimized for handling high-volume, timestamped data	InfluxDB is a time-series database designed to store and query metrics, events, and time-series data from various sources. It is particularly effective for storing and processing network telemetry data, including latency, packet loss, traffic volume, and device health metrics. Can be used to store time-series data from the network and provide high-performance querying and analysis for the NDT to track changes in network conditions over time.
Grafana [51]	Visualization and monitoring of data	Grafana is a popular open-source tool for visualizing time-series data and network metrics that integrates with databases like InfluxDB, Prometheus, and others to create real-time dashboards for network monitoring, offering interactive visualisation capabilities (e.g., graphs, charts, maps). Can be used to create visual dashboards and real-time monitoring views for the NDT, providing a live representation of the network's performance.

4.2 Data Pipelines

The NDT aims to link the physical and digital world by virtually representing the connected network elements. By creating real-time replicas of these elements and their surrounding network topology, the NDT can not only automatically plan the optimal way to manage and control the available resources but also plan the best implementation of network functions and services to be executed. This, of course, raises the complexity of the intended 6G architectures that aim to support the NDT; we need to implement a system that has the required MANO capabilities that can support the different types of devices connected to our network, as well as the means to control and optimize the necessary applications and services running on them. These applications and services will be supported by basic and functional models that are trained and stored on our 6G-TWIN NDT structure.

Throughout this document, one thing can be highlighted as the cornerstone of NDTs: Data. It is data that gives the NDT the information necessary to differentiate the connected devices, the capability to train models and host relevant simulations, and allows the support of the MANO control loops that automatically manage the NDT processes. For this reason, it was necessary to outline this important Distributed Data Exposure and Collection Framework behind the 6G-TWIN NDT in this deliverable.

For this, we started by characterizing the necessary features of the NDT, such as the available Sources from which the NDT will collect Data, the Data Collection framework interfacing between the Data Sources and the MANO component, the Data Harmonization and Data Processing procedures that standardize how the data is handled to be used and stored for the models and Simulation framework in the NDT. Enabling this data flow through the NDT, Data Pipelines are a pivotal element of the architecture.

The Data Pipeline in an NDT refers to the set of processes, technologies, and systems used to collect, transform, store, and analyze data from various sources and feed it into the NDT model. This allows for real-time monitoring, predictive analytics, and decision-making processes based on up-to-date network states, as well as supplying relevant data to the AI functions that support all of the automated and zero-touch management capabilities of the NDT. Table 8 below presents a collection of tools that our NDT framework can adopt to support the necessary Data Pipelines between the physical network and the NDT layer.

Table 8 Data pipelines for the NDT.

6G-TWIN framework	Data Pipeline	Description	Data Pipeline Tools
Telemetry Data Layer (TDL)	Data Collection Pipeline	The data collection pipeline's primary purpose is to gather raw data from the physical network environment, devices, and sensors in real time or in periodic intervals. This data forms the foundation of the NDT by ensuring that the virtual model reflects the current state of the network through: • **Real-Time Monitoring:** The data collected feeds into the NDT model in real-time to provide an up-to-date view of network health, performance, and topology. • **Diverse Data Sources:** It integrates data from a wide range of sources, such as network devices (routers, switches), IoT devices, and edge devices. These devices might provide telemetry data	**SNMP** [16], outlined in 3.1, for monitoring network device performance data. **NetFlow/IPFIX** [24], outlined in 3.1, for traffic analysis and data flow. Telemetry collection through **gRPC** [21], outlined in 3.1**, REST APIs** and **MQTT** [52]**,** outlined in , for edge/IoT devices. **Syslog** [53] for log aggregation.

		(e.g., CPU utilization, memory, bandwidth), performance metrics, event logs, and sensor data. This pipeline will help gather detailed insights into the health and performance of every component in the network, as well as providing data to actively highlight problems, enabling faster response times.	
	Data Ingestion Pipeline	The data ingestion pipeline is responsible for moving data from various sources to the appropriate storage and processing systems. The purpose of this pipeline is to manage the flow of data, ensuring it's reliably delivered for processing in a format suitable for analysis. This is done by: • Real-Time Data Handling (Streaming): For immediate processing and response needs, such as detecting network anomalies or adjusting bandwidth allocation, streaming data pipelines are used. This enables near-instantaneous decision-making. • Batch Data Handling: For less time-sensitive tasks like periodic reports, historical analysis, or in-depth machine learning model training, batch processing can be used. This pipeline will help enhance the scalability of our NDT, by processing and handling large volumes of network data that are collected from the network elements and supporting real-time and offline flexibility, by using real-time and batch processing.	**Apache Kafka** [48], as mentioned in the previous section, using distributed messaging platforms for real-time reporting. **AWS Kinesis** [54] or **Google Cloud Platform (GCP) Pub/Sub** [55], as cloud based solutions for messaging platforms.
Harmonisation Data Layer (HDL)	Data Transformation and Processing	Once data is ingested, it often requires transformation or preprocessing to ensure it's in a usable format for downstream analytics. The data transformation and processing pipeline cleanses,	**Apache NiFi** [43], as mentioned in the previous section, is a data ingestion and transformation tool to process and distribute

		structures, and enriches raw data from different sources before it's stored or analyzed, though: • **Data Normalization:** Ensures that data is in a consistent format, such as converting timestamp formats or normalizing traffic metrics (e.g., Mbps to Gbps). • **Anomaly Detection:** This pipeline may also include preliminary anomaly detection to identify data points that deviate significantly from expected values, signaling potential network issues. • **Enrichment:** Combining raw network data with other relevant datasets (e.g., geographical data, historical performance data) to provide context to the data and improve analytics. This pipeline ensures data consistency from different sources and devices, by harmonizing it into a common format. Processing helps clean up the data, handle missing values, and filter our irrelevant information, increasing reliability across the data collected.	data, perfect for scenarios where data needs to be processed as soon as it arrives. **Apache Airflow** [56], a tool used for managing complex workflows that require precise timing and coordination. It features task management capabilities, extensible programmability through the support of custom tasks and operators in Python, scheduling periodic tasks, monitoring of data workflows with options for alerts and horizontal scalability, important for the NDT use case.
	Data Storage Pipeline	The data storage pipeline is responsible for storing the transformed data in a way that ensures it is both retrievable and accessible for future analysis, querying, and modelling. Given the massive volume of network data, storage solutions must be scalable, fast, and resilient. Some methods that help achieve these requirements are: • **Efficient Querying:** Data must be stored in a manner that supports fast querying, as network data is often used for real-time analytics and decision-making. • **Time-Series Data Storage:** Network data, particularly	**Prometheus** [57], for efficiently monitoring and storing metrics in Time Series Databases. Data lakes like **Amazon S3** [58] and **Google Cloud Storage** [59] for storing raw and semi-structured data. Data warehouses like **Amazon Redshift** [60] or **Google BigQuery** [61] for structured data and analytics.

		performance metrics (e.g., bandwidth, latency), often needs to be stored with time-series indexing, allowing analysts to track changes over time and detect trends. • **Data Retention Policies:** Depending on the criticality of the data, this pipeline may also manage data retention, storing critical data for longer periods while archiving or discarding less important data. Through appropriate storing techniques, the NDT is able to optimize its performance, quickly being able to retrieve the right data for analysis. This optimization allows for the NDT to increase the amount of data over time, improving its scalability.	
	Data Analysis and Simulation Pipeline	The analysis and simulation pipeline uses the stored and processed data to provide actionable insights. The primary purpose is to leverage advanced analytics, machine learning, and simulation techniques to simulate the network's behaviour and predict potential issues. The mechanisms used are based on: • **Descriptive Analytics:** Describes what is happening in the network at a particular moment, such as generating reports on traffic patterns or network health. • **Predictive Analytics:** Uses machine learning models to forecast future network behaviours, such as congestion, device failures, or resource bottlenecks, and anticipates issues before they happen. • **Prescriptive Analytics:** Suggests actions or decisions based on the analysis, such as rerouting traffic or adjusting bandwidth to optimize	Data Science frameworks like **TensorFlow** [49] or **PyTorch** [62] for training and deploying predictive machine learning models. **Grafana** [51] for visualizing network data.

		performance and reduce latency. This enables our NDT to be proactive in its decision-making, by being capable of predicting potential issue, enabling automation mechanisms or manual operators to act on network issues before they impact its performance and user experience. This data analysis also helps optimize resource allocation, through improved network design and enhanced performance based on these network insights.	
	Data Management Pipeline	The management pipeline connects the data and insights from the NDT with the operational network management functions, enabling automation and real-time decision-making based on the NDT's recommendations. This pipeline allows for closed-loop operations and zero-touch maintenance services, where actions are taken automatically based on insights from the NDT. • **Automation:** The insights from the NDT can trigger automated actions within network management systems. For instance, predictive models might indicate a failure is likely in a specific device, prompting the system to reroute traffic or provision additional resources to prevent downtime. • **Alerting and Notifications:** If the NDT detects a potential issue (e.g., an impending network failure), it can send alerts to network operators or trigger automated remediation procedures. • **Dynamic Network Adjustments:** Based on real-time data, network parameters like traffic routing, Quality of Service (QoS), or resource	SDN Controllers like **ONOS** [63], **OpenDaylight** [64] or **RYU-SDN** [65] that dynamically adjust the network based on insights from the NDT. **NFV** [66] mechanisms and technologies, making the NDT capable of deploying virtualized network services that can be adjusted based on NDT insights.

	allocation may be adjusted automatically. By automating actions based on NDT predictions, this pipeline reduces the time and manual effort required to act on network issues, decreasing the response time required to take action. Given that automated response mechanisms are faster and more efficient than manual intervention, the risk of network downtime or lower levels of network performance are diminished.	

5 Data Communication Buses

This section explores the concept of data communication buses, which are essential for facilitating efficient data transfer and integration within the 6G-TWIN architecture. Robust communication frameworks become critical as networks evolve to support increasingly complex applications and services. This section outlines the various architectural models, paradigms, protocols, and platforms enabling seamless data exchange between different network components, ensuring that information flows smoothly and reliably across diverse systems.

5.1 Communication Paradigms

Communication paradigms define how data flows between producers, consumers, and intermediaries in a system (in our case, between the physical system, its digital replica, and connected systems or users). The choice depends on the twin's requirements for **real-time data exchange**, **scalability**, **reliability**, and **flexibility**. The primary paradigms include [67], [68]:

- **Publish-Subscribe Model [69]:** Data producers (publishers) send messages to a central broker within this model. Consumers (subscribers) receive messages based on topics or events they are subscribed to.
- **Event-Driven Architecture [69]:** Within this model, components communicate by generating and responding to events. The events are captured and propagated to interested systems.
- **Client-Server Model:** Within this model, a centralized server provides data or services to multiple clients upon request.
- **Data-Centric Model (e.g., DDS - Data Distribution Service):** Within this model, data producers and consumers share data directly, organized around a common data model. There is no central broker; data flows peer-to-peer based on QoS policies.
- **Microservices-Based Architecture:** Within this model, components of the digital twin are developed as independent services that communicate via APIs or a message bus.

Table 9 compares these four models, where we can conclude that selecting the right architecture often involves a combination of these models tailored to the twin's specific requirements.

- **Publish-Subscribe** and **Event-Driven** models are most effective for real-time, flexible, and scalable communication.
- **Data-centric models (e.g., DDS)** are ideal for low-latency, high-frequency synchronization in complex systems.
- Use **Client-Server** or **Microservices** for simpler setups or when modularity is essential.

Table 9 Comparison of network digital twin model architectures.

Model	Advantages	Drawbacks	Best Use Cases
Publish-Subscribe	• Enables decoupled communication between components (e.g., sensors/devices/.., twin models, analytics tools). • Supports dynamic scalability as new components (producers or consumers) can join without disrupting existing connections. • Ideal for real-time updates and event-based communication.	• Requires a reliable broker, adding complexity. • Latency can increase if the broker is overloaded.	Real-time updates, sensor data streaming, distributing analytics results.
Event-Driven	• Supports asynchronous communication, reducing system bottlenecks. • Aligns naturally with real-world events triggering updates in the digital twin (e.g., machine status changes or environmental shifts). • Facilitates integration with analytics and machine learning systems for event-triggered insights.	• Requires careful design of event-handling mechanisms to avoid missed or duplicate events. • Can become complex to manage as the number of events and components grows.	Event-triggered updates, anomaly detection, reactive maintenance workflows.
Client-Server	• Simple and well-understood architecture. • Useful for periodic data polling or centralized management of the twin.	• Limited scalability as the number of clients grows. • Less suitable for real-time, continuous updates compared to event-driven or publish-subscribe models.	Batch data requests, periodic reporting, centralized twin management.
Data-Centric (DDS)	• Low-latency communication with guaranteed delivery. • Scalable and robust for real-time systems.	• High complexity. • Requires precise QoS configuration.	Simulation-intensive twins, autonomous systems, industrial control twins.

	• Ensures consistency and synchronization between physical and digital systems.		
Microservices-Based	• Provides modularity and flexibility to scale or update individual components. • Enables integration of heterogeneous systems and technologies.	• Increased communication and orchestration overhead. • Higher complexity to maintain.	Multi-component twins, cloud-based deployments, integration with third-party tools or systems.

5.2 Protocols Supporting Communication Paradigms

Communication paradigms are realized through specific protocols tailored to various system needs. Table 10 provides a quick overview to help choose the most suitable protocol based on the specific needs of your NDT implementation [52], [70], [71], [72], [73], [74], [75], [76].

Table 10 Overview of protocols for network digital twin implementation.

Protocol	Best Suited For	Advantages	Drawbacks
MQTT	Real-time, lightweight communication with IoT devices	• Low overhead and bandwidth-efficient. • Supports QoS for reliable delivery. • Publish-subscribe model for flexibility.	• Limited interoperability with non-IoT systems. • Not ideal for complex data transactions.
AMQP	Reliable, secure, and efficient message delivery in complex systems	• Standardized protocol with strong queuing and routing. • Guarantees message delivery and acknowledgment. • Cross-platform compatibility.	• Higher overhead than MQTT. • More complex implementation and management.
HTTP/REST	Interoperability with web-based applications (based on client-server paradigm) or APIs.	• Widely used and easy to implement. • Stateless architecture simplifies scalability. • Ideal for exposing twin data to external applications.	• Not optimized for real-time communication. • Higher latency and bandwidth usage.
CoAP	Resource-constrained environments	• Lightweight and optimized for low-power devices. • Low latency via UDP.	• Limited functionality compared to HTTP or AMQP.

		• Supports multicast for efficient updates.	• Requires manual handling of reliability.
WebSockets	Real-time, bidirectional communication	• Full-duplex communication over a single TCP connection. • Ideal for real-time synchronization with frequent updates.	• Less standardized for IoT use cases. • Not optimized for unreliable networks.
DDS	High-performance, real-time systems	• Designed for low-latency and real-time needs. • Scalable with advanced QoS features. • Supports dynamic data discovery and sharing.	• High complexity and learning curve. • Overkill for simple digital twin implementations.
Kafka (binary over TCP)	High-volume data streaming, event-driven architectures	• Scalability • High throughput • Fault tolerance • Persistence • Event-driven	• Complexity • Latency • Resource-intensive • Setup overhead
Zenoh	Low-latency distributed systems	• Low-latency • Scalable • Flexible	• Newer • Infrastructure-specific

The best protocol for a data bus in a network digital twin environment depends on the system **requirements**. For **large-scale, high-throughput** data streams, **Kafka** is the optimal choice, while **Zenoh** offers **low-latency and distributed communication** for edge computing. For simpler environments with **low-bandwidth requirements**, **MQTT** is well-suited. **AMQP** excels in enterprise applications **requiring reliable, secure**, and transactional messaging. **REST** remains a viable option for **web services** but unsuitable for real-time streaming.

5.3 Data Management: Filtering, Prioritization, and Serialization

Effective data management ensures that digital twin systems focus on relevant information while minimizing overhead and latency. Key aspects include: (i) **Filtering Mechanisms:** Applied at the source or middleware, filtering includes threshold-based rules, content-based filtering, and ML-driven anomaly detection. (ii) **Prioritization Techniques:** Ensures timely delivery of critical data through QoS levels, traffic shaping, and priority-based scheduling. (iii) **Serialization and Deserialization**: Compact formats ensure faster processing and lower latency.

These mechanisms collectively optimize the performance and relevance of data streams within network digital twin systems. In the following, we detail the two concepts:

- To reduce noise and focus on relevant data streams, filtering mechanisms can be applied at various stages of the data flow in an NDT [67], [77]:
  - **Source-Level Filtering on the DT Connector:**
    1. Threshold-Based: Ignore data outside predefined ranges.
    2. Event-Driven: Transmit data only when specific conditions are met.
    3. Frequency Filtering: Send updates at defined intervals.
    4. Content-Based: Forward messages based on relevance.
  - **Middleware Filtering on the Data Bus:**
    1. Rule-based filtering by applying logical conditions.
    2. Temporal Aggregation: Average data over time windows.
    3. Statistical Summarization: Share only key metrics (e.g., min/max).
    4. ML-Based Filtering: Use models for anomaly detection or relevance classification.
    5. Semantic Filtering: Utilize context, ontologies, or knowledge graphs to focus on data relevant to specific simulations.
- To ensure that critical data is transmitted with higher urgency in an NDT, several prioritization techniques can be used [78]:
  - **Application Layer Prioritization**: Prioritize data at the application layer before transmission by the DT Connector.
  - **QoS:** Assign different priority levels to data packets (e.g., use QoS 2 for critical data in MQTT).
  - **Traffic Shaping:** Allocate bandwidth for high-priority data, ensuring critical data gets priority during congestion.
  - **Packet Tagging:** Use packet tags (e.g., DiffServ or MPLS) to classify data based on its urgency.
  - **Deadline Scheduling (DMS, EDF, LLF, etc.):** Prioritize time-sensitive data over less urgent data.
  - **Priority-Based Queues:** Sort data into different queues, with higher-priority data being transmitted first.

  These techniques help ensure that critical data is transmitted in a timely manner, improving the performance and reliability of the NDT system.

- Ensuring efficient data **serialization** and **deserialization** in an NDT involves selecting suitable methods and technologies that minimize overhead, reduce latency, and maintain data integrity while being adaptable to the twin's requirements. Formats like JSON and XML should be avoided because, although they are human-readable, they are verbose, which leads to increased data size and longer parsing times [70].

5.4 Monitoring and Performance Metrics

To assess the performance of the data bus, the following KPIs should be tracked [79], as shown in Table 11:

Table 11 KPIs for assessing data bus performance.

KPI	Description	Why It Matters
Latency	Time taken for data to travel across the system	Low latency is crucial for real-time decision-making.
Throughput	Volume of data transmitted per unit of time	High throughput is needed for handling large-scale data.
Packet Loss Rate	Percentage of lost packets during transmission	High packet loss impacts data integrity and system reliability.
Jitter	Variation in data latency	Consistent latency ensures stable communication.
Data Integrity	Ensuring accuracy and consistency of transmitted data	Reliable data ensures system accuracy.
Bandwidth Utilization	The proportion of available bandwidth being used	Efficient use of resources without overloading the system.
Reliability/Availability	System uptime and availability	Continuous availability is essential for system performance.
Queue Lengths and Buffering	Size of data queues and buffers	Long queues may introduce delays and increased latency.
Scalability	Ability to handle increased data volume without performance drop	Ensures the system can grow without losing efficiency.
Data Transfer Error Rate	Errors encountered during transmission	Low error rates ensure reliable data transmission.

To **implement monitoring tools** for real-time insights into the data bus performance and health, some of the following strategies should be considered:

- **Data Collection and Monitoring Agents:** Deploy distributed agents and network probes to gather performance data such as latency, packet loss, and throughput.
- **KPIs Tracking:** Use tools like Grafana or Kibana to visualize KPIs such as latency, throughput, and packet loss in real time.
- **Visualization and Reporting:** Use dashboards to display real-time and historical data, helping track trends and identify performance bottlenecks.
- **Alerting Systems:** Implement integrated alerting tools to notify when performance thresholds are exceeded.
- **Scalability and High Availability:** Utilize cloud-based monitoring for scalability and redundant systems to ensure continuous monitoring.

5.5 Protocols and Platforms: Integration and Performance Insights

Messaging systems like Kafka, **Zenoh**, **RabbitMQ**, and **NATS** [76] serve not only as communication tools but also as platforms that integrate messaging with infrastructure capabilities. Protocols like MQTT or AMQP are at the core of these systems, enabling efficient and reliable communication:

- **Kafka:** A distributed event streaming platform, Kafka handles high-throughput data streams and real-time analytics, which is ideal for large-scale digital twins.
- **Zenoh:** Combining protocol efficiency with platform-level features, Zenoh unifies communication, storage, and computation for low-latency edge computing.
- **RabbitMQ:** Implements AMQP and other protocols, offering robust message queuing and enterprise-grade reliability.
- **NATS: It** is a lightweight, open-source, cloud-based messaging system that supports various messaging models, including publisher-subscriber, request-reply, and messaging queue models, and is known for its simplicity, high performance, and scalability.

These platforms extend basic protocol capabilities to support advanced use cases, such as interoperability through middleware and protocol gateways. For example, Kafka can transform data formats or act as a bridge between systems using different protocols. In terms of performance, a recent study on the protocols mentioned above highlights that Zenoh performs well in terms of throughput and latency in certain configurations, making it a strong candidate for real-time digital twin applications [77]. However, Kafka remains the go-to option for large-scale streaming applications, especially in environments requiring reliable event processing across distributed systems. With its lightweight and high-performance design, NATS is ideal for cloud-native applications requiring low latency and high throughput. The choice of the platform and protocols to serve the data collection and messaging will depend on specific system requirements such as scalability, real-time performance, and infrastructure complexity.

6 Data Management

6G networks evolve to accommodate unprecedented complexity and heterogeneity, and efficient and secure data management becomes a cornerstone for their success. Within the NDT-enabled 6G architecture, data management is crucial, ensuring that diverse data sources are seamlessly integrated, processed, and utilized to support real-time decision-making, optimization, and orchestration. This section delves into the critical functions of data management within the 6G-TWIN management and orchestration layer, highlighting its significance in meeting the challenges of scalability, reliability, and security.

Data management is crucial to ensure data integrity, availability, and usability across a range of systems and applications when data is exposed to unauthorized parties. Data management is critical in today's dispersed, cloud-based, or multi-cloud scenarios, where data volume, complexity, and variety are constantly increasing. Exposed data may include sensitive data, such as Personally Identifiable Information (PII), which must be protected. To ensure secure data management, several requirements should be considered. Controlling who can access data and what actions they can perform is a critical component of data security. This prevents unauthorized users from accessing sensitive information. Role-based access control or multi-factor authentication can be used to securely manage user access across different systems. The integrity of the data is also a fundamental aspect of security; cryptographic hashes and checksums can verify that data has not been tampered with. In addition, to make sure that data follows data privacy regulations, techniques like data masking and anonymization can be utilized.

As systems grow, more entry points and data flows can result in more opportunities for malicious parties to exploit vulnerabilities, thus increasing the attack surface. A large scale and widespread exposure of sensitive data can affect many users and systems if security measures fail to scale with the system. Ensuring data security in distributed systems is crucial due to the wide surface area exposed by multiple nodes across different locations. In distributed systems, one requirement is to maintain data consistency across multiple nodes. Also, it is important to make sure that the sensitive data is not shared between distributed nodes.

Figure 14 illustrates the core components of the data management system, each designed to address specific aspects of the data lifecycle. The security module ensures that data remains protected throughout its lifecycle, employing robust encryption and access control measures. Federated learning facilitates distributed machine learning across multiple nodes, enabling collaborative intelligence without compromising data privacy. The scalability function ensures the framework can accommodate increasing volumes of data, users, and devices, maintaining performance efficiency even in high-demand scenarios.

Key processes such as frequency management and data cleaning are essential for ensuring data quality and relevance. While frequency management regulates data collection intervals to optimize resource usage, data cleaning eliminates inconsistencies and errors, ensuring accuracy in analytics and decision-making. Parallelization and update synchronization further enhance the system's capability to process large-scale data in real-time, ensuring that updates are consistently propagated across all network components.

Together, these components form a cohesive framework that underpins the successful operation of NDT within 6G architecture. By integrating these functions, the 6G-TWIN project addresses critical data processing, harmonization, and security challenges, paving the way for a robust and intelligent network management ecosystem. The subsequent sections will provide an in-depth analysis of these functions and their implementation within the 6G-TWIN framework.

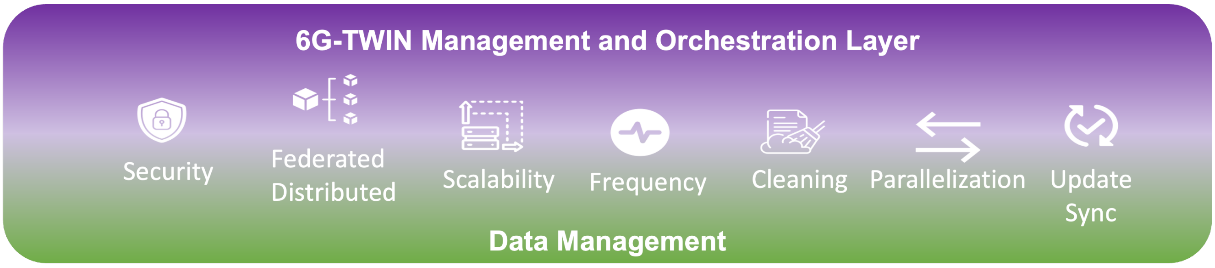


Figure 14 Data management functions inside the 6G-TWIN management and orchestration layer.

One of the management “plane” blocks should take care of data. This section will review techniques/enablers for performing data management and their extended/required capabilities.

6.1 Management System for Scalability, Reliability, and Parallelization

The data collection and exposure layer is a critical component that bridges all subsystems within an NDT [80], as shown in Figure 3. This layer serves as a point of ingestion of the original data and of return at the right time to direct the interactive optimization process resulting from their interaction. Existing works provide the functionality to manage different aspects of data, such as data cleaning, quality assessment, transformation, integration, and search, among others. These data management functionalities are either explicitly encompassed in a dedicated data ingestion layer or scattered in other components of the NDT [81].

The term “Big Data" is used to label data management according to different attributes as proposed by Curry et al. [82]: volume, velocity, and variety. Volume requires dealing with large scales of data within data processing. Velocity involves dealing with high-frequency streams of incoming real-time data (e.g., sensors, IoT). Variety implies handling data using differing syntactic formats (e.g., spreadsheets, XML), structured and unstructured data (e.g., tabular data, texts, videos), schemas, and meanings. As the field matured, other properties were included [83], among them Veracity and Value. Veracity refers to the truthfulness or reliability of the data, while Value is the measurement of data usefulness that determines the discovery of hidden values from the collected data [80].

Then, in the era of big data, efficient parallel and concurrent algorithms and implementation techniques are needed to meet the scalability and performance requirements entailed by scientific data analyses [84]. Challenges such as scalability and resilience to failure are foundational properties of any robust data system, ensuring its ability to handle growing demands while maintaining operational integrity. New Big Data problems relate to users handling too many files and/or working with very large files. Applications need fast movement and operations on that data, not to mention support to cope with an incredible diversity of data. Big Data issues also emerge from extensive data sharing, allowing multiple users to explore or analyze the same data set. All these demand a new movement and a new set of complementary technologies.

Therefore, scalability, reliability, and parallelization are the foundational properties of any robust data system, ensuring its ability to handle growing demands while maintaining operational integrity [85]. Below, we explore the significance of these properties, their measurement, and how key data ingestion platforms—Kafka, Zenoh, and RabbitMQ—address them.

6.1.1 Why Are These Properties Critical?

Scalability ensures a system can handle increased workloads—whether by adding more resources or efficiently utilizing existing ones. This is crucial as data volumes and system complexity grow, particularly in environments like IoT and cloud-native applications. Reliability ensures data is transmitted and processed without loss or corruption. This is critical for maintaining trust and accuracy in systems involving sensitive or critical data, such as financial transactions or healthcare records. Parallelization refers to the ability of a system to handle multiple operations concurrently, a necessity for modern applications that process high-throughput, low-latency workloads.

Scalability in a distributed system has three different dimensions. Size scalability refers to the ability to expand a system by adding more components (i.e., increasing capacity) without causing a decline in performance. However, challenges can arise when scaling, such as limitations in the system's computational capacity (e.g., CPUs), storage capacity (e.g., disk space), and network bandwidth (e.g., the connection between users and the system). If these resources are not adequately managed, they can negatively impact the system's ability to scale effectively.
Geographical scalability means different system components can lie geographically apart, but the user should not notice the delay between components. Geographical scalability is hard to implement with synchronous communication; hence, asynchronous technologies are typically used. Administrative scalability means that the system needs a safe way to handle multiple independent administrative domains that govern the system. Administrative scalability faces problems related to policies of resource usage, management, and security [86].

Reliability in a distributed system refers to its ability to function correctly and deliver its intended services even in the presence of faults or failures. A reliable distributed system ensures that data is not lost, operations are performed as planned, and failures are handled gracefully, maintaining consistency and availability as much as possible. The reliability of a system is also expressed by the message delivery guarantees. Reliability may include guarantees like **at-most-once, at-least-once, and exactly-once message semantics** for messaging systems. This allows developers to choose the level of reliability that suits their application requirements.

- **At most once**: Messages may be lost but are never redelivered. This corresponds to QoS level 0. QoS level 0 represents the best-effort delivery of the message.
- **At least once**: Messages are never lost but may be redelivered. This corresponds to QoS level 1.
- **Exactly once**: this is what people actually want, each message is delivered once and only once. This corresponds to QoS level 2.

It is worth noting that reliability in a system can be broken down into four key aspects: durability, availability, fault tolerance, and consistency. Durability refers to the guarantees that data will not be lost even in the event of system crashes or power failures, often implemented through persistent storage or replication. Availability ensures that the system remains operational and accessible, minimizing downtime and reducing Mean Time To Recovery (MTTR). Fault tolerance means that the system can continue operating correctly despite failures in components such as hardware, software, or network elements, typically achieved through replication, failover mechanisms, and redundancy. Consistency ensures that all parts of the system agree on the state of the data, which is crucial in systems requiring strong consistency (e.g., databases adhering to ACID – Atomicity, Consistency, Isolation, Durability- properties).

Finally, scalability and reliability cannot exist without parallelization [84], which refers to a system's ability to execute multiple tasks or processes simultaneously. The goal of parallelization is to improve performance, reduce execution time, and optimize resource utilization by leveraging the system's distributed nature. In a data ingestion layer, data should be divided into chunks that can be processed independently in parallel. Thus, data partitioning strategies are common in distributed data processing systems like MapReduce, Spark, or Flink. Moreover, the tasks and data are evenly distributed across nodes to prevent bottlenecks and optimize resource usage. Additionally, the data ingestion layer should coordinate the parallel execution of tasks to ensure correctness, especially for tasks that share dependencies or require aggregation of results. Combined, these properties form the backbone of a secure and distributed data management ecosystem, enabling resilience, performance, and adaptability.

The previous sections showed that Apache Kafka, Zenoh, and RabbitMQ are the most suitable platforms for realizing the data communication buses. In the next subsections, we will briefly summarize the scalability, reliability, and parallelization properties that are enforced by these three main platforms.

6.1.2 Kafka

Kafka is a distributed event-streaming platform known for its scalability, reliability, and support for parallelization. It can be deployed on a single workstation or a multi-node cluster, offering flexibility based on system requirements. Its architecture decouples producers and consumers, allowing them to operate independently, which is critical for achieving high scalability [74].

Most data communication platforms are configurable via config files following a key, value format as proposed by XML, JSON, and YAML. The parameters that can be configured vary depending on the communication platform. Kafka uses ZooKeeper to manage metadata, distribute partitions, and ensure fault tolerance by assigning master-slave roles to brokers. ZooKeeper is a centralized service in Kafka that manages cluster metadata, ensures synchronization, and provides robust coordination among distributed systems. It plays a critical role in maintaining the state of Kafka cluster nodes, topics, and partitions, thereby supporting Kafka's reliability and scalability.

Key functions of ZooKeeper in Kafka include controller election, which manages the leader-follower relationship among partitions and ensures seamless failover by electing a new controller when a node shuts down. It also handles the configuration of topics, maintaining details such as the number of partitions, replica locations, and leader preferences.

ZooKeeper enforces security and access management through Access Control Lists (ACLs), which regulate permissions for all topics in the cluster. Additionally, it tracks the membership of the cluster, maintaining an updated list of active brokers to ensure efficient resource allocation and system reliability.

Through these capabilities, ZooKeeper ensures that Kafka operates with high fault tolerance, consistent configuration management, and streamlined synchronization, which is essential for distributed data streaming environments.

Scalability in Kafka is achieved through partitioning topics and replicating these partitions across brokers. ZooKeeper maintains information on In-Sync Replicas (ISRs), ensuring fault tolerance by promoting a replica to the leader role if the current leader fails. Tools like MirrorMaker facilitate geo-replication, enabling horizontal scaling across geographically dispersed clusters while maintaining message ordering within partitions.

Reliability is a core aspect of Kafka, with guarantees provided through ISRs and configurable levels of message durability. Kafka offers three delivery semantics: at-most-once, at-least-once, and exactly-once [87]. Its transactional producer/consumer model supports exactly-once semantics, which is crucial for data consistency applications. Kafka retains messages in a durable log, allowing replay and recovery during failures.

Parallelization is inherent to Kafka’s design, with partitions enabling concurrent processing by multiple consumers in a consumer group. This design ensures high throughput and efficient load distribution. Kafka supports compression algorithms like Gzip and Zstandard to optimize storage and network usage, further enhancing its ability to handle high-volume data streams.

6.1.3 Zenoh

Zenoh [88] is a fully distributed system designed for efficient and flexible peer-to-peer communication, avoiding bottlenecks caused by centralized brokers, such as Kafka [77]. Its dynamic configuration capabilities allow for seamless integration across various network topologies, such as mesh or routed setups, without requiring changes to application logic. This adaptability extends to internet-scale networks, enabling Zenoh to scale easily without global re-initialization, making it suitable for geographically dispersed devices.

Zenoh achieves scalability by leveraging unicast communication, outperforming multicast in high-load scenarios. Its routing protocol ensures that a single producer can reliably send data to multiple consumers while maintaining a consistent reliability. While Zenoh's peer-to-peer model eliminates many constraints of centralized systems, its router-based configurations can also support persistence and enhanced data delivery guarantees, especially when integrating with protocols like MQTT. Zenoh's scalability has been tested [70], [89], showing that Zenoh is capable of handling the participation of new devices regardless of their geographic locations and without global re-initialization. Moreover, multicast communication, used in DDS systems can exhibit worse performance under a high load (a greater number of participants) than unicast communication, used in Zenoh [90].

Reliability in Zenoh is supported through a multi-layered protocol. The session protocol establishes reliable, bidirectional communication between runtimes, while the routing protocol handles data propagation. Zenoh offers "hop-to-hop" reliability by default, where data integrity is maintained during stable operations. However, during topology changes, data samples may be lost. For systems requiring higher reliability, Zenoh provides end-to-end and first-router-to-last-router strategies, ensuring data delivery even during disruptions at the cost of scalability and resource consumption [91].

Zenoh incorporates advanced flow control mechanisms to balance reliability, progress, and memory usage. It allows receiving applications to specify resending strategies and selectively propagate data reliably to specific subscribers. Meanwhile, producers and routers manage memory allocation for reliability and define congestion strategies, such as message dropping or blocking, to mitigate system slowdowns caused by misbehaving components.

Zenoh provides a series of APIs in different programming languages, which can be used to configure the session [92]. Moreover, just like Kafka, Zenoh uses a configuration file and plugins to control its infrastructure. Currently, JSON5 and YAML are the primary configuration formats, but additional support for other serialization formats may be added in the future. Within its plugins, we highlight the storage manager plugin [93], which allows Zenoh to store values associated with a set of keys (data in Zenoh jargon), allowing other nodes to query the most recent values associated with them. Through the configuration file, access control policies can be easily set up. Access control enables Zenoh instances to filter (allow or deny) messages depending on certain characteristics of individual messages and their respective source or destination. On the other hand, Authentication allows Zenoh instances to identify certain characteristics in other instances they connect to, which are used to match the remote instances with configured subjects in the ACL policies and apply the rules accordingly to the exchanged messages.

In summary, Zenoh offers a highly scalable and reliable framework for distributed communication, with the flexibility to tailor its behavior to diverse application requirements. Its separation of reliability and congestion control, combined with robust routing and failover mechanisms, ensures efficient data transmission and adaptability in dynamic environments.

6.1.4 NATS

NATS (Neural Autonomic Transport System) [76] is a lightweight, high-performance messaging system for distributed applications. NATS is built in the Go programming language and facilitates real-time communication between services, applications, and devices. It supports various messaging models, including publish-subscribe, request-reply, and messaging queue models. NATS is known for its simplicity, speed, and scalability, making it ideal for cloud-native systems, IoT, and AI-driven workloads.

NATS is designed with scalability in mind, allowing it to handle a growing number of tasks and connections efficiently. Its architecture supports clustering, where multiple NATS servers can work together to distribute the load and ensure high availability. This clustering capability enables NATS to scale horizontally, adding more servers to handle increased traffic without significant changes to the existing architecture. Additionally, NATS uses a lightweight protocol that minimizes overhead, further enhancing its scalability.

In terms of reliability, NATS provides several mechanisms to achieve high reliability, including message persistence and durable subscriptions. With NATS Streaming (JetStream), messages can be stored on disk, allowing for message replay and ensuring that no data is lost even if a subscriber is temporarily unavailable. Moreover, NATS’ publish-subscribe model decouples producers and consumers, allowing them to operate independently and in parallel. NATS also supports load balancing by distributing messages across multiple subscribers, ensuring that no single node becomes a bottleneck. This parallel processing capability allows NATS to manage high-throughput applications that require real-time data processing.

6.1.5 RabbitMQ

RabbitMQ achieves scalability by replicating broker instances in a master/slave topology, ensuring message availability even if the master broker fails. This replication is implemented using mirrored queues [94], which replicate messages to slave brokers and delete them once acknowledged. Quorum queues, a newer alternative to mirrored queues, offer improved reliability and throughput by retaining messages in memory and on disk, though with higher latency. Additionally, RabbitMQ supports cluster-wide replication of users, exchanges, and bindings, making all queues visible across the cluster.

Reliability in RabbitMQ is ensured through features like durable queues, persistent messages, and acknowledgments. Durable queues and exchanges allow for recovery after system restarts, while message persistence ensures that data survives broker failures if explicitly enabled by the publisher. Acknowledgments and confirms verify message delivery [95], with the latter used for persistent queues. RabbitMQ also offers lazy queues [96], which store messages in disk storage immediately to optimize memory usage and maintain reliability during high workloads [94].

RabbitMQ supports multiple QoS levels as defined by AMQP and MQTT protocols, including at-most-once and at-least-once delivery guarantees. Messages can be redelivered if an acknowledgment is not received, though this may result in duplicates. The platform also incorporates features like priority queues, allowing prioritized message processing to enhance throughput and adaptability in resource-intensive scenarios.

Parallelization is supported through RabbitMQ’s queueing model, where tasks can be distributed across consumers connected to different brokers. However, strict FIFO ordering is only guaranteed within individual queues, meaning cross-queue ordering should not be relied upon. RabbitMQ clusters facilitate efficient workload distribution, and the "shovel" and "federation" features extend their reach to distributed systems.

While RabbitMQ provides strong scalability, reliability, and parallelization, there are some limitations. It does not natively support inter-cluster message compression, and strict message ordering is not guaranteed beyond single queues. Despite these challenges, RabbitMQ remains a robust and versatile messaging system for a wide range of applications. Table 12 summarizes how Kafka, Zenoh, and RabbitMQ address the main foundational properties of any robust data ingestion layer.

Table 12 Summary of how Kafka, Zenoh and RabbitMQ handle scalability, reliability and parallelization.

Framework	Scalability	Reliability	Parallelization
Kafka	Kafka uses a partitioned log approach, distributing messages across brokers. Zookeeper manages broker metadata and enables horizontal scaling by adding brokers, ensuring high throughput for large-scale applications.	Kafka achieves reliability through in-sync replicas, ensuring data persistence across broker failures. Its replication mechanisms and idempotent producers support exactly once semantics for transactional processing.	Kafka's architecture, centered on topic partitioning, supports parallel processing by decoupling producers and consumers. Consumer groups efficiently distribute workload across multiple instances.
Zenoh	Designed as a peer-to-peer system, Zenoh scales flexibly by	Zenoh provides hop-to-hop reliability by default and supports stronger	Zenoh's protocol leverages automatic batching and

	establishing direct connections between nodes. It avoids centralized bottlenecks and supports dynamic network topologies for broad geographical reach.	guarantees like end-to-end reliability. This flexibility allows users to choose reliability levels suited to their needs.	fragmentation, optimizing data transfer and supporting concurrent operations across distributed nodes.
RabbitMQ	RabbitMQ employs mirrored queues and quorum queues to replicate messages across brokers in a master-slave topology. This ensures scalability while maintaining system consistency.	Durable queues and persistent messages enhance RabbitMQ’s fault tolerance. Acknowledgments and confirms ensure delivery, but it primarily supports at-least-once and at-most-once semantics.	RabbitMQ's queueing system supports concurrent processing, with features like prioritized queues enabling efficient resource allocation for high-throughput demands.
NATS	NATS supports horizontal scaling through clustering, allowing multiple servers to work together to handle increased traffic. Its lightweight protocol minimizes overhead, enhancing scalability.	NATS ensures high reliability with features like message persistence and durable subscriptions. NATS Streaming (JetStream) allows for message replay and fault tolerance, ensuring no data is lost.	NATS excels in parallelization by decoupling producers and consumers, enabling independent and concurrent operations. It supports load balancing by distributing messages across multiple subscribers, preventing bottlenecks.

Benchmarks for stream processing frameworks are already available [97], [98], [76]. However, these benchmarks should be interpreted in the context of the specific nature of the data being processed. For example, the data used in Adobe's evaluation may differ significantly from network data, which could affect the relevance of the results for other use cases.

6.2 Secured and Privacy-Preserving Data Management System

A high-level reference architecture and a functional architecture for NDT-enabled 6G are proposed respectively in D2.1[99] and in this document (Figure 3). In this section, NDT-enabled 6G architecture’s data management framework is investigated to secure the data and preserve privacy in the data-related processes. For this purpose:

- Vulnerabilities, threats, and possible attacks specific to data management processes are identified.
- Countermeasures are listed and analyzed in a manner specific to the identified threats.
- The practicality and feasibility of the existing countermeasures are discussed to enable security and privacy and preserve the NDT data management system.

Enabling NDT through 6G network orchestration provides advantages and new opportunities. However, it also widens the threat and attack surface by propagating the existing one and by bringing out new ones because of the NDT-6G integration. Therefore, the potential attack and threat surfaces are recognized and divided into categories to handle security and privacy threats. The definition of these surfaces is predicated on the idea that all elements, whether virtual or physical, are assets. Moreover, the NDT-enabled 6G network reference architecture's network topologies are regarded as separate entities.

The identification and description of threat and attack surfaces associated with data management, data collecting, and data exposure points inside a 6G network with NDT capabilities are discussed in this section.

The attack surfaces are separated in detail in the subsections that follow, with a focus on the data management lifecycle. To better understand the threats and attack surfaces mentioned, these surfaces are also depicted in a data flow diagram, which is presented in Figure 15 Figure 15 6G-TWIN data management lifecycle threat surface architecture [100]and is created considering the data collection and management framework and processes presented in the Figure 2.

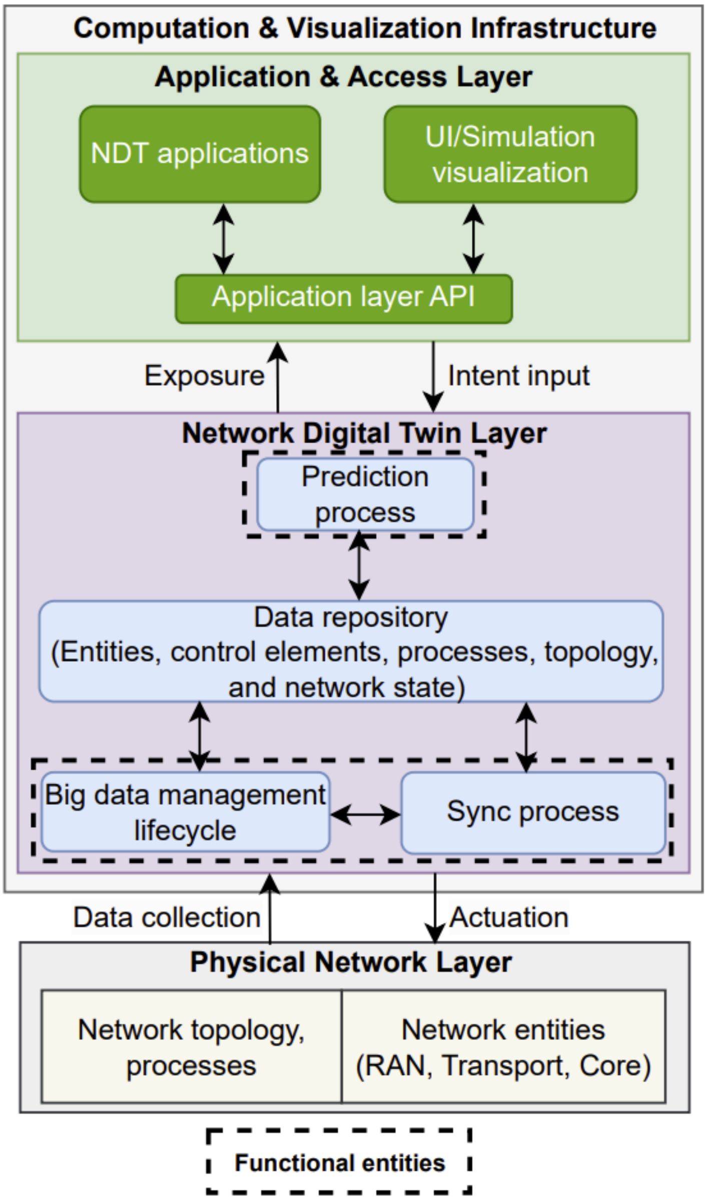

Computation & Visualization Infrastructure
Application & Access Layer
NDT applications
UI/Simulation visualization
Application layer API
Exposure
Intent input
Network Digital Twin Layer
Prediction process
Data repository (Entities, control elements, processes, topology, and network state)
Big data management lifecycle
Sync process
Data collection
Actuation
Physical Network Layer
Network topology, processes
Network entities (RAN, Transport, Core)
Functional entities

Figure 15 6G-TWIN data management lifecycle threat surface architecture [100].

The threat and attack surfaces presented in Figure 15 These threats are listed and briefly described by evaluating their requirements in terms of communication channels, associated protocols throughout their lifecycles, their components, and the corresponding NDT layer. The results are presented in Table 13 below.

Table 13 Threat surface of the data management framework.

Threat surface	Description	Possible Attacks
Physical Infrastructure of the Network	Includes various 6G components, actuators, sensors, IoT devices, and data collection elements—is susceptible to security breaches. Protecting these components is critical due to their essential role in supporting NDT in 6G networks. Since real-time data	Physical damage, tampering, information disclosure, unauthorized

	collection is necessary to enable accurate simulations and real-world operations, securing this infrastructure is fundamental to maintaining the integrity of NDT-enabled 6G systems.	access, single point of failure
Communication Interfaces	This surface encompasses connections and exposure points for data exchange between the physical network, the NDT, and APIs. The communication interface manages data flow from the physical network to the NDT, ensuring the NDT accurately mirrors the network's real-time state. It also facilitates data transfer from the NDT back to the physical network, enabling actions to be implemented based on simulations and predictions generated by the NDT.	Replay, eavesdropping, DoS, DDoS, spoofing, man in the middle
Application and Access Layer	Serves as the user interface for NDT, connecting with NDTs through application layer APIs to support various applications. This layer provides stakeholders with access to interfaces and simulation visualizations, making it one of the network's most exposed areas as it directly interacts with users and external systems. It typically includes UI modules, visualization tools, API gateways, and application-specific components accessing NDT functions. Communication within this layer often uses web-based interfaces and protocols that can be reached via internal or external networks.	Backdoor, Ransomware, API exploitation
Computation and Virtualization Infrastructure	This infrastructure comprises the NDT along with the application and access layers. An attack on this layer is appealing to attackers as it could ultimately impact the physical layer. This layer represents all digital assets within the NDT system.	Covers all attack surfaces from *application and access layer* and *functional entites*
Functional Entities	Implementing NDT relies on several computational methods, such as knowledge extraction, creation algorithms, and AI and ML techniques for prediction and learning. The functional entities in the NDT architecture are outlined with dashed lines in Figure 14 above.	Covers all attack surfaces from *synchronization, prediction* and *big data management processes*
Synchronization Process	This process ensures the virtual and physical components of the network are accurately synchronized in real-time, covering the timing, volume, and frequency of data flow between the physical network layer and the NDT layer. This synchronization process has two key elements. First one is the synchronization between the physical network and its NDT, and the second one is the dual synchronization across multiple NDTs.	Replay, time delay
Prediction Process	AI/ML models and algorithms play a key role in simulating, predicting, and optimizing network operations within the NDT. They process large data volumes to forecast network activity, identify potential issues, and propose solutions. To ensure model reliability, this data must be comprehensive, accurate,	Privacy attacks, model and data poisoning, white-black box attacks

	and reflective of real-world scenarios. Thus, the datasets for training and validation, the computational infrastructure for model processing, and the AI/ML algorithms are critical components of this threat surface.	
Big Data Management Life Cycle	Consists of four main stages: *(i)* data collection from diverse sources like sensors, IoT devices, and network elements, *(ii)* data storage in databases or cloud platforms, *(iii)* data processing and analysis to generate insights, and *(iv)* knowledge extraction and creation, which converts processed data into actionable insights for decision-making.	Impersonation, data breaching, malware injection, identity theft

After this point, comprehensive countermeasures are addressed and outlined associated with the data management processes' threat and attack surfaces in an NDT-enabled 6G network. However, it is useful to remember that balancing the enhancements in security and privacy with the overall system's utility is a key challenge. The countermeasures to identified threats in the literature can be listed and briefly described as follows:

- **Blockchain-integrated solutions:** Aims to address threats like data tampering, vulnerabilities in the synchronization layer, and risks in communication interfaces. This integration leverages blockchain's strengths, such as decentralized, trustless data storage, immutability, irreversibility, and transparency.
- **Decentralized and collaborative machine learning methods:** NDT collects data from various devices for security and anomaly detection and uses these data in machine learning processes for efficient analysis. Centralizing data collection and model training can create vulnerabilities. To mitigate this, decentralized methods like federated learning are suggested, as they allow for collaborative model training without relying on a central server.
- **Blockchain-integrated federated learning solutions:** Decentralized and collaborative machine learning solves some issues from central model training by enabling distributed collaboration. At the same time, it still relies on a central server to aggregate the global model, which introduces risks. To mitigate this, blockchain is proposed for securely transferring data in decentralized learning scenarios.
- **Use case-specific lightweight and scalable encryption algorithms and security protocols:** The NDT-enabled 6G architecture encompasses multi-domain heterogeneous environments with distinct security, privacy, or computational needs. As a result, different encryption schemes and new security protocols are proposed for the NDT architecture, customized for various interfaces and prioritized based on specific requirements.
- **Privacy-enhancing solutions:** NDTs depend on real-time data and advanced AI models to simulate, predict, and optimize network behavior. Several solutions are proposed to protect data management privacy, including anonymization, encryption, differential privacy, limiting model queries, homomorphic encryption, and safeguarding intellectual property rights.

7 Implementing the 6G TWIN Data Collection Framework

This section serves as an initial approach, outlining design decisions that shape the telemetry/data framework in 6G-TWIN based on the architecture proposed in Section 2 and technology enablers presented in Sections 3 to 5. In this document, the terms 'telemetry collection' and 'data collection' are used interchangeably, as they both refer to the process of gathering, transmitting, and processing data from diverse network components. In the following subsections, two approaches are presented: one focused on RAN and the other focused on cross-domain data collection.

7.1 Initial 6G-TWIN data collector for O-RAN

In 5G and 6G systems, data collection enables dynamic network adjustments based on real-time data analysis. For example, data collected from Distributed Units (DUs), Radio Units (RUs), and Centralized Units (CUs) help optimize resource usage, balance loads, and improve energy efficiency across the network. The integration of AI and ML further enhances telemetry and data collection roles, as AI algorithms use telemetry to automate network management, enabling near-instant adjustments to optimize performance and energy use. Looking ahead to 6G, the role of telemetry and data will expand significantly. Emerging technologies such as JCAS, RIS, and advanced spectrum management strategies depend on robust telemetry/data systems to function efficiently. Dynamic spectrum re-farming, for example, will rely heavily on real-time telemetry to make decisions regarding spectrum allocation and interference management. Without accurate and timely telemetry, implementing these innovations would be challenging.

Additionally, O-RAN architecture plays a critical role in advancing telemetry/data systems. O-RAN introduces the RIC, both in Near-RT and Non-RT forms, which use data collection to drive intelligent network decisions. These RICs ingest data from various sources in the RAN, process it using AI/ML models, and make real-time decisions that optimize network performance. Telemetry serves as the backbone of this decision-making process, ensuring that the network remains responsive and adaptive to changing conditions, including fluctuations in traffic, mobility patterns, and resource availability.

NDTs are becoming a crucial component of modern network management, closely intertwined with telemetry/data systems. As described previously, an NDT is a virtual representation of the physical network, including its components, performance metrics, and real-time status. Telemetry data provides the real-time information needed to keep the digital twin synchronized with the physical network, enabling operators to model, analyze, and predict network behavior. The integration of telemetry into NDTs allows network operators to simulate network scenarios and predict outcomes, offering a proactive approach to network optimization. This relationship is particularly valuable in complex environments, such as 5G and 6G networks, where real-time adjustments are needed to address changing traffic patterns, manage resources effectively, and optimize network health. By using data-driven NDTs, operators can perform what-if analyses, foresee potential issues, and implement solutions before they affect the physical network, thus improving efficiency and reducing downtime.

Furthermore, 6G networks are expected to integrate more diverse data sources, such as environmental sensors and positioning systems, which will feed additional telemetry/data into the system. This expanded scope of telemetry/data will enable networks to not only optimize performance but also address broader goals, such as reducing Electromagnetic Field (EMF) exposure and improving user experience in urban and rural environments.

7.1.1 Telemetry, Data Collection, and Control Framework in O-RAN

The Telemetry, Data collection and Control Framework is developed to streamline and organize telemetry data and provide translator control capabilities within O-RAN technology environments. This framework is particularly crucial for handling data from various segments of the O-RAN infrastructure and is structured around several key components.

At the heart of the system is the Telemetry Collector (TC), which plays a pivotal role in ensuring interoperability across different parts of the O-RAN setup and maps to the Telemetry Data Layer of 6G-TWIN architecture as shown in Figure 3 and explained in the subsequent paragraphs. Given that some implementations of the O-RAN interfaces such as E2, O1, F1, or A1 might not always be fully implemented O-RAN-compliant, the TC steps in to bridge these gaps. It translates or regenerates data from the RU and from the DU, making it indispensable for environments where O-RAN standards are not fully implemented and communication is needed with the CU or the RIC. Similarly, it translates RAN control messages to the RU/DU/CU entities that are not fully O-RAN compliant, such as new technologies like ISAC or Cell-Free Massive MIMO.

There are currently two primary trends in O-RAN deployment: a) systems that are fully compliant with O-RAN standards, featuring fully functional O-RAN interfaces, and b) those that are not fully compliant, lacking complete O-RAN interface integration or providing extensions that O-RAN does not specify. The telemetry/data collection framework is crucial for the latter as it facilitates the necessary translation between non-compliant interfaces and the rest of the O-RAN ecosystem. Additionally, raw data from the radio environment or abstracted data from the real world through sensing capabilities can be leveraged on the TGW and fed to the Near-RT RIC and Non-RT RIC for decision-making and RAN control. For example, the TC can collect input values from the Kafka bus and publish the calculated output metrics with the same timestamp via the Math Module, allowing other xApps to use them independently of their source.

Moreover, the telemetry and data collection framework simplifies the development process by abstracting the complexities of O-RAN interfaces for xApp/rApp application developers. Instead of requiring developers to interact directly with the intricate details of O-RAN interfaces and the specific type of messages from RUs and DUs, the framework offers processed and relevant information directly to the RIC. This abstraction eases the development process and ensures that developers can focus on creating applications without delving into the underlying O-RAN infrastructure details. This also applies to controlling messages, as the TC can handle abstracted messages, such as simplified handover messages written in some xApp. At the same time, the TC will generate the appropriate message structure needed for lower-layer components.

The technical requirements for the telemetry and data collection framework within O-RAN architecture focus on ensuring comprehensive data management and control across the network's various components, specifically categorized into two main areas: telemetry and data messages or metric messages and control messages.

- **Telemetry and Data Collection Messages or Metric Messages**: This category encompasses the capabilities required to handle and process data from the lower RU and CU components of the O-RAN standardization framework. Essentially, it involves collecting, preprocessing, and abstracting metrics reported by the User Equipment (UE) to the CU, DU, and RU. These metrics must be processed within the telemetry/data framework in a manner that is efficient and aligned with both O-RAN and 3GPP specifications. The preprocessing of these metrics is critical for transforming raw data into usable metrics on the xApp/rApps, ensuring that the data is standardized and interoperable across the various elements of the O-RAN ecosystem.
- **Control Messages:** The second key area, control messages, deals with integrating the operational capabilities of the radio components within the network. This includes functions such as activating or deactivating cells, triggering handovers, modifying physical layer characteristics, and managing mobility aspects. These control operations are vital for the dynamic management of the network, allowing for real-time adjustments to optimize performance and efficiency based on current network conditions and demands. The execution of these control messages is contingent upon the specific capabilities and configurations of the radio equipment in use, necessitating a flexible and adaptable telemetry/data framework that can accommodate a wide range of operational scenarios.

Together, these two areas of technical requirements underline the complexity and sophistication needed in the telemetry, data collection, and control framework to ensure seamless data integration and network control within the O-RAN architecture.

7.1.2Implementation: Interfaces and Data processing

A key innovation of the 6G-TWIN project lies in extending telemetry and data collection to include non-O-RAN-compliant elements. Many networks still contain legacy systems or proprietary devices that do not fully adhere to O-RAN standards. Traditional telemetry frameworks are often limited in integrating data from these non-compliant components. In 6G-TWIN, we introduce a flexible telemetry/data architecture that bridges the gap between O-RAN and non-O-RAN systems, allowing data from proprietary components, legacy infrastructure, or even custom-built radios to be collected and processed alongside standardized O-RAN components [101]. Another novel feature is the system's ability to handle more diverse data sources. As 5G technologies evolve, the network must process telemetry and data from new devices and systems, such as environmental sensors and user equipment with advanced sensing capabilities. These devices introduce new data streams that traditional RAN telemetry/data systems were not designed to handle. The extended telemetry and data collection framework can ingest and process this wider variety of data, enabling more sophisticated network optimizations, such as dynamic spectrum re-farming, real-time interference management, and energy-efficient network configurations.

Furthermore, advancements in programmable telemetry/data, such as the RANSight platform [102], are revolutionizing how telemetry/data is collected and exposed within the RAN. Programmable telemetry/data allows for highly customizable data collection pipelines, where network operators can specify the exact metrics they want to collect and how they want to process them. This flexibility enables more granular control over network operations and allows real-time adjustments based on precise, localized data. These advancements in programmable telemetry/data have far-reaching implications, especially for AI-driven network management, which relies on vast amounts of accurate and timely data to function effectively [103].

Lastly, recent updates to the O-RAN specifications have introduced enhanced support for telemetry/data collection and management. The latest revisions to the O1 and E2 interfaces have refined how telemetry/data is transmitted between network components and the RICs, improving the speed and accuracy of data transmission. These updates also include expanded support for new use cases, such as network slicing and autonomous Vehicle-to-Everything (V2X) communications. By building on these developments, this project ensures that the telemetry/data system is future-proof and capable of supporting the rapidly evolving demands of next-generation networks O-RAN Alliance.

The Telemetry and Data Collector Architecture inside the Accelleran's dRAX is designed to gather, process, and distribute telemetry and data across a diverse range of network components, both compliant and non-compliant with O-RAN standards. The architecture comprises three main blocks: Input Interfaces, 3GPP/O-RAN compliant Data processing, and Output Interfaces. These blocks work together to ensure seamless data collection, transformation, and dissemination, supporting real-time network optimization and decision-making processes, as shown in Figure 16.

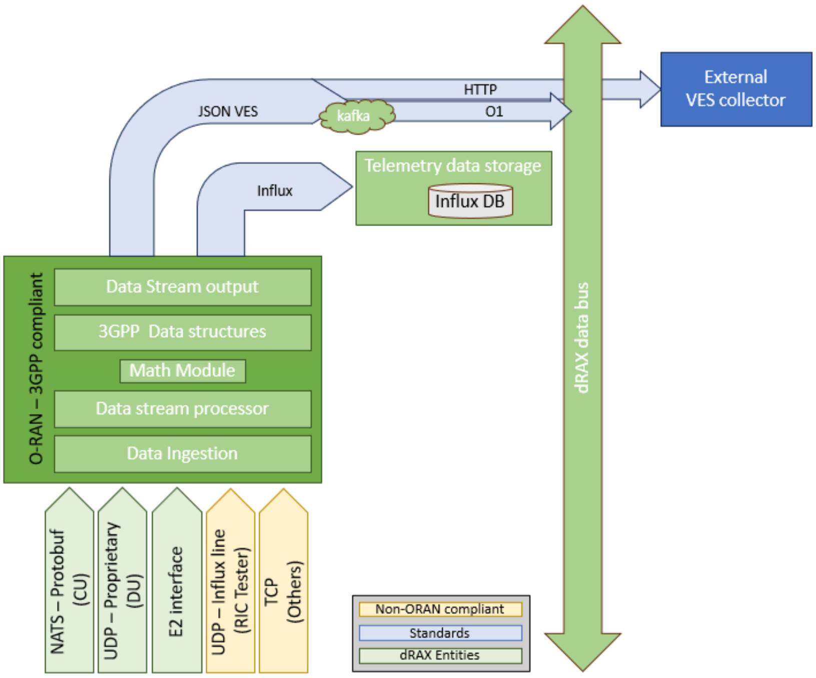


Figure 16 Accelleran Telemetry collector general architecture.

7.1.2.1 Input Interfaces

In the lower part of Figure 16, the input interfaces collect telemetry and data from various sources within the network, including O-RAN-compliant and non-O-RAN-compliant or legacy components. The following key input interfaces are integrated into the system:

- **E2 Interface:** This interface is a core component for collecting telemetry from O-RAN elements like the DU, CU, and RU. The E2 interface facilitates real-time data transmission between these RAN elements and the Near-RT RIC, enabling critical operations such as resource management, mobility control, and interference handling.
- **NATS Protobuf:** This interface is defined inside the dRAX entity and oversees quickly bringing information from the Accelleran CU to provide fast telemetry/data interaction.
- **Proprietary UDP:** This input interface supports high-speed data collection from custom-built or non-standardized components. It is tailored for specific use cases where existing protocols may not suffice, ensuring that proprietary network elements can still contribute telemetry/data to the system. It is part of the dRAX family interfaces that connect the RIC with the DU.
- **InfluxDB Line Protocol:** Used primarily for time-series data collection, this interface enables the integration of performance metrics and sensor data into the telemetry/data system. It is particularly useful for monitoring network health over time, feeding data into AI-driven decision-making processes. This is used to connect testers that do not support fully O-RAN compliant testers or KPI values that are not defined in 3GPP metrics.

- **TCP-based Input Interfaces:** These interfaces provide compatibility with legacy or non-O-RAN-compliant systems. The use of general-purpose TCP protocols allows the system to ingest telemetry/data from older network elements that do not follow O-RAN standards, ensuring backward compatibility.
- **NATS Messaging [76]:** NATS is used as a messaging protocol to facilitate real-time data transport between distributed components of the network. It is lightweight and scalable, providing low-latency delivery of telemetry/data across different parts of the architecture.

These input interfaces allow for a wide variety of telemetry/data sources to feed data into the system, ensuring that both traditional and modern network elements can be monitored and optimized in real time.

7.1.2.2 3GPP/O-RAN Compliant Data Processing

Once the telemetry and data are collected from the input interfaces, it enters the 3GPP/O-RAN compliant Data processing. This section is also known as the Telemetry data layer of Figure 3. The primary role of this component is to standardize and transform the data into formats that can be processed and acted upon by the network's AI-driven systems.

- **Data ingestion:** These components convert proprietary or non-standard data formats into formats compatible with 3GPP data structures and O-RAN-compliant systems. For instance, data collected from legacy equipment or custom-built devices might use different protocols. However, the translators ensure this data can still be utilized effectively within the broader network architecture.
- **Data Stream processor:** This processor is responsible for pre-processing the telemetry data and performing functions like filtering, aggregation, and normalization. These processes ensure that only the most relevant data is passed on to the decision-making components, reducing the processing load while maintaining high accuracy. One important module here is the mathematical module, which aggregates new metrics based on the other existing metrics for specific use cases. This preprocessing covers all the metrics in 3GPP standard data structure based on the 3GPP TS 28.552 and 3GPP TS 38.314 standards.
- **Data Stream Output:** These are the ones in charge of extending the 3GPP data sources into the needed output interfaces for the different parts of the dRAX components and external entities.

This block ensures that data from all sources is presented in a uniform, actionable format, which is then delivered to the output interfaces for distribution to the appropriate decision-making entities.

7.1.2.3 Output Interfaces

The output interfaces are responsible for disseminating the processed telemetry and data to various network consumers and external systems. The primary output interfaces include:

- **O1 VES (Kafka):** This interface streams telemetry data in real-time to event-driven systems using the VNF Event Streaming(VES) protocol over Kafka. It allows for integration with third-party systems that require real-time telemetry/data for analytics, monitoring, or optimization purposes AWS Documentation O-RAN SC.

- **HTTP for External VES Collectors:** External collectors and monitoring systems can access the telemetry data via HTTP-based interfaces. This makes the data easily consumable by external applications that might not be part of the O-RAN ecosystem but require access to network performance metrics or health data O-RAN SC.
- **Future Expansion for 6G Systems:** The architecture is designed with future-proofing in mind, supporting future enhancements such as WebSockets for 3GPP-compliant streaming and high-volume data flows. This ensures that the system can scale and adapt to the needs of emerging 6G technologies, which will require even more sophisticated telemetry/data systems.

These output interfaces ensure that telemetry/data is readily available to key consumers, such as the RICs (Near-RT and Non-RT), AI/ML systems, and external monitoring platforms, facilitating real-time network optimization and troubleshooting.

7.1.2.4 Scalability and Future Enhancements

The Telemetry and Data Collector Architecture is built with scalability in mind, designed to accommodate the increasing complexity and data volume of 6G networks. As technologies like NDT, RIS, and JCAS become more integrated into the network, the system must handle an even greater diversity of data sources.

To support these advancements, the architecture is modular, allowing for the easy addition of new protocols and interfaces as they become necessary. The system's ability to scale horizontally by adding new processing nodes and vertically by increasing its processing power ensures that it can grow with the demands of next-generation networks O-RAN Alliance.

Future iterations of the system will focus on a) connection and exposure towards the harmonization data layer and b) improving the integration of AI and ML into the decision-making process, enabling the network to optimize itself autonomously based on the telemetry data it collects as part of the zero-touch management approach of 6G-TWIN. This will further reduce operational overhead and improve network efficiency, particularly in dense, urban deployments where real-time data processing is critical.

7.2 Cross-domain Data Collection and Optimization

Recent advances in cross-optimization between the RAN and TN have shown promise for improving network performance [104]. However, significant gaps remain in the literature, particularly regarding how the RAN can leverage monitoring data from the TN to optimize resource management without direct control over the TN resources. Existing approaches for SDN-based mobile backhaul architectures [105] and multi-tenant dynamic slicing frameworks [106], [107] typically enable cross-domain resource provisioning through centralized control. These solutions are not suitable for environments where independent control of the RAN and TN. Moreover, the lack of standardization to integrate controllers across different domains hinders seamless coordination and interoperability, which is essential for efficient cross-optimization among decentralized network domains [108].

As described in Section 2.2 in D2.1 [99], Section 7.1, and later in Section 10.3, the control entities of both the RAN and the TN are already well established, each providing fine-grained control over each network domain. However, the interfaces and information exchange between these two entities is not specified. Since the RIC and the SDN controller of the TN operate in different areas of the network, our initial approach is to keep their decision-making processes independent so that they do not exert control over one another [109]. While the existing NBI of the SDN controller could be connected to the Service management and orchestration (SMO) framework to provide the necessary information to the RAN, this would also allow the SMO to control the TN, which is undesirable in federated domains, for example.

Table 14 provides an overview of the existing interfaces that enable information sharing between the RAN and the TN. This clearly shows 1) the lack of telemetry interfaces on the SDN controller for external entities and 2) the lack of a unified interface for this kind of information exchange. Consequently, we propose and discuss some architectural changes and interfaces required to enable information sharing between the RAN and the TN while maintaining independent control over each domain. Figure 17 depicts two interfaces that will allow this behavior: the East-West Interface (EWI) and the North-South Interface (NSI).

Table 14 Traditional interfaces used to share information between RAN and TN controllers.

From component	To component	Interface name
Near-RT RIC	Non-RT RIC	A1
Near-RT RIC	SDN Controller	Y1
Non-RT RIC	Near-RT RIC	A1
Non-RT RIC	SDN Controller	DME*
SDN Controller	Near-RT RIC	/**
SDN Controller	Non-RT RIC	/**

** The Non-RT RIC can indirectly expose enrichment information to external entities using the Data Management and Exposure (DME) of the SMO. However, this interface is not fully standardized yet. **SDN Apps can be used to collect and expose telemetry data to the SMO's EI interface, indirectly reaching the RIC.*

7.2.1 East-West Interface (EWI)

First, the EWI, as shown in Figure 17, is defined as an interface to expose and consume information between the SMO containing the Non-RT RIC and the TN SDN controller. We define the EWI in line with the existing standardization efforts of interfaces at the RIC, the SMO, and the SDN controller.

- **RIC:** The Near-RT RIC can expose data to external entities through the O-RAN Y1 interface [110]. This interface plays a key role in sharing RAN telemetry with both internal and external entities within the trust domain of the MNO. Functionally, the Y1 interface exposes various types of RAN analytics that can be used to support key tasks, such as traffic steering, mobility management, and interference management.
  To consume data from external entities, the RIC can use the external Enrichment Information (EI) interface of the SMO [111]. This interface allows the SMO to consume enrichment information from external sources and pass it to the Non-RT RIC via the internal Non-RT RIC-SMO interface. The process of gathering such enrichment information is thoroughly described in the O-RAN use cases specification [112], providing sequence diagrams of the required message exchange to collect enrichment information from external applications (e.g., a telemetry data App running on the TN SDN controller). Furthermore, the EI interface also allows the Non-RT RIC to exchange handshake messages (e.g., request and response messages) with external enrichment information sources. When required, enrichment information can also be exposed to the Near-RT RIC through the A1 interface of the NSI. Additionally, O-RAN is studying new external SMO terminations, such as the Data Management and Exposure (DME) interface, to expose data from the SMO and Non-RT RIC to external entities [113]. Although these terminations have not yet been fully standardized in the O-RAN specifications, they provide the functionality required to exchange enrichment information with external entities.
- **SDN:** On the SDN TN side, there is no standardized interface like O-RAN to consume data from and expose data to external entities. However, custom applications can be deployed in the application plane of the TN, with functionalities similar to the xApps and rApps of the RIC. To this end, as shown in Figure 17, we create two SDN Apps: the TN Telemetry Source App and the RAN Telemetry Sink App. The TN Telemetry Source App monitors the TN, collects enrichment information (e.g., packet loss, link throughput, etc.), and exposes this information to external entities. The SMO in the RAN can then connect to this App through the EI interface and consume data from the TN. The RAN Telemetry Sink App serves as an endpoint for the TN to collect telemetry data from the RAN, coming from both the Y1 interface of the Near-RT RIC and the DME of the SMO.

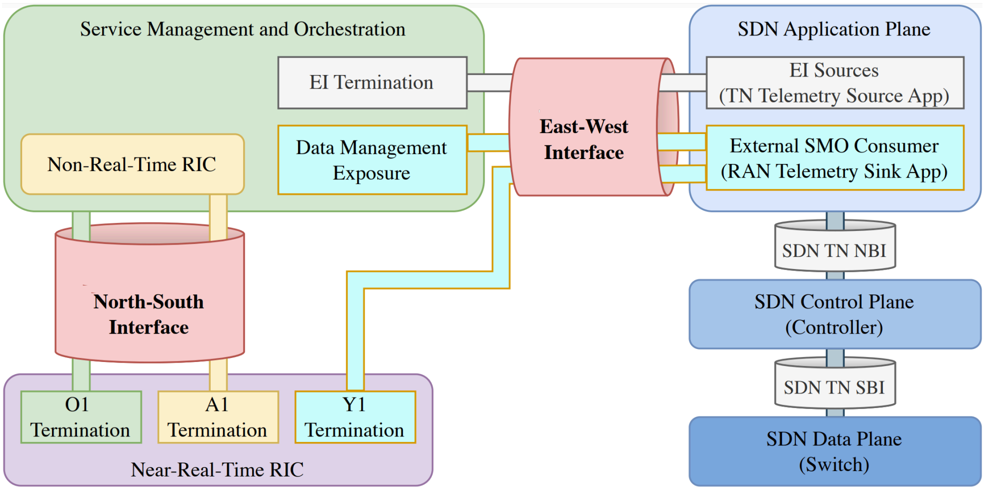


Figure 17 Definition of North-South and East-West Interfaces. These interfaces enable the RAN and the TN to exchange telemetry data with one another in a non-hierarchical way, meaning that neither one of the domains can control the other.

7.2.2North-South Interface (NSI)

To complement the EWI, the NSI is an interface allowing information sharing between the Non-RT RIC and the Near-RT RIC. As shown in Figure 17, the NSI is composed of two existing interfaces: 1) the O1 interface between SMO and Near-RT RIC [5]; and 2) the A1 interface between Non-RT RIC and Near-RT RIC [114]. The established O1 and A1 interfaces already enable the necessary data exchange between the Non-RT RIC and the Near-RT RIC, eliminating the need for additional components or interfaces. Nevertheless, the functions provided by these interfaces are essential for RAN-TN cross-optimization. They allow the Non-RT RIC to communicate TN-aware enrichment information and policies to the Near-RT RIC. By combining the NSI and the EWI, we achieve information exchange among the Near-RT RIC, the Non-RT RIC housed within the SMO, and the SDN controller of the TN.

7.2.3Cross-domain Optimization Using Multi-Domain Telemetry data

To showcase the benefits of sharing telemetry information between the RAN and the TN using well-defined interfaces, a Proof of Concept (PoC) of a TN-aware intelligent RIC on top of the Open5G@TheBeacon 5G SA testbed [115] was developed and tested. Figure 18 represents the experimental setup and the network architecture deployed on our testbed. As shown in this figure, we add three new major components on top of our existing testbed to enable a TN-aware intelligent RAN: 1) a telemetry-producing app for the TN that records various metrics; 2) a telemetry-consuming rAPP for the RIC that fetches the monitoring information from the TN; and 3) an xAPP that can dynamically (re)allocate PRBs for each slice in the network.

These components interact as follows. First, the TN telemetry producer, which can be deployed switch-by-switch, allows us to capture hop-by-hop performance metrics (link throughput and packet loss) throughout the TN. Our captured metrics are then saved in

PostgreSQL databases deployed on the same machine as the User Plane Functions (UPFs), allowing them to be consumed by external entities. Then, the telemetry consumer rAPP in the RIC periodically collects these metrics, analyzes them, and uses them to create policies regarding the allocation of PRBs. More specifically, our rAPP currently contains a simple algorithm that decides to reallocate radio resources whenever packet loss in the TN exceeds a user-defined threshold. Finally, these policies are executed by the PRB allocation xAPP, which is based on an existing FlexRIC xAPP that allocates PRBs at the slice level. This xAPP can dynamically reallocate PRBs between users based on the policies created by the rAPP.

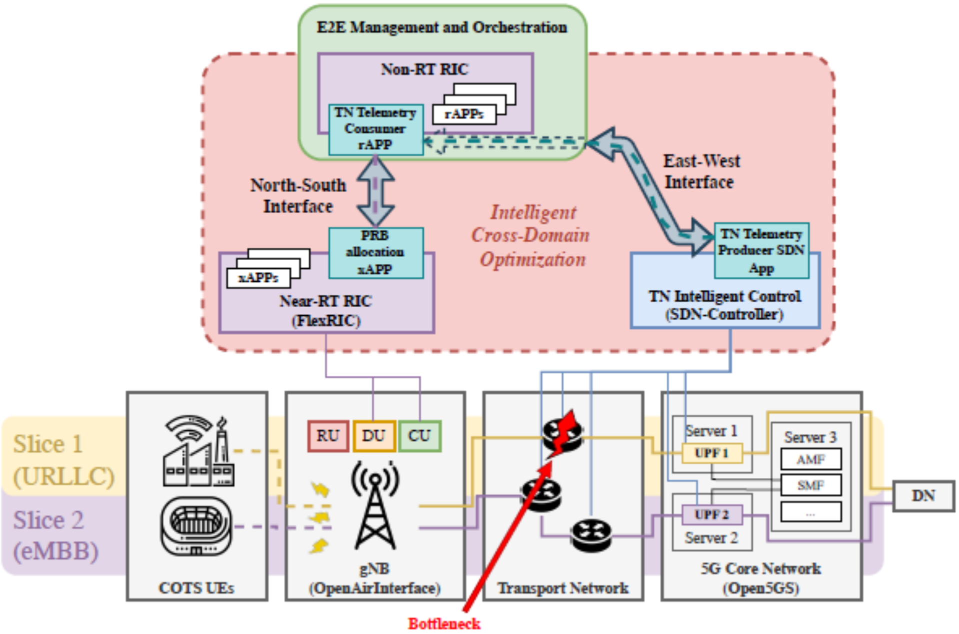


Figure 18 Experimental setup for TN-aware intelligent RAN experiment. Two UEs are connected to a single gNB over 5G New Radio (NR). Each UE is connected to a different slice, reaching its distinct UPF over different TNs.

Based on this experimental setup, a scenario with two User UEs, each using a different slice, is deployed. This scenario serves as initial experimentation to validate concepts that are required for 6G-TWIN use cases, specifically UC2 related to energy savings in cross-domain setups. UE 1 is assigned to a Ultra-Reliable and Low-Latency Communications (URLLC) slice called Slice 1, while UE 2 is assigned to an enhanced Mobile Broadband (eMBB) slice called Slice 2. Both UEs are connected to the same gNB, but their data planes run over two different TNs, each of which ends up in a different UPF. As shown in Figure 18, UPF 1 serves UE 1 on Slice 1, while UPF 2 serves UE 2 on Slice 2. The UPFs are connected to the same 5G CN, providing the remaining NFs required to run a functional 5G SA network. In this scenario, both UEs start with a fixed PRB allocation: 60 % of PRBs are allocated to UE 1, while the remaining 40 % are assigned to UE 2. This kind of fixed allocation is extremely useful in the context of network slicing, as resources can be dedicated to a specific slice to ensure QoS requirements. Then, we generate 80 Mbps of downlink traffic over UDP for each UE using iperf, after which we emulate a congestion problem in the TN by rate-limiting the interface of UPF 1 connected to the TN. This causes the throughput of UE 1 to drop and the packet loss at the TN to increase. Based on this sudden increase in TN packet loss, the RIC should dynamically reallocate radio resources since UE 1 cannot fully utilize its assigned PRBs anymore.

The results shown in Figure 19 present the throughput and packet loss of each slice, from the application level (left) and from the TN (right). Two important events are marked on the graphs: the time at which the congestion problem occurred in the TN, and the time at which the RAN dynamically adapted the assigned PRBs because of that problem. At the first mark, a bottleneck in the TN of Slice 1 causes the TN-level and application-level throughput of UE 1 to drop from 80 Mbps to 50 Mbps and the packet loss to increase to 40 %. As a result, the radio resources allocated to UE 1 are not being fully used, since the PRBs are allocated in a fixed way to each slice. As a result, the fixed amount of PRBs allocated to UE 1 cannot be fully used, resulting in inefficient use of radio resources.

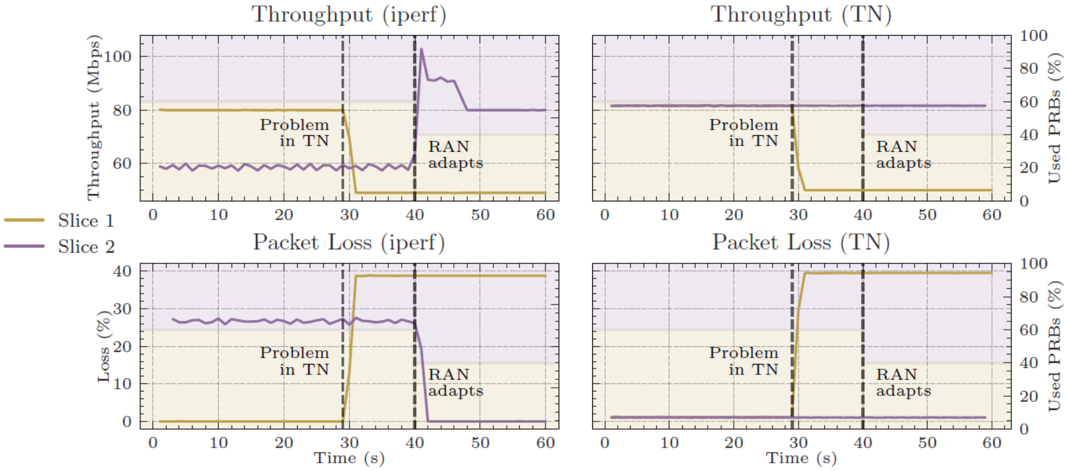


Figure 19 Throughput and packet loss results at the application level (left) and at the TN level (right) for the TN-aware intelligent RAN scenario. Each plot marks the time the TN congestion problem was induced and the time at which the RIC reallocates PRBs. The background colors depict the distribution of PRBs between users across the experiment run.

By the second mark, the RIC detects the sudden increase in packet loss using telemetry data from the TN and reallocates the PRBs, assigning more resources to UE 2. We observe a throughput peak of over 100 Mbps caused by buffered packets in the RAN, after which the throughput settles back to the 80 Mbps generated by the application. At the same time, the application-level packet loss of UE 2 significantly decreases. It is important to note that the packet loss of UE 2 was high in the first place because we use UDP, which does not have any congestion control mechanisms. By dynamically reallocating PRBs across the UEs, the RIC optimizes spectrum usage based on telemetry data from the TN.

These results empirically prove that the RAN can benefit from TN telemetry data. Similarly, a RAN-aware TN could make better management decisions using RAN information. For example, the TN could predict congestion based on the number of connected UEs or the average throughput provided by the RAN’s Y1 interface [116]. The TN can exploit this increased awareness to improve network management, which is crucial in network sharing.

8 Data Collection from the Viewpoint of Simulations

Besides system analysis and management, one purpose of collecting data is to create realistic simulations. A good overview of the relationship between NDTs and network simulations has been given in [117]. That means that measured data is used to set up simulations that match a real system as closely as possible. Two versions of realistic simulations are common in science:

- One scenario type is the repetition of former real situations in a simulation to use them as a baseline for optimization of, e.g., management strategies that help improve the management in similar future situations. A typical step for that purpose is the optimization of unmeasured or unknown model parameters to match specific quantities that are measured both in the real system as well as in the simulation, e.g., latency, throughput, energy consumption, or packet error rate. Simulation models for this kind of purpose are often trace-driven [118], [119], [120] or might at least use data of real maps [121] or satellite positions [122]. In vehicular applications also vehicle positions are typically tracked [123], [124], [125].
- The other scenario type is the online simulation of a system in real time, where the simulation runs in parallel to a real system. Simulations have already been coupled with the real networks [126]. One benefit of this scenario is that it can give insight into KPIs and other system properties that are not measurable in the real system but can be read from the simulation. To get realistic results, the system parameters must be updated when they change in the real system. Although in some interpretations, this application can be seen as the scope of a “digital twin” (sometimes called a “living digital twin”), it has been decided by the 6G-TWIN consortium to take this type of scenario out of scope during the projects’ duration, because, depending on the level of detail, many network simulations are not fast enough to be executed in real-time.

That means that in the 6G-TWIN project, simulations are only used to either (1) recapitulate former situations or (2) analyze what-if scenarios. Both can benefit from measurement data, but especially for the former one, measurement data is typically indispensable.

In this context, “measurement data” can be understood in a very different meaning:

- The most obvious kind of measurement data is data measured by network devices. Such data is typically available as measured time series. Examples are KPIs like throughput, latency, data rates, energy consumption etc.
- Beside these, system parameters like the number of UEs or software configuration parameters can be recorded. This is information that has not necessarily to be measured if it is known and can be edited by humans for setting up a simulation. However, in practice it is often not known a priori, e.g., at which time new UEs connect to the network or when configuration parameters are changed automatically by software to optimize or manage the system. Because of that, it can be easier to record such quantities automatically, which can be regarded as measurement.
- System parameters like the number and position of gNodeBs (gNBs), the bandwidth of wired connections etc. are often constant and do not change during measurements. However, this information is also needed to set up realistic simulations. That means

that, strictly speaking, this is no measurement data, but for example, basic models (see Figure 20) similarly use both sources of information, which makes it reasonable to handle this data like measurement data.

One key goal of the 6G-TWIN project is developing a simulation framework that combines formerly separated simulation models. For example, it shall be possible to connect a RAN model of one project partner with a core model of another project partner and a UE mobility model of a third project partner. These simulation components are distinguished into (1) basic model and (2) functional models. Basic models are a graph-based description of properties of a network, like topography, network elements, their capabilities, gNB positions, together with related measurement data. More details can be found in 6G-TWIN deliverable D2.2 [127]. Functional models, on the contrary, are special-purpose functionalities like analysis and optimization.

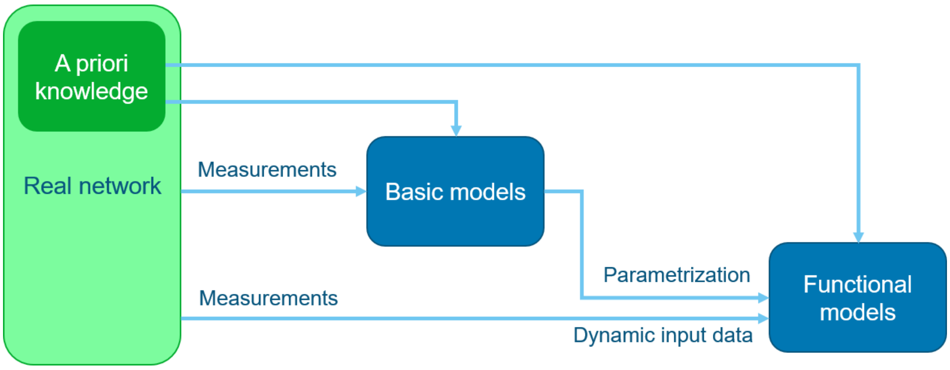


Figure 20 Creation of basic and functional models from real networks.

Besides traditional analytic or stochastic models, functional models can also be based on AI including ML. A typical property of ML is that it requires a vast quantity of data to match the model parameters to the measured data. AI models often contain artificial neural networks that contain thousands or millions of weights. These weights can be aligned with the input data only if the learning algorithm is provided with data tuples, approximately matching the parameter count, in a suitable order. That means the more complex an AI model is, i.e., the more weights it has, the more accurate it can represent reality, but also the more data is required to train the model. Therefore, if a complex AI model should match a real network, many measurements must be taken to train the model.

Independent of the model type, models can represent a real network only if knowledge of the real system exists that has been edited by humans or measured. Which kind of data is needed to parameterize a model strongly depends on a) the parameters of the model – these have typically to be known at the start-up of a simulation run but may be changeable during the simulation; b) the inputs of the model – these are typically fed into the simulation step by step during runtime, e.g., the number and position of UEs as a function of time; c) the outputs of the model – real measurements about these don't have to be known by the model itself, but often, it is necessary to compare the simulation model outputs with measured outputs, especially for model training, including a comparison of the model (simulation) outputs with the measured values. For example, if KPIs are measured and output from the simulation, both can be compared to evaluate the quality of the model.

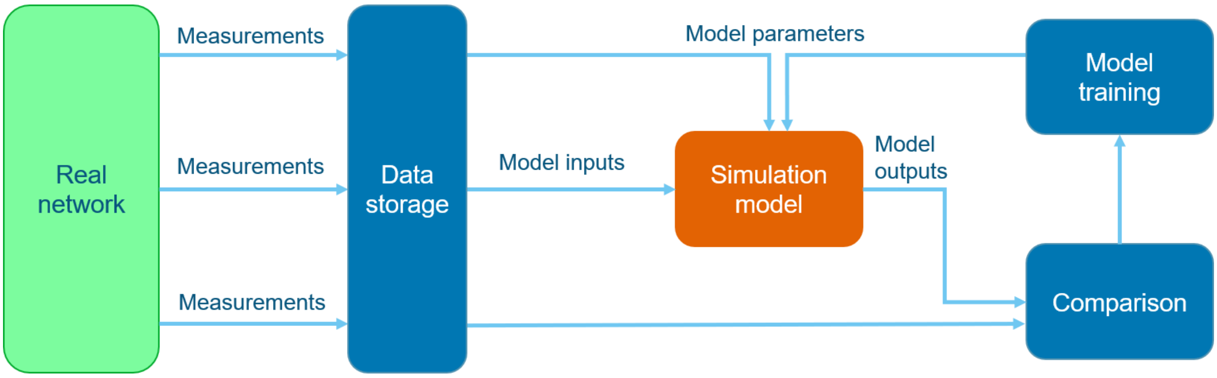


Figure 21 Using measurements for simulation models.

This makes it clear that there is not one unique list of data that must be measured. Instead, the needed data depends on the concrete model that is simulated. However, which model is simulated depends on the use case or scenario examined. Therefore, it can only be said that at least the following quantities are required for realistic simulations:

- Measurements (or manual edit) of system properties like number of gNBs, number of UEs, core network capabilities, topologies etc.
- Measurement of dynamic data like UE positions, required data rate per UE etc.
- KPIs of the scenario like latency, reached uplink and downlink data rate, etc.

Properties, scenarios, and KPIs of the use cases of the 6G-TWIN project can be found in deliverable D1.1. The upcoming deliverables define the basic and functional models, particularly D2.2 and D2.3 of 6G-TWIN, respectively. Thus, besides the use case definitions, the necessary set of parameters to be known cannot yet be fully given here.

9 Conclusions

Incorporating NDTs into 6G architectures is a transformative step toward addressing the inherent challenges of increasingly complex and dynamic wireless networks. As 6G networks strive to deliver ultra-reliable communication, low latency, and seamless integration of AI-driven applications, an NDT provides a virtual counterpart to physical networks, enabling real-time monitoring, predictive analysis, and automated management. This capability not only supports the orchestration of diverse network components but also enhances scalability, efficiency, and security in handling complex use cases such as teleoperated driving and energy optimization in dense deployments. This deliverable, D1.2, contributes significantly to the 6G-TWIN project by establishing a robust framework for scalable and secure data collection and exposure. The proposed data architecture supports the integration of various heterogeneous data sources, including TNs and edge/cloud infrastructures, creating a unified system for seamless interaction between physical and virtual network layers. By defining efficient data processing pipelines, telemetry frameworks, and harmonization processes, this deliverable ensures that the data infrastructure is equipped to handle large-scale, real-time network demands without performance degradation. The telemetry layer defined in this document not only facilitates the collection of diverse data types from distributed environments but also integrates with communication paradigms to ensure interoperability and high performance. Moreover, the harmonization layer processes and structures this data, enabling the simulation of real-world scenarios and providing actionable insights for network management and orchestration. These contributions form the backbone of a TRL 3 prototype, serving as a critical validation step toward implementing NDT in real-world 6G environments.

The findings of D1.2 are integral to the broader objectives of the 6G-TWIN project. They provide essential inputs for related tasks, such as decision-making in closed-loop analytics (Task 1.3) and data harmonization (Task 2.2), and extend prior work on data governance and privacy outlined in D2.1. Additionally, the security and privacy considerations mentioned in this deliverable ensure that the proposed frameworks align with the rigorous standards required for next-generation networks.

Looking forward, further research and iterative development are necessary to optimize the scalability, reliability, and efficiency of the NDT-enabled 6G architecture. These efforts will enable 6G-TWIN to fulfill its mission of creating an AI-native, interoperable, and intelligent 6G ecosystem that addresses the demands of increasingly complex and interconnected networks. With its forward-looking approach and innovative solutions, the project is poised to shape the future of wireless communication and set new benchmarks for network efficiency and resilience.

This document will be further elaborated and updated through a new deliverable, D1.5, provided at the end of 6G-TWIN and considering all the elements mentioned above.

10 Annex: Updates on Network Data Types and Interfaces

10.1 Introduction

Section 2 of D2.1 [99] introduces an initial data inventory and interface definition of the RAN and CN. However, the ambitious 6G Network will go beyond the traditional RAN-CN network domains, and mechanisms such as Software-Defined Networks (SDN) and Network Function Virtualization (NFV) have served as key enablers in evolving the typical mobile network approach of a telecommunication platform toward end-to-end radio, computing, and storage with native support for intelligence. This section will complement the data inventory and interface definition of D2.1 with the data sources and interfaces for the edge/cloud computing and the transport domain. In addition, an example of a cross-domain interface definition is presented to highlight the benefits of supporting the exchange of telemetry data between different domains for advanced network optimization.

10.2 Edge and Cloud Computing Domain

NFV has revolutionized the network management approach that virtualizes traditional network functions such as firewalls, routers, and load balancers. The primary objective of NFV is to make networks more agile and responsive to changing demands by leveraging the power of virtualization. With NFV, network functions are deployed as VNFs that run on virtualized infrastructure, which can be dynamically scaled, updated, or moved across different locations as needed. This adaptability allows operators to respond faster to customer needs, roll out new services rapidly, and optimize resource use.

10.2.1 NFV MANO Architecture

To coordinate and manage these virtualized network functions effectively, the NFV MANO framework was established and plays a fundamental administrative role. NFV MANO defines the architecture, components, and interfaces necessary for managing the lifecycle of VNFs, allocating infrastructure resources, and orchestrating complex network services.

The main functional components of NFV MANO Architecture, as depicted in Figure 22, are:

- **NFV Orchestrator (NFVO)**: It is responsible for the creation and administration of the VNF catalog, Network Services (NS) catalog, NFV Instances, and resources of Network Functions Virtualization Infrastructure (NFVI).
- **VNF Manager (VNFM)**: Responsible for lifecycle management of VNF instances, from instantiation to scaling and termination.
- **Virtualized Infrastructure Manager (VIM)**: Controls and manages the networking, storage, and computing resources within the NFVI.

- **The WAN Infrastructure Manager (WIM)** is a functional block that provides management of Multi-Site Connectivity Services (MSCS). The WIM establishes the connectivity between NFVI-Point of Presence (NFVI-PoP)

The NFV MANO architecture must be integrated with open APIs in existing systems to operate efficiently and effectively. The MANO layer uses templates for standard VNFs and allows users to select from available NFVI resources to deploy their platforms.

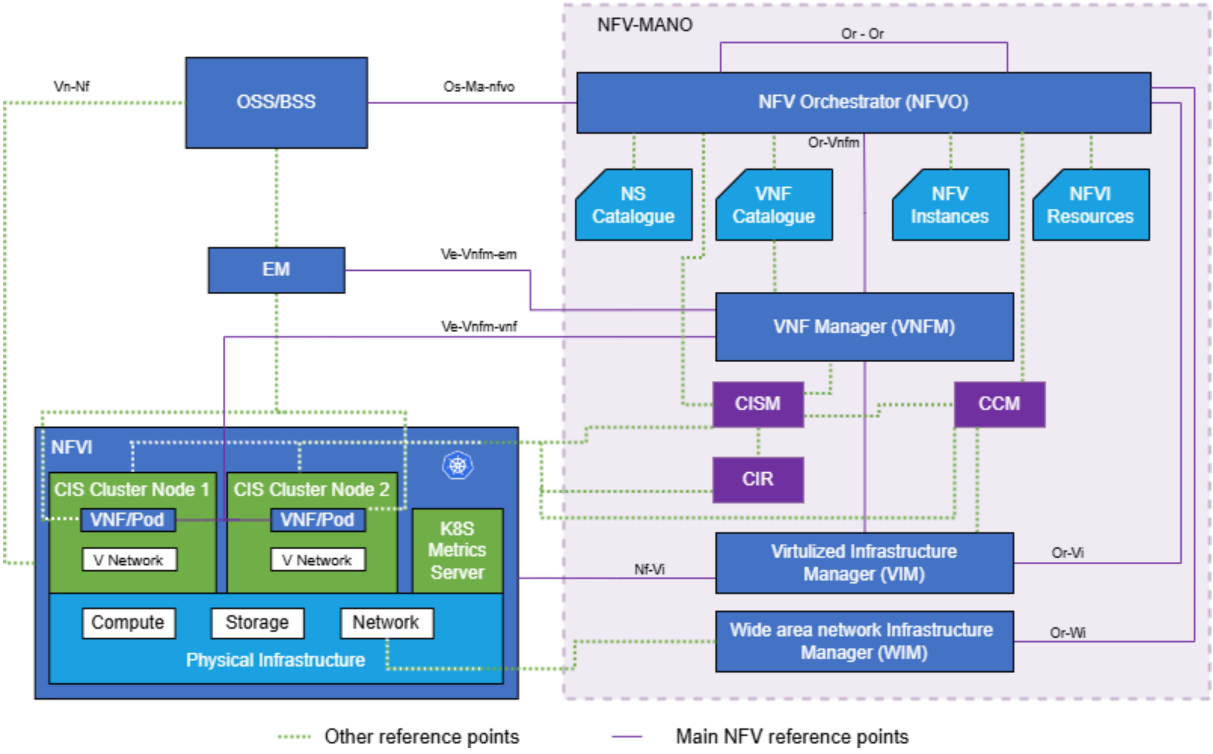


Figure 22 NFV MANO Architectural framework with containers support [128].

Other components of NFV MANO architecture are:

- **Operations Support Systems (OSS)** and **Business Support Systems (BSS)** are two key systems used by telecommunications providers and other service providers to manage their networks, operations, and customer interactions while they serve different purposes.
    - **OSS**: Covers the systems and software used to manage and operate the technical functions of a network. It focuses on managing the network and ensuring services are delivered with the required performance and reliability.
    - **BSS**: Includes systems that support customer-facing and revenue-related functions, such as billing, customer service, and order management. It focuses on managing the business aspects of the service providers' operations.
- **Element Management (EM):** It is a component that enables the configuration and customization of network elements, allowing operators to set parameters, deploy updates, and apply configurations to ensure each element operates as needed within

the network. It provides a layer of management directly over the network components, such as routers, switches, or VNFs.

- **Virtual Network Function (VNF):** is a software-based implementation of a specific network function such as firewall, router, load balancer and more. Each VNF is an individual component that runs within the NFV framework, Multiple VNFs can work together to form complete network services.
- **Network Function Virtualization Infrastructure (NFVI):** is a key component of the NFV architecture that describes the hardware and software components on which virtual networks are built. It includes compute, storage, and network resources, which are managed by the VIM.
- **Container Infrastructure Service (CIS):** is an instance or Container Infrastructure Service Management (CISM) instance (e.g., k8s), or both. A CIS cluster node can be either physical (e.g., a bare-metal server), or virtual (e.g., a virtual machine) [129].
- **Container Infrastructure Service Management (CISM):** is responsible for the deployment, monitoring, and lifecycle management of containerized workloads as MCIOs running in OS containers. The CISM function is exposing OS container management service [130].
- **Container Image Registry (CIR):** is responsible for storing and maintaining information of OS container software images [131].
- **CIS Cluster Management (CCM):** Is responsible for managing virtualized computers, storage and network resources for CIS clusters and further support the CIS cluster lifecycle management.

10.2.2 Data Types

In NFV MANO, three methods can be used to collect information [6]:

- **Status Counter (SC):** The entity receives a metric at each predetermined interval. A measurement is generated from processing (e.g. arithmetic mean, peak) all the samples received in the collection period.
- **Transparent Forwarding (TF):** The entity maintains a measurement count that stores the content of the metric that it received.
- **Object Mapping (OM):** The entity receives a metric for measured object A in the collection period and maps the received metric from measured object A to measured object B. A measurement is generated for measured object B by processing the metric(s), which may be mapped from one or more measured object(s) A to a single measured object B. It is noted that:
    - The source metric for measured object A and the target measurement for measured object B may or may not contain sub counters. How the mapping is done for the case that either of the source metric and target measurement contain sub counters is to be defined case by case in the trigger of the measurement definition.

  - Multiple source metrics for measured object A may be mapped to a single target measurement for measured object B. How the mapping is done for this case is to be defined in the trigger of the measurement definition

In order to collect performance data from VNFs, the use of Performance Monitoring (PM) Jobs, Connection Points (CP), and Service Access Points (SAP) is required. The PM Job acts as a task scheduler, specifying the metrics to be collected such as CPU usage, memory utilization, and network throughput along with the frequency of data collection (e.g., every 5 minutes) and the retention period for this data, which determines how long it will be stored for analysis or reporting.

The Connection Point (CP) serves as an interface within a VNF that enables connectivity between different virtual links or between VNFs and the underlying network infrastructure. CPs define how traffic flows into, out of, and within the VNF, ensuring seamless communication and data routing within the VNF environment.

The SAP represents an instance of a Network Service (NS) that interacts with external networks, services, or other NS instances. SAPs are composed of one or more VNFs or Physical Network Functions (PNFs) working together to deliver a specific end-to-end service, thus facilitating connectivity and service interaction across the network.

The Container Infrastructure Service (CIS) Is a group of clustered compute resources within the NFVI where resources like CPU, memory, and storage, are provided with high availability and scalability (e.g., k8s).

For collecting performance data, it is essential to consider the measured object types within NFV MANO, as summarized in Table 15.

Table 15 Description of the measured object types within NFV MANO.

Measured object	Description and variables
Virtual Compute	Used to collect and report the performance measurements for one or more instances of the Virtualized compute resource. Its variables are: • The *objectType*, when used in PM job or performance report, is equal to "VirtualCompute". • The *objectInstanceId,* when used in PM job or performance report, corresponds to *computeId* of the measured Virtualized compute resource.
VNF	Used to collect and report the performance measurements for one or more VNF instances. Its variables are: • The *objectType*, when used in PM job or performance report, is equal to "Vnf".

	• The *objectInstanceId*, when used in PM job or performance report, corresponds to *vnfInstanceId* that identifies the measured VNF instance.
VNF Component	Used to collect and report the performance measurements for one or more VNFC instances. Its variables are: • The objectType, when used in PM job or performance report, is equal to "Vnfc". • The objectInstanceId, when used in PM job or performance report, corresponds to the vnfInstanceId of the VNF instance that contains the measured VNFC instance. • The subObjectInstanceId, when used in PM job or performance report, corresponds to the vnfcInstanceId that identifies the measured VNFC instance.
Virtual Network	Used to collect and report the performance measurements for one or more instances of the virtual network. Its variables are: • The objectType, when used in PM job or performance report, is equal to "VirtualNetwork". • The objectInstanceId, when used in PM job or performance report, corresponds to networkResourceId of the measured virtual network instance.
Virtual Storage	Used to collect and report the performance measurements for one or more instances of the Virtualized storage resource. Its variables are: The objectType, when used in PM job or performance report, is equal to "VirtualStorage". The objectInstanceId, when used in PM job or performance report, corresponds to storage Id of the measured Virtualized storage resource.
Internal connection point of a VNF instance	Used to collect and report the performance measurements for one or more instances of the VNF internal CP. Its variables are: • The objectType, when used in PM job or performance report, is equal to "VnfIntCp". • The objectInstanceId, when used in PM job or performance report, corresponds to the vnfInstanceId of the VNF instance which the VNF internal CP belongs to. The subObjectInstanceId, when used in PM job or performance report, corresponds to the instance identifier of the measured VNF internal CP instance.
External connection point of a VNF instance	Used to collect and report the performance measurements for one or more instances of the VNF external CP. Its variables are:

	• The objectType, when used in PM job or performance report, is equal to "VnfExtCp". • The objectInstanceId, when used in PM job or performance report, corresponds to the vnfInstanceId of the VNF instance that exposes the measured VNF external CP instance. The subObjectInstanceId, when used in PM job or performance report, corresponds to the instance identifier of the measured VNF external CP instance.
Service Access Point	Used to collect and report the performance measurements for one or more SAP instances of an NS instance. Its variables are: • The objectType, when used in PM job or performance report, is equal to "Sap". • The objectInstanceId, when used in PM job or performance report, corresponds to theNsInstanceId of the NS instance that exposes the measured SAP. The subObjectInstanceId, when used in PM job or performance report, corresponds to the instance identifier of the measured SAP instance.
Cis cluster node	Used to collect and report the performance measurements for one or more CIS cluster nodes. Its variables are: • The objectType, when used in PM job or performance report, is equal to "CisClusterNode". • The objectInstanceId, when used in PM job or performance report, corresponds to cisClusterNodeId that identifies the measured CIS cluster node.
MCIO-C	Used to collect and report the performance measurements for one or more instances of a Compute Management, Control, Infrastructure, and Orchestration (MCIO) which represents one or more OS containers. Its variables are: • The objectType, when used in PM job or performance report, is equal to "Mcio-c". • The objectInstanceId, when used in PM job or performance report, corresponds to "ID of Compute MCIO" mapped to the resource handle's "resourceId" of the measured compute MCIO. NOTE: Pod is an example of Compute MCIO corresponding to a group of one or more OS containers in a K8S environment.

10.2.3 Interfaces

In NFV MANO, the interfaces are called reference points, which define a specific connection between two functional components. Reference points specify how these components interact, what types of messages they exchange, and the protocols they use to communicate, ensuring interoperability and a well-coordinated NFV environment. Table 16 lists the main interfaces used in NFV MANO.

Table 16 Main interfaces in NFV MANO.

Interface	Operation support
Os-Ma-nfvo	This reference point is used for exchanges between OSS/BSS and NFV Orchestrator, and supports the following [132]: • Network Service Descriptor and VNF package management. • Network Service instance lifecycle management: o Network Service instantiation. o Network Service instance update. o Network Service instance query. o Network Service instance scaling. o Network Service instance termination. • VNF lifecycle management. • Policy management and/or enforcement for Network Service instances. • Querying relevant Network Service instance and VNF instance information from the OSS/BSS. • Forwarding events.
Ve-Vnfm-em	This reference point is used for exchanges between EM and VNF Manager, and supports the following [133]: • VNF instantiation. • VNF instance query. • VNF instance update. • VNF instance scaling out/in, and up/down. • VNF instance termination. • Forwarding of configuration and events from the EMS to the VNFM. • Forwarding of configuration and events regarding the VNF from the VNFM to the EMS. NOTE: This reference point is only used if the EM is aware of virtualization.

Ve-Vnfm-vnf	This reference point is used for exchanges between VNF and VNF Manager, and supports the following: • VNF instantiation. • VNF instance query. • VNF instance update. • VNF instance scaling out/in, and up/down. • VNF instance termination. • Forwarding of configuration and events from the VNF to the VNFM. • Forwarding of configuration, events, etc. regarding VNF, from the VNFM to the VNF. • Verification that the VNF is still alive/functional.
Nf-Vi	• This reference point is used for exchanges between virtualization Infrastructure Manager and NFV Infrastructure, and supports the following: o Allocate VM with indication of compute/storage resource. o Update VM resources allocation. o Migrate VM. o Terminate VM. o Create connections between VMs. o Configure connection between VMs. o Remove connection between VMs. o Forwarding of configuration information, failure events, measurement results, and usage records regarding NFVI (physical, software, and virtualized resources) to the VIM.
Or-Vnfm	This reference point is used for exchanges between NFV Orchestrator and VNF Manager, and supports the following [134]: • NFVI resources authorization/validation/reservation/release for a VNF. • NFVI resources allocation/release request for a VNF. • VNF instantiation. • VNF instance query. • VNF instance update. • VNF instance scaling out/in, and up/down. • VNF instance termination. • VNF package query. • Forwarding of events, other state information about the VNF that may also impact the Network Service instance.
Or-Vi	This reference point is used for exchanges between the NFV Orchestrator and the VIM, and supports the following [135]: • NFVI resource reservation/release.

	• NFVI resource allocation/release/update. • VNF software image addition/deletion/update. • Forwarding of configuration information, events, measurement results, usage records regarding NFVI resources to the NFV Orchestrator.
Vi-Vnfm	This reference point is used for exchanges between the VNF Manager and the Virtualized Infrastructure Manager, and supports the following [128]: • NFVI resources reservation information retrieval. • NFVI resources allocation/release. • Exchanging configuration information between reference point pairs and sending such information to the VNF Manager for which the VNFM has subscribed.

10.3 Transport Network Domain

Traditional TN form the backbone of telecommunications, enabling the delivery of massive data streams across regional, national, and international boundaries. These networks rely on fixed, hardware-centric architectures composed of vendor-specific equipment and protocols. While reliable, this rigidity has led to several limitations. Operators must manage high costs associated with maintaining and upgrading the infrastructure, static network configurations, and complex operations reliant on proprietary systems. As data traffic continues to surge, driven by innovations like streaming services and cloud computing, the capacity for these legacy systems to scale efficiently is being severely tested [136].

A critical gap exists between the demands of modern network services and the capabilities of these static architectures. Traditional networks lack the programmability required to dynamically allocate resources, respond to fluctuating traffic patterns, or support the rapid deployment of new services. Moreover, the reliance on vendor-specific technologies often results in operational silos, increasing costs and hindering innovation. These inefficiencies have led network operators to seek transformative solutions that can address these challenges while reducing Operational Expenses (OpEx) and capital expenditures (CapEx)[137].

Transport Software-Defined Networking (T-SDN) emerges as a revolutionary approach to bridging this gap [138]. T-SDN leverages the principles of Software-Defined Networking (SDN) by decoupling the control and data planes, introducing centralized control, and fostering network programmability. Unlike its successful implementation in data centers, adapting SDN for TNs involves overcoming the inherent complexities of multi-layered, multi-domain, and multi-vendor systems. T-SDN promises to deliver dynamic resource management, seamless integration across technologies, and enhanced scalability, transforming TNs into agile, future-ready infrastructures.

10.3.1 Transport Networks Evolution

TN technologies evolve from early systems like Plesiochronous Digital Hierarchy (PDH) to advanced solutions such as Optical Transport Network (OTN). Each step reflects significant improvements in efficiency, reliability, and flexibility, meeting the demands of increasingly complex and data-intensive applications. In the following discussion, we will describe some of these technologies and examine their unique features.

Synchronous Optical NETwork/Synchronous Digital Hierarchy (SONET/SDH) [139]: These synchronous networks multiplex digital signals into optical signals, offering reliability and standardized interfaces. However, their static configurations and limited scalability make them less suitable for modern data demands.

PDH [140]: An older asynchronous technology, PDH lacks a global standard for interface design, making it less flexible and interoperable compared to SDH.

Multi-Service Transmission Platforms (MSTP) [141]: Evolving from SDH, MSTP integrates capabilities to handle multiple services, including Ethernet and ATM, within a single device. This evolution simplifies service delivery while retaining SDH's inherent advantages of reliability and protection mechanisms.

Packet Transport Networks (PTN) [142]**:** PTN introduces packet-based operations within TNs, creating a layer between IP services and optical transmission mediums. PTNs are designed to handle bursty traffic efficiently and are compatible with both legacy services and newer technologies like SDN. Their core strengths include robust VPN capabilities, scalable architecture, and low operational cost.

Wavelength Division Multiplexing (WDM) [143]: WDM enhances bandwidth efficiency by enabling simultaneous transmission of multiple data streams across different wavelengths of light in a single optical fiber.

Optical Transport Networks (OTN) [144]: OTN is a sophisticated, standards-driven framework that combines optical transmission with electrical processing to provide transparent, high-capacity, and efficient data transport. It supports features like transparent transmission of diverse client signals, advanced error correction and fault detection mechanisms, and scalability for dynamic high bandwidth demands.

Data Center Interconnect (DCI) [145]: DCI facilitates communication between data centers over long distances (up to 100 KMs) using high-capacity links, including WDM-based solutions, addressing challenges like scalability, low latency, and operational simplicity.

10.3.2 Transport Networks Architecture

As the demand for high-bandwidth, flexible, and scalable network infrastructures continues to rise, transport technologies are evolving to align with the latest advancements in network management, particularly SDN [146], [147], [138]. The integration of these technologies with SDN enables more dynamic, programmable, and efficient network control. This integration has been captured in the standardization efforts of the ONF architecture for TNs presented in [148]. Figure 23 presents the generic SDN framework (right) and an example of two controllers connected to their own Network Elements (NEs).

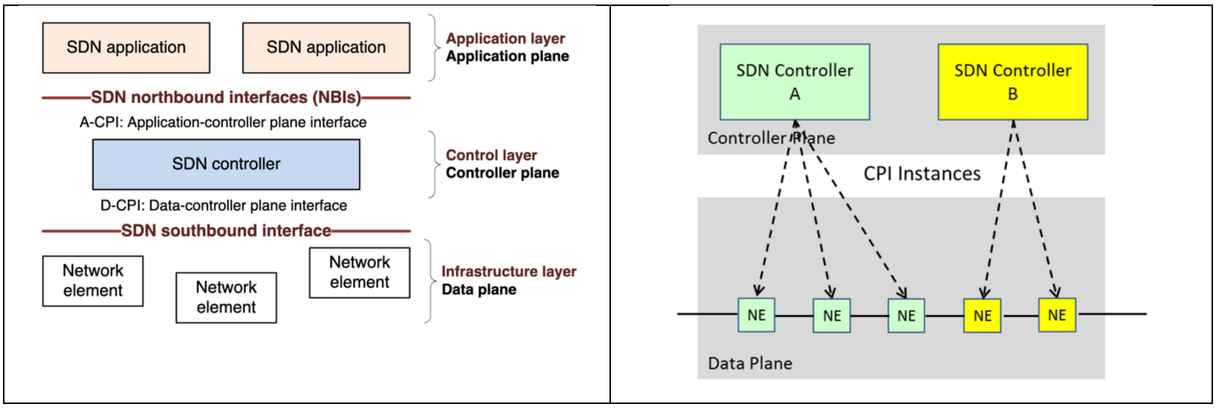


Figure 23 SDN components [147] (right) and two controllers managing their respective NEs [148] (left).

The ONF Information Modeling project has developed a **Core Information Model (CIM)** [149] to manage and control network components in a technology-independent manner. The CIM is extensible, allowing for the inclusion of technology-specific fragments for standards like OTN and Ethernet. These extensions can then be mapped to specific control interfaces.

10.3.3 Data Types

In the context of TN, various data types can be collected to monitor and manage network performance effectively. In modern TNs, several elements can serve as telemetry data generators [150]:

- Optical devices including transceivers (including the power-efficient DSP), amplifiers, ROADMs, and interrogators.
- Packet nodes, including IP over WDM (IPoWDM) white box switches and SmartNIC/DPU, providing telemetry data/metadata using In-band telemetry or post-card telemetry.
- Computing nodes, e.g., providing statistics about CPU, memory usage, from Kubernetes etc.
- SDN Controllers/Orchestrator.
    - RIC, collecting and aggregating data from wireless devices.
    - Transport SDN Controller / Orchestrator generating data about network events associated to network operation.

Similarly, the following elements serve as consumers of telemetry data:

- **Telemetry collector:** a dedicated element designed to efficiently process telemetry data, typically adopted in centralized/hierarchical scenarios, collecting data from large number of producers, useful for network-level close loop operations.

- **SmartNIC/DPU:** enabling local processing of received telemetry data, taking advantage of embedded CPU/GPU resources. It would enable decentralized scenarios, collecting data from a limited number of producers, useful for node-level (local) close-loop operations.

In general, control (e.g., controller) and data (e.g., switch) network elements can generate telemetry data at different levels to provide insights into the operational status, and performance metrics within the TN [151].

- **Network Performance Data:** Network performance data is crucial for assessing the efficiency and reliability of TNs. This data helps operators understand how well the network is performing under varying conditions and traffic loads.
  - **Bandwidth Utilization:** Measures the amount of bandwidth being used compared to the total available bandwidth.
  - **Latency:** The time taken for data to travel from the source to the destination.
  - **Packet Loss Rates:** The percentage of packets that fail to reach their destination, indicating potential issues in the network.
  - **Jitter:** The variation in packet arrival times, which can affect the quality of real-time applications.
- **Operational Data:** Operational data provides insights into the functioning of network devices and their configurations. This information is essential for troubleshooting and ensuring optimal performance.
  - **Device Status:** Information on the operational state of network devices, including uptime and error rates.
  - **Configuration Data:** Details about the settings and configurations of network devices, which can impact performance and reliability.
- **Telemetry Data:** Telemetry data is essential for real-time monitoring and management of network performance. This data allows operators to respond quickly to issues as they arise.
  - **SNMP Traps and Alarms:** Alerts generated by network devices indicating issues or changes in state.
  - **Performance Counters:** Metrics collected from network interfaces, such as throughput and error counts.
- **Traffic Data:** Traffic data provides insights into the flow of data through the network, helping operators manage congestion and optimize resource allocation.
  - **Flow Data:** Information about data flows through the network, including source and destination IP addresses, protocols used, and volume of data transferred.
- **Environmental Data:** Monitoring environmental conditions is crucial for maintaining the health and performance of network equipment.
  - **Temperature and Power Levels:** Data on the operational environment of network devices, which can affect their performance and longevity.

10.3.4 Interfaces

To collect data from the networks and control the devices, interfaces in the TNs play a crucial role in enabling communication between different network components, ensuring interoperability, and facilitating the programmability and automation essential for modern network operations. As TNs evolve to embrace technologies like SDN, these interfaces have become more sophisticated, bridging the gap between various layers of the network architecture and supporting multi-domain, multi-layer, and multi-vendor environments.

TNs rely on several key interfaces to ensure efficient communication and interoperability between their diverse components. These interfaces are categorized into Northbound Interfaces (NBI), Southbound Interfaces (SBI), and sometimes East/West Interfaces depending on the level of interaction and the entities involved [148].

10.3.4.1 *Northbound Interfaces (NBI)*

Northbound Interfaces connect the control plane to the application plane, allowing high-level applications to interact with the network's underlying infrastructure. These interfaces abstract the complexity of the network, providing a simplified and unified view to applications and operators. Among its functionalities, we can enumerate:

- Enable centralized network management and orchestration by exposing topology, resource availability, and performance metrics.
- Support the deployment of new services and the enforcement of policies without direct interaction with individual devices.
- Allow automation and network optimization through programmatic access.
- Examples:
- Transport API (TAPI): Defined by the Open Networking Foundation (ONF), TAPI provides standard APIs for network resource discovery, service configuration, and lifecycle management.
- RESTful APIs: Frequently used due to their simplicity and compatibility with modern software frameworks.
- Network Abstraction: Applications can use northbound APIs to execute complex tasks like path computation, service provisioning, and performance monitoring.

10.3.4.2 *Southbound Interfaces (SBI)*

Southbound Interfaces are the link between the control plane and the data plane, facilitating communication between SDN controllers and network devices. SBIs allow the control plane to program devices, gather operational data, and manage the physical network's underlying infrastructure. Among its functionalities, we can enumerate:

- Translate abstracted, high-level instructions from the control plane into specific configurations for network elements (e.g., routers, switches, optical equipment).
- Provide real-time state and performance updates from the data plane to the control plane.
- Enable dynamic resource allocation and fault recovery through centralized control.
- Examples:

- OpenFlow: A foundational SBI protocol for SDN, OpenFlow has been extended to support transport-specific requirements like circuit switching and optical layers.
- NETCONF/YANG: Widely used for device configuration and state monitoring, especially in optical and TNs.
- PCEP (Path Computation Element Protocol): Used in SDN/GMPLS hybrid architectures for dynamic path computation and optimization.
- Vendor-Specific Interfaces: Many legacy systems still rely on proprietary SBIs, though open standards are gradually replacing these.

10.3.4.3 *East/West Interfaces*

East/West Interfaces facilitate peer-to-peer communication between controllers in distributed or flat architectures. These interfaces are vital in multi-domain networks where different controllers must coordinate to ensure seamless operation. Among its functionalities, we can enumerate:

- Enable exchange of topology and state information between domains.
- Support inter-controller coordination for service provisioning, fault management, and resource allocation.
- Allow for interoperability in heterogeneous environments.
- Examples:
- Controller-to-controller protocols for sharing routing and topology information.
- Early proposals for East/West protocols include inter-domain solutions for Elastic Optical Networks (EON) and multi-domain SDN orchestration, though standardization is still emerging in this area.

Co-funded by the European Union

11 References

[1] Y. Wu, K. Zhang, and Y. Zhang, "Digital Twin Networks: A Survey," *IEEE Internet Things J.*, vol. 8, no. 18, pp. 13789–13804, Sep. 2021, doi: 10.1109/JIOT.2021.3079510.

[2] 6G-TWIN Project, "Deliverable D1.1: 6G-TWIN architecture and technical foundations (initial)," 6G-TWIN Project, 2024. [Online]. Available: https://6g-twin.eu/resources/#deliverables

[3] 6G-TWIN Project, "Deliverable D1.1: 6G-TWIN architecture and technical foundations (initial)," 6G-TWIN Project, 2024. [Online]. Available: https://6g-twin.eu/resources/#deliverables

[4] IETF, "Data Collection Requirements and Technologies for Digital Twin Network," IETF, Draft, Jul. 2023. [Online]. Available: https://www.ietf.org/archive/id/draft-zcz-nmrg-digitaltwin-data-collection-03.txt

[5] O.-RAN Alliance, "O-RAN O1 Performance Measurements Specification," O-RAN Alliance, Oct. 2024. [Online]. Available: https://specifications.o-ran.org/specifications

[6] E. G. NFV-IFA 027, "Network Functions Virtualisation (NFV) Release 5; Management and Orchestration; Performance Measurements Specification V.5.2.1," ETSI, Sep. 2024.

[7] ETSI, "Fifth Generation Fixed Network (F5G); Data Models of Telemetry for Access Network," ETSI, Group Specification, Jun. 2023. [Online]. Available: {https://www.etsi.org/deliver/etsi_gs/F5G/001_099/016/01.01.01_60/gs_F5G016v010101p.pdf}

[8] 3GPP, "TS 38.314 - 5G NR; Layer 2 measurements," 3GPP, Technical Specification, May 2024.

[9] 3GPP, "TS 28.552 - Management and orchestration; 5G performance measurements," 3GPP, Technical Specification, Sep. 2024.

[10] 3GPP, "TS 32.404 - Telecommunication management; Performance Management (PM); Performance Measurements; Definition and template," 3GPP, Technical Specification, Sep. 2011.

[11] H. Song, F. Qin, P. Martinez-Julia, L. Ciavaglia, and A. Wang, "Network Telemetry Framework." May 2022. doi: 10.17487/RFC9232.

[12] ISO/IEC 10746, "Information technology — Open Distributed Processing — Reference model: Overview," ISO/IEC, 1998. [Online]. Available: https://www.iso.org/standard/20696.html

[13] H. Song, F. Qin, P. Martinez-Julia, L. Ciavaglia, and A. Wang, "Network Telemetry Framework." May 2022. doi: 10.17487/RFC9232.

[14] H. Isah, T. Abughofa, S. Mahfuz, D. Ajerla, F. Zulkernine, and S. Khan, "A Survey of Distributed Data Stream Processing Frameworks," *IEEE Access*, vol. 7, pp. 154300–154316, 2019, doi: 10.1109/ACCESS.2019.2946884.

[15] ITU-T, "Digital twin network - Requirements and architecture," ITU, Recommendation, 2022. [Online]. Available: https://www.itu.int/rec/T-REC-Y.3090-202202-I/en

[16] J. Case, M. Fedor, M. Schoffstall, and J. Davin, "A Simple Network Management Protocol (SNMP)." 1990. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc1157

[17] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman, "Network Configuration Protocol (NETCONF)." 2011. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc6241

[18] Cisco Systems, “Introduction to Data Models - Programmatic and Standards-Based Configuration.” [Online]. Available: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/166/b_166_programmability_cg/configuring_yang_datamodel.pdf

[19] Rayka, “What is NETCONF Protocol?” [Online]. Available: https://rayka-co.com/lesson/what-is-netconf-protocol/

[20] Cisco Systems, “RESTCONF Programmable Interface.” [Online]. Available: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/166/b_166_programmability_cg/restconf_prog_int.pdf

[21] gRPC, “OpenTelemetry Metrics.” [Online]. Available: https://grpc.io/docs/guides/opentelemetry-metrics/

[22] gRPC, “gRPC Documentation.” [Online]. Available: https://grpc.io/

[23] OpenTelemetry, “Metrics API: MeterProvider.” [Online]. Available: https://opentelemetry.io/docs/specs/otel/metrics/api/#meterprovider

[24] IBM, “IPFIX Overview.” [Online]. Available: https://www.ibm.com/docs/en/npi/1.3.1?topic=insight-ipfix-overview

[25] P. Phaal, S. Panchen, and N. McKee, “InMon corporation’s sFlow: A method for monitoring traffic in switched and routed networks,” 2001. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc3176

[26] sFlow, “sFlow Overview.” [Online]. Available: https://sflow.org/sFlowOverview.pdf

[27] IBM, “NetFlow.” [Online]. Available: https://www.ibm.com/topics/netflow

[28] B. Claise, “Cisco Systems NetFlow Services Export Version 9.” [Online]. Available: https://www.rfc-editor.org/rfc/rfc3954

[29] M. Bjorklund, “YANG-a data modeling language for the network configuration protocol (NETCONF),” 2010.

[30] OASIS, “Topology and Orchestration Specification for Cloud Applications Version 1.0 (TOSCA).” [Online]. Available: https://docs.oasis-open.org/tosca/TOSCA/v1.0/TOSCA-v1.0.html

[31] M. Pattaranantakul, R. He, Z. Zhang, A. Meddahi, and P. Wang, “Leveraging Network Functions Virtualization Orchestrators to Achieve Software-Defined Access Control in the Clouds,” *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 1, pp. 372–383, Jan. 2021, doi: 10.1109/TDSC.2018.2889709.

[32] R. Bless, “A lower-effort per-hop behavior (LE PHB) for differentiated services,” 2019.

[33] H. Song and J. Gong, “Data-Networking Pipeline for In-network Intelligence Query (DNP4IQ).” [Online]. Available: https://datatracker.ietf.org/doc/html/draft-song-opsawg-dnp4iq-01

[34] F. Brockners, S. Bhandari, and T. Mizrahi, “Data Fields for In Situ Operations, Administration, and Maintenance (IOAM).” [Online]. Available: https://www.rfc-editor.org/rfc/inline-errata/rfc9197.html

[35] J. Song, R. Huang, T. Zhou, Z. Li, and R. Pang, “Postcard-based On-Path Flow Data Telemetry.” [Online]. Available: https://datatracker.ietf.org/doc/html/draft-song-ippm-postcard-based-telemetry-07

[36] B. Claise, A. Johnson, and J. Quittek, “Packet Sampling (PSAMP) Protocol Specification.” [Online]. Available: https://www.rfc-editor.org/rfc/rfc5476.txt

[37]Microsoft, “Azure Digital Twins.” [Online]. Available: https://learn.microsoft.com/en-us/connectors/azuredigitaltwins/

[38]Inkwell Data, “Digital twins for industrial IoT with Altior.” 2020. [Online]. Available: https://inkwelldata.com/wp-content/uploads/2021/03/Digital-twins-for-industrial-IoT-with-Altior-Part-2.pdf

[39]S. L. Feibish, Z. Liu, and J. Rexford, “Compact Data Structures for Network Telemetry,” *ArXiv Prepr. ArXiv231102636*, 2023.

[40]J. Liu, P. Xun, and B. Wang, “Relevant Backtracking: An Efficient Telemetry Data Collection Method for Data Center Networks,” in *2023 15th International Conference on Communication Software and Networks (ICCSN)*, IEEE, Jul. 2023, pp. 100–107. doi: 10.1109/ICCSN57992.2023.10297330.

[41]M. Polverini, S. Sardellitti, S. Barbarossa, A. Cianfrani, P. D. Lorenzo, and M. Listanti, “Reducing the In band Network Telemetry overhead through the spatial sampling: Theory and experimental results,” *Comput. Netw.*, vol. 242, p. 110269, Apr. 2024, doi: 10.1016/j.comnet.2024.110269.

[42]P. Foroughi, F. Brockners, and J.-L. Rougier, “ADT: AI-Driven network Telemetry processing on routers,” *Comput. Netw.*, vol. 220, p. 109474, Jan. 2023, doi: 10.1016/j.comnet.2022.109474.

[43]Apache, “Apache NiFi.” 2024. [Online]. Available: https://nifi.apache.org/

[44]Talend, “Talend Data Integration.” 2024. [Online]. Available: https://www.talend.com/

[45]Apache, “Apache Camel.” 2024. [Online]. Available: https://camel.apache.org/

[46]MuleSoft, “MuleSoft Anypoint Platform.” 2024. [Online]. Available: https://www.mulesoft.com/platform/enterprise-integration

[47]dbt, “Data Build Tool.” 2024. [Online]. Available: https://www.getdbt.com/

[48]Apache, “Apache Kafka.” 2024. [Online]. Available: https://kafka.apache.org/

[49]Google, “Tensorflow.” 2024. [Online]. Available: https://www.tensorflow.org/

[50]InfluxData, “Influx DB.” 2024. [Online]. Available: https://www.influxdata.com/

[51]Grafana Labs, “Grafana.” 2024. [Online]. Available: https://grafana.com/

[52]N. Q. Uy and V. H. Nam, “A comparison of AMQP and MQTT protocols for Internet of Things,” in *2019 6th NAFOSTED Conference on Information and Computer Science (NICS)*, IEEE, Dec. 2019, pp. 292–297. doi: 10.1109/NICS48868.2019.9023812.

[53]R. Gerhards, “The Syslog Protocol,” IETF, Standard, Mar. 2009. [Online]. Available: {https://datatracker.ietf.org/doc/html/rfc5424}

[54]Amazon, “Amazon Kinesis.” 2024. [Online]. Available: https://aws.amazon.com/kinesis/

[55]Google, “What is Pub/Sub?” 2024. [Online]. Available: https://cloud.google.com/pubsub/docs/overview

[56]Apache, “Apache Airflow.” 2024. [Online]. Available: https://airflow.apache.org/

[57]Prometheus, “Prometheus.” 2024. [Online]. Available: https://prometheus.io/

[58]Amazon, “Amazon S3.” 2024. [Online]. Available: https://aws.amazon.com/s3/

[59]Google, “Google Cloud Storage.” 2024. [Online]. Available: https://cloud.google.com/storage

[60] Amazon, “Amazon Redshift.” 2024. [Online]. Available: https://aws.amazon.com/redshift/

[61] Google, “Google BigQuery.” 2024. [Online]. Available: https://cloud.google.com/bigquery/

[62] PyTorch, “PyTorch.” 2024. [Online]. Available: https://pytorch.org/

[63] Open Networking Foundation, “Open Networking Operating System.” 2024. [Online]. Available: https://opennetworking.org/onos/

[64] The Linux Foundation, “OpenDaylight.” 2024. [Online]. Available: https://www.opendaylight.org/

[65] RYU, “RYU SDN.” 2024. [Online]. Available: https://ryu-sdn.org/

[66] European Telecommunications Standards Institute, “Network Functions Virtualisation (NFV).” 2024. [Online]. Available: https://www.etsi.org/technologies/nfv

[67] G. Baldoni, J. Quevedo, C. Guimarães, A. de la Oliva, and A. Corsaro, “Data-Centric Service-Based Architecture for Edge-Native 6G Network,” *IEEE Commun. Mag.*, vol. 62, no. 4, pp. 32–38, Apr. 2024, doi: 10.1109/MCOM.001.2300178.

[68] H. Serrano-Magaña, A. González-Potes, V. Ibarra-Junquera, P. Balbastre, D. Martínez-Castro, and J. Simó, “Software Components for Smart Industry Based on Microservices: A Case Study in pH Control Process for the Beverage Industry,” *Electronics*, vol. 10, no. 7, p. 763, Mar. 2021, doi: 10.3390/electronics10070763.

[69] Altexsoft Software R&D Engineering, “Event-Driven Architecture and Pub/Sub Pattern Explained.” 2021. [Online]. Available: https://www.altexsoft.com/blog/event-driven-architecture-pub-sub/

[70] J. Zhang, X. Yu, S. Ha, J. Pena Queralta, and T. Westerlund, “Comparison of DDS, MQTT, and Zenoh in Edge-to-Edge and Edge-to-Cloud Communication for Distributed ROS 2 Systems,” *ArXiv E-Prints*, p. arXiv-2309, 2023.

[71] P. André, F. Azzi, and O. Cardin, “Heterogeneous communication middleware for digital twin based cyber manufacturing systems,” in *Service Oriented, Holonic and Multi-agent Manufacturing Systems for Industry of the Future: Proceedings of SOHOMA 2019 9*, Springer, 2020, pp. 146–157.

[72] A. De Benedictis, F. Rocco di Torrepadula, and A. Somma, “A Digital Twin Architecture for Intelligent Public Transportation Systems: A FIWARE-Based Solution,” in *International Symposium on Web and Wireless Geographical Information Systems*, Springer, 2024, pp. 165–182.

[73] P. Singh and N. Kaur, “A Review: Cloud Computing using Various Task Scheduling Algorithms,” *Int. J. Comput. Appl.*, vol. 142, no. 7, pp. 30–32, May 2016, doi: 10.5120/ijca2016909931.

[74] J. Kreps, N. Narkhede, J. Rao, and others, “Kafka: A distributed messaging system for log processing,” in *Proceedings of the NetDB*, Athens, Greece, 2011, pp. 1–7.

[75] J. Dantas and J. Almeida, “AMQP: Advanced Message Queuing Protocol for Message-Oriented Middleware Systems,” *J. Comput. Sci. Technol.*, vol. 31, no. 1, pp. 69–87, 2016.

[76] S. T and S. N. K, “A study on Modern Messaging Systems- Kafka, RabbitMQ and NATS Streaming,” Dec. 08, 2019, *arXiv*: arXiv:1912.03715. doi: 10.48550/arXiv.1912.03715.

[77] W.-Y. Liang, Y. Yuan, and H.-J. Lin, “A performance study on the throughput and latency of zenoh, mqtt, kafka, and dds,” *ArXiv Prepr. ArXiv230309419*, 2023.

[78]S. Dilek, K. Irgan, M. Guzel, S. Ozdemir, S. Baydere, and C. Charnsripinyo, “QoS-aware IoT networks and protocols: A comprehensive survey,” *Int. J. Commun. Syst.*, vol. 35, no. 10, p. e5156, 2022.

[79]F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the internet of things,” in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, ACM, Aug. 2012, pp. 13–16. doi: 10.1145/2342509.2342513.

[80]J. B. Correia, M. Abel, and K. Becker, “Data management in digital twins: a systematic literature review,” *Knowl. Inf. Syst.*, vol. 65, no. 8, pp. 3165–3196, Aug. 2023, doi: 10.1007/s10115-023-01870-1.

[81]F. Tao and M. Zhang, “Digital Twin Shop-Floor: A New Shop-Floor Paradigm Towards Smart Manufacturing,” *IEEE Access*, vol. 5, pp. 20418–20427, 2017, doi: 10.1109/ACCESS.2017.2756069.

[82]E. Curry, “The Big Data Value Chain: Definitions, Concepts, and Theoretical Approaches,” in *New Horizons for a Data-Driven Economy*, Springer International Publishing, 2016, pp. 29–37. doi: 10.1007/978-3-319-21569-3_3.

[83]M. Al-Mekhlal and A. A. Khwaja, “A Synthesis of Big Data Definition and Characteristics,” in *2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, IEEE, Aug. 2019, pp. 314–322. doi: 10.1109/CSE/EUC.2019.00067.

[84]C. Dobre and F. Xhafa, “Parallel Programming Paradigms and Frameworks in Big Data Era,” *Int. J. Parallel Program.*, vol. 42, no. 5, pp. 710–738, Oct. 2014, doi: 10.1007/s10766-013-0272-7.

[85]J. Nuikka, “Comparison of Cloud Native messaging technologies,” Master’s Thesis, Tampere University, 2021.

[86]M. van Steen and A. S. Tanenbaum, “A brief introduction to distributed systems,” *Computing*, vol. 98, no. 10, pp. 967–1009, Oct. 2016, doi: 10.1007/s00607-016-0508-7.

[87]A. Kafka, “Message Delivery Semantics.” 2024. [Online]. Available: https://kafka.apache.org/documentation/#semantics

[88]A. Corsaro *et al.*, “Zenoh: Unifying Communication, Storage and Computation from the Cloud to the Microcontroller,” in *2023 26th Euromicro Conference on Digital System Design (DSD)*, IEEE, Sep. 2023, pp. 422–428. doi: 10.1109/DSD60849.2023.00065.

[89]C.-S. Shih, H.-J. Lin, Y. Yuan, Y.-H. Kuo, and W.-Y. Liang, “Scalable and Bounded-time Decisions on Edge Device Network using Eclipse Zenoh,” in *2022 IEEE 28th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, IEEE, Aug. 2022, pp. 170–179. doi: 10.1109/RTCSA55878.2022.00024.

[90]K. Peeroo, P. Popov, and V. Stankovic, “Exploring the Effects of Multicast Communication on DDS Performance,” *ArXiv Prepr. ArXiv220909001*, 2022.

[91]Zenoh, “Zenoh Reliability, Scalability and Congestion Control.” 2021. [Online]. Available: https://zenoh.io/blog/2021-06-14-zenoh-reliability/

[92]Zenoh, “Configuration.” 2024. [Online]. Available: https://zenoh.io/docs/manual/configuration/

[93]Zenoh, “Storage manager plugin.” 2024. [Online]. Available: https://zenoh.io/docs/manual/plugin-storage-manager/

[94] RabbitMQ, “RabbitMQ 4.0 Documentation.” 2024. [Online]. Available: https://www.rabbitmq.com/docs

[95] RabbitMQ, “Reliability Guide.” 2024. [Online]. Available: https://www.rabbitmq.com/docs/reliability

[96] RabbitMQ, “Classic Queues Operating in ‘Lazy’ Queue Mode (A Lazy Queue).” 2024. [Online]. Available: https://www.rabbitmq.com/docs/lazy-queues

[97] J. Karimov, T. Rabl, A. Katsifodimos, R. Samarev, H. Heiskanen, and V. Markl, “Benchmarking Distributed Stream Data Processing Systems,” in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, IEEE, Apr. 2018, pp. 1507–1518. doi: 10.1109/ICDE.2018.00169.

[98] F. A. Ahmed, J. Ye, and J. Arthur, “Evaluating Streaming Frameworks for Large-Scale Event Streaming.” 2019. [Online]. Available: https://blog.developer.adobe.com/evaluating-streaming-frameworks-for-large-scale-event-streaming-7209938373c8

[99] 6G-TWIN Project, “Deliverable D2.1: Data governance, privacy and harmonization,” 6G-TWIN Project, 2024. [Online]. Available: https://6g-twin.eu/resources/#deliverables

[100] A. Sayın, B. Mohammedreza, R. Fuladi, and U. Gülen, “Secure and privacy preserved data management system for 6G Network Digital Twin,” in *Computing Conference*, 2025.

[101] O-RAN Alliance, “69 New or Updated O-RAN Technical Documents Released since November 2023.” 2024. [Online]. Available: https://www.o-ran.org/blog/69-new-or-updated-o-ran-technical-documents-released-since-november-2023

[102] A. Kak, V.-Q. Pham, H.-T. Thieu, and N. Choi, “RANSight: Programmable Telemetry for Next-Generation Open Radio Access Networks,” in *GLOBECOM 2023-2023 IEEE Global Communications Conference*, IEEE, 2023, pp. 5391–5396.

[103] M. Polese, L. Bonati, S. D’oro, S. Basagni, and T. Melodia, “Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges,” *IEEE Commun. Surv. Tutor.*, vol. 25, no. 2, pp. 1376–1411, 2023.

[104] F. Z. Morais, C. A. da Costa, A. M. Alberti, C. B. Both, and R. da Rosa Righi, “When SDN meets C-RAN: A survey exploring multi-point coordination, interference, and performance,” *J. Netw. Comput. Appl.*, vol. 162, p. 102655, 2020.

[105] H. M. Do, M. A. Gregory, and S. Li, “SDN-based wireless mobile backhaul architecture: Review and challenges,” *J. Netw. Comput. Appl.*, vol. 189, p. 103138, Sep. 2021, doi: 10.1016/j.jnca.2021.103138.

[106] M. R. Raza, M. Fiorani, A. Rostami, P. Öhlen, L. Wosinska, and P. Monti, “Dynamic Slicing Approach for Multi-Tenant 5G Transport Networks [Invited],” *J. Opt. Commun. Netw.*, vol. 10, no. 1, p. A77, Jan. 2018, doi: 10.1364/JOCN.10.000A77.

[107] S. Matoussi, I. Fajjari, N. Aitsaadi, and R. Langar, “User-Centric Slice Allocation Scheme in 5G Networks and Beyond,” *IEEE Trans. Netw. Serv. Manag.*, vol. 20, no. 4, pp. 4268–4282, Dec. 2023, doi: 10.1109/TNSM.2023.3284206.

[108] A. Rostami *et al.*, “Orchestration of RAN and Transport Networks for 5G: An SDN Approach,” *IEEE Commun. Mag.*, vol. 55, no. 4, pp. 64–70, Apr. 2017, doi: 10.1109/MCOM.2017.1600119.

[109] A. Troch *et al.*, “Optimizing Radio Resource Allocation in 5G using Transport Network-Aware Intelligent RAN,” in *2024 IEEE Latin-American Conference on Communications*

(LATINCOM), IEEE, Nov. 2024, pp. 1–6. doi: 10.1109/LATINCOM62985.2024.10770646.

[110] O.-RAN Alliance, “O-RAN Y1 interface: Type Definitions,” O-RAN Alliance, Technical Specification, Oct. 2024. [Online]. Available: https://specifications.o-ran.org/specifications

[111] O.-RAN Alliance, “O-RAN Non-RT RIC: Functional Architecture,” O-RAN Alliance, Technical Specification, Jun. 2021. [Online]. Available: https://specifications.o-ran.org/specifications

[112] O.-RAN Alliance, “O-RAN Use Cases Detailed Specification,” O-RAN Alliance, Technical Specification, Oct. 2024. [Online]. Available: https://specifications.o-ran.org/specifications

[113] O.-RAN Alliance, “O-RAN Decoupled SMO Architecture,” O-RAN Alliance, Technical Specification, Oct. 2024. [Online]. Available: https://specifications.o-ran.org/specifications

[114] O.-RAN Alliance, “O-RAN A1 interface: General Aspects and Principles,” O-RAN Alliance, Technical Specification, Oct. 2024. [Online]. Available: https://specifications.o-ran.org/specifications

[115] J. F. Nunes Pinheiro, C. M e Silva, and J. M. Marquez-Barja, “5GOpen@ TheBeacon: the 5G testbed,” in *Networks and Communications (EuCNC), European Conference on*, 2023, pp. 1–2.

[116] O.-RAN Alliance, “O-RAN Y1 interface: General Aspects and Principles,” O-RAN Alliance, Technical Specification, Oct. 2024. [Online]. Available: https://specifications.o-ran.org/specifications

[117] S. Si-Mohammed, A. Bardou, T. Begin, I. G. Lassous, and P. Vicat-Blanc, “NS+NDT: Smart integration of Network Simulation in Network Digital Twin, application to IoT networks,” *Future Gener. Comput. Syst.*, vol. 157, pp. 124–144, Aug. 2024, doi: 10.1016/j.future.2024.03.038.

[118] A. Diaz, P. Merino, and A. Salmeron, “Obtaining Models for Realistic Mobile Network Simulations using Real Traces,” *IEEE Commun. Lett.*, vol. 15, no. 7, pp. 782–784, Jul. 2011, doi: 10.1109/LCOMM.2011.060111.102514.

[119] P. Owezarski and N. Larrieu, “A trace based method for realistic simulation,” in *2004 IEEE International Conference on Communications (IEEE Cat. No.04CH37577)*, IEEE, 2004, pp. 2236-2239 Vol.4. doi: 10.1109/ICC.2004.1312915.

[120] E. Harinda, H. Larijani, and R. M. Gibson, “Trace-Driven Simulation for LoRaWan868 MHz Propagation in an Urban Scenario,” in *2019 UK/ China Emerging Technologies (UCET)*, IEEE, Aug. 2019, pp. 1–5. doi: 10.1109/UCET.2019.8881854.

[121] W. Zheng, A. Ali, N. Gonzalez-Prelcic, R. W. Heath, A. Klautau, and E. M. Pari, “5G V2X communication at millimeter wave: rate maps and use cases,” in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, IEEE, May 2020, pp. 1–5. doi: 10.1109/VTC2020-Spring48590.2020.9128612.

[122] M. Franke and C. Sommer, “Toward Space-Air-Ground Integrated Network Simulation with 4D Topologies,” in *2024 19th Wireless On-Demand Network Systems and Services Conference (WONS)*, IEEE, Jan. 2024, pp. 61–68. doi: 10.23919/WONS60642.2024.10449583.

[123] H. Schippers, S. Böcker, and C. Wietfeld, “Data-Driven Digital Mobile Network Twin Enabling Mission-Critical Vehicular Applications,” in *2023 IEEE 97th Vehicular*

Technology Conference (VTC2023-Spring), IEEE, Jun. 2023, pp. 1–7. doi: 10.1109/VTC2023-Spring57618.2023.10200830.

[124] C. Celes, A. Boukerche, and A. A. F. Loureiro, “Mobility Trace Analysis for Intelligent Vehicular Networks,” *ACM Comput. Surv.*, vol. 54, no. 3, pp. 1–38, Apr. 2022, doi: 10.1145/3446679.

[125] N. Goebel, R. Bialon, M. Mauve, and K. Graffi, “Coupled simulation of mobile cellular networks, road traffic and V2X applications using traces,” in *2016 IEEE International Conference on Communications (ICC)*, IEEE, May 2016, pp. 1–7. doi: 10.1109/ICC.2016.7511126.

[126] M. Franke and F. Klingler, “Poster Abstract: HiL meets Commodity Hardware – SimbaR for coupling IEEE 802.11 Radio Channels,” in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, IEEE, May 2021, pp. 1–2. doi: 10.1109/INFOCOMWKSHPS51825.2021.9484559.

[127] 6G-TWIN Project, “Deliverable D2.2: Basic models (initial),” 6G-TWIN Project, 2024. [Online]. Available: https://6g-twin.eu/resources/#deliverables

[128] E. G. NFV-IFA 006, “Network Functions Virtualisation (NFV) Release 5; Management and Orchestration; Vi-Vnfm reference point - Interface and Information Model Specification V5.2.1,” ETSI, Dec. 2023.

[129] E. G. NFV-IFA 036, “Network Functions Virtualisation (NFV) Release 5; Management and Orchestration; Requirements for the management of virtualized resources V.5.1.1,” ETSI, Jun. 2024.

[130] E. G. NFV-IFA 010, “Network Functions Virtualisation (NFV) Release 5; Management and Orchestration; Functional requirements specification V.5.2.1,” ETSI, Oct. 2024.

[131] E. G. NFV-IFA 040, “Network Functions Virtualisation (NFV) Release 5; Management and Orchestration; Requirements for service interfaces and object model for OS container management and orchestration specification V5.1.1,” ETSI, Jun. 2024.

[132] E. G. NFV-IFA 013, “Network Functions Virtualisation (NFV) Release 5; Management and Orchestration; Os-Ma-nfvo reference point - Interface and Information Model Specification V.5.1.1,” ETSI, Jun. 2024.

[133] E. G. NFV-IFA 008, “Network Functions Virtualisation (NFV) Release 5; Management and Orchestration; Ve-Vnfm reference point - Interface and Information Model Specification V5.1.1,” ETSI, Jun. 2024.

[134] E. G. NFV-IFA 007, “Network Functions Virtualisation (NFV) Release 5; Management and Orchestration; Or-Vnfm reference point - Interface and Information Model Specification V5.1.1,” ETSI, Jun. 2024.

[135] E. G. NFV-IFA 005, “Network Functions Virtualisation (NFV) Release 5; Management and Orchestration; Or-Vi reference point - Interface and Information Model Specification V5.1.1,” ETSI, Jun. 2024.

[136] M. Mezhoudi and Y. Hu, “Economical analysis of NG-optical backbone transport network,” in *2011 20th Annual Wireless and Optical Communications Conference (WOCC)*, IEEE, Apr. 2011, pp. 1–6. doi: 10.1109/WOCC.2011.5872304.

[137] A. Triki, A. Gravey, and P. Gravey, “CAPEX and OPEX saving in SDN-compliant sub-wavelength switching solution,” in *2015 International Conference on Photonics in Switching (PS)*, IEEE, Sep. 2015, pp. 262–264. doi: 10.1109/PS.2015.7329020.

[138] R. Alvizu *et al.*, “Comprehensive survey on T-SDN: Software-defined networking for transport networks,” *IEEE Commun. Surv. Tutor.*, vol. 19, no. 4, pp. 2232–2283, 2017.

[139] J. Marshall, R. Gallagher, and R. Cole, “Managing network flexibility in a SONET/SDH environment,” in *1989 IEEE Global Telecommunications Conference and Exhibition’Communications Technology for the 1990s and Beyond’*, IEEE, 1989, pp. 1495–1499.

[140] X. Gong, X. Chang, and Y. Wu, “The Research and Implementation of PDH Network Management Software for Telecom Access Network,” in *2012 International Conference on Industrial Control and Electronics Engineering*, IEEE, Aug. 2012, pp. 131–135. doi: 10.1109/ICICEE.2012.43.

[141] X. Wang, “Fusion networking of PTN/MSTP from different equipment providers by employing QinQ technique,” in *2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, IEEE, 2016, pp. 241–244.

[142] Z. Yong, Y. Jian-Li, and W. Cui-Xian, “Design and research on PTN technology in local transport network,” in *2011 IEEE 3rd International Conference on Communication Software and Networks*, IEEE, May 2011, pp. 479–482. doi: 10.1109/ICCSN.2011.6013876.

[143] M. Herzog, M. Maier, and M. Reisslein, “Metropolitan area packet-switched WDM networks: A survey on ring systems,” *IEEE Commun. Surv. Tutor.*, vol. 6, no. 2, pp. 2–20, 2004, doi: 10.1109/COMST.2004.5342236.

[144] R. Vaez-Ghaemi, “Optical Transport Networks (OTN) Test,” White Paper, 2008. [Online]. Available: https://www.viavisolutions.com/sites/default/files/support/OTN%20Testing%20White%20Paper.pdf

[145] C. Xie and B. Zhang, “Scaling Optical Interconnects for Hyperscale Data Center Networks,” *Proc. IEEE*, vol. 110, no. 11, pp. 1699–1713, Nov. 2022, doi: 10.1109/JPROC.2022.3178977.

[146] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, “Software-Defined Networking: A Comprehensive Survey,” *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015, doi: 10.1109/JPROC.2014.2371999.

[147] ONF, “SDN Architecture, ONF TR - 521,” Open Networking Foundation, 2016. [Online]. Available: https://opennetworking.org/wp-content/uploads/2014/10/TR-521_SDN_Architecture_issue_1.1.pdf

[148] ONF, “SDN Architecture for Transport Networks, ONF TR - 522,” Open Networking Foundation, Mar. 2016. [Online]. Available: https://opennetworking.org/wp-content/uploads/2014/10/SDN_Architecture_for_Transport_Networks_TR522.pdf

[149] N. Davis and K. Lam, “ONF Core Model; Introduction to models, guidelines and tooling from ONF Open Information Model & Tooling project.” 2018. [Online]. Available: https://opennetworking.org/wp-content/uploads/2018/12/ONF-Core-Model.pdf

[150] SEASON Project, “Deliverable D4.1: First Design and Implementation of Control Plane Infrastructure,” 2024. [Online]. Available: https://www.season-project.eu/wp-content/uploads/Deliverables/SEASON-D4.1-v1.0.pdf

[151] J. Farmer, B. Lane, K. Bourg, and W. Wang, “Chapter 16 - Performance Management,” in *FTTx Networks*, J. Farmer, B. Lane, K. Bourg, and W. Wang, Eds., Boston: Morgan Kaufmann, 2017, pp. 377–401. doi: https://doi.org/10.1016/B978-0-12-420137-8.00016-0.

Co-funded by the European Union