UNIVERSIDAD POLITÉCNICA DE MADRID
Escuela Técnica Superior de Ingenieros de Telecomunicación



Contribution to Network Operation and Management Models Using Digital Twins

D O C T O R A L T H E S I S

Submitted for the degree of Doctor by:

Daniel González Sánchez
M.Sc. in Telematic Services and Network Engineering

Madrid, 2025



UNIVERSIDAD POLITÉCNICA DE MADRID
Escuela Técnica Superior de Ingenieros de Telecomunicación
Departamento de Ingeniería de Sistemas Telemáticos

Doctoral Degree in Telematic Systems Engineering

Contribution to Network Operation and Management Models Using Digital Twins

D O C T O R A L T H E S I S

Submitted for the degree of Doctor by:

Daniel González Sánchez
M.Sc. in Telematic Services and Network Engineering

Under the supervision of:
Dr. Luis Bellido Triana

Madrid, 2025

Title: Contribution to Network Operation and Management Models Using Digital Twins

Author: Daniel González Sánchez

Doctoral Programme: Ph.D. in Telematic Systems Engineering

Thesis Supervision: Dr. Luis Bellido Triana, Associate Professor, Universidad Politécnica de Madrid (Supervisor)

External Reviewers:

Thesis Defense Committee:

Thesis Defense Date:

Agradecimientos

Quisiera agradecerles a mis padres por toda la confianza y dedicación que han depositado en mí durante la etapa académica y por todo el apoyo y cariño incondicional que me han profesado a lo largo de mi vida. No puedo evitar acordarme primero de mi madre, una mujer maravillosa, amable e inteligente que ha sido, es y será siempre mi inspiración como persona. Destacar también la fortaleza y atención de mi padre, además de su enorme labor de guía, referencia y apoyo en mi día a día. También hago aquí mención especial a mis hermanas, Luzía y Marina, que son un pilar esencial en mi vida y que consiguen sacar lo mejor de mí en todo momento. No lo digo con toda la frecuencia que debería pero lo dejo aquí por escrito: "¡Os quiero mucho!".

Cómo no, agradecer a todos los integrantes de mi grupo de investigación GIROS que me abrieron las puertas hace cinco años y me acogieron como a uno más de la familia. Todos ellos, antes o después, han aportado su granito de arena durante este periodo de investigación, contribuyendo a que pueda sacar esta tesis doctoral adelante. Quiero destacar a David y a Luis que fueron los primeros en acogerme y han ejercido de "padres profesionales" a lo largo de toda esta etapa, echándome un cable con todo el agrado del mundo ante cualquier tipo de duda o problema al que me haya tenido que enfrentar. Y en especial, quiero expresar toda mi gratitud a Luis, mi director de tesis, por haberme sabido guiar en todo momento, por el apoyo y grado de formación recibidos y por todos los consejos que me ha ofrecido durante el desarrollo de esta tesis doctoral.

También quiero acordarme de los compañeros de Telefónica Innovación Digital, con quienes ha sido un verdadero placer colaborar durante estos años en múltiples proyectos de investigación. Su apoyo ha sido fundamental para orientar la temática de esta tesis y consolidar los avances obtenidos. En esta misma línea, deseo expresar mi agradecimiento a los participantes de los distintos proyectos europeos en los que he tenido la oportunidad de colaborar (5Growth, 5G-CLARITY, PALANTIR, aerOS y ACROSS), cuyas valiosas discusiones técnicas han ayudado a perfilar y enriquecer las contribuciones de esta tesis.

Por último, agradecer de corazón a demás familiares y amigos que con sus ánimos y muestras de ayuda y cariño han logrado que siga adelante en los buenos y no tan buenos momentos.

¡Muchísimas gracias a todos! Esta tesis también es vuestra.

Madrid, 09/05/2025
~ Dani

Abstract

The growth of telecommunications networks, driven by increasingly complex next-generation technologies and new resource-intensive services, has made network monitoring and maintenance more critical than ever for network operators. In this sense, the Digital Twin paradigm is emerging as a promising technology to mirror a network's whole operational and management lifecycle to optimize its performance without compromising the real-world network.

Digital Twins are widely used in industry and research to accurately model the behavior of complex and dynamic systems. The application of the term Digital Twin to the field of telecommunication networks, also known as Network Digital Twin (NDT), has also gradually attracted attention to modeling the behavior of real networks through a digital representation of them that is continuously updated with the real network performance, configuration, and status data. As a result, an NDT system enables network operators to design maintenance and analytics solutions, model optimization systems, troubleshoot issues, or plan network configuration updates, thus promoting efficient network operations and management.

In this regard, an NDT can be envisioned as an indispensable part of global network management systems in the future, covering the entire lifecycle management of real networks without compromising their daily operations, supporting the application of innovative technologies, and increasing the network's level of automation and intelligence. However, several challenges have been identified in the design and implementation of NDT systems to efficiently control and manage modern telecommunication networks, such as the dynamic modeling of real network behavior within the NDT system or the design of mechanisms for data collection, storage, and management, taking into account that data is the cornerstone of NDT construction.

In line with this, the overall objective of this Doctoral Thesis is to contribute to the understanding and development of Network Digital Twins applied to the scope of autonomous network management, facilitating interoperability with external systems requiring information from network infrastructures and enabling on-demand management and operation tasks for network operators. In this sense, this thesis proposes an NDT solution supported by a new data integration methodology and a prototypical architecture to translate information from the network management domain into a formal, data-driven, and interoperable model. This NDT solution is built on the principles of model-driven network management and the use of the NGSI-LD standard specification defined in the ETSI Industry Specification Group on Context Information Management (ETSI ISG CIM), which facilitates the modeling of Digital Twins. The NDT proposal enables both the exposure and handling of the capabilities of real networks in terms of configuration and operational status, thus facilitating network operation and monitoring. By applying this NDT approach, the resulting work provides benefits in terms of network control and management to facilitate the integration of network automation systems.

Resumen

La monitorización y el mantenimiento de la red son tareas cada vez más cruciales para los operadores, impulsadas por los avances en las redes de telecomunicaciones, fruto de tecnologías de próxima generación cada vez más complejas y de nuevos servicios con alta demanda de recursos. En este sentido, el paradigma de los Gemelos Digitales o *Digital Twin* emerge como una tecnología prometedora para reflejar todo el ciclo de vida operativo y de gestión de una red con el fin de optimizar su rendimiento sin comprometer a la propia red.

Los Gemelos Digitales están siendo ampliamente adoptados en la industria y la investigación para modelar con precisión el comportamiento de sistemas complejos y dinámicos. La aplicación del término Gemelo Digital en el campo de las redes de telecomunicaciones, conocido como Gemelo Digital de Red o *Network Digital Twin* (NDT), también ha atraído gradualmente la atención de la industria y la investigación para modelar el comportamiento de las redes a través de una representación digital de las mismas que se actualiza continuamente con los datos reales de rendimiento, configuración y estado de la red. Como resultado, un sistema NDT permite a los operadores de red diseñar soluciones de mantenimiento y análisis, modelar sistemas de optimización, solucionar problemas o planificar actualizaciones en la configuración de la red, fomentando así la eficiencia en las operaciones y la gestión de la red.

En este contexto, un NDT puede considerarse una pieza clave de los futuros sistemas globales de gestión de redes, ya que abarca todo el ciclo de vida de gestión de las redes sin afectar a su funcionamiento diario, mejoran el nivel de automatización e inteligencia de las redes y facilitan la implementación de tecnologías innovadoras. Sin embargo, se han identificado varios desafíos en el diseño e implementación de sistemas NDT para el control y la gestión eficiente de las redes de telecomunicaciones modernas. Entre ellos se encuentran el modelado dinámico del comportamiento de las redes dentro del sistema NDT y el diseño de mecanismos para la recopilación, almacenamiento y gestión de datos de la red, considerando que los datos serán un elemento fundamental en la construcción de un NDT.

En línea con lo anterior, el objetivo general que persigue esta tesis doctoral es contribuir a la comprensión y el desarrollo de Gemelos Digitales de Red aplicados a la gestión autónoma de redes, facilitando la interoperabilidad con sistemas externos que requieran información de infraestructuras de red y permitiendo la gestión y operación de la red bajo demanda por parte de los operadores de red. En este sentido, se propone una solución de NDT basada en una nueva metodología de integración de datos y un prototipo de arquitectura que traduce la información del dominio de gestión de redes a un modelo formal, basado en datos e interoperable. Esta solución de NDT se fundamenta en los principios de la gestión de redes basada en modelos y en el uso de la especificación estándar NGSI-LD definida por ETSI en el grupo de especificación ETSI ISG CIM (*ETSI Industry Specification Group on Context Information Management*) que facilita el modelado de Gemelos Digitales. Esta propuesta de NDT permite tanto la exposición como la gestión de las capacidades que ofrecen las redes reales en términos de configuración y de estado operativo, facilitando así su operación y monitorización. Al aplicar este enfoque de NDT, el trabajo resultante aporta beneficios en el control y la gestión de redes, facilitando la integración de sistemas de automatización de redes.

Table of Contents

Agradecimientos . . . iii
Abstract . . . v
Resumen . . . vii
List of Figures . . . xiii
List of Tables . . . xix
Abbreviations and Acronyms . . . xxi

1 Introduction **1**
1.1 Contextualization and Motivation . . . 2
1.2 Objectives . . . 3
1.3 Methodology . . . 4
1.4 Activities Developed within the Context of this Doctoral Thesis . . . 5
1.4.1 Participation in competitive R&D&I projects . . . 5
1.4.2 Publications related to the research work of the Doctoral Thesis . . . 8
1.4.3 Other international publications . . . 11
1.4.4 Technical documents . . . 11
1.4.5 Teaching activities . . . 12
1.5 Document Structure . . . 14

2 State of the Art **17**
2.1 Digital Twins . . . 18
2.1.1 Distinction between Digital Twins, simulations, and emulations . . . 20
2.1.2 Benefits and enablers of Digital Twins . . . 21
2.1.3 Challenges of Digital Twins . . . 22
2.1.4 Data integration techniques, approaches, and architectures . . . 23
2.1.5 Data modeling in Digital Twins . . . 31
2.2 Network Digital Twins . . . 39
2.2.1 NDT standardization activities . . . 40
2.2.2 NDT benefits and challenges . . . 43
2.2.3 NDT components and architecture . . . 44
2.2.4 NDT and Digital Map process . . . 47
2.2.5 Enabling technologies for building an NDT . . . 49
2.2.6 NDT for data model-driven network management . . . 50
2.3 Emerging Trends in Network Operations and Management . . . 52
2.3.1 Model-driven network management and telemetry . . . 54
2.3.2 Intent-Based Networking for network automation . . . 63
2.3.3 Advancing network automation with Artificial Intelligence and Knowledge Graphs . . . 66

3 Modeling an NDT Solution for Network Operations and Management 69
3.1 Designing an NGSI-LD Compliant NDT Solution . . . 70
3.1.1 Alignment of NDT elements and architecture with the NGSI-LD standard 70
3.1.2 NGSI-LD Digital Twins for network operations and management . . . 72
3.2 A New Methodology for Modeling an NDT based on the NGSI-LD Standard 74
3.2.1 Data integration using the NGSI-LD standard . . . 75
3.2.2 YANG to NGSI-LD translation conventions . . . 79
3.2.3 Descriptive NDT architecture . . . 82
3.2.4 Network automation using the Descriptive NDT and the NGSI-LD standard . . . 85
3.3 Overview of the NDT Proposal . . . 96

4 Descriptive NDT System for Network Operations and Management 97
4.1 A Programmatic Digital Mapping System for Network Operations and Management . . . 98
4.1.1 Descriptive NDT Generator Plane . . . 98
4.1.2 Data Integration Plane Pipeline . . . 103
4.1.3 Query processing and rewriting for model-driven network management purposes . . . 106
4.2 Case Study: Descriptive NDT for Network Operations and Management . . . 111
4.2.1 IETF-based Descriptive NDT model . . . 111
4.2.2 OpenConfig-based Descriptive NDT model . . . 115
4.2.3 Descriptive NDT model for NetFlow v9 protocol . . . 119
4.2.4 Application of the NGSI-LD Descriptive NDT for network operations and management purposes . . . 123
4.3 Overview of the Descriptive NDT System . . . 134

5 Experimental Results and Evaluation 135
5.1 Experimental Testbed . . . 136
5.2 Performance of the Descriptive NDT Prototype for Network Data Acquisition 139
5.2.1 Interface telemetry performance experiments . . . 140
5.2.2 NetFlow monitoring performance experiments . . . 144
5.2.3 Discussion on experimental results . . . 146
5.3 Performance of the Descriptive NDT Prototype for Completing Network Operations . . . 148
5.3.1 Latency performance results for NETCONF operations . . . 151
5.3.2 Latency performance results for gNMI operations . . . 154
5.3.3 Latency performance results for NetFlow monitoring . . . 156
5.3.4 Discussion on experimental results . . . 158

6 Conclusions and Future Work 161
6.1 Conclusions . . . 162
6.2 Future Work . . . 165

References 169

Annexes **185**
A.1 Descriptive NDT Architecture and Prototype Design 186
A.2 Data Formats within the NDT Data Integration Pipeline 188
A.2.1 Data formats for NETCONF case 188
A.2.2 Data formats for gNMI case 197
A.3 Handling YANG Configuration and Operational Data in Descriptive NDT solution 202
A.3.1 Handling of NMDA compliant IETF models 202
A.3.2 Handling of non-NMDA compliant IETF models 203
A.3.3 Handling of OpenConfig models 204
A.4 Examples of Query Processing Data Integration Workflows 206
A.4.1 Querying operational status data from a network device interface . . 206
A.4.2 Subscription to telemetry statistics from a network device interface . 206
A.4.3 Querying configuration data from a network device 209
A.4.4 Updating the configuration of a network device interface 211
A.5 Open-Source Software Development Activity 216
A.5.1 Prototype implementation of Descriptive NDT 216
A.5.2 Semantic Data Aggregator and CANDIL Data Fabric 216
A.5.3 OpenAPI Specification for NGSI-LD and artifacts 217
A.5.4 Containerlab testbeds for analyzing network telemetry and management mechanisms based on YANG 217

List of Figures

2.1 Digital Twin general overview. 20
2.2 General architecture of a data integration system (based on [40]). 25
2.3 General architecture of a data materialization system (based on [40]). 27
2.4 General architecture of a data virtualization system (based on [40]). 28
2.5 General architecture of a hybrid data integration system (based on [40]). . . 29
2.6 UML diagram of NGSI-LD meta-model (based on [56]). 35
2.7 Distributed architecture showing interactions between components using the NGSI-LD API. The Context Registry is depicted in red color as it can be optionally deployed outside the Context Broker [84]. 36
2.8 High-level architecture for Digital Twin systems according to the NGSI-LD standard [11]. 39
2.9 Network Digital Twin core elements (based on [7]). 44
2.10 Reference Architecture of Network Digital Twin (based on [7]). 46
2.11 Application of NDT architecture for network operations and management. . 47
2.12 General insight on Digital Map process. 48
2.13 Closed-loop paradigm applied to network operations and management. . . . 54
2.14 The data model–driven management components (based on [61]). 56
2.15 The server architecture in a model-driven network management system (based on RFC 6244 [131]). 58
2.16 Impact analysis for *ietf-interfaces* YANG data model. 60
2.17 Intent-Based Networking on a closed-loop system for network management purposes. 65

3.1 NDT core elements compliant with the NGSI-LD standard. 71
3.2 Prototypical architecture of NDT compliant with the NGSI-LD standard. . . 72
3.3 General insight on NGSI-LD compliant Digital Twins for network management domain. 74
3.4 General insight on NGSI-LD compliant Descriptive Twin for network management domain. 75
3.5 Data materialization with the NGSI-LD standard. 76
3.6 Data virtualization with the NGSI-LD standard. 77
3.7 General architecture of a data materialization system using NGSI-LD for MDT data sources. 78
3.8 General architecture of a data virtualization system using NGSI-LD for MDT data sources. 80
3.9 Descriptive NDT architecture compliant with the NGSI-LD standard and the NDT core elements [127]. 83

3.10 UML diagram of NGSI-LD model for network operations and management based on data materialization approach. 87
3.11 Data materialization with GET RPC operation. 89
3.12 Data materialization with SUBSCRIBE RPC operation. 90
3.13 Data materialization with SET RPC operation. 91
3.14 UML diagram of NGSI-LD model for network operations and management based on data virtualization approach. 92
3.15 Data virtualization with GET RPC operation by interacting with the NGSI-LD Context Broker. 93
3.16 Data virtualization with GET RPC operation by interacting with the NGSI-LD Context Source. 93
3.17 Data virtualization with initial step for triggering SUBSCRIBE RPC operations. 94
3.18 Data virtualization with SUBSCRIBE RPC operation by interacting with the NGSI-LD Context Broker. 94
3.19 Data virtualization with SUBSCRIBE RPC operation by interacting directly with the NGSI-LD Context Source. 95
3.20 Data virtualization with SET RPC operation. 96

4.1 A prototypical implementation of the Descriptive NDT architecture. It is based on a programmatic digital mapping system tailored to MDT network assets. 98
4.2 Descriptive NDT Generator Plane functionality. 99
4.3 OAS Schema for *InterfaceStatistics* NGSI-LD entity type. 100
4.4 OAS Schema for *inOctets* NGSI-LD property. 101
4.5 NGSI-LD context associated with *InterfaceConfig* entity type. 102
4.6 NGSI-LD metadata associated with *InterfaceConfig* entity type. 102
4.7 Details of a *YANGIdentity* entity associated with the case of network interface type as specified by the *ietf-interfaces* and *iana-if-type* standardized IETF YANG models. 103
4.8 NGSI-LD context associated with *YANGIdentity* entity type. 104
4.9 Data Integration Plane Pipeline (Phase 1). Network Operator executing RPC commands. 104
4.10 The Data Integration Plane Pipeline (Phase 2). Network Operator collecting configuration, operational state, and telemetry data. 105
4.11 Query processing and rewriting general workflow for performing GET RPC model-driven network management operations. 107
4.12 Query processing and rewriting general workflow for performing SUBSCRIBE RPC model-driven network management operations. 108
4.13 Query processing and rewriting general workflow for performing GET CONFIG RPC model-driven network management operations. 109
4.14 Query processing and rewriting general workflow for performing SET RPC model-driven network management operations. 110
4.15 UML diagram of NGSI-LD domain model for managing network topology and interfaces based on IETF RFCs 8345 and 8343 (NMDA compliant). 111
4.16 UML diagram of NGSI-LD domain model for network interface IP management based on IETF RFC 8344 (NMDA compliant). 113

4.17 UML diagram of NGSI-LD domain model for managing network topology and interfaces based on IETF RFCs 8345 and 7223 (non-NMDA compliant). . . . 114
4.18 UML diagram of NGSI-LD domain model for network interface IP management based on IETF RFC 7277 (non-NMDA compliant). 115
4.19 UML diagram of NGSI-LD domain model for managing network interfaces according to OpenConfig consortium. 117
4.20 UML diagram of NGSI-LD domain model for network interface IP management according to OpenConfig consortium. 118
4.21 YANG module for NetFlow version 9. The YANG tree representation is displayed in two columns for better readability. 120
4.22 Augmentation of YANG module for NetFlow version 9 with aggregated KPIs. 121
4.23 NGSI-LD domain model for network flows according to NetFlow v9 protocol. 122
4.24 Request body for a NGSI-LD update entity operation to configure network interface using a data materialization approach. 125
4.25 Request body for a NGSI-LD update entity operation to configure network interface using a data virtualization approach. 126
4.26 Request body for a NGSI-LD update entity operation to configure IPv4 addressing of a network interface using a data materialization approach. . . . 126
4.27 Request body for a NGSI-LD create entity operation to configure IPv4 addressing of a network interface using a data virtualization approach. . . . 127
4.28 NGSI-LD subscription to interface traffic telemetry statistics. 128
4.29 NGSI-LD subscription to interface telemetry statistics. 129
4.30 NGSI-LD subscription to the operational status of network interfaces. 131
4.31 NGSI-LD subscription to receive notifications of new Level 2 neighbors discovered on a network interface. 132
4.32 NGSI-LD subscription to network flows. 133

5.1 Experimental testbed for validating the Descriptive NDT capabilities and performing the performance analysis. 139
5.2 Measuring the latency performance for network data acquisition. 140
5.3 YANG to NGSI-LD translation latency. 141
5.4 YANG to NGSI-LD translation and instantiation latency. 142
5.5 YANG to NGSI-LD translation and instantiation latency for NECONF and gNMI comparing performance of the Orion-LD and Scorpio NGSI-LD Context Brokers. 143
5.6 Throughput experiments for NETCONF notifications. 143
5.7 Throughput experiments for gNMI notifications. 144
5.8 YANG to NGSI-LD translation and instantiation latency performance for NetFlow v9. 145
5.9 Throughput experiments for NetFlow notifications. 146
5.10 Measuring the latency for network request operations when interacting with the NGSI-LD Context Broker. 149
5.11 Measuring the latency for network request operations when interacting with the NGSI-LD Context Source. 149

5.12 Measuring the latency of network data reception when interacting with the NGSI-LD Context Source. . . . 150
5.13 Measuring the latency of network data reception when interacting with the NGSI-LD Context Broker. . . . 150
5.14 Mean execution time for completing NETCONF RPC query operations. . . . 152
5.15 Mean execution time for replies generated from NETCONF RPC query operations. 153
5.16 Mean evaluation time for notifications generated from NETCONF RPC subscribe operations. . . . 153
5.17 Mean execution time for completing gNMI RPC query operations. . . . 154
5.18 Mean execution time for replies generated from gNMI RPC query operations. 155
5.19 Mean evaluation time for notifications generated from gNMI RPC subscribe operations. . . . 156
5.20 Measuring the latency for the reception of network flow data when interacting with the NGSI-LD Context Broker. . . . 157
5.21 Measuring the latency for the reception of network flow data when interacting with the NGSI-LD Context Source. . . . 157
5.22 Mean evaluation time for integrating notifications about NetFlow monitoring data. . . . 158

A.1 Descriptive NDT architecture. . . . 186
A.2 Descriptive NDT prototype design. . . . 187
A.3 Operational data about network interface retrieved by NETCONF protocol in XML encoding format (non-NMDA architecture). . . . 188
A.4 Configuration data about network interface retrieved by NETCONF protocol in XML encoding format (non-NMDA architecture). . . . 189
A.5 Operational data about network interface retrieved by NETCONF protocol in NGSI-LD data format (non-NMDA architecture). . . . 190
A.6 Configuration data about network interface retrieved by NETCONF protocol in NGSI-LD data format (non-NMDA architecture). . . . 191
A.7 Configuration data about IPv4 addressing of network interface retrieved by NETCONF protocol in NGSI-LD data format (non-NMDA architecture). . . . 191
A.8 Operational data about network interface retrieved by NETCONF protocol in XML encoding format (NMDA architecture). . . . 192
A.9 Configuration data about network interface retrieved by NETCONF protocol in XML encoding format (NMDA architecture). . . . 193
A.10 Operational and configuration data about network interface retrieved by NETCONF protocol in NGSI-LD data format (NMDA architecture). . . . 194
A.11 Telemetry notification about network interface statistics counters retrieved by NETCONF protocol in XML encoding format. . . . 195
A.12 Telemetry notification about network interface statistics counters retrieved by NETCONF protocol in NGSI-LD data format. . . . 196
A.13 Operational data about network interface retrieved by gNMI protocol in JSON encoding format. . . . 197
A.14 Configuration data about network interface retrieved by gNMI protocol in JSON encoding format. . . . 197

A.15 Operational data about network interface retrieved by gNMI protocol in NGSI-LD data format. 198
A.16 Configuration data about network interface retrieved by gNMI protocol in NGSI-LD data format. 199
A.17 Telemetry notification about network interface statistics counters retrieved by gNMI protocol in JSON encoding format. 200
A.18 Telemetry notification about network interface statistics counters retrieved by gNMI protocol in NGSI-LD data format. 201
A.19 Example of mapping NMDA-based IETF YANG data model to NGSI-LD. . 203
A.20 Example of mapping non-NMDA IETF YANG data model to NGSI-LD. . . 204
A.21 Example of mapping OpenConfig YANG data model to NGSI-LD. 205
A.22 Query processing and rewriting workflow for retrieving operational status data from a network interface using NETCONF for non-NMDA compliant network devices. 207
A.23 Query processing and rewriting workflow for retrieving operational status data from a network interface using NETCONF for NMDA compliant network devices. 207
A.24 Query processing and rewriting workflow for receiving notifications about telemetry statistics from a network interface using NETCONF for non-NMDA compliant network devices. 208
A.25 Query processing and rewriting workflow for receiving notifications about telemetry statistics from a network interface using NETCONF for NMDA compliant network devices. 209
A.26 Query processing and rewriting workflow for retrieving configuration data from a network interface using NETCONF for non-NMDA compliant network devices. 210
A.27 Query processing and rewriting workflow for retrieving configuration data from a network interface using NETCONF for NMDA compliant network devices. 211
A.28 Query processing and rewriting workflow for updating the configuration of a network interface using NETCONF for non-NMDA compliant network devices (data materialization). 212
A.29 Query processing and rewriting workflow for updating the configuration of a network interface using NETCONF for NMDA compliant network devices (data materialization). 213
A.30 Query processing and rewriting workflow for updating the configuration of a network interface using NETCONF for non-NMDA compliant network devices (data virtualization). 214
A.31 Query processing and rewriting workflow for updating the configuration of a network interface using NETCONF for NMDA compliant network devices (data virtualization). 215

List of Tables

2.1 Basic operations of the NGSI-LD API. 36
2.2 Comparison of NETCONF, RESTCONF, and gNMI. 61
2.3 RPC basic operations in NETCONF, gNMI, and RESTCONF. 61

3.1 High-level mapping from YANG to NGSI-LD. 80
3.2 Data type conversion from YANG to NGSI-LD. 82

5.1 The classification of NETCONF, gNMI, and NetFlow v9 protocol capabilities in router models from different manufacturers. 137

Abbreviations and Acronyms

3GPP 3rd Generation Partnership Project

AI Artificial Intelligence

API Application Programming Interface

ARP Address Resolution Protocol

CBOR Concise Binary Object Representation

CIM Context Information Management

CLI Command Line Interface

CoAP Constrained Application Protocol

DBMS Data Base Management System

DCMI Dublin Core Metadata Initiative

DIT Departamento de Ingeniería de Sistemas Telemáticos

DNS Domain Name System

DT Digital Twin

ETL Extract, Transform, and Load

ETSI European Telecommunications Standards Institute

ETSIT Escuela Técnica Superior de Ingenieros de Telecomunicación

EVPN Ethernet Virtual Private Network

gNMI gRPC Network Management Interface

gRPC Google Remote Procedure Call

GUI Graphical User Interface

HTTP Hypertext Transfer Protocol

IANA Internet Assigned Numbers Authority

IBN Intent-Based Networking

IEEE Institute of Electrical and Electronics Engineers

IETF Internet Engineering Task Force

IoT Internet of Things

IP Internet Protocol

IRTF Internet Research Task Force

ISG Industry Specification Group

ITU International Telecommunication Union

JSON JavaScript Object Notation

JSON-LD JSON Linked Data

KNE Kubernetes Network Emulation

LLM Large Language Model

LPG Labeled Property Graph

M2M Machine-to-Machine

MDT Model-Driven Telemetry

MTTD Mean Time To Detect

MTTR Mean Time To Repair

NDP Neighbor Discovery Protocol

NDT Network Digital Twin

NDTaaS Network Digital Twin as a Service

NFV Network Function Virtualization

NGSI-LD Next Generation Service Interfaces Linked Data

NMDA Network Management Datastore Architecture

NMS Network Management System

OAM Operations, Administration, and Maintenance

OAS OpenAPI Specfication

OWL Web Ontology Language

RAG Retrieval-Augmented Generation

RDF Resource Description Framework

RDFS RDF Schema

RFC Request For Comments

RPC Remote Procedure Call

SDN Software-Defined Networking

SDO Standards Development Organizations

SNMP Simple Network Management Protocol

TCP Transmission Control Protocol

TMF TeleManagement Forum

TSN Time-Sensitive Networking

UML Unified Modeling Language

UPM Universidad Politécnica de Madrid

URI Uniform Resource Identifier

URL Uniform Resource Locator

URN Uniform Resource Name

VRRP Virtual Router Redundancy Protocol

W3C World Wide Web Consortium

XML eXtensible Markup Language

XMPP eXtensible Messaging and Presence Protocol

YANG Yet Another Next Generation

ZSM Zero-touch network and Service Management

Chapter 1

Introduction

This chapter emphasizes the importance of the subject matter, validity, and timeliness of this Doctoral Thesis. The research challenge is presented, including the contextualization and motivation for its development, as well as the objective and purpose of the work. This chapter also describes the planning, methodology, and structure of the document. In addition, the research and academic activities developed in the context of the related dissertation are enumerated.

Table of Contents

1.1 Contextualization and Motivation 2
1.2 Objectives 3
1.3 Methodology 4
1.4 Activities Developed within the Context of this Doctoral Thesis 5
1.4.1 Participation in competitive R&D&I projects 5
1.4.2 Publications related to the research work of the Doctoral Thesis . 8
1.4.3 Other international publications 11
1.4.4 Technical documents 11
1.4.5 Teaching activities 12
1.5 Document Structure 14

1.1 Contextualization and Motivation

A Digital Twin is conceived as a dynamic virtual model or virtual representation of a real system, process, or service aimed at replicating its behavior, components, properties, and interactions within a specific environment to provide a digitalized, augmented, and processable view [1]–[3]. Since the term Digital Twin was first introduced, it has been adopted across various industry sectors and engineering disciplines, where it is becoming increasingly relevant to model complex and dynamic systems [4].

The digital transformation of society and industry has led to the emergence of novel network applications with complex requirements that traditional network management solutions cannot easily satisfy. At the same time, the exponential proliferation of connected devices is making modern networks more dynamic and heterogeneous. As a result, communication networks are becoming more complex and expensive to manage. To address these difficulties, the concept of Digital Twin has been proposed as a key enabler for the efficient control and management of modern telecommunication networks [5], [6].

In this sense, the Network Digital Twin (NDT) paradigm is emerging as a new technology that efficiently analyzes, diagnoses, simulates, and controls real networks using data and models within a virtualized, digital network. This NDT paradigm is built by applying the concept of Digital Twins across various network management systems and applications, as stated by standards development organizations (SDOs) in telecommunications, such as the Internet Engineering Task Force (IETF) [7] and the International Telecommunication Union (ITU) [8].

Therefore, the concept of NDT considers modeling the behavior of a real network through a dynamic virtual representation of it, which is continuously updated with the real network performance, configuration, and health status data [9]. The use of the NDT paradigm enables the rapid deployment of innovative network technologies and a wide range of applications such as network operation, maintenance, optimization, network visualization, intent verification, and network self-tuning devices [10].

Other European telecommunication standardization organizations, such as ETSI, through different Industrial Specification Groups (ISG), provide alternative definitions and applications of Digital Twins. The ETSI ISG cross-cutting Context Information Management (CIM) focuses on the characterization of context information primarily in the Internet of Things (IoT) domain and provides an alternative definition of Digital Twin as the digital counterpart of a real-world asset, which can be a physical asset or a concept. Furthermore, it provides a systematic classification of Digital Twin types according to their purpose of use, including application use cases [11]. On the other hand, the ETSI ISG Zero Touch Network and Service Management (ZSM) provides a vision of an NDT that is very aligned with that given by IETF, explaining how an NDT can help with network automation and service management, and describing different examples of use cases related to autonomous networks [12], [13].

In the context of an NDT, accurate modeling of the real network in virtual reality has a significant impact on the performance of the NDT. Therefore, there are several challenges to effectively promote the use of NDT solutions as efficient systems for network operations and management purposes, such as the difficulties in mapping real network information (e.g.,

configuration data and operational state of network infrastructures) to unified data models used by the NDT paradigm [14]–[16], but also with data collection, storage, and management processes, considering data as the cornerstone of NDT construction [5].

To overcome these challenges, this Doctoral Thesis proposes a new methodology for modeling an NDT solution by translating data from network management sources into a knowledge-based data integration system that facilitates the interaction with the configuration and operational status of real networks. By applying this NDT approach, the resulting work offers benefits in terms of network control and management to facilitate the application of network automation systems.

All this context provides the opportunity to dive into the world of data model-driven network management and Digital Twins. As a result of this research, a series of research questions are posed, which will be answered throughout this Doctoral Thesis:

1. *What should be the design principles of a Digital Twin architecture that facilitate its application in managing, operating, and automating telecommunications networks?*
2. *How can the harmonization of heterogeneous network data sources be guaranteed in the design of Digital Twins covering the formalization and use of standardized data representations and communication interfaces for integrating real network data?*
3. *What could be the interoperable mechanism of the Digital Twin solution that would allow external systems to operate on the real network and have control over the status and configuration of the network itself?*

Therefore, the motivation of this Doctoral Thesis is to answer these questions by achieving the objectives listed below.

1.2 Objectives

The overall objective of this Doctoral Thesis is to contribute to the understanding and development of Digital Twins in the field of network operations and management, facilitating interoperability with external systems to acquire information from real network infrastructures and enabling their on-demand operation and management to network operators and administrators. Consequently, the following specific objectives have been formulated:

1. Study the principles that Network Digital Twins must comply with and the barriers that currently exist. This study is fundamental to detect the needs and challenges of Network Digital Twins focusing in the network operations and management ecosystem and to be able to contribute with an architecture design that addresses these challenges.
2. Propose a methodology for modeling and designing a Network Digital Twin architecture for network operations and management purposes. The proposal should allow integrating real network management data, as well as providing representation models and standard communication interfaces that facilitate its interoperability with third party applications.
3. Evaluate the proposed architecture by implementing a functional prototype based on open-source code that facilitates its reproducibility to the research community. The

prototype must be functional, scalable, extensible, and adhere to the standards of formal data representation and communication interfaces proposed in the reference architecture. In addition, special attention will be given to automating the collection of data from heterogeneous network sources, as well as to managing and integrating the associated context information.

4. Evaluate the Network Digital Twin proposal by means of application use cases for network operations and management purposes that will allow the validation of its functional requirements and additional capabilities and benefits.

5. Realize an experimental performance evaluation of the Network Digital Twin solution, in order to discuss the obtained results and provide suggestions to improve the prototype implementation in future work.

1.3 Methodology

In order to answer the questions raised and meet the objectives set, the Doctoral Thesis is divided into three phases, one of documentation and the other two experimental. The first phase consists of a continuous review of the literature about network automation trends and Digital Twins. The second phase consists of the proposal and validation of an architecture and solution for the development of Network Digital Twins focusing on network data collection purposes. The last phase studies the functionality of the Network Digital Twin solution for completing network operations and management purposes, and enabling the interoperability with external systems through the use of standardized data representations and communication interfaces.

The following list of tasks is defined as the methodology to accomplish these phases and the aforementioned set of objectives:

- Task 1: Identify the needed, challenges, and issues related to the Network Digital Twin paradigm in the context of network operations and management (Objective 1).
- Task 2: Study and analyze proposals and contributions in the literature and standardization bodies related to the application of Network Digital Twin in the scope of network operations and management (Objective 1).
- Task 3: Design a methodology for building an architecture of a Network Digital Twin system to cover aspects of network operations and management (Objective 2).
- Task 4: Development of a reference implementation of the Network Digital Twin solution following the design architecture (Objective 3).
- Task 5: Apply the Network Digital Twin solution to solve different case studies in the area of network operations and management (Objective 4).
- Task 6: Realization of performance evaluation analysis of the Network Digital Twin implementation (Objective 5).

This Doctoral Thesis has been carried out in collaboration with the Research Group on Networking and Communication Service Virtualization (GIROS) of the Department of Telematic Systems Engineering at the Universidad Politécnica de Madrid (UPM). With more than 25 years of experience in national and European research projects, GIROS is a consolidated research group at UPM, whose work focus is to contribute to improving the efficiency, performance, management, and quality of experience of new services and trends in telecommunication networks.

In the last few years, coinciding with the start of this Doctoral Thesis, GIROS has been involved in various research projects focused on data-driven network management, exploring and analyzing ways or solutions to the challenge of handling the monitoring of data from a multitude of heterogeneous data sources available in the network management domain. As a result, this Doctoral Thesis has been framed in the context of different European and national research projects, seeking solutions that facilitate the autonomous operations and management of networks through the application of new paradigms focused on improving network automation.

These research projects have served as a working and validation environment, as well as a means of analysis and study supporting the contributions made in the thesis period. During this period, these projects have allowed the validation of the proposals made, the testing of the developments carried out, and the dissemination of some of the results obtained through scientific publications in international journals. In line with this, the following Section 1.4 includes a summary of the activities in terms of participation in research projects and scientific publications related to the research work of this thesis.

Throughout this Doctoral Thesis report, definitions, explanations, research results, and real use cases are included in the field of telecommunications network management. More specifically, the application context is designed to facilitate the automation of network operations and management processes through the use of the Network Digital Twin paradigm.

1.4 Activities Developed within the Context of this Doctoral Thesis

This section presents the results of the scientific and teaching activities carried out throughout the Doctoral Thesis, highlighting participation in competitive research, development, and innovation projects, publications related to the research work of the thesis, as well as collaboration in university teaching courses and supervision of final degree projects during this period.

1.4.1 Participation in competitive R&D&I projects

During the Ph.D. studies related to this Doctoral Thesis, participation in numerous national and European competitive research, development, and innovation projects took place, details of which are listed below:

- 5Growth: "5G-enabled Growth in Vertical Industries"
  - Reference: 856709
  - UPM Principal Investigator: David Fernández Cambronero
  - Start date - End date: June 2019 - February 2022
  - Funding Agency: European Union's Research and Innovation Programme Horizon 2020
  - Scope (National / International): International
- 5G-CLARITY: "Beyond 5G Multi-Tenant Private Networks Integrating Cellular, Wi-Fi, and LiFi, Powered by Artificial Intelligence and Intent Based Policy"
  - Reference: 871428
  - UPM Principal Investigator: David Fernández Cambronero
  - Start date - End date: November 2019 - February 2023
  - Funding Agency: European Union's Research and Innovation Programme Horizon 2020
  - Scope (National / International): International
- Red Temática Go2Edge: "Diseñando las futuras redes, sistemas y servicios de computacion seguros en el extremo"
  - Reference: RED2018-102585-T
  - UPM Principal Investigator: Luis Bellido Triana
  - Start date - End date: January 2020 - December 2022
  - Funding Agency: Ministerio de Ciencia, Innovación y Universidades
  - Scope (National / International): National
- ECTICS: "Infraestructuras de computación en el borde de la red para el soporte de servicios innovadores en las redes de campus"
  - Reference: PID2019-105257RB-C21
  - UPM Principal Investigators: David Fernández Cambronero and Luis Bellido Triana
  - Start date - End date: June 2020 - November 2023
  - Funding Agency: Ministerio de Ciencia e Innovación - Agencia Estatal de Investigación
  - Scope (National / International): National
- PALANTIR: "Practical Autonomous Cyberhealth for resilient SMEs & Microenterprises"
  - Reference: 883335

  - UPM Principal Investigator: Luis Bellido Triana
  - Start date - End date: September 2020 - August 2023
  - Funding Agency: European Union's Research and Innovation Programme Horizon 2020
  - Scope (National / International): International
- PAGODA: "Plataforma Automática de GestiOn Dinámica distribuidA"
  - Reference: Plataforma Inteligente Distribuida del Expediente de CPP del CDTI 1/2020 AB
  - UPM Principal Investigator: Luis Bellido Triana
  - Start date - End date: May 2021 - February 2023
  - Funding Agency: Proyecto de investigación mediante contrato con Bosonit según artículo 83 y financiado por el CDTI
  - Scope (National / International): National
- B5GEMINI: "Beyond 5G Enhanced Management through dIgital twiNs based on artificial Intelligence"
  - Reference: B5GEMINI-INFRA (TSI-063000-2021-81)
  - UPM Principal Investigator: José Ignacio Moreno Novella
  - Start date - End date: April 2022 - November 2024
  - Funding Agency: Ministry of Economic Affairs and Digital Transformation of the Spanish Government and the European Union-NextGenerationEU (Recovery, Transformation and Resilience Plan - PRTR), UNICO-5G I+D Program
  - Scope (National / International): National
- aerOS: "Autonomous, scalablE, tRustworthy, intelligent European meta Operating System for the IoT edge-cloud continuum"
  - Reference: 101069732
  - UPM Principal Investigator: Luis Bellido Triana
  - Start date - End date: September 2022 - August 2025
  - Funding Agency: European Union's Horizon Europe research and innovation program
  - Scope (National / International): International
- ACROSS: "Automated zero-touch cross-layer provisioning framework for 5G and beyond vertical services"
  - Reference: 101097122

  - UPM Principal Investigator: Luis Bellido Triana
  - Start date - End date: January 2023 - December 2025
  - Funding Agency: European Union's Horizon Europe research and innovation program
  - Scope (National / International): International
- ETSI Specialist Task Force (STF) 676 on Usage Domains Mapping: "OpenAPI specification for NGSI-LD API"
  - Reference: Work Item Reference: DGS/CIM-047 (Ref. Body ISG CIM)
  - UPM Principal Investigator: Luis Bellido Triana
  - Start date - End date: February 2024 - December 2024
  - Funding Agency: European Telecommunications Standards Institute (ETSI)
  - Scope (National / International): International

1.4.2 Publications related to the research work of the Doctoral Thesis

This subsection lists the articles written and published during the Ph.D. studies that are directly related to the research area pursued in the Doctoral Thesis. These include publications in international conferences and workshops, as well as publications in international journals indexed in the field of telecommunications.

International indexed journals

- I. D. Martinez-Casanueva, D. Gonzalez-Sanchez, L. Bellido, D. Fernandez and D. R. Lopez, "Toward Building a Semantic Network Inventory for Model-Driven Telemetry," IEEE Communications Magazine, vol. 61, no. 3, pp. 60-66, March 2023, doi: 10.1109/MCOM.001.2200222.
  - Summary: This paper proposes to leverage the NGSI-LD standard, defined by ETSI ISG CIM for managing context information across information systems, to collect context information related to Model-Driven Telemetry (MDT) obtained from network infrastructure equipments and to integrate it into a network inventory. A reference architecture for this network inventory is designed and validated through a prototype. In addition, a case study is presented to illustrate how the context information gathered in the network inventory can be used to support the development of network monitoring applications.
  - Additional information: This work has been partially funded by the European Union's Research and Innovation Programme Horizon 2020 under Grant Agreement No. 871428 (5G-CLARITY project) and by the Spanish Ministry of Economy and Competitiveness, and the Spanish Ministry of Science and Innovation through the ECTICS project (PID2019-105257RB-C21) and Go2Edge (RED2018-102585-T).

  - Journal Impact Factor: JCR Q1 - 8.3 (2023)

- Moreno-Sancho, A.A., Pastor, A., Martinez-Casanueva, I.D., González-Sánchez D., and Triana L.B., "A data infrastructure for heterogeneous telemetry adaptation: application to Netflow-based cryptojacking detection," Annals of Telecommunications, vol. 79, pp. 241–256, October 2023. doi: https://doi.org/10.1007/s12243-023-00991-6.

  - Summary: This paper proposes a monitoring system based on data modeling and low-latency data processing that facilitates the integration of heterogeneous information sources for network telemetry normalization and aggregation. As a use case, the paper describes a cryptomining detection system based on network traffic flows processed by a Machine Learning engine. The results show how the monitoring system in this use case is leveraged to obtain aggregated information that improves the performance of the cryptomining detection model.

  - Additional information: The conference paper presented in 2023 at the "26th Conference on Innovation in Clouds, Internet and Networks (ICIN)" was selected to be published in a special issue of the Springer Journal "Annals of Telecommunications". The result of this paper is an extension of the previous conference paper. This article, carried out in collaboration with Atos Research and Innovation and Telefónica Innovación Digital, received funding from the European Union's Research and Innovation Programme Horizon 2020 under Grant Agreement No. 883335 (PALANTIR project) and No. 952644 (FISHY project).

  - Journal Impact Factor: JCR Q3 - 1.8 (2023)

- I. D. Martinez-Casanueva, L. Bellido, D. González-Sánchez, and D. Lopez, "CANDIL: A federated data fabric for network analytics," Future Generation Computer Systems, vol. 158, pp. 98-109, April 2024, doi: https://doi.org/10.1016/j.future.2024.04.013.

  - Summary: This paper proposes a solution for integrating data from heterogeneous data sources, focusing mainly on network management aspects. The solution defines an ontology that captures and defines network management concepts originally modeled with YANG, together with a reference architecture for integrating and consuming the data by building a federated knowledge graph and using the NGSI-LD standard proposed by ETSI ISG CIM.

  - Additional information: This work was partially supported by the European Union's Horizon Europe research and innovation program under Grant Agreement No. 101069732 (aerOS project) and by the Spanish Ministry of Economy and Competitiveness, and the Spanish Ministry of Science and Innovation through the ECTICS project (PID2019-105257RB-C21).

  - Journal Impact Factor: JCR Q1 - 6.2 (2023)

- Daniel González-Sánchez, Luis Bellido, Ignacio D Martinez-Casanueva, David Martínez-García, David Fernández, and Diego R. Lopez, "Toward Building a Digital Twin for Network Operations and Management," IEEE Open Journal of the Communications Society, vol. 6, pp. 2583-2598, 2025, doi: 10.1109/OJCOMS.2025.3549873

  - Summary: This paper proposes a new knowledge-based data integration system for modeling Network Digital Twin (NDT) solutions. By relying on automated mechanisms, standard interfaces, and standard data models, the resulting NDT solution enables exposing the capabilities of real networks in terms of configuration and operational status in a formal and interoperable model, thus facilitating network operations and management.
  - Additional information: This work was supported in part by the European Union's Horizon Europe Research and Innovation Program under Grant 101069732 (aerOS) and Grant 101097122 (ACROSS), and in part by the Ministry of Economic Affairs and Digital Transformation of the Spanish Government and the NextGenerationEU (Recovery, Transformation and Resilience Plan - PRTR) Projects of the UNICO-5G I+D Program under Grant TSI-063000-2021-79 (B5GEMINI-AIUC) and Grant TSI-063000-2021-81 (B5GEMINI-INFRA).
  - Journal Impact Factor: JCR Q1 - 6.3 (2023)

International conferences and workshops

- D. González-Sánchez et al., "Model-Driven Network Monitoring Using NetFlow Applied to Threat Detection," 2022 IEEE 8th International Conference on Network Softwarization (NetSoft), Milan, Italy, 2022, pp. 450-455, doi: 10.1109/NetSoft54395.2022.9844107.
  - Summary: This workshop conference paper focuses on the possibility of applying a data model-driven approach to network flow collection using YANG data models. The paper proposes a specific YANG data model for the NetFlow protocol, a protocol widely used in today's network management for threat detection. It also presents a reference architecture that enables the collection, aggregation, and delivery of NetFlow-based monitoring data modeled with YANG. This architecture aims to demonstrate the benefits of NetFlow monitoring data structured according to a YANG model for Artificial Intelligence engines that analyze network flows for threat detection, leveraging the semantics, flexibility, and scalability provided by YANG models.
  - Additional information: This article, carried out in collaboration with Telefónica Innovación Digital, received funding from the European Union's Research and Innovation Programme Horizon 2020 under Grant Agreement No. 883335 (PALANTIR project) and No. 871808 (INSPIRE-5Gplus project) and by the Spanish Ministry of Economy and Competitiveness, and the Spanish Ministry of Science and Innovation through the ECTICS project (PID2019-105257RB-C21) and Go2Edge (RED2018-102585-T).
- A. A. Moreno-Sancho, A. Pastor, I. D. Martinez-Casanueva, D. González-Sánchez and L. B. Triana, "A data infrastructure for heterogeneous telemetry adaptation. Application to Netflow-based cryptojacking detection," 2023 26th Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), Paris, France, 2023, pp. 105-112, doi: 10.1109/ICIN56760.2023.10073490.

  - Summary: This conference paper proposes a monitoring system based on data modeling and low-latency data processing that facilitates the integration of heterogeneous information sources for network telemetry normalization and aggregation. As a use case, the paper describes a cryptomining detection system based on network traffic flows processed by a Machine Learning engine. The results show how the monitoring system in this use case is leveraged to obtain aggregated information that improves the performance of the cryptomining detection model.
  - Additional information: This article, carried out in collaboration with Atos Research and Innovation and Telefónica Innovación Digital, received funding from the European Union's Research and Innovation Programme Horizon 2020 under Grant Agreement No. 883335 (PALANTIR project) and No. 952644 (FISHY project).

1.4.3 Other international publications

Here is an article elaborated and published during Ph.D. studies that is not directly related to the research area of this Doctoral Thesis.

- C. M. Lentisco, L. Bellido, D. González-Sánchez, I. D. Martínez-Casanueva, D. Fernández and I. Soto, "Extending the MEC Mobility Service to Support Cache Prefetching," 2022 18th International Conference on the Design of Reliable Communication Networks (DRCN), Vilanova i la Geltrú, Spain, 2022, pp. 1-4, doi: 10.1109/DRCN53993.2022.9758012.
  - Summary: This conference paper proposes to extend the standard AMS service in Multi-Access Edge Computing (MEC) environments for application mobility to support cache prefetching of multimedia content based on mobility predictions. This prefetching mechanism allows multimedia content to be cached in advance at a target MEC site before a handover situation occurs, thus improving the perceived latency and continuity of multimedia services.
  - Additional information: This work was supported by the Spanish Ministry of Economy and Competitiveness, and the Spanish Ministry of Science and Innovation in the context of ECTICS project (PID2019-105257RB-C21) and Go2Edge (RED2018-102585-T).

1.4.4 Technical documents

This subsection contains information about a technical document prepared in cooperation with the European Telecommunications Standards Institute (ETSI) as a result of the work developed during the Ph.D. studies.

- ETSI, "OpenAPI Specification for NGSI-LD API," GS CIM 047 v1.1.2, ETSI, Dec. 2024.
  - Summary: This technical document formally describes the OpenAPI Specification for the standard NGSI-LD API for Context Information Management defined by ETSI Industry Specification Group (ISG) Context Information Management

(CIM)[1.1]. The document outlines the design and evaluation strategies for the OpenAPI Specification implementation, as well as its general structure and content. Practical examples are also included throughout the document to help readers understand the usability of OpenAPI.

  - Additional information: This ETSI ISG CIM Group Specification (GS) document has been produced as a result of the work carried out during the participation in the ETSI Specialist Task Force (STF) 676 project on Usage Domains Mapping: "OpenAPI specification for NGSI-LD API".

1.4.5 Teaching activities

This subsection presents the university teaching activities related to the courses and final degree projects supervised during the Doctoral Thesis period at the Universidad Politécnica de Madrid.

Teaching courses given during the Doctoral Thesis period

- Fundamentos de los Sistemas Telemáticos
  - University degree: Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación
  - Academic years: 2021-2025
- Centros de Datos y de Provisión de Servicios
  - University degree: Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación
  - Academic years: 2022-2025
- Computación en la Nube
  - University degree: Grado en Ingeniería y Sistemas de Datos
  - Academic years: 2022-2025
- Computación en Nube y Virtualización de Redes y Servicios
  - University degree: Máster Universitario en Ingeniería de Telecomunicación
  - Academic years: 2021-2025
- Arquitectura de Redes y Servicios en la Nube
  - University degree: Máster Universitario en Ingeniería de Redes y Servicios Telemáticos
  - Academic years: 2024-2025

[1.1]The OpenAPI Specification for NGSI-LD defined by ETSI ISG CIM is available in a repository of the CIM GitLab organization within the official ETSI Forge. The repository URL is as follows: https://forge.etsi.org/rep/cim/ngsi-ld-openapi.

- Redes Definidas por Software, Virtualización y Servicios Avanzados de Red
  - University degree: Máster Universitario en Ingeniería de Telecomunicación
  - Academic years: 2021-2025
- Redes Definidas por Software y Virtualización de Redes
  - University degree: Máster Universitario en Ingeniería de Redes y Servicios Telemáticos
  - Academic years: 2021-2025

Final degree projects supervised during the Doctoral Thesis period

- *Diseño de un sistema de monitorización de redes basado en modelos. Integración de mecanismos de telemetría mediante el protocolo NETCONF.*
  - Author: Ripoll Cabrera, Andrés Manuel
  - University degree: Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación
  - Grade: 8.5
  - Exam date: July 6, 2022
- *Implementación de un sistema de automatización y orquestación de flujos de integración y entrega continuas para el desarrollo, despliegue y control de servicios para redes 5G.*
  - Author: Gil Zubeldia, Unai
  - University degree: Máster Universitario en Ingeniería de Redes y Servicios Telemáticos
  - Grade: 10
  - Exam date: July 19, 2023
- *Diseño e implementación de un sistema de telemetría como servicio para la monitorización de funciones de red virtualizadas.*
  - Author: Encinas Alonso, Aitor
  - University degree: Máster Universitario en Ingeniería de Telecomunicación
  - Grade: 10 (M.H.)
  - Exam date: July 20, 2023
- *Desarrollo de escenarios de validación para el análisis de datos de telemetría sobre arquitecturas de red basadas en topologías Leaf and Spine.*
  - Author: Vicente Albertos, Mario Pedro

  - University degree: Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación
  - Grade: 9.5
  - Exam date: July 16, 2024

- *Evaluación e implementación de una solución de partición de red para dar soporte a servicios en redes de comunicación 5G.*
  - Author: Coiduras Gállego, José María
  - University degree: Máster Universitario en Ingeniería de Redes y Servicios Telemáticos
  - Grade: 8.5
  - Exam date: September 20, 2024

- *Desarrollo de escenarios virtuales didácticos para el estudio comparativo de soluciones de red en Kubernetes.*
  - Author: Martín Luengo, Roberto
  - University degree: Máster Universitario en Ingeniería de Telecomunicación
  - Grade: 9.5
  - Exam date: September 26, 2024

1.5 Document Structure

This dissertation raises and develops the topics and objectives introduced previously, following the structure and organization of the chapters defined below.

Chapter 2: State of the Art

The second chapter includes the state of the art of this Doctoral Thesis, where insights into technical details and research activities related to Digital Twins and new trends in the scope of network operations and management are introduced.

Chapter 3: Modeling an NDT Solution for Network Operations and Management

The third chapter introduces the Network Digital Twin solution proposed in this Doctoral Thesis, providing design guidelines and a methodology on how to model a Network Digital Twin system for network operations and management purposes.

Chapter 4: Descriptive NDT System for Network Operations and Management

This chapter includes the prototype implementation details of the Network Digital Twin solution defined in Chapter 3. It also includes a case study to show different examples of network operations and management requirements that can be addressed by this Network Digital Twin solution.

Chapter 5: Experimental Results and Evaluation

This chapter provides the experimental performance results and evaluation of the Network Digital Twin prototype implementation described in Chapter 4.

Chapter 6: Conclusions and Future Work

This final chapter discusses the conclusions and future work derived from the development of this Doctoral Thesis.

Annexes

This dissertation includes additional annexes with details about the design and implementation of the Network Digital Twin solution presented in this document. In addition, the software development activities related to the work of this thesis are also included.

Chapter 2

State of the Art

This chapter is intended to provide the necessary background for a comprehensive understanding of this Doctoral Thesis. It offers insights into the research activities and technical considerations relevant to the implementation and use of Digital Twins in the domain of network operations and management. First, the concept of Digital Twin is described, identifying its benefits and challenges, and focusing on data integration and data modeling techniques. Then, the suitability of the Digital Twin concept in the networking domain is analyzed, identifying standardization and research activities, in addition to specifying its key components, as well as the benefits and challenges encountered. Finally, the study analyzes emerging trends in the scope of network operations and management that are important to consider when modeling Digital Twin solutions, with the ultimate goal of achieving the autonomous networking milestone.

Table of Contents

2.1 Digital Twins 18
2.1.1 Distinction between Digital Twins, simulations, and emulations . . 20
2.1.2 Benefits and enablers of Digital Twins 21
2.1.3 Challenges of Digital Twins 22
2.1.4 Data integration techniques, approaches, and architectures 23
2.1.5 Data modeling in Digital Twins 31
2.2 Network Digital Twins 39
2.2.1 NDT standardization activities 40
2.2.2 NDT benefits and challenges 43
2.2.3 NDT components and architecture 44
2.2.4 NDT and Digital Map process 47
2.2.5 Enabling technologies for building an NDT 49
2.2.6 NDT for data model-driven network management 50
2.3 Emerging Trends in Network Operations and Management . . . 52
2.3.1 Model-driven network management and telemetry 54
2.3.2 Intent-Based Networking for network automation 63
2.3.3 Advancing network automation with Artificial Intelligence and Knowledge Graphs 66

2.1 Digital Twins

The idea of Digital Twin (DT) technology was first voiced in 1991, with the publication of the "Mirror Worlds" book, by David Gelernter [17]. The central metaphor of the book is the computer as a digital mirror of the world, suggesting that the progress of computing will inevitably produce a single vast, distributed computer system containing a complete mirror image of reality [18]. However, Dr. Michael Grieves (then on faculty at the University of Michigan) is credited with first applying the concept of Digital Twins to manufacturing in 2002 for product lifecycle management and formally announcing the concept of Digital Twin [19], [20]. But the term Digital Twin was coined from a NASA employee named John Vickers, who first used it in 2010 as part of a technology road map report for the space agency. However, the core idea of using a Digital Twin as a means to study a physical object actually dates back much earlier, to the dawn of the digital age. In fact, it can be rightfully said that NASA pioneered the use of Digital Twin technology during the space exploration of the Apolo missions in the late 1960s, when each voyaging spacecraft was exactly replicated in an earthbound version that was used for study, simulation, and safety precaution purposes by NASA personnel serving on flight crews [21]. Finally, Grieves and Vickers provide the first formal definition of Digital Twin [22] as *"a set of virtual information constructs that fully describes a potential or actual physical manufactured product from the micro-atomic level to the macro-geometrical level"*.

Typically, a Digital Twin is conceived as a dynamic virtual model or representation of a real system, process, or service aimed at replicating their behavior, components, properties, and interactions within its particular environment in order to provide a digitalized, augmented, and processable view [2], [3], [20]. Since the Digital Twin term was first introduced, it has been applied in various industry sectors and engineering fields where it is becoming increasingly relevant for modeling complex and dynamic systems [4]. In this regard, the authors in [23] conducted an industrial case study to analyze how the industry perceived and defined the Digital Twins between 2002 and 2019. They define Digital Twin as a virtual dynamic representation of a physical system which is connected to it over the entire lifecycle for bidirectional data exchange. Unlike Grieves and Vickers' original definition, which only includes an automated connection from the real system to its Digital Twin representation, this definition includes the bidirectional exchange of data between the real system and the Digital Twin, allowing remote control and operation of the real system, as well as new opportunities for insight collaboration between the real and digital instances.

Based on the given definitions of a Digital Twin term, one might identify a common understanding of Digital Twins, as digital counterparts of physical objects. Within these definitions, the terms Digital Model, Digital Shadow, and Digital Twin are often used synonymously. However, the given definitions differ in the level of data integration between the physical and digital counterpart. Some digital representations are modeled manually and are not connected with any physical object in existence, while others are fully integrated with real-time data exchange.

In this sense, the authors in [21], [24] propose a classification of Digital Twins into these three subcategories, according to their level of data integration:

1. *Digital Model, where there is no automated connection and data exchange with the physical asset. As a result, changes in the physical asset do not immediately affect the Digital Model and vice versa.*
2. *Digital Shadow, where there is an automated one-way data flow from the physical asset to the digital object, but not the other way around. Then, a change in the state of the physical asset causes a change in the state of the Digital Shadow, but not vice versa.*
3. *Digital Twin, where the data flows between the physical asset and the Digital Twin instance in both directions in an automated manner. In this case, the Digital Twin may also control the physical asset. Therefore, a change in the state of the physical asset directly results in a change in the state of the Digital Twin and vice versa.*

With the increasing importance of Digital Twins, the International Organization for Standardization (ISO) also published the ISO 23247 series [25], defining a framework to support the creation of digital twins of observable manufacturing elements, including personnel, equipment, materials, manufacturing processes, facilities, environment, products, and supporting documents. ISO states that a Digital Twin assists with detecting anomalies in manufacturing processes to achieve functional objectives such as real-time control, predictive maintenance, in-process adaptation, Big Data analytics, and Machine Learning. A Digital Twin monitors its observable manufacturing element by constantly updating relevant operational and environmental data. The visibility into process and execution enabled by a Digital Twin enhances manufacturing operation and business cooperation. An interesting aspect of ISO 23247 is the notable omission of bidirectional communication. Then, this standard primarily emphasizes the monitoring aspect of a Digital Twin but not in the actuation phase. In addition, ISO in collaboration with the International Electrotechnical Commission (IEC) publishes the technical document ISO/IEC-2023 [26] to establish terminology and describe concepts in the field of Digital Twin, including Digital Twin terms and definitions, functional view of Digital Twin, and Digital Twin stakeholders.

The Digital Twin Consortium [27], which helps organizations innovate throughout the digital engineering lifecycle and builds on the promise of Digital Twins, defines a Digital Twin as an integrated data-driven virtual representation of real-world entities and processes, with synchronized interaction at a specified frequency and fidelity. In addition, the Digital Twin Consortium states that the Digital Twin needs to provide the following capabilities:

- *Digital Twins are motivated by outcomes, driven by use cases, powered by integration, built on data, enhanced by physics, guided by domain knowledge, and implemented in reliable and trustworthy information technology (IT) systems.*
- *Digital Twin systems transform business by accelerating and automating holistic understanding, continuous improvement, decision making, and interventions through effective action.*
- *Digital Twin systems are built on integrated and synchronized IT systems, use real-time and historical data to represent the past and present, and simulate predicted futures.*
- *Digital Twin prototypes use data to model and simulate predicted futures before being integrated into IT systems and before synchronization with the real-world process.*

Thus, we can conclude that a Digital Twin is commonly defined as a virtual instance of a real system that is continually updated with the latter's performance, maintenance, and health status data throughout the lifecycle of the real system [2]. In addition, the Digital Twin could also act as a control instance, where a configuration or change performed on the Digital Twin could be reflected on the real system.

Figure 2.1 shows this view of the Digital Twin paradigm, where there is a bidirectional link between the real-world asset and the corresponding Digital Twin instance for both the data flow from the real system to the Digital Twin and the control flow from the Digital Twin to the real system.

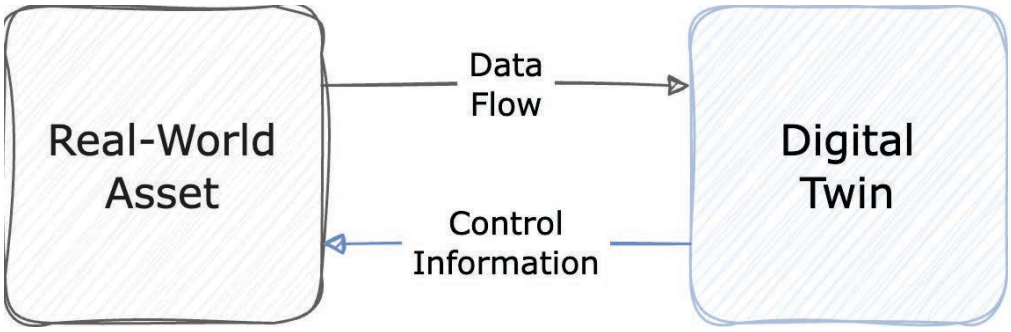


Figure 2.1: Digital Twin general overview.

2.1.1 Distinction between Digital Twins, simulations, and emulations

A Digital Twin is a virtual representation of a physical object or a system designed to accurately reflect it. It spans the object's lifecycle, is updated from real-time data, and uses simulation, Machine Learning, and reasoning to help make decisions [28]. After being provided with the relevant data, the digital model can be utilized to conduct various simulations, analyze performance problems, and create potential improvements. The ultimate objective is to obtain valuable knowledge that can be used to improve the original physical entity.

Although both simulations and Digital Twins utilize digital models to replicate the various processes of a system, a Digital Twin is actually a virtual environment, making it much richer for study. The difference between a Digital Twin and a simulation is largely a matter of scale. While a simulation typically studies one particular process, a Digital Twin can run any number of useful simulations to study multiple processes. A simulation models a system or process to understand its behavior under specific conditions. Then, simulations are static snapshots in time that represent predefined scenarios. However, a Digital Twin goes beyond this. It is a living representation of a specific physical or real asset. Unlike a simulation, a Digital Twin is continuously updated using real-time data (e.g., data from sensors reflecting its unique and most up-to-date state) [28], [29].

The differences do not end there. Simulations usually do not benefit from having real-time data. But Digital Twins are also designed around a two-way flow of information that occurs when physical object sensors provide relevant data to the system processor (i.e., the Digital

Twin instance) and then happens again when insights created by the processor (i.e., again the Digital Twin instance) are shared back with the original source physical object.

A similar case occurs with emulation systems, where the focus is to accurately replicate the behavior and characteristics of real environments, making use of hardware and software technologies. Although emulation systems are really useful for testing and validating complex systems, such as communication network scenarios, by attempting to create realistic test environments or testbeds, they fail to provide in real time the feedback of information to the physical system that is covered by Digital Twins. This last feature of Digital Twins makes them really important solutions to address the autonomous management of real systems with fast decision-making, response optimization, problem detection, or change planning.

In summary, simulations tell us how things should work, emulations help us reproduce or imitate things, and Digital Twins show us how things are working right now. Digital Twins can address more issues from many more viewpoints than standard simulations and emulations can, offering greater potential for improving products and processes. This is due to the fact that Digital Twins have better and constantly updated data related to a wide range of areas, combined with the added computing power of having a virtual environment.

2.1.2 Benefits and enablers of Digital Twins

The main advantage of Digital Twins is that they can accurately model complex systems. Nowadays, engineering and manufacturing predominantly use Digital Twins to provide accurate virtual representations of objects and simulations of operational processes. Digital Twin technology holds great potential for a wide range of industrial activities and can radically change the face of manufacturing. Digital Twin applications include enabling smart manufacturing in Industry 4.0 [5], improving the performance of complex engineering products, and modeling physical interactions. In this sense, we can highlight the following general benefits offered by Digital Twin technology [28], [30]:

- Better research and development: The use of Digital Twins enables more effective research and design of products, with an abundance of data created about likely performance outcomes. That information can lead to insights that help companies make needed product refinements before starting production.
- Greater efficiency: Even after a new product has gone into production, Digital Twins can help mirror and monitor production systems, with an eye to achieving and maintaining peak efficiency throughout the entire manufacturing process.
- Product end-of-life: Digital Twins can even help manufacturers decide what to do with products that reach the end of their product lifecycle and need to receive final processing, through recycling or other measures. By using Digital Twins, they can determine which product materials can be harvested.

The main aspects of Digital Twins are data acquisition, data modeling, data management and storage, and data processing. In the literature, it has been argued that the growth of Digital Twins is due to the evolution of technology, especially the technologies considered as enablers for the development of Digital Twins, including the Internet of Things (IoT),

Artificial Intelligence (AI), Extended Reality, and Cloud Computing [30], [31]. Digital Twins can take advantage of these four technologies to collect and store real-time data, obtain information to provide valuable insights, and create a digital representation of a real asset.

2.1.3 Challenges of Digital Twins

According to [32]–[34], the main challenges in building and maintaining Digital Twins can be summarized as the following aspects:

1. Data acquisition and processing. Digital Twins require collecting and processing large amounts of data, both in batches and in real time, which implies the need for distributed systems running in the Cloud Computing systems using Big Data techniques.
2. Data integration from multiple and heterogeneous data sources. Digital Twins interact with a variety of heterogeneous sources, using different protocols, different data formats, and receiving the modeled information in different ways.
3. High-fidelity modeling. The integration of data sources is a major challenge and requires specific technologies for information homogenization, as well as the definition of data models and ontologies that allow the representation of any entity regardless of its origin.
4. Massive data storage, high availability, and standardized communication interfaces. The integration of interdisciplinary models poses challenges in terms of data storage, information exchange between systems, and data processing and availability. Therefore, it is advisable to use standardized communication protocols to ensure interoperability between all components of a Digital Twin infrastructure.
5. Real-time, bidirectional communication between the Digital Twin systems and the real systems. The Digital Twin receives real-time data from real assets, and the analysis results are used to control the real systems in real time. Due to various shortcomings such as large amounts of data, network transmission delay, and model analysis time, it can be difficult for a Digital Twin to achieve real-time two-way communication with the real world ecosystem.
6. Unified development platforms and tools. In addition to standard data representation models and data exchange interfaces, the components and tools used to build Digital Twins must be consistent. Due to different formats, protocols and standards, current tools may be hard to integrate. Therefore, in the future, the development of universal design and development platforms and tools for Digital Twins is required.
7. Ensure the security of the systems involved in data exchange. Digital Twins encompass a large number of roles and, in some cases, sensitive information. Therefore, systems are required to ensure the security of information (i.e., confidentiality, integrity, availability, and authenticity). Authentication and authorization systems must be implemented to protect data, in addition to ensuring that all communications are encrypted.
8. Environmental coupling technologies. It is common to treat the Digital Twin as an isolated system, lacking connection to the external environment. The mechanism explaining how real objects interact with the environment has not been fully embodied

in the scope of virtual models and Digital Twins. It is somewhat interesting to provide mechanisms for the Digital Twin to communicate with the outside world, thereby enhancing its capabilities and those of the other systems. To achieve this, it is necessary to provide mechanisms for connection and information exchange without negatively affecting information security or compromising any of the systems involved.

The following subsections provide a detailed description of data integration and data modeling trends related to and applicable to Digital Twins. Other considerations related to data acquisition techniques and standardized communication interfaces required to design and build Digital Twin solutions within the scope of network management will be analyzed in Section 2.2. In addition, the Digital Twin solution proposed in this research, which is presented in Chapter 3, takes into account the need for real-time and bidirectional communication with external systems, and also involves a unified set of different software tools.

2.1.4 Data integration techniques, approaches, and architectures

The digital revolution produces massive, heterogeneous, and isolated data. In the context of Digital Twins, effectively leveraging this data diversity requires data integration systems to break down data silos and create a common information space. Such systems ensure not only seamless and standardized access to data, but also enable interoperability, facilitating meaningful interpretation and utilization. This common information space allows the integrated data sources to speak the same language, empowering data analysts with comprehensive tools for dataset exploration and for discovering the implicit knowledge within integrated datasets. The Digital Twin paradigm can be considered as an opportunity and a comprehensive solution that provides data integration and analytics mechanisms, allowing the collection and management of data from multiple data sources and systems [35]–[37].

Specifically, semantic data integration goes beyond the data integration process of combining data retrieved from independent sources to provide a standardized and interoperable structure [38]. Semantic data integration is the process of combining and consolidating heterogeneous data into valuable and meaningful information by linking them to create a richer global representation [39]. Unfortunately, semantic data integration faces several challenges, the foremost being data heterogeneity. Another critical challenge is data querying, which becomes increasingly complex as data integration systems scale up to accommodate diverse and evolving data sources.

Therefore, data integration is the process of combining data retrieved from multiple independent sources to provide users with a unified and integrated view of the data, known as the global schema [40], [41]. A schema provides a formal description of the data, specifying the data types, their properties, and how they relate to each other. The global schema provides a general formal representation for data sources of different idiosyncrasies, regardless of their local representation schemas. In the process of transforming between two schemas, there are two main data integration techniques that can be applied:

- Schema Matching: It involves identifying possible connections or correspondences between elements, typically attributes, in two schemas. The primary goal of schema matching is to discover relationships and similarities between the components of different

schemas, enabling integration and interoperability between them. When presented with two schemas, a resolution to the schema matching problem constitutes a set of matches. Essentially, it is a group of pairings that indicate how elements in one schema align with or relate to elements in the other schema. To address semantic conflicts, one can identify semantic correspondences among elements in the different schemas by accurately discerning the meaning of schema labels. This process is known as semantic schema matching [42]. Ontologies or meta-models can be considered as schemas for knowledge and databases. Therefore, techniques developed for semantic schema matching often leverage ontologies as domain knowledge for semantic annotation, a process of determining the meaning of schema labels in the ontology. Matchings can be used as input to schema mapping algorithms, identifying the creation of mapping rules in schema mapping techniques [43].

- Schema Mapping: It is the process of establishing explicit transformation based on semantic connections between schemas. Its main objective is to define a formal specification that describes how data from one schema can be transformed or mapped to another schema. Given a source schema S and a target schema T, a schema mapping M is a specification that dictates how data instances from S can be converted or related to instances in T [43]. Different mechanisms and algorithms for schema mapping have been proposed and they fall into four different categories:

  1. Global As View (GAV) [44]: It defines a global schema in advance, which is an overview of the local schemas. Therefore, data integrated according to a local schema is mapped directly to the global schema. Moreover, queries made on the global schema are easily translated into queries on local schemas. However, it is not a flexible mechanism because any change in the local schema can significantly impact the global schema, requiring costly updates.

  2. Local As View (LAV) [45]: It takes the opposite approach to GAV by defining each local schema as a view over a global schema. This makes the global schema independent of the different local data sources to be integrated, allowing greater flexibility. However, since local schemas provide only fragments of the overall schema, query processing becomes more complex and costly.

  3. Global and Local As View (GLAV) [46]: It combines aspects of both GAV and LAV, enabling flexible mappings between local and global schemas. Instead of requiring a strict one-to-one correspondence, GLAV allows transformation rules that relate elements from local schemas to the global schema. This category balances flexibility and efficiency, making it particularly useful in complex data integration scenarios where structured mappings guide dynamic transformations.

  4. Both As View (BAV) [47]: It treats both global and local schemas as views over an implicit, underlying schema. This approach provides a high degree of flexibility but requires complex reconciliation mechanisms to ensure consistency between different schema representations. It is commonly used in ontology-based data integration and Knowledge Graph alignment transformations, where schemas evolve dynamically based on an underlying conceptual model.

Conventionally, a data integration system is defined as the following expression (Expression 2.1):

$$I = \langle G, S, M \rangle, \tag{2.1}$$

where: i) G is the global schema, representing the unified view, such as an information model, relational schema, or an ontology; ii) S is the set of data sources that are represented by the set of local schemas (i.e., $S_1, \ldots, S_n$); and iii) M represents the mapping that specify correspondences between the local schemas (i.e., S) and the global schema (i.e., G).

Data integration can be done physically (i.e., materialized), virtually, or in a hybrid way. Regardless of the approach category, in general, an integration system architecture is composed of three main layers, as depicted in Figure 2.2:

1. Data Layer: Contains the data sources to be integrated.
2. Wrapper Layer: This layer provides access to the different data sources, extracts data from them, and represents the data in the global schema. It also facilitates interaction between the Data Layer and Warehouse/Mediator Layer in the data integration system architecture.
3. Warehouse/Mediator Layer: Contains the necessary elements to query the different data sources through a global schema that can be either materialized in a data warehouse or virtualized through a mediator system. The external applications can then interact with the data warehouse or mediator system to retrieve the integrated data.

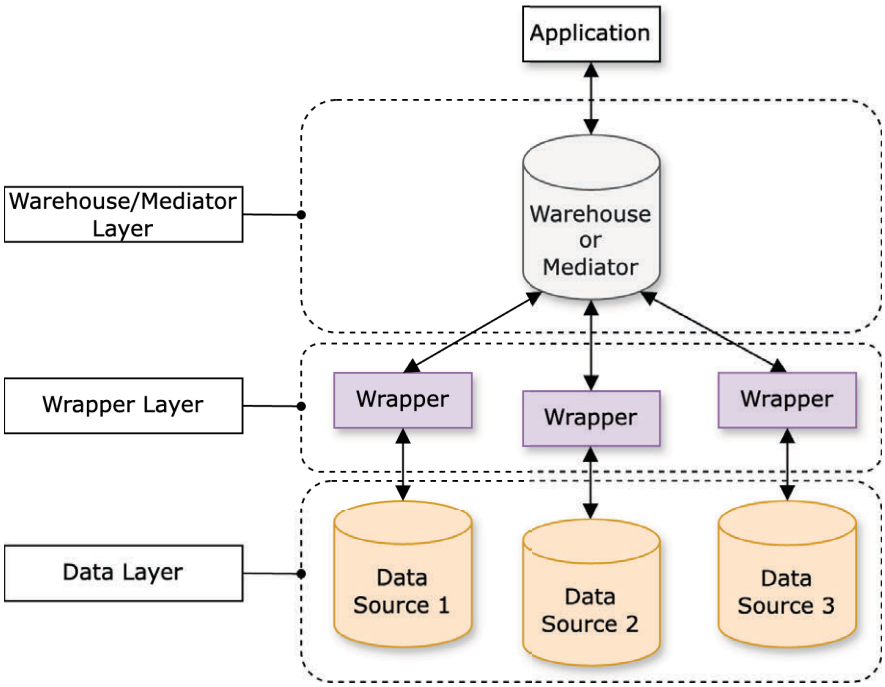


Figure 2.2: General architecture of a data integration system (based on [40]).

The following subsections detail the architecture of materialization, virtualization, and hybrid integration systems, respectively.

Architecture of the data materialization approach

In the data materialization approach, the data sources are physically integrated, and the global schema representation is fully materialized (see Figure 2.3). Indeed, there is a need to implement a data warehouse via a Data Base Management System (DBMS) that reproduces all the source data according to the global schema. In this case, requests to the global schema of the warehouse are based on centralized query processing techniques. As a result, the user interacts with the warehouse through direct data query requests. Data integration is then based on the overall warehouse schema, providing an integrated view of sources. Since the integration is performed by the Extract, Transform, and Load (ETL) tools, which perform a multiphase process, the data integration process is divided into four main steps that correspond to those of the ETL approach [48]: i) data extraction from sources; ii) data transformation at the structural and semantic levels of the target global schema; iii) data integration; and iv) load and storage of integrated data in the target system (i.e., the data warehouse system).

It is important to note that the communication takes place in two different phases: i) an offline communication phase for the materialization of the data from the time it is extracted from the sources until it is loaded into the warehouse system; and ii) an online communication phase where external applications can retrieve the data already materialized in the warehouse system.

Then, the data materialization approach involves the static transformation of distributed data into a data warehouse and is linked to an ETL approach. The main advantage of this approach is that query processing is centralized and fast, as the integrated data is consolidated into a single source of truth, similar to the case of a database. Therefore, a rewriting phase of queries is not necessary, making the system more reactive and efficient. Furthermore, system performance no longer depends on the performance of each source or communication delay.

However, these materialized approaches have major disadvantages, as they are expensive in terms of implementation and maintenance, and ETL processing is time-consuming. They require extra storage since data are replicated, which implies additional costs for storage and maintenance [49]. Furthermore, there is no guarantee that the data loaded into the data warehouse is up-to-date due to the desynchronization between the offline and online phases discussed above. The materialized data may rapidly become outdated if the data source is regularly updated. This can be addressed by running the data extraction and transformation processes periodically. However, in the context of large datasets, a compromise must be found between the costs of materialization in terms of time, memory, and CPU and the freshness of the available data in the warehouse system.

Architecture of the data virtualization approach

In data virtualization approaches, the global schema is entirely virtual and, therefore, not materialized. In this case, all data remain in local sources and are accessed through an intermediate infrastructure component, usually called a mediator, that contains the global schema representation. The objective of this approach is to provide users with the illusion of querying a homogeneous and centralized system, while the sources are distributed and

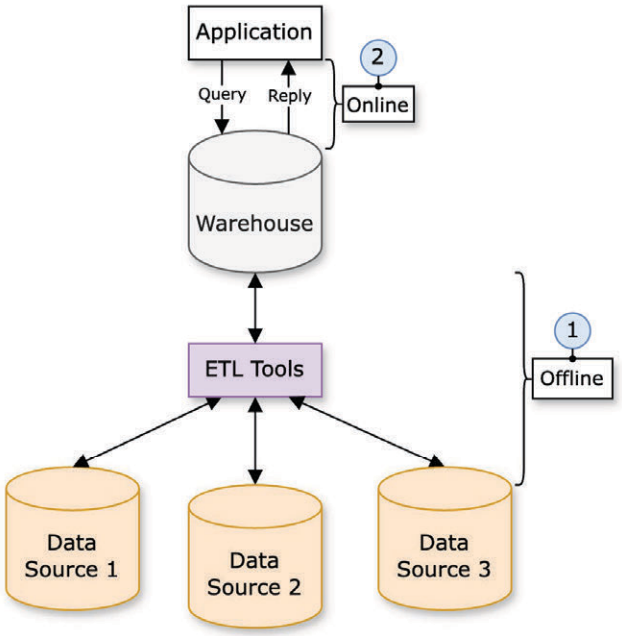


Figure 2.3: General architecture of a data materialization system (based on [40]).

autonomous. It is also proposed as a solution to the challenge of data storage. In data virtualization approaches, one of the most common architectures is the mediator-wrapper architecture. The mediator-wrapper architecture was proposed by Wiederhold to integrate data from a selection of independent sources [50]. It is composed of three layers, as shown in Figure 2.4: mediator, wrappers, and data sources.

The mediator provides a common interface to the users that is used to formulate queries. Specifically, at the mediator level, the global schema provides a vocabulary for expressing requests and describing sources through a set of abstract views on them. As a model of the field of application, information models or ontologies can be used as global schemas. In fact, they provide a structured vocabulary to support the expression of requests. Besides, the mediator establishes a connection with the accessible data sources by describing their content in a consistent and uniform manner, in a global data catalog.

Wrappers are tools that allow mediators to access the content of the sources in a uniform language. They reformulate a request in a specific query language accepted by each source. They also transform query responses into the global schema defined by the mediator. Unlike data materialization, a data virtualization approach establishes communication in only one online phase, as queries from the external applications trigger the reformulated subqueries to retrieve the data from the sources on demand. Thus, these virtual data integration approaches allow data to remain in its original sources and be accessed on the fly using a query language. Hence, the integrated data are always up-to-date, making the query results more sound and complete despite the evolving nature of sources. Such approaches solve the problem of real-time integration of heterogeneous data and have the advantage of avoiding the cost of data materialization.

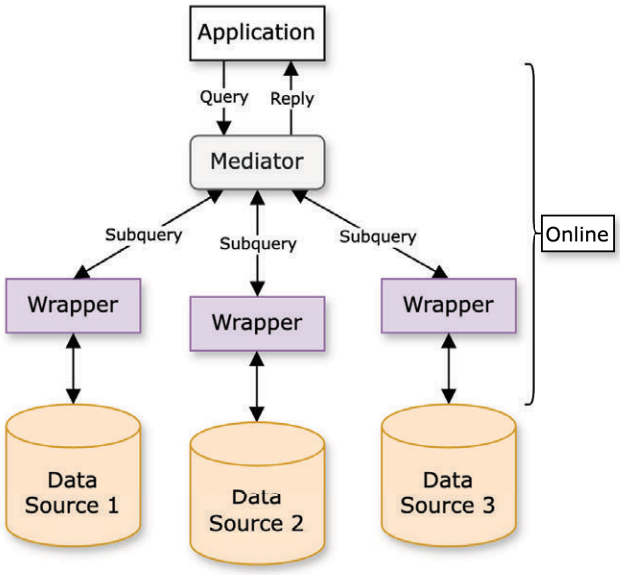


Figure 2.4: General architecture of a data virtualization system (based on [40]).

Despite the numerous proposals for data virtualization approaches found in the literature, several limitations can be noted. Many of the proposals lack automation, which requires significant human intervention, particularly in data mapping and modeling. For instance, the semantic annotation process is mostly done manually by human annotators. Moreover, most of the approaches deal with particular data formats. Although recently proposals aim to integrate heterogeneous data formats, there is still a lack of schematic and semantic interoperability among different data sources. In data virtualization systems based on a mediator-wrapper architecture, the use of wrappers facilitates the integration of distributed data sources providing heterogeneous formats (e.g., CSV, XML, JSON, database encoding formats, etc.). However, several wrappers need to be written to convert the query into the format of each data source and then convert the returned result into the common format of the mediator. Furthermore, wrappers are language-specific. Consequently, if the language of the global schema changes, all wrappers should be updated.

Architecture of the hybrid data integration approach

Hybrid approaches have been developed to create more flexible data integration systems. They involve a combination of data virtualization and partial data materialization [51] (see Figure 2.5). Hence, in hybrid approaches, a portion of the data is queried on demand, similar to data virtualization approaches, while the other part of the data is extracted and stored in a data warehouse.

Consequently, to develop a hybrid data integration system, the data integrator expert should answer several questions, including:

- *Which query results to materialize and which to virtualize?*
- *Where to materialize the data?*
- *How and when to refresh materialized data?*

- *How to adapt query processing in the presence of materialized and virtualized data?*

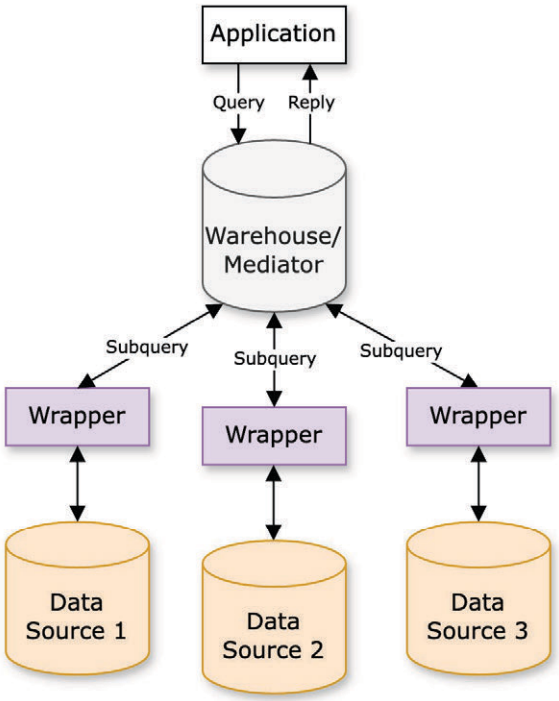


Figure 2.5: General architecture of a hybrid data integration system (based on [40]).

The main advantage of hybrid data integration approaches is to take advantage of the strengths of both materialization and virtualization approaches while mitigating their respective shortcomings. In particular, hybrid data integration offers a trade-off between query response time and data freshness within a data integration system. On the one hand, the inclusion of partially materialized data contributes to enhanced query performance. Pre-computing and storing selected portions of data allow queries to be answered more quickly, as the relevant information is readily available without the need for complex real-time computations or extensive data retrieval. On the other hand, virtual integration ensures that data remain up-to-date. Importantly, by striking a balance between pre-materialized and virtualized data significantly reduces query response times, providing users with faster and more efficient access to the data they need. Moreover, this hybrid model emphasizes the integrity and completeness of query responses in light of any data changes or updates, thereby providing a robust and comprehensive solution for data integration.

Query processing and rewriting

Data querying is a critical challenge when developing a data integration system. Once a data integration system is established, end users can submit queries according to the global schema. However, query processing, also called query answering, differs depending on the data integration approach. In the data materialization approach, a copy of all data that correspond to the global schema is saved in a single global database. Thus, a query performed by external users or applications against the global schema is answered by simply running the

query over the global database. In contrast, in a data virtualization approach, data are kept in the sources, and thus a query against the global schema has to be translated to corresponding queries against the local schemas of the relevant data sources. Here, query processing makes use of mappings by decomposition and reformulation of the query into subqueries over data sources and then recomposition of partial answers for a unified response for the original query. Hence, query processing methods vary depending on the mapping approach chosen in the data virtualization system.

In addition, query processing is performed differently depending on the schema mapping category. In the GAV approach, the global schema is expressed in terms of the data sources, with each global relation defined as a view over the local relations. Queries made on the global schema are straightforwardly translated into queries on local schemas, making it a suitable approach when the global schema is stable and well-defined. This simplifies query rewriting, but makes the system less adaptable to changes in local schemas. In LAV, query processing offers query optimization since local schemas are expressed in terms of the global schema, enabling the system to select a minimal number of data sources relevant to a query. However, query rewriting becomes more complex and costly, requiring dynamic resolution of correspondences across multiple data sources. The hybrid approaches (i.e., GLAV and BAV) express both a global schema and local schemas by using LAV and GAV query rewriting processes. This reduces LAV complexity while ensuring the scalability of the global schema. With data materialization, the schema mapping approach affects data transformation and integration processes, but not query rewriting, since queries run directly on the materialized data. On the other hand, with data virtualization, schema mapping directly affects query rewriting because data remains in its original sources.

Given that data are distributed across various sources, a distributed query processing strategy decomposes the original query into relevant subqueries for each data source and later joins local results. Distributed querying typically involves five main steps [52]:

- Query parsing: It is the first task performed in any query processing system. This task aims to ensure that the query is correctly specified and to convert the query into an internal pattern in order to facilitate the manipulation of the query for later steps, particularly query rewriting and query planning.
- Source selection: Identifies the relevant data sources for each query. In fact, all data sources do not necessarily contain data that are relevant for a given query. Thus, selecting relevant sources prevents sending useless queries.
- Query rewriting: It decomposes the original query into a set of subqueries that will be distributed to the different data sources. It both ensures the preservation of the original query semantics and takes into account the performance of the subqueries to be processed. Rewriting is usually expressed in the same language as used for both the original query and views and is called exact if it is equivalent to the original query.
- Query planning: It involves generating a plan that efficiently computes a user query from the relevant information sources. This plan consists of data retrieval actions in distributed information sources and data manipulation operations. Essentially, it organizes subqueries to create an optimal evaluation plan that minimizes the overall

execution time of the query. As a general rule, the most selective queries should be processed first.

- Query execution: It is intended for the execution of the query plan and the collection of the results generated from all the subqueries.

An important aspect of query processing is query optimization. The goal of query optimization is to choose the best execution strategy for a given query under the given resource constraints [52]. Optimization can occur at various stages of the query processing pipeline and is typically categorized as static or dynamic. Static optimization refers to the optimization processes that occur before query execution, such as source selection, query planning, and query rewriting. These techniques aim to improve the accuracy and completeness of the query response. In contrast, dynamic optimization involves techniques performed during the query execution phase to reduce processing time and communication costs between the query engine and remote data sources.

2.1.5 Data modeling in Digital Twins

One of the key aspects of Digital Twins is to ensure high-fidelity data modeling. Data models allow for the representation of the real world in a virtual environment. The Digital Twin systems use data models to capture the current state of real objects and manage them in their virtual counterpart. They do not require complex equations or algorithms but perform processing based on current information and historical information of the real systems.

In data modeling, there is a need to define the data schema and data format. The authors in [34], [53] indicate that the most widely used standards for modeling Digital Twins are the Digital Twin Definition Language (DTDL) [54], Asset Administration Shell [55], Next Generation Service Interface-Linked Data (NGSI-LD) [56], OGC SensorThings API [57], and the Web of Things [58]. In addition, serialization formats include AutomationML, UML, SysML, Extensible Markup Language (XML), JavaScript Object Notation (JSON), JSON Linked Data (JSON-LD), and the various Resource Description Framework (RDF) serialization formats.

All of the aforementioned solutions for modeling Digital Twins focus their attention on modeling intelligent systems in the IoT domain. However, the NGSI-LD standard proposed by ETSI embraces the possibility of integrating data sources from multiple domains and has recently been analyzed as a good opportunity to model Digital Twin solutions oriented to network infrastructure data sources [59].

The content of this section will specify key paradigms related to data modeling, such as Knowledge Graphs, Ontologies, and Data Fabric, considering the importance of governing the knowledge of integrated and modeled data in Digital Twin solutions. But first, some considerations of the distinction between data models and information models are discussed. Finally, since the purpose of this Doctoral Thesis is to model a Digital Twin solution focused on data sources belonging to the network infrastructure, the use of the NGSI-LD standard is proposed, which will be detailed below.

Information Models vs. Data Models

In order to avoid confusion throughout the document, this section makes a distinction between information models and data models [60], [61]. The main purpose of an information model is to model managed objects at a conceptual or abstract level, independently of any specific implementations or protocols used to transport the data. The degree of detail of the abstractions defined in the information model depends on the modeling needs of its designers. Information models can be implemented in different ways and mapped on different protocols. Then, they are protocol neutral. Another important characteristic of an information model is that it defines relationships between managed objects. Information models can be defined in an informal way, using natural languages such as English, or, alternatively, they can be defined using a formal language or a semi-formal structured language such as Entity-Relationship diagrams or Unified Modeling Language (UML) diagrams, the latter standardized by the Object Management Group (OMG) [62]. Information models such as these allow objects and the relationships between them to be represented in a standard graphical form, providing an easy way to get an overview of the underlying management model. Organizations may use the contents of an information model to delimit the functionality that can be included in a data model.

Data models, conversely, are defined at a lower level of abstraction, providing a more concrete and detailed representation. They include implementation and include protocol-specific details. Since conceptual models can be implemented in different ways, multiple data models can be derived from a single information model. Most of the management models standardized to date are data models. For example, there are several standardized data modeling languages for defining managed objects in network management. One of these is the use of Management Information Base (MIB) [63] modules defined by using the Structure of Management Information (SMI) [64] language, commonly used by the Simple Network Management Protocol (SNMP) [65]. Another example of a standardized data modeling language that has been adopted by the network management industry is YANG , which will be discussed in more detail later. Both are standardized by the Internet Engineering Task Force (IETF).

Although information models and data models serve different purposes, it is not always possible to precisely define which level of detail belongs to an information model or to a data model, leaving a gray area where they overlap.

Knowledge Graphs, Ontologies, and Data Fabric paradigms

A Knowledge Graph is a semantically rich data model to store, organize, and understand connected entities [66], [67]. A Knowledge Graph contains three essential elements: i) entities, which represent the data of the organization or domain area; ii) relationships, which show how the data entities interact with or relate to each other, providing context for the data; and iii) an organizing principle that captures meta-information about core and relevant concepts.

Then, Knowledge Graphs model knowledge of a domain as a graph with a network of entities and relationships. The main function of Knowledge Graphs is to connect and contextualize data in a way that makes it more explicit and usable for common applications. It then

implements a semantic layer that connects all of the context data by mapping it to a conceptual model (i.e., an information model). Knowledge Graphs help organizations make critical decisions based on harmonized knowledge models and data from siloed source systems. Knowledge Graphs are most often associated with data integration due to the huge value created by their data standardization and semantic modeling capabilities [68].

The term Knowledge Graph was introduced in 2012 when Google shared their vision of "things, not strings" [69]. In this proposal, real-world entities and their relationships with other entities were formally represented in a graph along with the categorization and features of these entities (i.e., the knowledge about reality). In this regard, the conceptualization of reality in a formal and explicit way that can be understood by machines is known as an ontology.

An ontology is a formal description of knowledge as a set of concepts within a domain and the relationships that hold between them [70]. It ensures a common understanding of information and makes explicit domain assumptions, thus allowing organizations to make better sense of their data.

Knowledge Graphs connect ontologies with the collected data, providing a powerful combination that gives contextualized understanding of the data, as the meaning of the data (i.e., the concepts) travels with the data themselves. Another advantage of using Knowledge Graphs is the flexibility of the graph structure, which allows for accommodating any other types of data structures such as CSV or JSON [71]. Based on these fundamental aspects, Knowledge Graphs enable the integration of linked data from heterogeneous data silos, and therefore, the realization of a Data Fabric platform.

The Data fabric paradigm defines a new data infrastructure architecture that enables integrating data from heterogeneous data sources [72]. The Data Fabric implements a semantic knowledge layer that enables connecting silos of data, thus providing unified access to all data in a distributed environment. This layer simplifies data access by abstracting consumers from the location and underlying complexities of data sources, such as data access protocols, schemas, and encoding formats. However, the mechanisms, standards, and technologies that are more suited for the implementation of a Data Fabric architecture are still under exploration [73]. Further exploring on knowledge management, [74] and [75] identify Knowledge Graphs, based on the Semantic Web standards, as enablers for a semantic-based data integration in the Data Fabric.

Traditionally, the standard Resource Description Framework (RDF) [76] data model defined in the World Wide Web Consortium (W3C) Semantic Web has been considered the standard graph data model and, therefore, most Knowledge Graph implementations have relied on RDF and related standards such as Web Ontology Language (OWL) [77]. However, during the last decade the Labeled Property Graph (LPG) data model has gained popularity with the emergence of graph databases like Neo4j [78]. The benefits of the LPG data model in terms of query performance and optimized storage have made it an interesting choice for the implementation of Knowledge Graphs [79].

Standards developing organizations like the W3C or ETSI have shifted their focus towards property graphs. In this regard, the W3C recently chartered the RDF-star Working Group [80],

which aims at providing interoperability between RDF and property graphs. The specifications produced within RDF-star are set to evolve the RDF standard in order to enable making statements about statements. The group is looking to finalize all specifications by 2025, along with the first commercial and open-source implementations compliant with RDF-star. On the other hand, ETSI started the ISG CIM group to create the NGSI-LD standard, which specifies a protocol for the exchange of contextualized data between smart applications. This NGSI-LD standard is described in detail below.

Both W3C RDF-star and ETSI NGSI-LD focus on enabling complex Knowledge Graphs, grounded on semantics captured by ontologies, that allow for making statements about statements. However, when it comes to building and interacting with the Knowledge Graph, the NGSI-LD API based on REST facilitates the adoption of the NGSI-LD standard, as opposed to W3C SPARQL, which defines a query language and protocol with a harder learning curve. But more importantly, the NGSI-LD API standardizes relevant interactions not covered by the W3C standards, such as subscriptions (i.e., periodic or based on events) to the Knowledge Graph or the concept of a Context Source, which provides the basis towards implementing data virtualization. For these reasons, NGSI-LD has been selected as the standard to implement the knowledge-based data integration platform that supports the NDT solution in this thesis.

NGSI-LD standard and Digital Twins

Digital Twins rely heavily on data, both consumed and produced by them, to provide a digitalized, augmented, and actionable view of real-world assets. In line with this view, recent reports from ETSI's Industry Specification Group on Context Information Management (ETSI ISG CIM) analyze the feasibility of building Digital Twins and propose different use cases and applications [11]. ETSI CIM considers the exchange of context information across systems as a crucial enabler for smart applications and has standardized the NGSI-LD specification to allow these applications to gather information from multiple sources, integrate the information, and eventually generate derivative information or make decisions. To this end, the NGSI-LD standard specification comprises two main features: i) the NGSI-LD information model [81] as a means to represent and describe context information; and ii) a REST-based application programming interface (API) [56] as a means to exchange context information between systems.

The NGSI-LD information model builds on the LPG data model and the JSON-LD data format, where entities have relationships with other entities, and entities and relationships can have properties with particular values that provide additional characteristics. NGSI-LD defines these foundational classes "entity, property, relationship" in the so-called NGSI-LD meta-model, which represents the formal basis for building property graphs. In addition, the NGSI-LD information model enriches the LPG model by adding support for second-level logic, which allows the definition of properties-of-properties, relationships-of-properties, and relationships-of-relationships. Figure 2.6 shows the UML diagram for the NGSI-LD meta-model. The associations between the UML classes are used to interrelate them while keeping the structure and semantics defined by the NGSI-LD information model. All elements considered in an NGSI-LD information model must be URIs (Uniform Resource Identifiers),

including the identifiers of the NGSI-LD entities. In this regard, JSON-LD provides the concept of context entries, which allow translation between expanded full URIs and compacted short forms of URIs. The NGSI-LD information model is grounded on Semantic Web standards, allowing graph data to reference ontologies defined in languages such as RDF Schema (RDFS) [82], OWL, or Simple Knowledge Organization System (SKOS) [83], which can be leveraged to apply semantic reasoning purposes. In addition, the NGSI-LD standard refers to domain-specific models when information models introduce the specific entity types required for a particular domain. Hereinafter, the terms NGSI-LD domain model and NGSI-LD model are used interchangeably throughout the document.

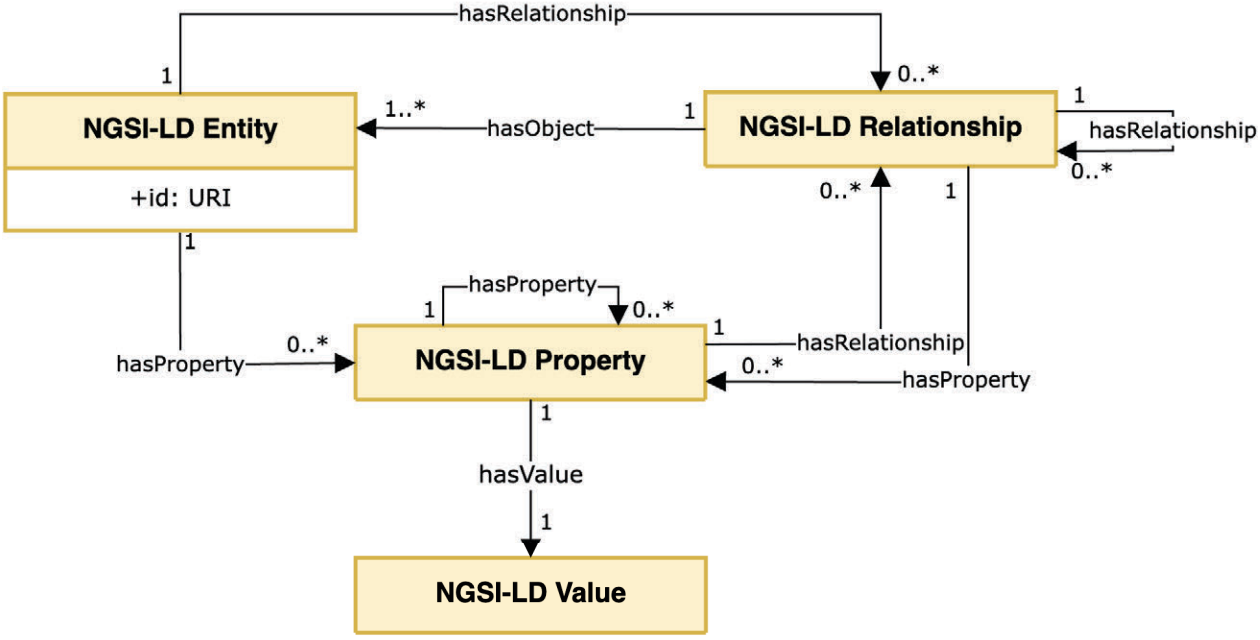


Figure 2.6: UML diagram of NGSI-LD meta-model (based on [56]).

Based on the structure of the NGSI-LD information model, the NGSI-LD API enables the exchange of context information. Table 2.1 shows the list of NGSI-LD API basic operations supported by an NGSI-LD Context Broker, as defined by the ETSI Group Specification CIM 009 document [56].

The NGSI-LD standard does not define a reference architecture for this API, but suggests prototypical architectures such as centralized, distributed, or even federated, as depicted in Figure 2.7 [84]. The NGSI-LD standard introduces multiple architectural roles that exchange context information using the NGSI-LD API. The main architectural role is the Context Broker, which stores and exposes the context information. Depending on the architectural approach, the Context Producer and Context Source roles register, provide, and update the context information they can offer through the Context Broker. Context Producers store context information in the Context Broker, whereas Context Sources expose context information on demand when requested by the Context Broker. Note that Context Brokers can register themselves as Context Sources of other Context Brokers, thus building a federated architecture. Meanwhile, the Context Consumer role discovers and queries relevant context information and subscribes to notifications.

Table 2.1: Basic operations of the NGSI-LD API.

Operation	HTTP Method	Description	Routes
Create	POST	Creates a new entity.	/ngsi-ld/v1/entities
Read	GET	Retrieves information of one or more entities or attributes.	/ngsi-ld/v1/entities, /ngsi-ld/v1/entities/{id}
Update	POST / PATCH / PUT	Replaces entities or updates one or more properties of an existing entity.	/ngsi-ld/v1/entities/{id}, /ngsi-ld/v1/entities/{id}/attrs, /ngsi-ld/v1/entities/{id}/attrs/{attrId}
Delete	DELETE	Deletes an entity or its attributes.	/ngsi-ld/v1/entities/{id}, /ngsi-ld/v1/entities/{id}/attrs/{attrId}
Subscribe	POST	Subscribes to entity updates.	/ngsi-ld/v1/subscriptions
Register	POST	Registers NGSI-LD Context Source.	/ngsi-ld/v1/csourceRegistrations

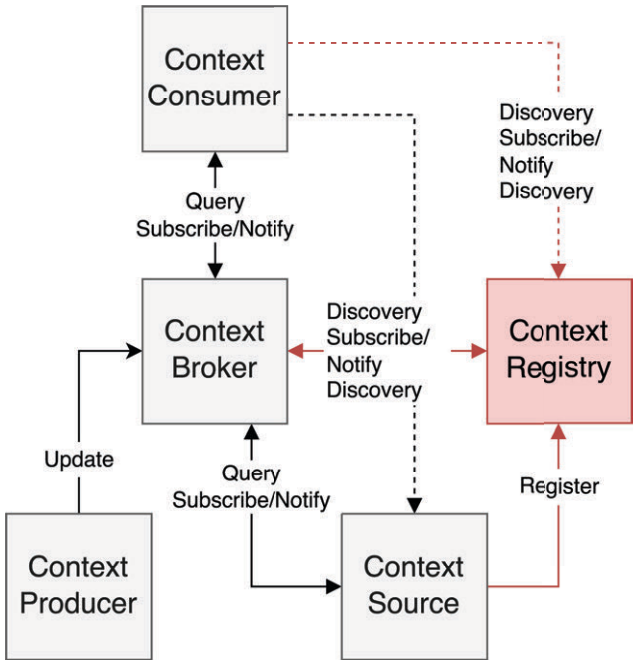


Figure 2.7: Distributed architecture showing interactions between components using the NGSI-LD API. The Context Registry is depicted in red color as it can be optionally deployed outside the Context Broker [84].

The choice of architecture depends on the scale and specific requirements of the application environment from which context information needs to be extracted. Given that the NGSI-LD standard supports integrating context information from multiple data sources, this thesis proposes using it to cover network monitoring sources. This approach aims to facilitate comprehensive network operations and management purposes considering the nature of the sources and the scale of the network infrastructure. Consequently, NGSI-LD domain-specific models for network operations and management will be covered.

NGSI-LD has been adopted by the European FIWARE[2.1] initiative for managing context information in intelligent applications, including Digital Twins. In addition, Digital Twins have been systematically classified by ETSI CIM according to their capabilities as follows:

- Descriptive Twin: The Descriptive Twin informs about the current state of the real-world asset. The Digital Twin case presents past and current values of some of the real-world asset characteristics. These characteristics can be static (e.g., a building geometry) or dynamic (e.g., sensor measurements). The connection between the real-world asset and the Digital Twin is bidirectional, so a change made in the Descriptive Twin is reflected (in real time or not) on the real-world asset (i.e., actuation). The Descriptive Twin capability does not restrict itself to metrics collected from the real-world asset and can be augmented with computational functions. Descriptive Twins are natively handled by any NGSI-LD compliant Context Broker. Some augmentation functions (e.g., time series aggregation functions) are also natively included in the NGSI-LD specification.
- Predictive Twin: The Predictive Twin extends the Descriptive Twin capability by providing predictions on the way the real asset could evolve in the future. The predicted future values can be computed on demand or stored within properties of the Digital Twin entities to be consumed, as any other temporal value, through the NGSI-LD Temporal API. Any past, current, and future values of the real asset can then be consumed by external applications using the same NGSI-LD API. The Predictive Twin capability is handled by an NGSI-LD Context Broker coupled with a prediction model, which may require the availability of additional context information. Predictions are not ensured to be completely accurate, but informative about possible futures. The NGSI-LD specification offers different ways to categorize the values of Digital Twin properties based on their sources or other metadata. For instance, it supports grouping attributes or using multi-attributes, which would enable the inclusion of confidence levels in predictions.
- Prospective Twin: The Prospective Twin builds upon the predictive capabilities. It is meant to help users of the Digital Twin to evaluate the impact that actions would have on the real asset if applied. This allows running a *what-if* analysis by executing the prediction model over the current description of the asset, which is altered to represent the intended action.

[2.1] FIWARE is a platform, driven by the European Union, for the development and global deployment of future Internet applications. Among the key differentiators of FIWARE is the development of a software ecosystem around the NGSI-LD standard specified by the ETSI ISG CIM. The official FIWARE website is available at the following link: https://www.fiware.org/

- Prescriptive Twin: The Prescriptive Twin goes one step further. It aims at identifying actions to be taken over the real asset for it to reach a target state. Identified actions could be simply provided to the user or actuated directly on the real asset, thus implementing an autonomous behavior of the asset.
- Diagnosis Twin: The Diagnosis Twin is meant to help understand what happened in case of a real asset malfunction, or what is happening when a real asset deviates from its expected operation. It builds on a diagnostic model that collects additional data related to the real asset evolution (e.g., log files).

A representation of the architecture relevant for handling the Digital Twin capabilities described above is shown in Figure 2.8. To build the Digital Twin system, several components have been identified:

- The NGSI-LD Context Broker is the central component. It contains the graph instance of the Digital Twin.
- The connection with the real-world asset is handled by system adapters. These components are in charge of connecting the NGSI-LD API with the interface of the real-world asset. The complexity of the system adapters can vary. The simplest one would publish information from an external system into the Context Broker, supporting Descriptive Twin capabilities.
- Additional services are optionally deployed to realize the Digital Twin capabilities and interface with the Context Broker through the NGSI-LD API:
  - Augmentation functions provide additional descriptive information in the Digital Twin system based on the observed state, as well as other information which could be made available to the Context Broker. These could be built upon NGSI-LD embedded capabilities such as aggregation functions (e.g., to provide the average of a time series) or an external service.
  - Prediction models are consuming information about the real-world asset and possibly other information to provide predictions. They can be executed on demand or triggered when new information about the Digital Twin is made available (i.e., using the NGSI-LD subscription mechanisms). No assumption is made about the type of prediction model, which can be a mathematical model, a model based on Machine Learning, or some other options. The description of the current state of the Digital Twin could be altered to run predictions on a hypothetically modified version of the Digital Twin to execute a prospective analysis.
  - A prescriptive service can be deployed to provide recommendations related to the Digital Twin management actions. For example, this service can interact with the prediction model by testing different alterations to identify the optimal operating point of a real-world asset.

The communication flow across these services and the triggering of the corresponding processes can be managed using a pure data model-driven approach based on the NGSI-LD API. However, more complex flows would require new approaches to orchestrate the execution of the services.

A Digital Twin solution using the NGSI-LD standard is, therefore, part of the innovative work on which this Doctoral Thesis focuses. To the best of our knowledge, there are no other proposals in the literature that attempt to delve into the use of the NGSI-LD standard for the construction of Digital Twin solutions for network operations and management.

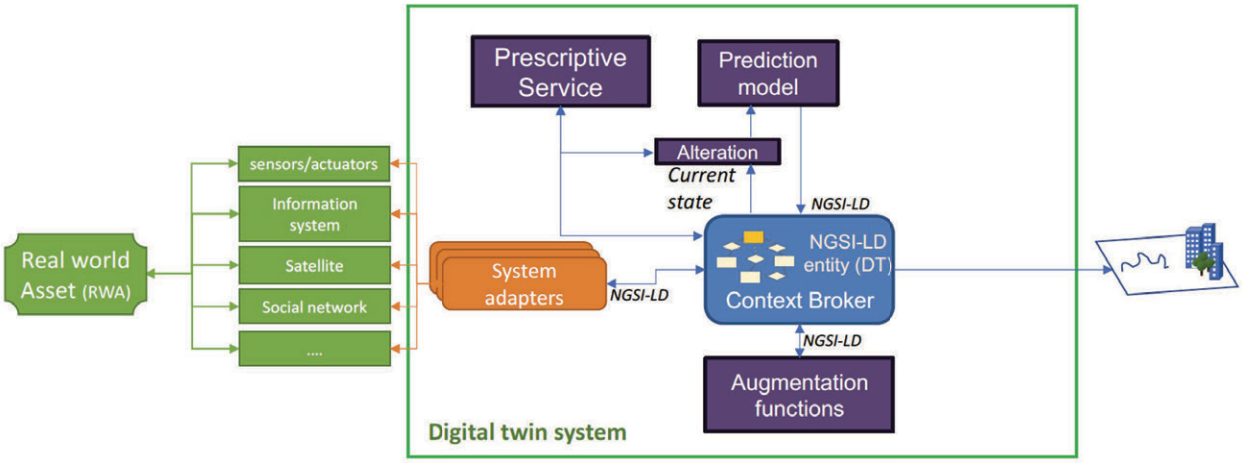


Figure 2.8: High-level architecture for Digital Twin systems according to the NGSI-LD standard [11].

2.2 Network Digital Twins

The digital transformation of society and industry has led to the emergence of novel network applications with complex requirements that traditional network management solutions cannot easily fulfill. At the same time, the number of connected devices is growing enormously, making modern networks very dynamic and heterogeneous. As a result, communication networks are becoming more complex and expensive to manage.

To deal with the complexity, cost, heterogeneity, and dynamics of modern telecommunication networks, the concept of Digital Twin has been extended as a promising technology and key enabler for efficient network control and management [1], [5], [6]. A Network Digital Twin (NDT) is thus conceived as a dynamic virtual model or representation of a real network that can be used to analyze, diagnose, simulate, and control the behavior of the network. NDT technology facilitates the rapid deployment of innovative network solutions with a wide range of applications such as network operation, maintenance, optimization, visualization, or intent verification [10].

NDT development has already raised the interest of standards development organizations such as ITU-T and IETF, which identify requirements, propose architectures, and discuss use cases [7], [8]. In the future, NDT can become an indispensable part of overall network management, covering the entire lifecycle of real networks without compromising their daily operations, facilitating the application of innovative technologies and increasing their level of automation and intelligence. However, several challenges have been identified in the design and implementation of NDT systems to efficiently control and manage modern telecommunication

networks. These challenges include dynamically reflecting the behavior of the real network within the NDT representation [85], providing an adequate mapping mechanism of data retrieved from the real network to unified data models used by the NDT considering the heterogeneity and idiosyncrasies of the network monitoring data sources [14]–[16], and designing mechanisms for data collection, storage, and management, taking into account that data is the cornerstone of NDT construction [5].

This section introduces insights into standardization and research activities related to the use of Digital Twins in the network management domain.

2.2.1 NDT standardization activities

The use of NDT solutions for monitoring, managing and optimizing different aspects of a network infrastructure has been a popular research topic in the last years, with a significant number of scientific articles and proposals available in the state of the art. However, Network Digital Twins are still in the early stages in network-related standardization bodies. Currently, there are no standardized specifications available and only a few Standards Development Organizations (SDOs) have initiated activities or discussions on the topic. This subsection aims at summarizing such existing activities in different SDOs.

NDT paradigm at the Internet Engineering Task Force (IETF)

The Network Digital Twin topic has been recently addressed by the IETF community in the Network Management Research Group (NMRG). This is one of the IETF's research groups within the Internet Research Task Force (IRTF), and its purpose is to provide a forum for the exploration of network management technologies and solutions that are not yet sufficiently understood for engineering work within the IETF. The NMRG is interested in discussing future Digital Twin-related technologies, preparing the needed new standards, and proving concepts with prototype implementations. What is discussed in this group can therefore be considered a kind of pre-standardization step in the field of NDT for network management purposes.

The NMRG draft (working document) on *"Network Digital Twin: Concepts and Reference Architecture"* [7] presents an overview of the concepts of the Network Digital Twin, provides a reference architecture, and a list of challenges and benefits for such technology. The document underlines the difference between a Network Digital Twin and a traditional network management system, the first being an innovative and interactive virtual-real mapping and a data-driven approach to build closed-loop network automation. The draft also defines four key elements for the NDT: 1) data collection, to maintain historical and real-time information on the real network; 2) models, to develop a comprehensive representation of the data collected; 3) standardized interfaces between the real network and the NDT platform in order to implement the closed-loop automation system and between the NDT platform and the application layer leveraging it; and 4) a mapping mechanisms allowing to identify the NDT elements with reference to the real network elements. The NMRG also lists some fundamental challenges for the design of an NDT, such as the scalability of the system, interoperability, or the complexity of managing real-time requirements. Other drafts discussed in the NMRG

address data collection requirements or *what-if* design aspects for NDT systems. However, these are only individual drafts presented for discussion by individual people or companies and are not yet endorsed by the research group.

NMOP (Network Management Operations) is another recently formed working group at the IETF. It also focuses on the NDT paradigm. It aims to address operational issues faced in the deployment and integration of network management technologies, especially data models and management protocols, in the context of the ambition of operators to implement more automation. NMOP addresses the modeling of network topologies in the NDT area [15], [86].

NDT paradigm at the European Telecommunications Standards Institute (ETSI)

The Zero-touch network and Service Management (ZSM) Industry Specification Group (ISG) within ETSI recently completed a work item on Network Digital Twins in the technical document ETSI Group Report (GR) ZSM 015 [12]. The scope of the work is to investigate the application of NDT for automation of zero-touch network and service management, and to introduce scenarios that can benefit from it.

The ETSI GR ZSM 015 document includes a list of possible use cases that can be resolved through the use of the NDT paradigm, such as savings in radio network energy, prediction of network slicing risk, prediction of signaling storms in the network, and improvement of Machine Learning model training. Furthermore, the document provides a mapping of the NDT features to the ZSM architecture and a list of additional potential capabilities to integrate into ZSM to support NDT use cases.

There is also a complementary ETSI Group Specification (GS) ZSM 018 document [13] that specifies extensions and new capabilities to support and integrate NDT technologies with the ZSM framework reference architecture to improve end-to-end zero-touch network and service management and automation. This document also documents important NDT principles and defines related NDT use cases to derive specific requirements.

In addition, recent reports from ETSI's Industry Specification Group on Context Information Management (ETSI ISG CIM), analyze the feasibility of building Digital Twins and propose different use cases and applications [11]. Moreover, as previously mentioned in Subsection 2.1.5, Digital Twins have been systematically classified by ETSI CIM according to their capabilities. The different Digital Twin cases can be encompassed in a holistic approach for better performance results. This classification of Digital Twins can be adapted to the field of network operations and management, a topic that will be discussed in detail in Chapter 3.

NDT paradigm at the International Telecommunication Union (ITU)

ITU Telecommunication standardization sector (ITU-T) released in early 2022 a document related to Network Digital Twins (in this document referred to as "Digital Twin Network"), as Recommendation Y.3090 *"Digital Twin Network – Requirements and architecture"* [8]. This document has a very similar structure to the draft under discussion in IETF, presenting a very similar high-level reference architecture and describing the same four main common features for a Network Digital Twin (i.e., data, models, interfaces, mapping). The document also lists a set of functional requirements (i.e., efficient data collection, unified data repository

and data models, open and standardized interfaces, Digital Twin's lifecycle management), as well as a set of service requirements (i.e., compatibility with various vendor equipment, scalability, reliability, security, and privacy).

Recommendation Y.3090 from ITU-T also lists some example of use cases that would benefit from the NDT. These include, for instance, network maintenance and operation, improved efficiency of network optimization through simulation on the NDT, simplification in measuring or indirectly estimating network parameters, support for the implementation of Intent-Based Networking, or detection of attack patterns for security purposes.

Furthermore, a very recent new work has been initiated in ITU-T Study Group 11, under ITU-T Q.STDN *"Signaling and protocol for Digital Twin Network supported for the IMT-2020 network and beyond"* [87]. This work aims to specify the interface for data exchange among digital twin network, third-party application, and physical network, including procedures for network simplification, automation, resiliency and full lifecycle operation and maintenance with the assistance of a Network Digital Twin.

Finally, in ITU-T Study Group 20, three recommendations have recently been approved not specifically to Network Digital Twin, but with some potential reuse for Network Digital Twin, as they investigate federations [88], [89] and information exchange [90] aspects for Digital Twins in smart city scenarios.

NDT paradigm at the 3rd Generation Partnership Project (3GPP)

Several tentatives were made to start a study item on *"Management aspects of Digital Twin Network"* in the 3GPP SA5 group (responsible in 3GPP for Management, Orchestration and Charging for 3GPP systems) [91]. The objective of the proposal included the identification of the potential use cases and corresponding requirements, as well as the set of features of the 3GPP system that could be managed and/or monitored through the Digital Twin. The first version of this study item proposal was submitted in 2021. However, until now the proposal has not yet reached a sufficient consensus to be approved within 3GPP SA5 and is not considered an official 3GPP study item.

NDT paradigm at the TeleManagement Forum (TMF)

The TeleManagement Forum is a global industry association for service providers and their suppliers in the telecommunications industry. This organization works on Digital Twin topics with a focus on closed-loop automation and decision intelligence in TR284E [92], TR284G [93], IG1307 [94], and IG1310A [95] technical draft documents.

The TR284E document will cover industry definitions of Digital Twin concepts, types, use cases, and advances in the use of closed loops. The value of Digital Twin and closed-loop automation is that intelligent workflows can now dynamically learn from the environment they operate in and adapt workflow behaviors intelligently in a way that realizes goals. In addition, the TR284G document applies the paradigm of Digital Twin and closed-loop automation for autonomous network operation. Meanwhile, the purpose of the IG1307 and IG1310A technical documents is to define a common vocabulary to model knowledge domains and a reference architecture for the Digital Twin for decision intelligence.

There is also a TMF catalyst program that facilitates the generation and validation of new concepts. One of them is the C22.0.386 Catalyst program entitled "*Digital twin network application in 5G network operation*" [96]. This program considers a Network Digital Twin Architecture and scenarios regarding Network Digital Twin in 5G network slice for telemedicine and smart factories use cases.

2.2.2 NDT benefits and challenges

The use of the NDT paradigm has become increasingly relevant in network management as a basis for advanced emulation solutions for optimization, troubleshooting, and network upgrade planning. Once an NDT is defined, it can be configured to operate independently of the real network to perform *what-if* analysis or optimization tests, which can then be run in real time on the virtual network model without compromising the real network [5]. Unlike conventional network emulation and simulation solutions, the NDT offers an interactive virtual-real mapping mechanism and a data-driven approach to enable closed-loop network management and automation. The NDT leverages real-time network data for high-fidelity condition simulation, minimizes the risk of applying configuration changes, and improves network reliability for automated fault detection and repair. This makes the NDT a powerful tool for network management compared to traditional emulation approaches that have low fidelity and lack interactive interfaces with the real network [7]. The key benefits of an NDT can be summarized as follows:

- Using real-time data to establish high-fidelity twins, the effectiveness of network simulation is higher; then the simulation cost will be relatively low.
- The impact and risk on real networks are low when automatically applying configuration and policy changes after the full analysis and required verifications (e.g., service impact analysis) are performed within the NDT.
- The faults of the real network can be automatically captured by analyzing real-time data, then the correction strategy can be distributed to the real network elements after conducting adequate analysis within the NDT to complete the closed-loop automatic fault repair.

Compared with other industrial sectors, Digital Twin in the networking field has unique characteristics. On the one hand, network elements and systems have a higher level of digitalization, which implies that data acquisition and virtual-real communication are relatively easy to achieve. On the other hand, networks can include a wide variety of elements and topologies, and their size is characterized by the number of nodes and links that they contain. However, the rate at which the network size grows usually does not keep pace with the increasing demand for services, especially when end-to-end services that span across multiple administrative domains are deployed. Therefore, when building an NDT system, several challenges need to be considered, such as how to deal with the collection and storage of network data and the generalization and scalability of the NDT system (e.g., a large-scale network can generate a large amount of data, thus impacting the data storage cost), interoperability due to possible inconsistencies between multiple Digital Twin instances, data modeling difficulties, complexity of managing real-time requirements, and security risks [5], [7], [97].

These difficulties extend the challenges associated with Digital Twin solutions, which were described in Subsection 2.1.3.

To address the challenges listed above, it is important to agree on a unified architecture of NDT that defines the main functional components and interfaces. But even with such an architecture, key enabling technologies, such as including data acquisition, data storage, data modeling, interface standardization, and security assurance, need to be further investigated.

2.2.3 NDT components and architecture

Given the IETF's standardization efforts to model NDT solutions in the network management domain, we will discuss their approach in detail. As defined by the IETF community, an NDT can be described as a digital counterpart of a real network that captures its attributes, behavior, and interactions and is continually updated with the network's performance, maintenance, and health status data throughout its lifecycle [7]. The NDT is considered to be founded on five core elements as depicted in Figure 2.9:

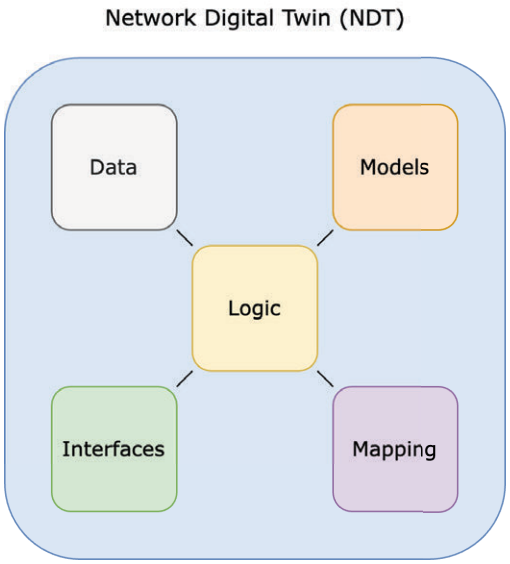


Figure 2.9: Network Digital Twin core elements (based on [7]).

- Data: An NDT should maintain historical data and real-time data (e.g., configuration data, operational state data, topology data, trace data, metric data, process data, etc.) about its real network that are required by the models to represent and understand the states and behaviors of the network. Data can be characterized as a single source of truth and is stored in a data repository, which provides timely and accurate data service support for building various models.
- Models: Models provide a comprehensive representation of the data retrieved from real network assets and their behaviors, reflecting the configuration, state, or use of the different network elements and resources. Models provide information on how the real network operates and generate reasoning data that may be utilized in operational decision-making for various use case-based network applications.

- Interfaces: Standardized interfaces ensure the interoperability of the NDT with real network operations systems. There are two main types of communication interfaces:
  - Interfaces between the NDT and the real network infrastructure that are used for managing and configuring the network, as well as for collecting real-time data.
  - Interfaces between the NDT and external applications, allowing applications to request actions from the NDT and the NDT to expose real network data and capabilities to applications.
- Mapping: An interactive and real-time data mapping process between the NDT and the real network and between NDT instances. Such mappings provide good visibility of the network status, making the NDT suitable to analyze and understand what is going on in the real network. It also facilitates the use of the NDT to optimize the performance and maintenance of the real network.
- Logic: An NDT facilitates optimal resource allocation and configuration, enhancing efficiency and performance. It can enable comprehensive troubleshooting maintenance and control by diagnosing issues using the NDT. Moreover, an NDT plays a crucial role in planning and deployment, allowing the simulation of new designs and configurations to anticipate their effects before implementation.

Based on the definition of NDT core elements, an NDT architecture is depicted in Figure 2.10. The functional components of the NDT architecture can be stated as follows: a Data Repository component, a Service Mapping Models component, and an NDT Management component. These key components can be placed within a single network management domain and provide services to operational applications, such as Software-Defined Networking (SDN) controllers or network emulation applications, within that domain or in other network management domains. They can also be placed in each network management domain and coordinate with each other to provide services to operational applications. One or more NDT instances may be maintained and operated on behalf of a given real network.

A summary of each of the functional components of the NDT architecture is described below:

- The Data Repository component is responsible for collecting, storing, and managing network data. It collects and updates real-time and historical operational, configuration, and telemetry data of the different network elements through the appropriate real network-facing input interfaces (e.g., data collection interface and intent interface), as well as from other operational system components. It also supplies data services (e.g., fast retrieval or conflict handling) to a Service Mapping Models component through appropriate output interfaces (e.g., a query interface).
- Service Mapping Models complete data modeling and provide data or other functional model instances supporting various network applications. There are two main types of models:
  - Basic models refer to network element models and network topology models used to reflect the basic configuration, environment information, operational state, and topology of the network and its elements.

  - Functional models refer to various data models or other models used to generate information supporting network analysis, emulation, diagnosis, prediction, assurance, etc. Functional models can be built and extended depending on the type of network, serving single or multiple network domains, or according to their specific functions. In the latter case, functional models and the information they generate can support various tasks, such as state monitoring, traffic analysis, security assessment, fault diagnosis, or quality assurance. In addition, they can cover different stages of network lifecycle management, such as planning, construction, maintenance, optimization, and operation. Functional models can also be divided into general models and specialized models. Multiple models can be combined to create a model for more specific application scenarios. New applications may require new functional models that do not yet exist. When a new model is needed, the Service Mapping Models subsystem can help create new models based on data retrieved from the Data Repository.

- The NDT Management component effectively manages the NDT operation and its subcomponents, serving applications that use the information generated by the NDT. It manages the session-based operation of the NDT, managing the lifecycle of operations providing services to associated applications. In addition, this component monitors the performance and resource consumption of the NDT and controls various operational aspects of the NDT, including the management of topology, configuration, performance, and security.

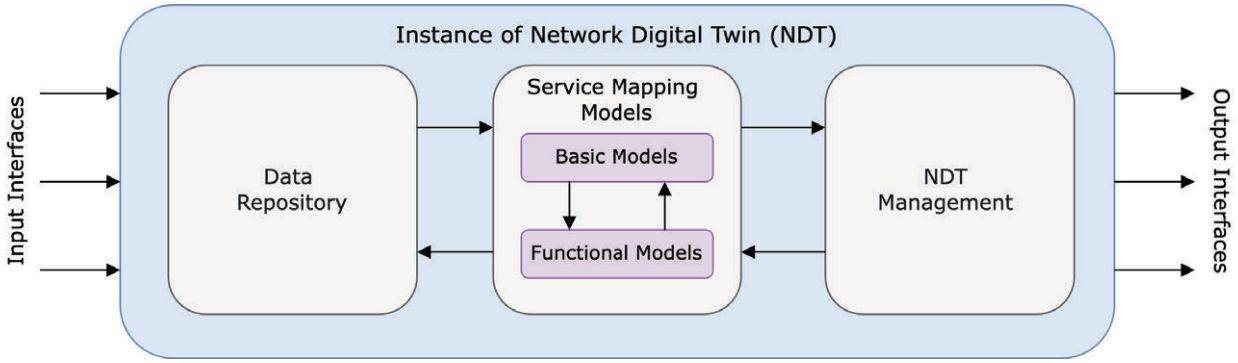


Figure 2.10: Reference Architecture of Network Digital Twin (based on [7]).

The real network can span across a single network administrative domain or multiple network administrative domains. The real network may include physical entities, such as hardware devices from physical network infrastructures, as well as virtual entities, such as virtual networks that employ Network Function Virtualization (NFV) and SDN paradigms. Together, these entities carry traffic and provide network services. All or a subset of the network elements in the real network deliver network data, either directly or via other systems, to the NDT through the appropriate interfaces. In addition, network elements may receive control inputs from operational systems that involve the NDT through specific interfaces. The interfaces might vary as a function of the specific NDT use case. The number of interfaces is also determined by specific NDT use cases.

As depicted in Figure 2.11, various external applications (e.g., network operations, administration, and maintenance (OAM)) can run efficiently over a NDT platform to implement either conventional or innovative network operations, at low cost and with less impact on the service on real networks. Network applications make requests to accomplish network management tasks that need to be addressed by the NDT. Such requests are exchanged through northbound interfaces using intent inputs, so that the requested network tasks are served on the appropriate NDT instances. Then, the NDT can perform network operations and management mechanisms within the real network as requested by service applications to configure the network accordingly or retrieve valuable operational and telemetry data. Finally, the result of the network operation or the collected network data can be delivered and exposed to the corresponding external application.

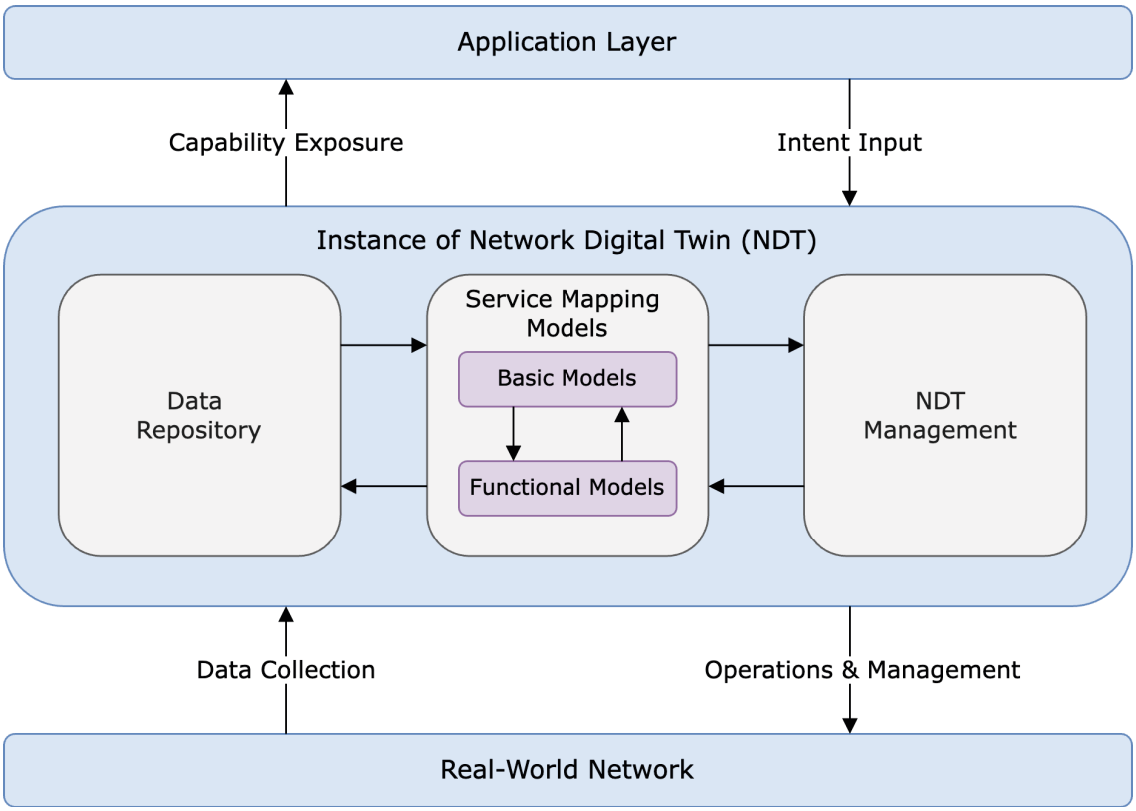


Figure 2.11: Application of NDT architecture for network operations and management.

2.2.4 NDT and Digital Map process

As determined in previous sections, an NDT can be described as a digital counterpart of a real network that captures its attributes, behavior, and interactions, and is continuously updated with the network's performance, maintenance, and health status data throughout its lifecycle [7]. In summary, it has also been determined that NDT is based on four core elements: i) historical and real-time data from real networks; ii) models for comprehensive representation of the data; iii) communication interfaces between the NDT and the real network, and between the NDT and external applications; and iv) an interactive, real-time

data mapping process between the NDT and the real network. These four pillars provide the information needed to implement the logic of the NDT to analyze, diagnose, emulate, and control the real network throughout its lifecycle using optimization algorithms, management methods, Machine Learning models, and expert knowledge.

In consequence, an NDT should be able to correlate all models and data to topological entities in the twin network. In this regard, the Digital Map[2.2] concept aims to provide a topological model representing the mapping between the real network and its NDT, facilitating a comprehensive and up-to-date view of the entire network [14], [15]. The Digital Map defines the key topological entities within the network (e.g., network, nodes, links, termination points), their role and particular properties in the network, and the relationships between the different entities. Figure 2.12 depicts the Digital Map process, highlighting in red the mapping for a router device and one of its network links along with the termination points, all within a network topology.

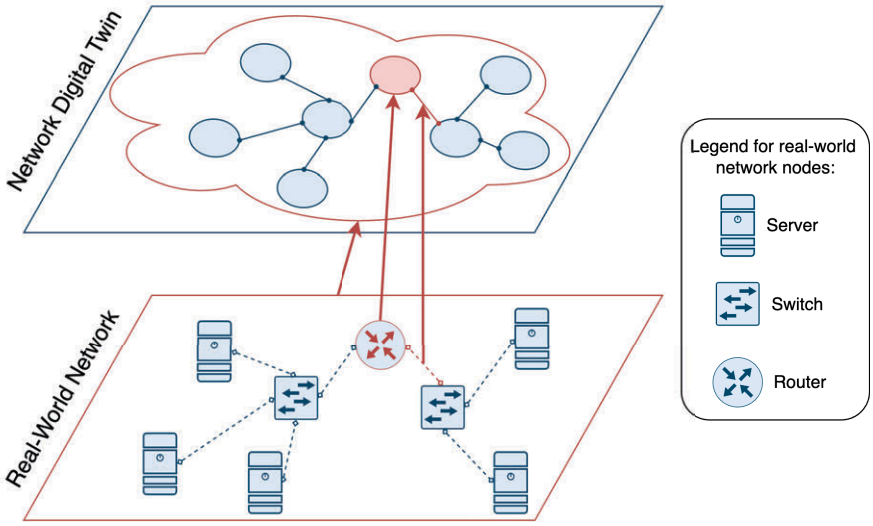


Figure 2.12: General insight on Digital Map process.

Connecting all models and data in the network management ecosystem, the Digital Map process acts as an enabler of closed-loop automation. This Digital Map process poses a set of functional requirements and suggests several application use cases, such as network design, traffic engineering, or building closed-loop network automation and NDT systems [15], [86]. The NDT solution presented in this thesis takes into account the aforementioned core elements and a digital mapping mechanism to integrate monitoring data from real networks into a knowledge-based interoperable model for efficient management purposes.

[2.2]During the IETF 121 meeting held in Dublin in November 2024, it was proposed to replace the term Digital Map with the acronym SIMAP (Service & Infrastructure Maps) to cover a broader scope of application of the term. The details of this new term are beyond the scope of this work, but are available in the following IETF draft: https://datatracker.ietf.org/doc/html/draft-ietf-nmop-simap-concept-03

2.2.5 Enabling technologies for building an NDT

Building an NDT requires several enabling technologies that address the key challenges and follow the reference architecture. These technologies include the following:

- Data collection: Data collection forms the foundation of an NDT by gathering information from heterogeneous sources using protocols such as SNMP [65], NETCONF [98], IPFIX [99], and network telemetry mechanisms [100]. Efficient data collection should be lightweight and tailored to the needs of NDT applications. Mechanisms such as Model-Driven Telemetry based on YANG data models [101] enable real-time subscriptions to telemetry data, while innovative techniques such as sketch-based measurement can be used to acquire more complex network data, such as network performance data. The heterogeneous and large-scale collected data must be stored in repositories, where it is processed, transformed, and made accessible through services such as fast search, batch processing and conflict resolution to support various network modeling and decision-making solutions.

- Network modeling: The basic network element models and topology models help generate a virtual model of the network according to the network element configuration, operation data, network topology relationship, link state and other network information. Then, network models are essential for simulating, monitoring, and optimizing network behavior. In this sense, the IETF SIMAP [86] proposal provides a foundation for multi-layer topologies. Small-scale networks can use simulation tools like NS-3 [102] and Mininet [103] or emulation tools like GNS3 [104] and EVE-NG [105]. However, these approaches face limitations in scalability and performance. For large-scale networks, mathematical and representational abstraction methods such as network calculus, Knowledge Graphs, and formal verification can improve the efficiency of building network models.

  However, a major challenge for the future is to improve the extensibility and accuracy of the models. Bottleneck structures provide an alternative modeling approach, offering a computational graph that efficiently represents network topology and flow relationships, enabling optimization without relying on resource-intensive simulations [106]. Network functional models, such as AI and Machine Learning techniques used to build data-driven digital network representations, further enhance the predictive capabilities of network models to realize the dynamic evolution of network performance evaluation and intelligent decision making [107]. To train these network models, it is possible to use diverse data coming from real networks, dedicated network testbeds, or network simulation tools, which allow cover a wide representation of potential scenarios that a network operator desires to mimic. In this context, deep learning techniques are of interest as they enable building accurate digital models of complex network environments [5]. However, challenges remain in improving generalization, interpretability, and real-time response.

- Network visualization: Network visibility is crucial for mapping real networks to their Digital Twin representation. Network visibility technology can help users understand the internal structure of the network and extract valuable information hidden within it. Visualization techniques, such as hierarchical, heuristic and force-directed layouts,

enhance the intuitive representation of network structures. The related topology data can be acquired using the data models defined in Request For Comments (RFCs) 8345 [108], 8346 [109], and 8944 [110]. Meanwhile, NDT systems can choose different data integration methods to realize the dynamic mapping of real networks and virtual representation or visualization. In this regard, query processing methods can express queries across diverse data sources by using a data virtualization approach and rewriting the queries in the native query language supported by the data sources, while keeping the data stored in the local data sources. Therefore, using standard representations and query processing techniques, NDT systems can dynamically interact with network data, providing interactive insights and improving network management.

- Interfaces: Interoperability within an NDT system relies on standardized interfaces. Three key interface categories exist: i) Network-facing interfaces, such as SNMP and NETCONF, facilitate communication between the physical network and its Digital Twin; ii) Application-facing interfaces, often RESTful APIs, enable integration with external applications; and iii) Internal interfaces, such as XMPP [111] or HTTP/3.0 [112], support high-speed data exchange within the NDT system. Standardized and open interfaces ensure interoperability and prevent vendor lock-in.
- Twinning management: Efficient twinning management ensures that Digital Twins contribute to network optimization, prediction, and automation. Twinning management involves orchestrating multiple twin instances, managing collaboration between NDT stakeholders, and detecting and resolving conflicts through policy-based mechanisms. Additionally, energy-efficient twinning strategies focus on reducing the power consumption of both network operations and the NDT system itself, contributing to sustainable networking practices.

In summary, these enabling technologies provide the essential building blocks for building an NDT, addressing data acquisition, modeling, visualization, interoperability, and management to create an effective and scalable digital representation of real networks.

2.2.6 NDT for data model-driven network management

The NDT paradigm has gained relevance in network management as a foundation for advanced emulation, optimization, troubleshooting, and upgrade planning. Once defined, an NDT can operate independently of the real network to perform real-time *what-if* analyses and tests on a virtual model without disrupting the real network [5], [113]. Unlike traditional emulation tools, NDT provides an interactive virtual-real mapping and a data-driven approach that enables high-fidelity simulations using real-time data, reduces the risk of configuration changes, and improves automated fault detection and repair [7]. These features make NDT an enabler for more efficient, reliable, and intelligent network management.

However, several challenges need to be addressed. The current amalgam of monitoring data sources makes it necessary to define standard interfaces and representations to be used for the NDT [5], [9]. Furthermore, both accuracy in the network data collection process and complete synchronization between the real network and its NDT are required [114].

Within the network management domain, the YANG data modeling language and network management protocols such as NETCONF or gNMI are the standards for managing configuration and obtaining telemetry data from network devices [61]. However, with the proliferation of YANG data models in the network industry, a new approach is needed to have a standard representation that encompasses not only the network data but also its semantics. In this regard, Knowledge Graphs are a promising solution that combines data integration with knowledge representation [67]. In addition, by linking different data sources, Knowledge Graphs enable intelligent automation systems to explore the causes and effects of network behavior [115]. Several works propose using Knowledge Graphs in NDT [85], [116], [117], but they are not based on formal and standardized data models for network management.

Network management systems also require a data integration technology capable of consolidating data from diverse sources into a unified view, providing a comprehensive data infrastructure with an interoperable interface for data consumption [118], [119]. In [84] we introduced CANDIL, a federated data fabric for integrating data from distributed systems, mainly focused on networking aspects. CANDIL defines an ontology that captures network concepts originally modeled with YANG, along with a reference architecture for ingesting and integrating data in a federated Knowledge Graph.

Therefore, a data integration approach simplifies data access and management by abstracting consumers from the location and underlying complexities of monitoring sources, such as protocols, data schemas, and encoding formats, becoming an ideal solution for NDT data representation. In this sense, the authors in [85] propose an NDT data integration solution based on a Knowledge Graph to provide topological, device, and network statistics information, but tightly dedicated to the management of SDN-based networks. Alternatively, NetGraph [16] presents a NDT data integration platform for the automated and intelligent operation and management of data center networks with network devices supporting YANG and non-YANG based monitoring mechanisms. However, when integrating data from network devices, NetGraph uses the data accessible in the configuration files as the knowledge baseline, regardless of the formal data models followed by the network data sources. The authors of NetGraph highlight the challenges of data modeling for NDT due to the vast amount of data sources and the need for accurate data mapping solutions between the real network and its Digital Twin representation. On the other hand, the authors in [120], [121] present a prototype NDT solution that covers the exchange of network management data between real networks and their Digital Twin emulations to partially mimic the status and operation of the real network, also based on YANG models and using the gNMI management protocol. However, the solution focuses on direct data exchange without providing formal representation and integration mechanisms for network management data.

Currently, various modern network automation solutions facilitate network operations and management while aligning with the principles of Network Digital Twins. Platforms such as NetBox [122], Nautobot [123], and InfraHub [124] serve as network source-of-truth systems, enabling efficient infrastructure modeling and automation. NetBox, originally developed for IP address management and data center infrastructure management, provides a robust data model but lacks built-in event-driven automation. Nautobot, a fork of NetBox, extends its functionality with enhanced programmability through a plugin ecosystem and dynamic

integration with automation frameworks, making it more suitable for adaptive network operations. InfraHub, a cloud-native alternative, prioritizes scalability and real-time data processing, supporting seamless orchestration across distributed environments.

While these platforms do not natively support direct interoperability with model-driven network management protocols such as NETCONF or gNMI, they can integrate with automation frameworks and SDN controllers that leverage these protocols. NetBox relies on external automation tools such as Ansible [125] or Terraform [126], while Nautobot enhances interoperability through its extensibility and plugin ecosystem. Meanwhile, with its focus on real-time automation, InfraHub facilitates integration with model-driven network management clients for telemetry and configuration tasks. As a result, these solutions can serve as foundational components for implementing Network Digital Twins, enabling automated, data-driven network management and optimization.

This Doctoral Thesis proposes a data integration solution for network data sources supporting YANG based on low-level analysis of standardized YANG data models. The solution integrates automatic data parsing and mapping mechanisms that allow the digital representation of data modeled in YANG into an interoperable NDT system for network management purposes [127]. To the best of our knowledge, this is the first Digital Twin proposal focused on the network management domain that proposes a high-fidelity data integration mechanism through the use of standard representations and communication interfaces, which is a crucial task to adequately reflect real network behavior in the NDT paradigm.

2.3 Emerging Trends in Network Operations and Management

Telecommunication networks are constantly changing and evolving, and their infrastructures are becoming increasingly complex. Continuous technological advances, exponential growth in the density of network devices and connected users, and the massive demand for services are driving the growth of dynamic and heterogeneous networks. Consequently, network operators and administrators face new operational and management challenges to ensure optimal network performance. This requires transforming the management and monitoring techniques currently used in telecommunication networks.

In this context, the concept of NDT is widely accepted as a promising technology and a key enabler for efficient control and management of modern telecommunication networks. The application of NDT in the field of network management has gradually attracted the attention of researchers and industry professionals. As explained in Section 2.2, NDT involves modeling the behavior of a real network through a virtual representation of the network that reflects its entire operational and management lifecycle, to optimize its performance without compromising the real network itself. Thus, NDT solutions promote advanced network operations and management mechanisms.

One of the challenges identified in the implementation of NDT solutions is to design efficient mechanisms for collecting and managing real network data based on standard communication interfaces and data representations that can be applied to the wide variety of monitoring

sources available. In turn, one of the main challenges in network monitoring is to deal with the heterogeneity of the available monitoring data sources, ranging from network device monitoring based on traditional management protocols such as SNMP [65], to network monitoring based on traffic flows using protocols such as NetFlow [128] and IPFIX [99], to monitoring information related to virtualized network functions for 5G/6G network services. In addition, each monitoring data source involves the use of different management protocols, as well as different formats and models to represent the data, which adds another layer of difficulty.

In the field of network management, the Model-Driven Telemetry (MDT) paradigm has gained attraction. MDT is designed for automatic network control based on the collection and manipulation of network configuration and operational data according to structured data models of monitored network devices. Therefore, MDT facilitates the automation of closed-loop systems that allow dynamic network monitoring to gain knowledge about network status and network configuration, ultimately enabling efficient and automated network management that can be applied through NDT solutions [129].

The goal of a closed-loop network system is that the network adapts dynamically to changes, proactively resolves issues, and maintains optimal performance without manual intervention. This closed-loop system is illustrated in Figure 2.13, where the continuous feedback loop between the monitoring and configuration workflows is orchestrated by a model-driven network management system (i.e., Model-Driven NMS) that acquires the relevant knowledge for automated control of the managed network.

Model-driven network management based on standard YANG data model solutions and network management protocols such as gNMI or NETCONF is gaining momentum in the telecommunications industry. YANG is a data modeling language that allows the definition of data models for network configuration and operational status. However, due to the proliferation of YANG data models in the industry, defined by different network equipment vendors, standardization organizations such as IETF, or consortia such as OpenConfig, there is nowadays an open research line studying solutions that allow the collection of data modeled according to formal and standard representations.

Autonomous network management requires the collection and aggregation of information from a variety of data sources that need to be combined to support an autonomous decision-making process for network operations and management. A key aspect of this is the management of the metadata that characterizes the network data sources and the collected network monitoring data to gain detailed knowledge of the network status. This requires data integration technologies that can combine data from heterogeneous sources to provide a complete data infrastructure with an interoperable interface to consume the data. Ideally, data access and management should be simplified by abstracting consumers from the underlying complexities of data sources, such as location, access protocols, data models, and encoding formats, which would facilitate the acquisition and representation of network data in NDT solutions.

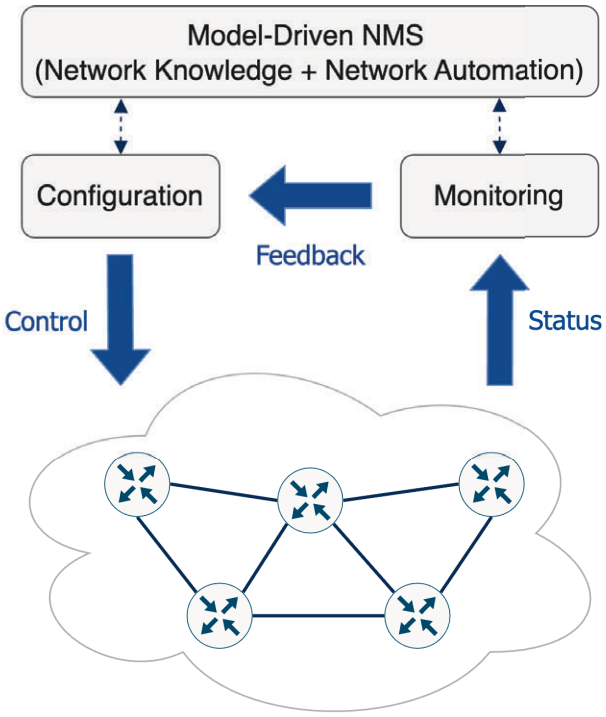


Figure 2.13: Closed-loop paradigm applied to network operations and management.

2.3.1 Model-driven network management and telemetry

Model-driven network management provides a mechanism for managing remote data sources whose information is structured according to formal and standardized data models (e.g., operational, configuration, and telemetry data from network devices) and for transmitting the information in a structured format to remote destinations for monitoring and analysis. As it turns out, model-driven management is based on the idea of using modeling languages to formally describe data sources, defining the semantics, syntax, structure, and constraints of management objects.

Data model–driven management simplifies automation and specifies that telemetry must be data model-driven, a process known as Model-Driven Telemetry (MDT) [61]. MDT is specifically designed to manage data from network equipment in real time and is critical to continuous and autonomous operation of the network itself. Increasing the frequency of collection of detailed data available from network devices through telemetry leads to improved efficiency of monitoring capabilities, providing greater insight into the state of the network, and therefore improved problem detection and resolution. The MDT mechanisms can help achieve better performance across the network infrastructure, including more efficient bandwidth utilization, comprehensive risk assessment and control, and increased scalability.

Network industry efforts, such as the IETF Telemetry Framework [100], clarify that network telemetry is all about network data that: i) can help provide insight into the current state of the network, including network devices, forwarding, control, and management planes; ii) can be generated and collected through a variety of techniques, including but not limited to

network instrumentation and measurements; and iii) can be processed using a wide variety of data analytical techniques for purposes ranging from service assurance to network security. Therefore, telemetry transforms the monitoring process into a data analytics proposition, enabling the rapid and efficient extraction and analysis of massive amounts of data to improve decision-making. MDT greatly facilitates the automation of the closed-loop paradigm, in which events extracted from network devices are fed back to the network for appropriate reconfiguration. Consequently, MDT is gaining attention from industry and vendors as a monitoring mechanism based primarily on the YANG data modeling language.

YANG data modeling language

YANG[2.3] is now a widely used data modeling language in standardization organizations, such as the IETF, and consortia, such as OpenConfig, and in the network management solutions these institutions propose, including the NETCONF, RESTCONF, and gRPC Network Management Interface (gNMI) network management protocols. YANG is a flexible and extensible data modeling language, rich in semantics, that structures data according to a tree hierarchy in the so-called YANG modules. In addition, YANG is independent of the data encoding format and transport protocol.

The different components of data model-driven management based on the YANG data modeling language are shown in Figure 2.14. Once YANG models are specified and deployed on devices, a client (e.g., a network management system, application, orchestrator, controller, etc.) can select a particular encoding format (i.e., XML, JSON, Protobufs, etc.) and transport protocol (i.e., NETCONF, RESTCONF, or gNMI) to perform this type of management.

Based on the encoding or serialization format and network management protocol selections, an application can generate behavior or programming code to access the YANG objects on managed devices (e.g., in Python, Java, C++, Go, or basically any programming language). In addition, platforms such as network orchestrators can deliver service automation through multiple protocols and encodings, leveraging code generation in their automation.

YANG is a language for modeling state data, configuration, Remote Procedure Calls (RPCs), and notifications that are transported over different network management protocols. The first specification (i.e., specification 1.0) of YANG dates back to 2010 and is documented in RFC 6020 [130]. Six years later, in August 2016, the specification 1.1 of YANG is completed, collected in RFC 7950 [101], which corrects some errors and ambiguities of the first version.

The original intent of the specification of YANG as a data modeling language is related to the NETCONF protocol. In fact, the document that contains the first YANG specification is called *"YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)"* [130].

In terms of usage, a specification written in the YANG language is known as a YANG module, and a set of such modules is known as a YANG model. A YANG model focuses on specifying the data that a client manipulates and observes using standard operations.

[2.3]As a curiosity, YANG is not an acronym, and its initials are jokingly interpreted as *Yet Another Next Generation (data modeling language)*.

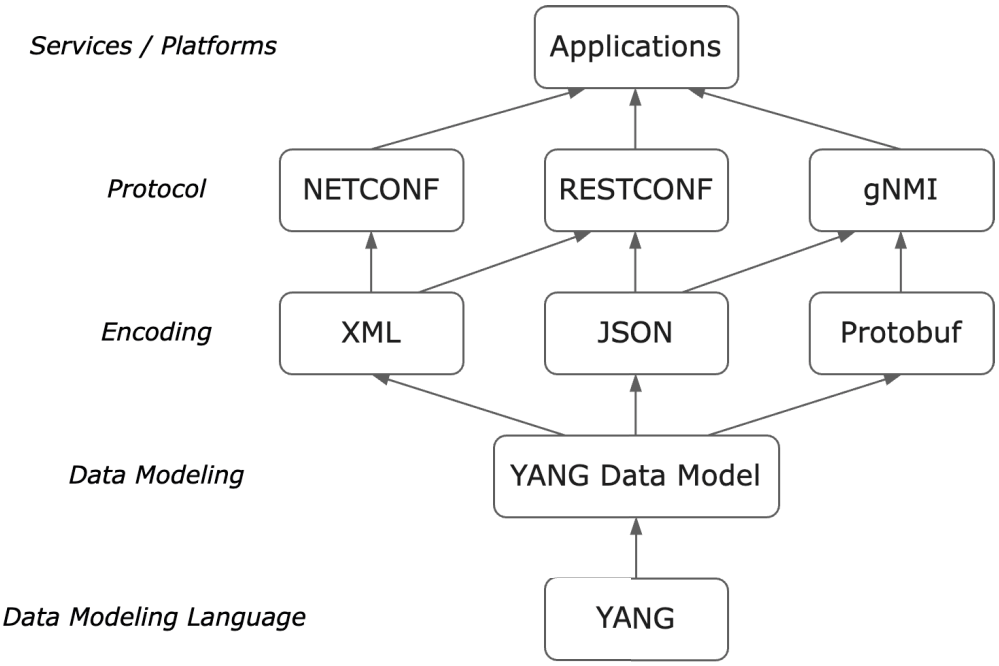


Figure 2.14: The data model–driven management components (based on [61]).

Each YANG module defines the hierarchy of data that can be used by protocol operations to fully describe the data sent from clients to servers. The way YANG models the hierarchical organization of data is through a schema tree in which each node has a name. YANG provides a concise description of each of these nodes and the relationships between them.

In addition to models and modules, YANG also structures data using submodules. These modules can import definitions from other external modules as well as from other submodules. The hierarchy can be augmented by limiting the cases where this is desired by specifying conditions to be met.

Since the description of YANG is certainly abstract, perhaps an example like the one shown below can help to better understand its operation and syntax. A YANG module consists of three different parts:

1. Headers: Describe the module and provide necessary information about the module itself. They correspond to the lines: *namespace*, *prefix*, *organization*, *contact*, and *description*. The *namespace* uniquely identifies the module, so there cannot be two identical namespaces for different modules.
2. Revision: Provides information about the date on which a new version of the YANG model was released.
3. Definition: The main part of the module where the data model is defined.

```
module network-model {
  namespace "urn:example:network";
  prefix "net";
  organization "Example Corp";
  contact "support@example.com";
  description "A generic YANG model for network configurations.";

  revision "2025-02-27" {
    description "Initial version of the model.";
  }

  container network {
    description "Network configuration.";

    leaf network-name {
      type string;
      description "Name of the network.";
    }

    leaf enabled {
      type boolean;
      default true;
      description "Indicates whether the network is enabled.";
    }

    list devices {
      key "device-id";
      description "List of devices in the network.";

      leaf device-id {
        type uint32;
        description "Unique identifier for the device.";
      }

      leaf device-type {
        type string;
        description "Type of device.";
      }

      leaf-list ip-addresses {
        type string;
        description "List of IP addresses assigned to the device.";
      }
    }
  }
}
```

In addition, the example illustrates the four main node types employed in YANG data models:

- *Container* nodes: Used to group related nodes in a single tree. A container node can include multiple nodes of any type (i.e., *leaf*, *leaf-list*, *container*, and/or *list* nodes).

- *Leaf* nodes: Define a simple scalar variable of a particular type. These types of nodes can only consist of a particular value and cannot contain child nodes.
- *Leaf-list* nodes: a sequence of leaf nodes.
- *List* nodes: Define a sequence of list entries, where each entry follows a defined structure and is uniquely identified by a key value. A *list* node can have any number of child nodes of any type (i.e., *leaf*, *leaf-list*, *container*, or *list* nodes).

The YANG language allows for specifying within a YANG data model which data is meant for configuration and which data represents the operational state in a YANG server (e.g., a network device). YANG data nodes marked as configuration represent the data that a YANG client can write to apply configuration to the YANG server. YANG data nodes marked as operational state data represent data that a YANG client can only read from the server. To store these different types of data, YANG servers implement databases called YANG datastores.

In the typical YANG-based solution, the client and the server are driven by the content of the YANG modules. The server architecture includes module definitions as metadata that is available to the model-driven network management protocol engine as server capabilities [61], [131]. This engine processes incoming requests, uses the metadata to parse and verify the request, performs the requested operation, and returns the results to the client, as shown in Figure 2.15.

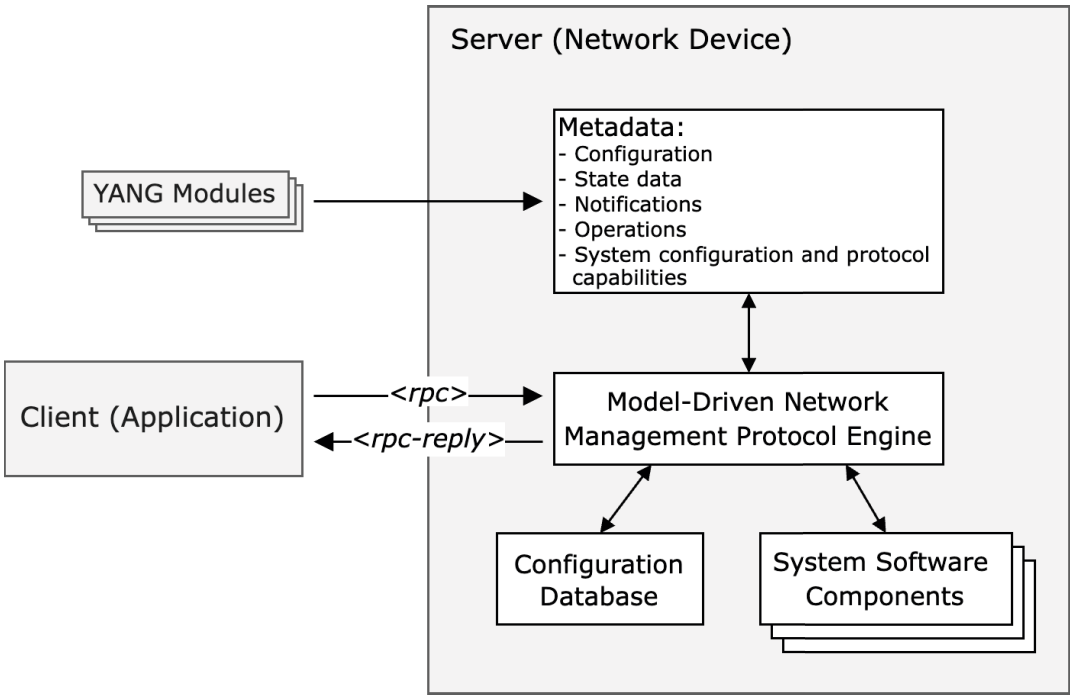


Figure 2.15: The server architecture in a model-driven network management system (based on RFC 6244 [131]).

YANG modules must be defined to model the specific problem domain. These modules are then loaded, compiled, or coded into the server. The typical client-server interaction sequence

is summarized below. The client opens a NETCONF session with the server and exchanges messages to identify supported capabilities and YANG modules. Among the capabilities, the server reports the protocol version, the supported encoding format, available YANG models, as well as additional features (e.g., datastore support or the possibility of XPath filters for operations). The client then sends a YANG-defined operation inside a *<rpc>* command, encoded in the particular format supported by the server's network management protocol. The server receives, validates, and executes the operation, updating the datastore if necessary. Next, the server generates and sends a response in a *<rpc-reply>* message, including any requested data or errors. Finally, the client receives, parses, and processes the response as needed, and closes the session established with the server.

Note that there is no requirement that the client or server process the YANG modules in this way. The server can hardcode the content of the data model rather than using a generic engine to process the content. Alternatively, the client can be tailored to the specific YANG model, rather than being driven generically.

There are currently three approaches to defining YANG models in the industry:

- Standards Development Organizations (SDOs): SDOs are responsible for developing technical standards to meet the needs of a specific group. Standardization processes undergo rigorous review by multiple people, which makes the time to complete a model specification long. The first organization to standardize a data model, and still very important today, is the IETF. Later, the Institute of Electrical and Electronics Engineers (IEEE) and the International Telecommunication Union (ITU) also began to define various data models.
- Consortia, forums, and open-source projects: They are responsible for defining data models related to specific use cases and proposing complete solutions. They focus on defining the use cases that the industry needs and that are not covered by any standards body. By not having to go through the rigorous processes that SDOs have, the definition of data models is faster and more agile. In this group, we can find Opendaylight, MEF Forum, Broadband Forum, OpenConfig, and Open ROADM. Among all of them, OpenConfig is the one that has the most relevance in the industry.
- Vendors: In this case, equipment vendors define their own data models based on their proprietary implementations. This makes interoperability between different vendors difficult.

Figure 2.16 shows the impact of the *ietf-interfaces* YANG data model, which has been standardized by the IETF in RFC 8343 [132] for the management of network interfaces, in different organizations and vendors that define their own YANG data models that extend the *ietf-interfaces* YANG data model[2.4]. Due to the heterogeneity that exists in today's networking industry, it is necessary to rely on standardized and open data models (i.e., data models regulated by SDOs, consortia, or open-source projects) that facilitate access to operational and configuration data of network devices. This avoids the use of proprietary data models and facilitates network interoperability.

[2.4]This YANG impact analysis was provided by the YANG Catalog utility: https://www.yangcatalog.org/

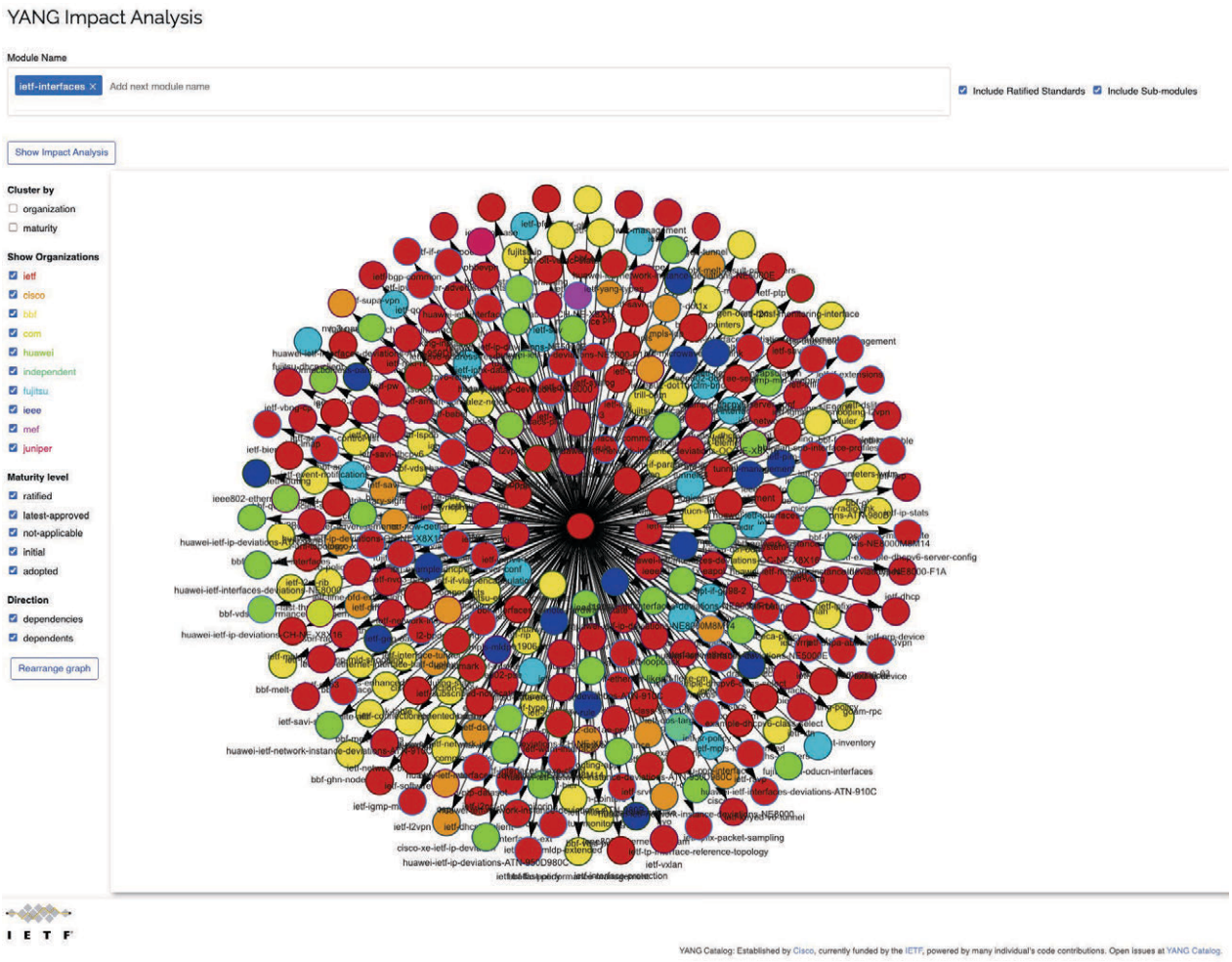


Figure 2.16: Impact analysis for *ietf-interfaces* YANG data model.

Model-driven network management protocols

As a quick introduction before diving into the different protocols, Table 2.2 provides a comparison of the most common protocols used in data model-driven management: NETCONF, RESTCONF, and gNMI. And Table 2.3 provides the common RPC operations supported by these model-driven network management protocols.

The NETCONF protocol, specified by the IETF in RFC 6241 [98], is one of the main management protocols for YANG-modeled network devices. NETCONF defines a set of operations that a client, often in the role of a network manager, can send to a NETCONF server, which is typically the managed network device. NETCONF uses RPC mechanisms to perform management operations. The most common management operations that NETCONF allows include: i) monitoring the operational status of the managed devices; ii) updating the configuration of managed devices; and iii) subscribing to notifications about changes in the operational status of the managed devices.

Table 2.2: Comparison of NETCONF, RESTCONF, and gNMI.

	NETCONF	**gNMI**	**RESTCONF**
Message and Payload Encoding	XML	JSON or Protobuf	JSON or XML
Operation Semantics	NETCONF specific. Network-wide transactions.	gNMI specific. Single-target, single-shot, sequenced transactions.	RESTCONF specific, based on HTTP. Single-target, single-shot transactions.
RPC Mechanism	NETCONF specific, based on XML	gRPC	REST style
Protocol Stack	SSH / TCP / IP	HTTP2 / TLS / TCP / IP	HTTP / TLS / TCP / IP

Table 2.3: RPC basic operations in NETCONF, gNMI, and RESTCONF.

Operation	**Description**	**NETCONF**	**gNMI**	**RESTCONF**
Create	Creates a new resource or configuration entry.	`<edit-config>`	`SetRequest`	`POST`
Read	Retrieves configuration or operational state data.	`<get>`, `<get-config>`	`GetRequest`	`GET`
Update	Modifies existing configuration data.	`<edit-config>`	`SetRequest`	`PATCH`
Delete	Removes configuration data.	`<edit-config>`	`SetRequest`	`DELETE`
Subscribe	Streams telemetry notifications.	`<create-subscription>`	`SubscribeRequest`	Not explicitly supported*

* RESTCONF does not provide a native `SUBSCRIBE` operation, but it supports event stream notifications via `GET` using Server-Sent Events (SSE), as defined in RFC 8650 [133].

In the requests and responses of the different RPC calls, the NETCONF protocol transports configuration and operational data in the XML encoding format. NETCONF supports network-wide transactions, allowing multiple devices to be configured in a single operation. Its rollback mechanism ensures that if a change fails on one device, all changes across the network are reverted, maintaining consistency. This makes it ideal for large-scale network configurations where atomicity and reliability are critical.

In addition, the original RFC for NETCONF has been extended to support the Network Management Datastore Architecture (NMDA) [134], [135] which defines a unified architecture for defining YANG data models. The idea behind this architecture is that a single YANG data model can have multiple instantiations in different YANG datastores depending on the purpose of the data (e.g., configuration or operational). When interacting with YANG servers that support NMDA, network management protocols such as NETCONF can select a particular datastore on the YANG server. Therefore, YANG clients see a single YANG data model, whose configuration or operational state data can then be selected by specifying the target datastore. NMDA-compatible devices can implement additional datastores that contain YANG data structured with the same YANG data model, such as *candidate* (i.e., configuration data stored in the device but not applied) or *startup* (i.e., configuration data stored in the device that will be applied upon the next reboot).

Before NMDA came on the scene, network management protocols could not select a particular YANG datastore when interacting with network devices. To overcome this limitation, YANG data models would follow a split layout to enable identifying the datastore that contains the data. These data models are divided into two branches: one containing configuration data and the other one containing the operational data.

Then, with NMDA, NETCONF can interact with specific datastores, improving the YANG data management. However, traditional retrieval mechanisms still relied on polling, which can be inefficient for monitoring dynamic network changes. To address this, YANG Push was standardized in RFC 8639 [136] and RFC 8641 [137], enabling subscription-based streaming of YANG datastore updates. This allows network devices to proactively send configuration or operational state changes to subscribers, either periodically or upon changes. Unlike traditional polling mechanisms, this approach reduces network overhead and ensures near real-time synchronization of the network state. However, implementing YANG Push is challenging and can be resource intensive, particularly on constrained devices. To address this, the YANG Push Lite proposal [138] introduces a simplified subscription mechanism with reduced complexity, making it more suitable for lightweight deployments. Combined with NMDA, these advancements enable clients to subscribe efficiently to specific datastores and receive structured data in a unified way.

The gNMI (gRPC Network Management Interface) [139] is another protocol for managing YANG-modeled devices that originated from the OpenConfig consortium [140], a group of network operators led by Google. OpenConfig is an open-source project built around a community of network vendors, network operators, and cloud providers. The goal of the project is to define and implement a common, vendor-independent software layer for managing network devices. Another contribution of OpenConfig is the creation of standard YANG data models for network management.

While OpenConfig continues to evolve the definition of YANG models, Google developed gNMI, a unified management protocol to transmit telemetry and manage the configuration of network devices through gRPC-based remote procedure calls. Then, gNMI is the management protocol, and gRPC is the underlying RPC framework. Messages for management operations using the gNMI protocol are encoded in formats such as JSON and Protobuf. Like the NETCONF protocol, gNMI allows subscription operations to receive notifications of operational information about the device. These subscriptions can be periodic or change-based.

Currently, there are many vendors of network devices that support the gNMI protocol, such as Arista Networks, Cisco, and Nokia. Although gNMI also uses YANG, it does not implement the NMDA architecture in the same way as NETCONF. Instead, it provides management methods better suited for real-time telemetry and batch configuration via gRPC. Unlike NETCONF, gNMI allows multiple configuration changes or telemetry requests to be processed in a single transaction, reducing session overhead and improving efficiency, but only for a single device at a time. Therefore, gNMI does not support network-wide transactions like NETCONF, but enables the ordered execution of transactions.

Meanwhile, the RESTCONF protocol [141] is another network management protocol that allows access to data defined in YANG. The idea of RESTCONF is not to replace NETCONF, but to provide an HTTP-based alternative that follows the principles of REpresentational State Transfer (REST). In the case of RESTCONF, RPC operations follow the REST style and are implemented using the Hypertext Transfer Protocol (HTTP) methods GET, POST, PUT, PATCH, and DELETE. Therefore, RESTCONF is a transport protocol based on REST that operates on a per-resource basis using HTTP methods, supporting both JSON and XML encoding formats. Like NETCONF, RESTCONF supports the NMDA architecture for manipulating datastores on servers and a set of operations for accessing those datastores [142].

In RESTCONF, each request targets a single device and a specific resource, making it simpler but lacking built-in transactional mechanisms. However, it is a suitable alternative for lightweight API-driven network management tasks. Therefore, a key difference in RESTCONF is that it establishes a single independent connection between the client and server per transaction, without maintaining an active session, so it is stateless. In contrast, in NETCONF, clients typically connect and manipulate the datastore with a series of operations until they validate those operations at a given moment, which are session-oriented and stateful. This behavior is not possible with RESTCONF, as it would require the server to store a particular state of the client, which is not compatible with REST principles.

2.3.2 Intent-Based Networking for network automation

As network management protocols continue to evolve, interesting use cases continue to emerge. In the near term, network automation is a prime candidate to take advantage of the programmatic interfaces exposed by modern network devices. Network automation is not just about automating the configuration of network devices using programmatic management protocols, but also about streamlining the network management process to collect operational

data for orchestrating automated diagnostic, remediation, and troubleshooting tasks[2.5] [143]. In this sense, model-driven network management simplifies automation and specifies that management, configuration, and telemetry processes must be driven by data models [144].

Another trend for network automation is Intent-Based Networking (IBN), which can be summarized as the ability to manage the network through a business language (i.e., the intent) without focusing on low-level details. IBM goes hand-in-hand with autonomous automation through a closed-loop orchestration system capable of responding to events [143].

Historically, managing networks in a prescriptive mode by focusing on detailing the necessary configuration steps was common. Contrary to the prescriptive approach, the intent-based approach focuses on the high-level business policies to accomplish Service Level Agreements (SLAs) and on what is expected from the network. In other words, the prescriptive approach focuses on *how*, while the intent-based approach focuses on *what* [61].

Therefore, intent-based means that users can input their abstract intent to the network, instead of detailed policies or configurations on the network devices. RFC 9315 [145] clarifies the concept of *intent* and provides an overview of Intent-Based Networking functionalities. The key characteristic of an IBN system is that user intent can be automatically ensured by continuously adjusting policies and validating real-time situations, with networks constantly learning and adapting.

Therefore, IBN adds the most value to network management by continuously learning, adapting, and optimizing, based on a feedback loop or closed-loop system, as shown in the following steps:

1. Decomposition of intent (the *what*) to network configuration (the *how*).
2. Network configuration, once the *what* is identified. Based on the data model–driven management and a good set of YANG data models, a network manager role translates the service model (e.g., YANG model for the delivery of Layer 3 VPN service, as reflected in RFC 8299 [146]) into a series of network device configurations.
3. Monitoring with model-driven telemetry mechanisms provides a real-time view of the state and configuration of the network. Any fault, configuration change, or even behavior change is directly reported to the network manager role.
4. A data analytics system correlates and analyzes the impact of the new network state for service assurance purposes, isolating the root cause issue even before the degradation occurs. From there, the next network optimization is deduced, almost in real time, before going back to step 1 to apply new optimizations resulting from new intents.

This constant feedback loop, described in four steps, is the foundation for autonomous networks that constantly learn and adapt. It allows moving away from reactive network management, where a network fault or a customer call triggers the troubleshooting, to constant

[2.5]It is important to distinguish between automation and orchestration terms. On the one hand, network automation refers to the automation of a single task or a small number of closely related tasks associated with network management operations. Orchestration, on the other hand, refers to the transformation of automated tasks into a broader workflow.

monitoring focused on the SLAs. The combination of predictive analytics and AI, combined with continuous learning and adaptation, is the key enabler of IBN. From here, the next logical step is the self-healing networks. Then, the network automation can enforce steps 2 and 3 in order to realize Intent-Based Networking. This closed-loop system with IBN applicable to the network management context is shown in Figure 2.17.

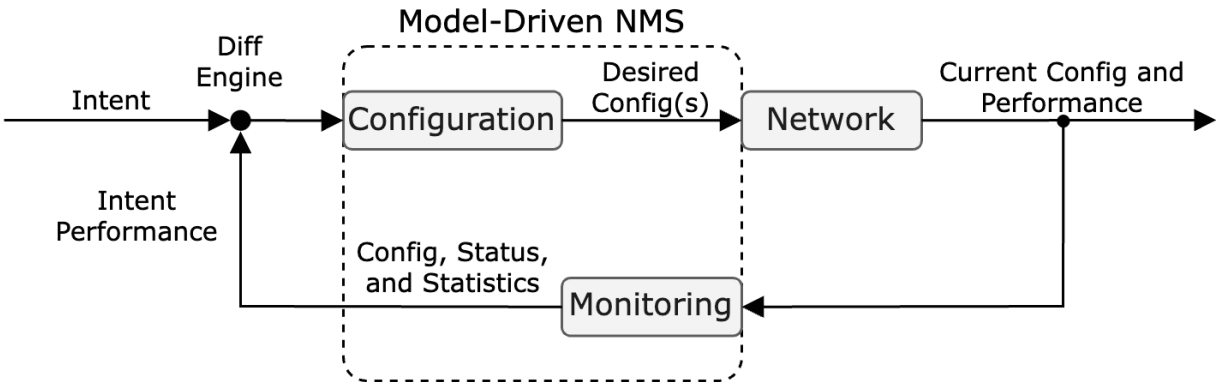


Figure 2.17: Intent-Based Networking on a closed-loop system for network management purposes.

Network operators would like to concentrate on the configuration of the network as a whole, rather than on individual network devices. Intent is then the high level of abstraction for managing the networks, where a particular intent is broken down into different network configurations. The Network Management System (NMS) runs the network-wide configuration transactions across the relevant devices by using model-driven network management mechanisms that allow to push and pull configurations to and from the network.

In order to close the loop, each of the following steps is equally important. When the NMS monitoring function reads the state of the network, it leverages model-driven telemetry capabilities to separate the configuration from other data related to operational status and statistical counters. The *Diff Engine* component then compares the intent performance with the original intent to see how well the current strategy for implementing the intent works, and it also compares the actual network configuration to the desired one. If a change in strategy is required or desired from the original intent, the NMS computes a new desired configuration and sends it to the network. It should be noted that the resulting network configuration transactions must describe a set of changes that, taken together and applied to the current configuration, must be consistent and make sense. In other words, they are consistent configurations, not atomic sequences of changes.

This paradigm shift enables the realization of full network automation to the required level in Zero-touch network Service Management (ZSM) architectures [147], [148]. The ZSM architecture proposed by ETSI defines a network with properties such as self-healing, self-configuration, and self-optimization. To achieve these qualities, networks must constantly sense and interpret the changes required to maximize performance. Much of the network management is end-user and application driven, so a fully autonomous network management system should be able to interpret the user's intent and configure the network accordingly.

Intent-Based Networking can be seen as the end goal of network automation, and a compulsory step to reach it is to close the loop with model-driven telemetry for service assurance. The big question at this point is how do we evolve existing networks towards this vision of intent-based networks, self-healing networks, or self-driving networks.

To answer this big question, Benoit Claise, chair of the IETF Operations and Management Area (OPSAWG) and the IETF Network Management Operations (NMOP) working groups, defines a hierarchy of needs to achieve Intent-Based Networking [149]. Each level of the hierarchy builds upon foundational capabilities and represents a prerequisite for the next level.

At the base is data model-driven management, which unifies configuration and operational telemetry to establish network observability. Building on this, the next level focuses on analytics to support troubleshooting, emphasizing the reduction of Mean Time To Detect (MTTD) and Mean Time To Repair (MTTR). Since issues must be accurately and promptly identified before they can be resolved, reducing MTTD is often more critical than MTTR.

The following level introduces assurance automation, which involves monitoring the health of network components and features in order to streamline diagnostics by excluding fault-free elements. A higher level incorporates service context, linking technical issues with affected service instances and clients. This step is an essential step for service-centric assurance and is formalized in RFC 9417 [150], which defines an architecture for service assurance in IBN.

At the top of the hierarchy lies the full realization of IBN, in which automation is driven by intent and made possible through closed-loop control and the intelligent processing of telemetry.

IBN can be envisaged in an NDT context to show how NDT improves the efficiency of deploying network automation processes. Several rounds of adjustment and validation can be emulated on the NDT platform instead of directly impacting real networks during the testing phase. Therefore, NDT can be an important enabler platform for implementing IBN systems and fostering their deployment.

2.3.3 Advancing network automation with Artificial Intelligence and Knowledge Graphs

Emerging network automation systems, including closed-loop approaches based on the IBN paradigm and AI solutions leveraging Large Language Models (LLMs), are expected to significantly improve network troubleshooting and service assurance capabilities.

The development of domain-specific LLMs and the incorporation of advanced AI techniques such as federated learning and reinforcement learning can further enhance the capabilities of network systems. In this regard, there are recent research activities dedicated to the analysis and use of LLMs in the context of network management. The authors in [151], [152] propose LLMs to realize Intent-Based Networking solutions that pursue fully automated network intelligence. Other works [153], [154] propose the use of LLMs to facilitate and increase the accuracy and effectiveness of network management tasks through the use of AI assistants that employ Retrieval-Augmented Generation (RAG). RAG enhances the information retrieval

capabilities of Generative AI models by providing source information from external data stores to augment the generated responses [155]. Some relevant efforts are also being made in standardized bodies such as ETSI ZSM to provide information on enablers and capabilities to pursue AI-based network and service automation [156].

However, AI often lacks the domain-specific knowledge required for complex networking tasks. Although LLMs can interpret input and generate responses, they rely on probabilistic associations learned from large general-purpose datasets, which can lead to linguistically accurate but contextually inappropriate responses for specific networking situations. In essence, LLMs do not possess the in-depth understanding of network infrastructure, protocols, and dynamics that comes naturally to experienced human operators [157].

Specifically, the authors in [158] propose the use of Generative AI for network configuration using NETCONF and various LLMs. They emphasize that LLMs have deficiencies in understanding YANG modeled data and in solving complex network management and configuration tasks, highlighting the need for new research activities in training LLMs extensively in the YANG language. In addition, there are some initial efforts for the integration of Generative AI platforms in network operating systems to enable network management based on natural language processing capabilities. In this regard, Nokia has implemented an LLM-based tool for network management that offers personalized assistance [159], [160]. It effectively summarizes the status of the network and identifies configuration errors, which helps troubleshooting. However, the LLM has been tailored only to the Nokia's SR Linux network operating system [161] and currently focuses more on guidance rather than implementing network automation tasks.

To address these limitations, leveraging and combining the strengths of Generative AI and Knowledge Graph solutions paves the way for creating more accurate, efficient, and reliable AI-driven tools with context-aware capabilities in the particular application domain [162], [163]. In this regard, there are some initial proposals, such as the work in [153], which explores the combined use of LLMs and Knowledge Graphs in the context of network management, highlighting the need for a detailed analysis of more advanced Knowledge Graph solutions to accomplish complex network management tasks.

In addition, the authors in [164] propose MAYA, a solution to automatically perform the alignment of YANG models from different vendors and organizations in order to find semantic similarities between the concepts modeled on YANG. The authors also highlight the potential of applying Knowledge Graph solutions to represent YANG-modeled data, taking into account that the YANG models are interconnected with each other [165].

The achievement of network automation might be possible if and only if we provide the network knowledge as reflected at the beginning of Section 2.3, based on an additional semantic layer, with an ontology of the business and network logic, organized as a Knowledge Graph [166]. This raises the need to adapt the notion of the Semantic Web and the Knowledge Graph to the networking world. Standardization bodies such as the IETF are already beginning to work in this respect [167], [168].

The application of Knowledge Graphs for network management has been widely explored in areas such as fixed networks, mobile networks, or SDN-based control, focusing on operations

such as service assurance or capacity delivery [85], [115]–[117]. Nevertheless, none of these approaches have followed standard and well-known methodologies for ontology development, thus producing isolated ontologies tailored to specific use cases. The results are low-grade ontologies that lack interoperability with other existing ontologies and miss the recognition by a broader community, therefore, limiting their adoption in other use cases in network management. Other approaches like NORIA [169] focus on risk management by detecting anomalies and incidents in the network based on a new ontology and the implementation of a Knowledge Graph. In this case, the NORIA ontology has been developed following the well-known Linked Open Terms (LOT) methodology [170], which ensures quality aspects such as ontology reuse. However, when formalizing core concepts in network management, such as network device or network interface, NORIA does not align with the standard YANG data models developed within the IETF.

In [84] we introduced CANDIL, a Context-Aware Network Data Integration Loom. CANDIL proposes a federated data fabric for integrating data from distributed systems, mainly focused on networking aspects. The CANDIL Data Fabric builds on technologies and standards like the NGSI-LD standard to implement a Knowledge Graph that enables data integration. To achieve this, CANDIL follows a standard methodology to develop an ontology that captures network concepts and their mapping with the data. CANDIL is a first approximation on how to formalize a Knowledge Graph solution integrating concepts extracted from the network management domain that have already been standardized by the IETF in the form of YANG data models. Inspired by this proposal, this Doctoral Thesis presents a Network Digital Twin solution based on a knowledge system that allows the integration of network management data modeled by YANG data models.

Chapter 3

Modeling an NDT Solution for Network Operations and Management

This chapter introduces the Network Digital Twin solution proposed in this Doctoral Thesis, which follows a data model-driven approach for the sake of achieving autonomous network management. First, design guidelines are outlined for accommodating the key elements and architecture of the NDT using the NGSI-LD standard. And second, a methodology is provided for addressing these guidelines in order to build an NDT system for network operations and management.

Table of Contents

3.1 Designing an NGSI-LD Compliant NDT Solution 70

3.1.1 Alignment of NDT elements and architecture with the NGSI-LD standard 70

3.1.2 NGSI-LD Digital Twins for network operations and management . 72

3.2 A New Methodology for Modeling an NDT based on the NGSI-LD Standard 74

3.2.1 Data integration using the NGSI-LD standard 75

3.2.2 YANG to NGSI-LD translation conventions 79

3.2.3 Descriptive NDT architecture 82

3.2.4 Network automation using the Descriptive NDT and the NGSI-LD standard 85

3.3 Overview of the NDT Proposal 96

3.1 Designing an NGSI-LD Compliant NDT Solution

In this section, we present how to design an NDT solution that is consistent with the IETF approach reviewed in Chapter 2 by using the NGSI-LD standard. First, we provide design guidelines for making the IETF's vision of NDT components and architectures compatible with the NGSI-LD standard. And second, we provide some insights on how to approximate the Digital Twin classification provided by the NGSI-LD standard for network operations and management purposes in the context of NDT.

3.1.1 Alignment of NDT elements and architecture with the NGSI-LD standard

This subsection explains how to make the NDT core elements and the reference architecture proposed by the IETF compatible with the NGSI-LD standard. The interpretation of the core elements of NDT to be compliant with the NGSI-LD standard, which are depicted in Figure 3.1, can be determined as follows:

- Data: Configuration, operational, and telemetry data retrieved from YANG data models are instantiated as NGSI-LD modeled data within the NDT instance by following data integration approaches.
- Models: NDT data are modeled according to the NGSI-LD meta-model by building domain-specific models compliant with the source YANG data models (e.g., building the NGSI-LD model for the network interface management domain).
- Interfaces: There are two main interfaces: 1) MDT network management protocols, such as NETCONF or gNMI, for the interaction between NDT and real network; and 2) NGSI-LD API for the interaction between external applications and NDT and for the interaction of NDT instance subcomponents.
- Mapping: The mapping mechanisms map the network monitoring data to a property graph data model based on the NGSI-LD standard. There are two main mapping mechanisms to accomplish this process: 1) mapping between YANG and NGSI-LD data models using schema matching techniques; and 2) mapping between YANG and NGSI-LD modeled data using schema mapping techniques.
- Logic: By coordinating the above NDT elements, the automation logic and the lifecycle of the NDT are facilitated to perform the orchestration of management and operational functions within the real network.

An NDT prototypical architecture compatible with the NGSI-LD standard to cover network monitoring and configuration purposes is depicted in Figure 3.2. The northbound interface to interact with the NDT instance and retrieve NDT data is implemented by the NGSI-LD API. In such a case, an NMS in the application layer can decompose the desired intent into different instructions or inputs through interactions using the NGSI-LD API. Meanwhile, the southbound interface used by the NDT instance to interact with the real-world network is implemented by an MDT network management protocol to apply the appropriate operation and management actions for network configuration and monitoring purposes.

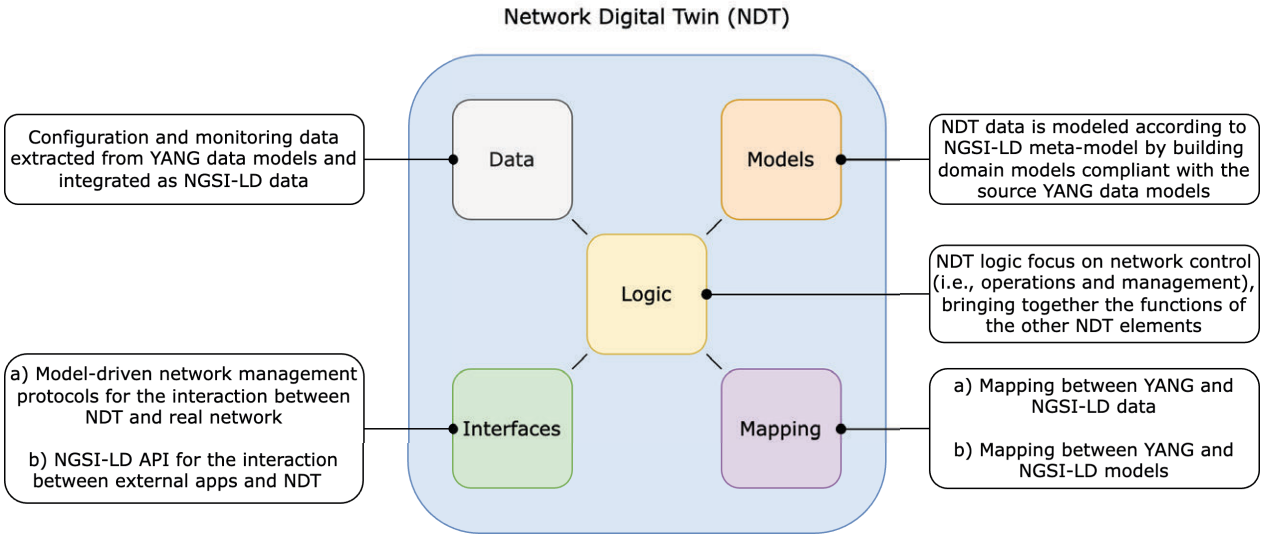


Figure 3.1: NDT core elements compliant with the NGSI-LD standard.

The network capabilities in terms of the performance, status, and configuration data reflected in the NDT instance can then be exposed to the NMS at the application layer, also using the NGSI-LD API.

In addition, the NGSI-LD standard provides the capabilities needed to implement the NDT instance as follows:

- For the Data Repository component, the NGSI-LD standard provides data integration mechanisms for processing and storing real-time and historical data from real networks. Depending on the requirements of the service application, the data integration approach could provide historical data storage or forward the real-time data directly to the service application. Subsection 3.2.1 provides a detailed description of the data integration approaches used in the developed Network Digital Twin solution.
- In order to support the Service Mapping Models building block functionalities, the NGSI-LD standard supports representation of basic and functional information models based on Labeled Property Graphs with linked data and semantic annotation functionalities. Depending on the capabilities offered by the NGSI-LD based Digital Twin system, the NDT could provide different information. For basic information about the network topology, configuration, and operational data of network elements, the Descriptive Twin case is the candidate. For other functional information, such as diagnosis or predictions, different NDT cases such as Diagnosis Twin or Predictive Twin should be selected. Subsection 3.1.2 provides a detailed description of the different types of Digital Twins listed according to the systematic classification given by ETSI CIM adapted to the scope of network operations and management.
- NDT management and control are facilitated by the NGSI-LD RESTful API, which provides HTTP CRUD operations to trigger and manage the lifecycle of network automation mechanisms such as network monitoring and configuration.

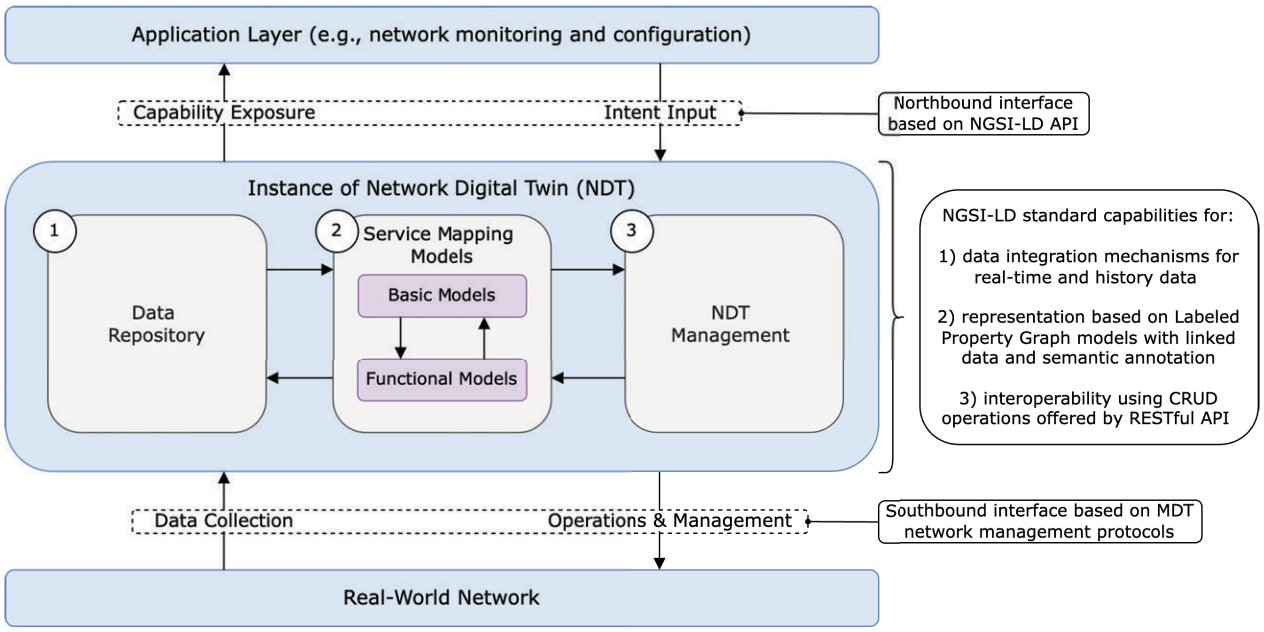


Figure 3.2: Prototypical architecture of NDT compliant with the NGSI-LD standard.

Section 3.2 describes a methodology for modeling an NDT based on the NGSI-LD standard that conforms to the specified design elements and architecture.

3.1.2 NGSI-LD Digital Twins for network operations and management

As introduced in Chapter 2, Digital Twins have been systematically classified by ETSI CIM according to their capabilities. The adaptation of this classification of Digital Twins in the area of network operations and management can be seen as follows:

- **Descriptive Twin**: In this first Digital Twin case, the Digital Twin is presented as an interface that informs about the current state of the real-world assets. Then, the Digital Twin can be queried and configured as if it would be the real-world element. Integrating this vision within the networking domain, if the asset is a network device (e.g., a router device), the Digital Twin may be the digitalized representation about its operational status and configuration information for interoperability in network management systems. Also, the interoperability between the real asset and the Digital Twins may be possible, so that the real-world status is represented in the Digital Twin and a change in the Digital Twin can be reflected on the real asset. Apart from that, augmentation functions are considered in order to provided an augmentable vision of the reality. In this term, network metrics augmented with computed functions, such as time series aggregation functions, can be used to calculate additional network key performance indicators (KPIs).
- **Predictive Twin**: It extends the Descriptive Twin case by providing predictions on the way the real-world asset could evolve in the future. In this sense, the predicted

future values can be computed on demand or stored as properties of the Digital Twin. External applications with prediction models can then compute these values as temporal values using the NGSI-LD Temporal API. These prediction models may require the availability of additional context information or aggregated functions in order to perform future predictions. Applying this twin case in the networking domain, based on the current state of a network, an external system based on a simulator, a Machine Learning model or a mathematical analytical model could predict its next performance states (i.e., end-to-end latency, bandwidth, congestion, etc.) using additional information, such as network traffic or throughput at each moment by means of an aggregation function used by the Descriptive Twin.

- **Prospective Twin**: It builds upon the prediction capabilities of the Predictive Twin. It helps to evaluate the impact that different actions would have on a real-world asset. Then, it allows running *what-if* scenarios by analyzing prediction models over the current state of real assets modified to represent an intended action. In this sense, the Digital Twin could be altered to run predictions on a hypothetically modified version of reality, thus performing the prospective analysis for hypothesis generation. In the networking world, for example, this twin case could be used to evaluate the impact of high traffic rates based on the current state of the network and variations in traffic flows.
- **Prescriptive Twin**: It aims at identifying actions to be taken over the real asset for it to reach a target state. For this case, a prescriptive service can be deployed in order to provide recommendations related to the Digital Twin management. For example, this service can interact with the prediction model and test different alterations to identify an optimum functionality for the real-world asset. Within the networking domain, it may be a particular case to solve a network optimization problem. When combined with actuation, this functionality provides automated management that will help to reach the closed-loop automation paradigm.
- **Diagnosis Twin**: It helps to understand what happens in case of a real-world asset malfunction or deviation from its expected status. This Digital Twin case builds on a diagnostic model that collects additional data related to the status evolution of the real-world asset. Within the networking domain, it would help to detect outliers o anomalies to raise alerts in case the network deviates from its anticipated state.

As can be seen from the Digital Twin classification established by ETSI CIM, all Digital Twin cases can provide added value in the area of NDT, as was also analyzed by the Innovative Optical Wireless Network Global Forum in its technical report *"Network Digital Twin Use Case"* [59].

As illustrated in Figure 3.3, the cornerstone on which all types of identified twins are based is the digitalization of the networking configuration and state information, and therefore the main pillar for this is the functionality of the first case of Digital Twin, which is the Descriptive Twin. The Descriptive Twin serves as an interface that informs about the past and current state of real-world assets. In addition, when bidirectional communication between real-world assets and the Digital Twin is possible, not only the real-world state can be represented in the Digital Twin, but changes in the Digital Twin can also be applied to real-world assets. Thus,

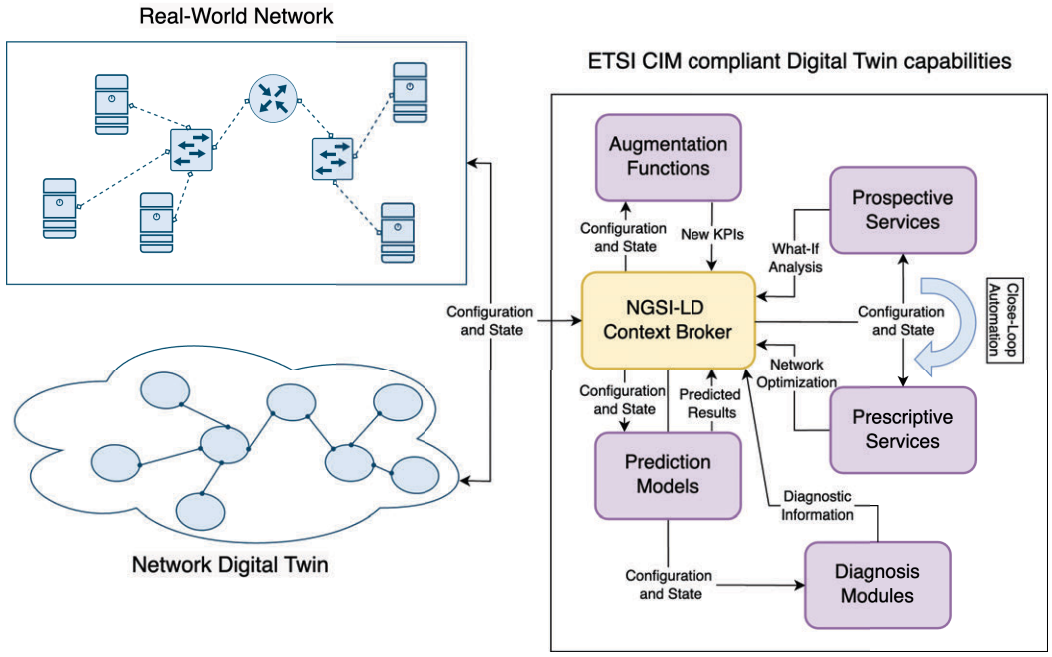


Figure 3.3: General insight on NGSI-LD compliant Digital Twins for network management domain.

the Digital Twin can be queried and configured as if it were a real-world element. Transferring this Digital Twin case to the network management domain, as illustrated in Figure 3.4, the asset is a real-world network and the NDT is a digitalized representation of the network topology with its operational status and configuration. Thus, the Descriptive NDT functions as a conceptual map of the real-world network assets, providing a continuous representation of network management data according to a knowledge-based model. The Descriptive NDT can also offer added value through computational capabilities, such as the calculation of network KPIs using time-series aggregation functions on the monitored data.

As discussed before, other Digital Twin cases that would add value to network management, such as providing predictions, running *what-if* scenarios, or diagnosing possible malfunctions, are based on the Descriptive Twin vision. The Descriptive NDT using the NGSI-LD standard is therefore the starting point on which this work focuses. The following Section 3.2 describes a methodology for modeling a Descriptive NDT solution based on the NGSI-LD standard.

3.2 A New Methodology for Modeling an NDT based on the NGSI-LD Standard

This section proposes a new methodology for modeling a Descriptive NDT using the NGSI-LD standard. First, approaches for integrating YANG data from the network management domain using NGSI-LD are introduced. Then, generic rules are proposed to facilitate the translation between YANG and the NGSI-LD meta-model, providing a mapping mechanism between network management data and its representation in the NDT. Finally, a reference architecture for modeling a Descriptive NDT is presented.

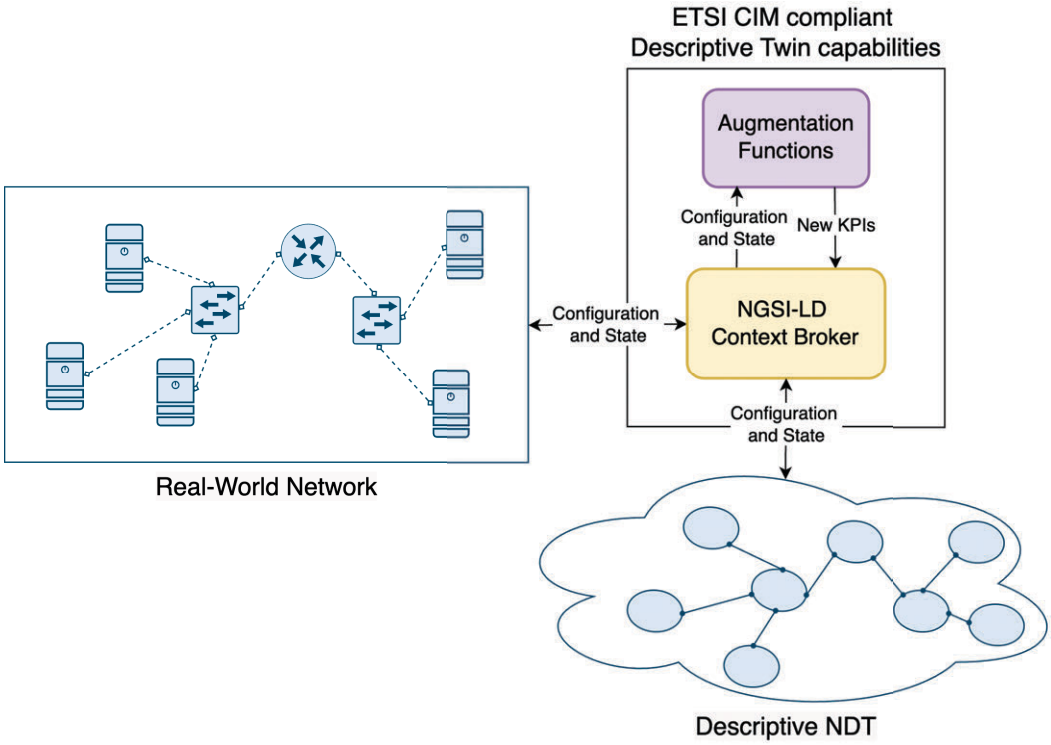


Figure 3.4: General insight on NGSI-LD compliant Descriptive Twin for network management domain.

3.2.1 Data integration using the NGSI-LD standard

As described in detail in Chapter 2, data materialization and data virtualization are two of the main approaches used to integrate data from different sources [40], [118], [119]. On the one hand, the data materialization approach proposes a target data store that is synchronized with the data sources and provides consumable historical data. When there is a need for a high-performance information backend for data integration, such as in NDT systems, this data materialization approach could be the ideal solution. On the other hand, data virtualization offers a suitable interface for querying and manipulating information without concerns about the physical location of data sources, providing the most up-to-date results without replicating information. It could be the most efficient solution when the data does not need additional pre-processing mechanisms and can be directly accessible and consumable.

Figure 3.5 shows the workflow to address the data materialization approach with the NGSI-LD standard. In this approach, the architectural roles are compatible with the vision of a centralized architecture that is reflected in the NGSI-LD standard. The Context Producer is responsible for collecting data from a Data Source, translating it into the NGSI-LD meta-model representation, and then replicating the data within a Context Broker. The Context Broker stores the data and provides access to interested Data Consumers that can either query for new data or subscribe to data changes using the NGSI-LD API. The main drawbacks of this approach are potential synchronization issues with the Data Source to ensure that the data remains up to date, and the limitation of Context Broker storage due to data replication.

However, the approach benefits from both the support for historical data records within the Context Broker and low end-to-end latency from data collection to data integration.

Figure 3.6 shows the workflow to address the data virtualization process to query and subscribe to virtualized data through an NGSI-LD Context Broker. In this approach, the use of architectural roles is compatible with the distributed architecture vision of the NGSI-LD standard. The Context Source acts as an intermediary system that implements the NGSI-LD API to provide context information from particular Data Sources. The Context Broker redirects requests from Data Consumers to the Context Source so that the Context Source provides the virtualized representation of the data collected from the Data Sources to the Data Consumers without storing the resulting data in the Context Broker. The main advantage of this approach is that it provides efficient access to the data, avoiding data replication. The main disadvantage of this approach is the additional delay in processing consumer requests to deliver the data.

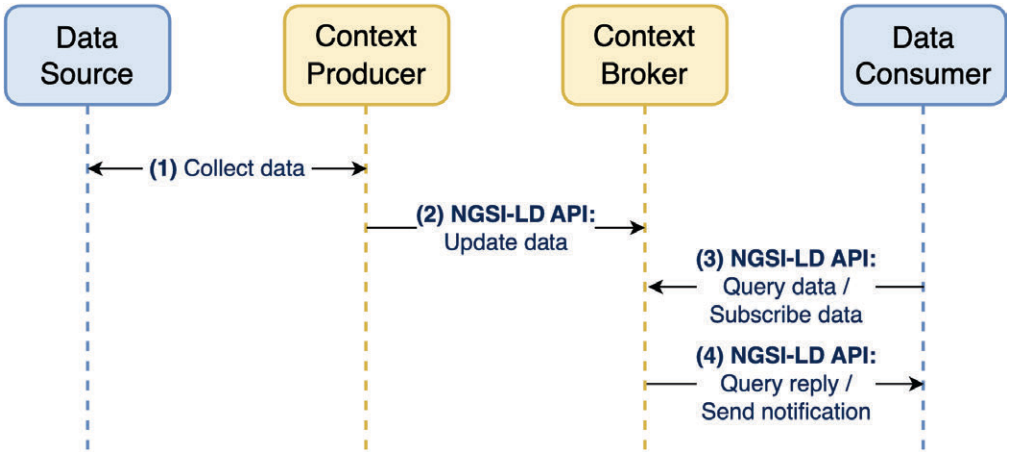


Figure 3.5: Data materialization with the NGSI-LD standard.

The Descriptive NDT solution proposed in this work can mirror the operational status and configuration of real networks using both data integration approaches to support an advanced network management system implementing closed-loop automation. The data materialization approach is identified as an ideal option for collecting network configuration and operational data over time that can be used, for example, to train and feed external Machine Learning systems to capture network behavior for performance analysis. Meanwhile, the data virtualization approach is an efficient solution to retrieve monitoring data on demand and request network management actions. Both approaches can be combined to create a more flexible data integration system for network operations and management.

The following describes in detail the data integration architecture for both data materialization and data virtualization, with a focus on integrating data from network monitoring sources that support model-driven network management protocols.

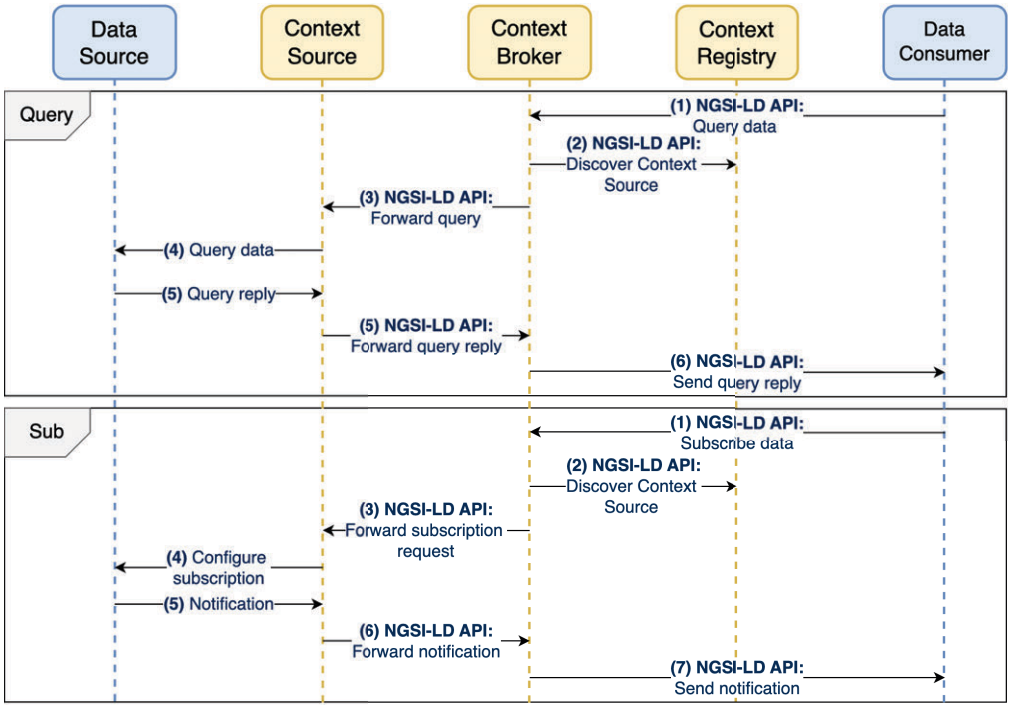


Figure 3.6: Data virtualization with the NGSI-LD standard.

Data materialization architecture using NGSI-LD

Figure 3.7 depicts the general architecture of a data materialization system using the NGSI-LD standard for network data sources supporting MDT and YANG data modeling language. The global schema is defined in terms of NGSI-LD information models particularized for the original local schemas available within the data sources (i.e., YANG data models). There are three main roles in this architecture:

1. *MDT Data Sources*: Data sources supporting YANG data modeling language and model-driven network management protocols such as NETCONF or gNMI. To enable the NGSI-LD Context Producer to communicate with accessible MDT Data Sources, external applications must provide communication contact information to the NGSI-LD Context Broker (see Subsection 3.2.4 for more details).

2. *NGSI-LD Context Producer*: It enables communication with MDT Data Sources by using RPC operations for both control and collection of YANG modeled data. In the case of data collection, the NGSI-LD Context Producer allows the data to be translated into the global schema (i.e., the resulting NGSI-LD information model). Finally, the NGSI-LD Context Producer stores the data in the NGSI-LD Context Broker by using the NGSI-LD API to complete the data materialization process. Therefore, the Context Producer needs to implement clients for both model-driven network management protocols and for the NGSI-LD API, in order to extract and materialize the data.

3. *NGSI-LD Context Broker*: It provides the data warehouse functionality with a replication of the data coming from the MDT Data Sources corresponding to the global NGSI-LD information models. Once the data is materialized, data consumers, such as external

applications, can interact with the NGSI-LD Context Broker through direct data query requests using the NGSI-LD API.

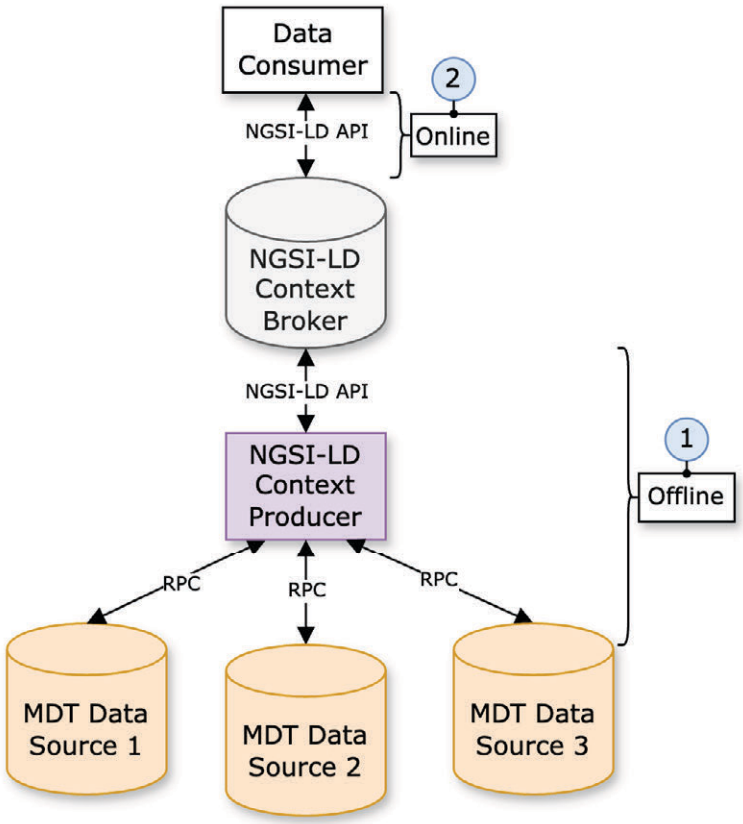


Figure 3.7: General architecture of a data materialization system using NGSI-LD for MDT data sources.

As is common with data materialization approaches, the processes involved in communicating with the data source and the transformation performed by the NGSI-LD Context Producer to complete the data integration in this particular implementation (i.e., the offline phase) are independent of the query processing activity initiated by the external applications (i.e., the online phase).

Therefore, the external application will not be able to consume data from the NGSI-LD Context Broker until the data integration process based on data materialization has been initiated and completed.

Data virtualization architecture using NGSI-LD

Figure 3.8 depicts the general architecture of a data virtualization system using the NGSI-LD standard for network data sources supporting MDT and the YANG data modeling language. The global schema is also defined in terms of NGSI-LD information models particularized for the original local schemas available within the data sources (i.e., the YANG data models). There are four main roles in this architecture:

1. *MDT Data Sources*: Data sources supporting YANG data modeling language and model-driven network management protocols such as NETCONF or gNMI.
2. *NGSI-LD Context Source*: It enables the reformulation of requests coming from NGSI-LD API to network management operations in the form of RPCs for collecting of YANG modeled data from MDT Data Sources. In addition, the NGSI-LD Context Source transforms the responses of RPC operations into the representation of the NGSI-LD information model. Finally, the NGSI-LD Context Source sends the data to the NGSI-LD Context Broker by using the NGSI-LD API in order to forward it to the Data Consumer without data replication. The Context Source needs to implement clients for model-driven network management protocols and for NGSI-LD API, in order to extract and virtualize the data. It also requires the implementation of server-side distribution operations from the NGSI-LD API to be able to register as a Context Source dedicated to MDT Data Sources within the Context Broker. Moreover, external data consumers can use the NGSI-LD API to formulate queries directly to the Context Source. This allows the Context Source to send the data in the NGSI-LD global schema directly to the Data Consumer without having to go through the Context Broker.
3. *NGSI-LD Context Broker*: It provides the NGSI-LD API that is used by external data consumers to formulate queries. Specifically, it provides information models for expressing requests and describing the data coming from the MDT Data Sources. Similar to the data materialization architecture, the NGSI-LD Context Broker allows external applications to initiate establishment sessions with accessible MDT Data Sources by formally describing the communication contact information (see Subsection 3.2.4 for more details). In addition, based on queries formulated by external applications, the Context Broker provides the requested data from the MDT Data Sources on the fly and is modeled according to the NGSI-LD information models.
4. *NGSI-LD Context Registry*: It allows NGSI-LD Context Source to register as a Context Source dedicated to MDT Data Sources within the NGSI-LD Context Broker. It also allows data consumers to discover which Context Sources that provide MDT source data are registered within a Context Broker. Depending on the implementation of the Context Broker, the NGSI-LD Context Registry functionality may be integrated into the NGSI-LD Context Broker or in an independent component.

Following this architecture, data remain in the MDT Data Sources and can be accessed on demand by external applications that interact directly with the NGSI-LD Context Broker via the NGSI-LD API, which is a fully online query processing mechanism.

3.2.2 YANG to NGSI-LD translation conventions

The translation conventions, based on [171], provide the rules for mapping YANG data node types to the elements of the NGSI-LD meta-model. The YANG data modeling language specifies different types of data nodes [101], where each node is represented in a schema tree and can be instantiated in a data tree. Meanwhile, the NGSI-LD information model [81] builds on the Labeled Property Graph model and the JSON-LD format, where entities are the top-level entries. Entities can have properties with particular values and relationships

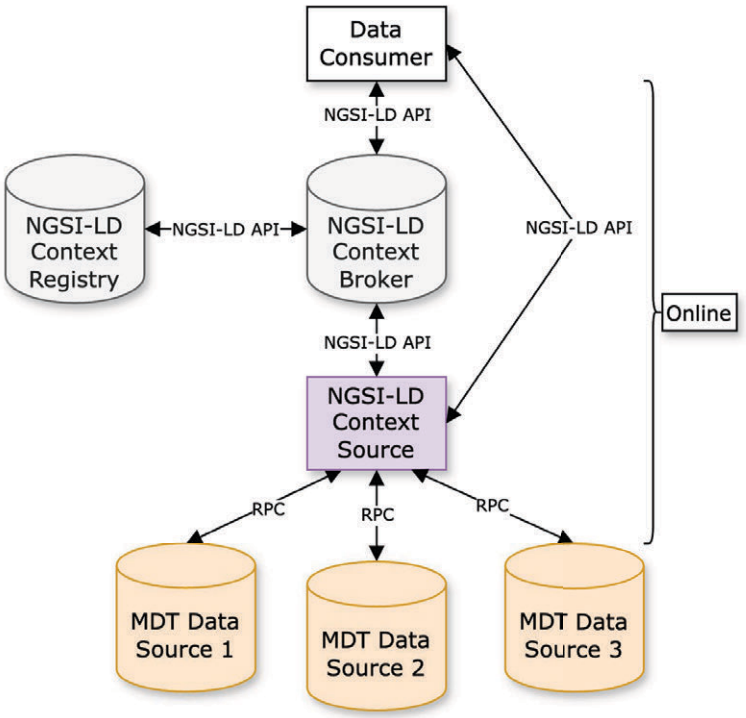


Figure 3.8: General architecture of a data virtualization system using NGSI-LD for MDT data sources.

with other entities, and properties and relationships can themselves also have second-level properties and relationships that provide additional characteristics or metadata.

The transformation process between YANG and NGSI-LD takes as input the effective YANG schemas, which is the compiled set of YANG data models, so the mapping rules are defined at the YANG level. This allows for seamless translation of any YANG data to NGSI-LD regardless of the encoding format (e.g., XML or JSON) and the network management protocol (i.e., NETCONF or gNMI) used for transporting YANG data. Table 3.1 shows a high-level mapping of the main YANG data node types to the core elements of the NGSI-LD meta-model.

Table 3.1: High-level mapping from YANG to NGSI-LD.

YANG Data Nodes	NGSI-LD Elements
Container	Entity
List	Entity
Identity	Entity
Leaf	Property or Relationship
Leaf-List	Property or Relationship

In general, YANG *container* maps to NGSI-LD *entity* whenever the *container* contains child data nodes and does not have a YANG *list* as its only child. If the *container* has a YANG list as its only child, then the *container* can be omitted since it does not have semantic meaning,

so it is not mapped to the NGSI-LD *entity*. As a result, this design choice makes the graph more compact, which improves usability. In a similar way, YANG *list* maps to NGSI-LD *entity*, where there will be an NGSI-LD *entity* for every member of the YANG *list*. In both cases of YANG data node types, if the YANG *container* or YANG *list* is the child of another YANG data node (either *container* or *list*), the resulting NGSI-LD *entity* type will include an NGSI-LD *relationship* pointing to the NGSI-LD *entity* type of the parent YANG data node, thus capturing the hierarchical connection between both YANG data nodes.

YANG *identity* introduces a special case as it represents an abstract concept that may inherit from other YANG base identities. In this sense, YANG identities will map to NGSI-LD entities of a new entity type called *YANGIdentity*. This NGSI-LD entity of type *YANGIdentity* has the context information to represent the identifier, source YANG namespace, and description as NGSI-LD properties, and a possible NGSI-LD relationship to point to a broader or parent YANG *identity*. It is important to note that each concept within the resulting NGSI-LD entity corresponding to a YANG *container* or *list* that matches to an NGSI-LD entity of type *YANGIdentity* results in an NGSI-LD relationship pointing to the corresponding object identifier of the *YANGIdentity* entity.

Meanwhile, YANG *leaf* and YANG *leaf-list* will map to NGSI-LD attribute (i.e., NGSI-LD *relationship* or NGSI-LD *property*), but in the case of YANG *leaf-list* the mapping NGSI-LD attribute is an array. The corresponding NGSI-LD attribute will be part of an NGSI-LD entity type that represents the parent YANG data node (either *container* or *list*) of the child YANG *leaf* or YANG *leaf-list*. When the YANG *leaf* or YANG *leaf-list* uses the *leafref* data type, it references to another YANG data node type, and it will map to an NGSI-LD *relationship* pointing to the resulting NGSI-LD entity type that contains the NGSI-LD attribute corresponding to the target YANG data node type. A similar case occurs when the YANG leaf or *leaf-list* uses the *identityref* data type that references a YANG *identity*, where it will map to an NGSI-LD *relationship* pointing to an NGSI-LD entity of type *YANGIdentity*. However, when the data type of the YANG *leaf* or YANG *leaf-list* is considered a literal, then it is mapped to an NGSI-LD *property*.

Table 3.2 captures the main data types considered in NGSI-LD when translating from the different YANG built-in data types. It should be noted that the NGSI-LD built-in data types are based on the JSON-LD data encoding format, which follows JSON syntax and can be validated using JSON Schema. In YANG, there are several *Integer* built-in types that represent signed and unsigned integers of different sizes, and all of them map to the *Integer* data type in NGSI-LD. Decimal numbers, boolean values, and string data types in YANG have their corresponding data types in NGSI-LD, emphasizing that if a YANG string contains a regular expression pattern, the mapping to NGSI-LD will also contain the same string pattern. Note that for representing bit sets and binary data, the candidates in NGSI-LD are lists of strings and strings, respectively, and for enumeration data types in YANG representing a set of bounded values, the corresponding data type in NGSI-LD is a string with the determined value. Note also that the empty YANG data type maps to a literal in NGSI-LD represented by an empty string, since the *null* value is reserved for deleting NGSI-LD attributes. In addition, there are custom data types defined in YANG for representing timestamps, which require a special mapping to a string with *date-time* format in NGSI-LD.

Table 3.2: Data type conversion from YANG to NGSI-LD.

YANG Built-in Data Types	NGSI-LD Built-in Data Types
Integer	Integer
decimal64	Number
string (regex pattern, if any)	String (format pattern, if any)
boolean	Boolean
enumeration	String
bits	String[]
binary	String
empty	String
timestamp (custom data type)	String (format date-time)

Because NGSI-LD is based on JSON-LD, all elements considered in an NGSI-LD information model must be URIs. In this sense, the associated JSON-LD URI for a converted NGSI-LD element type is generated by combining the namespace of the original YANG module that defines the corresponding YANG data node with its XPath.

As explained earlier, the YANG data models specify which data is intended for configuration and which data represents the operational state on a YANG server. In this regard, Annex A.3 provides details on how to handle YANG configuration and operational state data with the NGSI-LD standard.

Additional details of the YANG to NGSI-LD translation process, covering both mapping rules and data type conversion, are available in [171]. The aforementioned translation conventions from YANG to NGSI-LD define the schema matching process of the data integration solution proposed in this thesis. To cover these YANG to NGSI-LD translation conventions, there is a need for a technical solution tool capable of parsing YANG modeled data and matching it with its own YANG model to identify its schema and data tree structures in order to apply the mapping rules accordingly. This solution will allow completing the schema mapping process of the data integration proposal. The following subsection describes in detail a prototypical NDT framework architecture for automatically performing these mapping rules. Our solution can be seen as a first approximation or reference implementation to cover the translation process from YANG to NGSI-LD for a new model-driven network management proposal based on the NGSI-LD standard.

3.2.3 Descriptive NDT architecture

One of the main concerns when implementing an NDT system is the management and knowledge of real network data. We address this by defining a Descriptive NDT solution that provides automatic mechanisms for mapping data that is related to the network management domain to a common, formal, and interoperable knowledge model based on the NGSI-LD standard. By applying the translation conventions and the data integration approach mentioned in the previous subsections in a fully automatic way, this solution can provide a programmatic system for building a Descriptive NDT.

Figure 3.9 depicts the reference and high-level architecture of our Descriptive NDT solution[3.1], which is already published in [127]. This architecture encompasses the four core components considered for NDT solutions, providing 1) real-time and historical NDT data; 2) models for characterizing the NDT data; 3) interfaces to establish communication with the real network and with the NDT system; and 4) mapping mechanisms for the exchange of data between the real network and the NDT system.

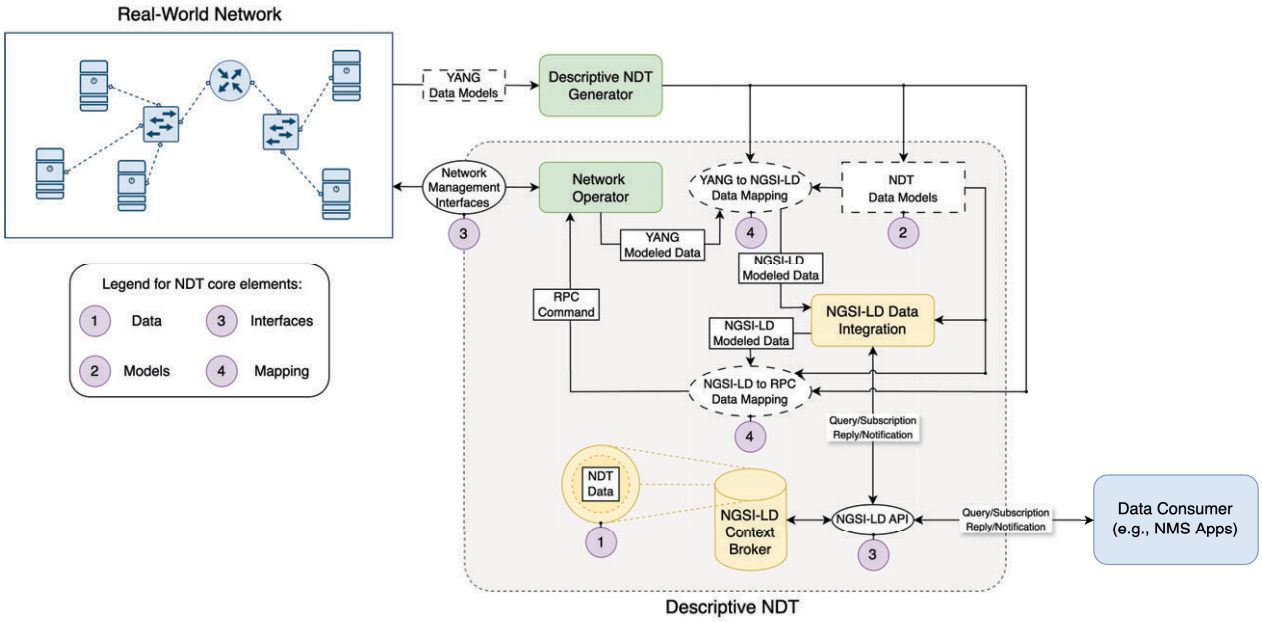


Figure 3.9: Descriptive NDT architecture compliant with the NGSI-LD standard and the NDT core elements [127].

The proposed architecture supports MDT and YANG-based management mechanisms using a data integration approach. Once the YANG Data Models that characterize the configuration and operation of the real networks are selected, the Descriptive NDT Generator component will automatically generate the models and mapping mechanisms to cover the Descriptive NDT solution for performing the translation between real network data and NDT data.

The data integration solution combines schema matching and schema mapping techniques for translation between YANG modeled data and NGSI-LD modeled data using the predefined transformation conventions or rules described in Subsection 3.2.2. This combined approach enables a flexible and extensible data integration mechanism by facilitating the interoperability between models and the automation within the data transformation process. The schema mapping process of the data integration solution follows a Global-Local-As-View (GLAV) [46] strategy because it defines a set of generalized and structured translation rules that map YANG data nodes to NGSI-LD elements. These translation rules serve as an intermediate level of abstraction that allows different YANG modeled data to be translated into corresponding NGSI-LD modeled data while maintaining a structured and reusable mapping framework. This means that even though different source data schemas based on YANG produce different

[3.1] An extended view of the Descriptive NDT architecture diagram is provided in Annex A.1.

target data schemas based on NGSI-LD, they are all derived using the same predefined schema-matching logic, demonstrating the flexibility and consistency of the data integration approach. Therefore, the global schema (i.e., NGSI-LD information models) provides a conceptual structure, while local schemas (i.e., YANG data models) define how specific data instances fit into the target NGSI-LD representation.

The resulting Descriptive NDT includes the following architectural components:

- *Network Operator:* The Network Operator enables the application of network automation processes for both network monitoring and configuration purposes. For monitoring purposes, the Network Operator enables the collection of management data such as topology, configuration, and operational data from real-world networks using network management interfaces based on the YANG data modeling language, such as NETCONF or gNMI. On the other hand, for configuration purposes, the Network Operator allows the application of desired configuration changes into the real network using the network management interfaces.
- *YANG to NGSI-LD Data Mapping:* This data mapping mechanism allows parsing the YANG Modeled Data collected by the Network Operator and mapping it to the NGSI-LD Modeled Data by applying the translation conventions and following the NDT Data Models accordingly.
- *NDT Data Models:* These models are automatically generated data schemas that provide the knowledge baseline for the NGSI-LD information models associated with each YANG data model. Depending on the element of the NGSI-LD meta-model (i.e., whether it is an entity, property, or relationship), a specific type of data schema is defined, including any particularities specified within the associated YANG data nodes, such as descriptions, data types, mandatory field selection, regular expressions, formats, units, thresholds, or default values. Finally, this component generates the base models to address the complete data translation pipeline between the real network and the NDT.
- *NGSI-LD to RPC Data Mapping:* This data mapping mechanism allows parsing the NGSI-LD Modeled Data provided by a Data Consumer to update the real network configuration or to start the monitoring of a telemetry data source. The parsed data is then mapped to RPC commands which are orchestrated by the Network Operator component to apply the desired configuration on a network node or to set up the telemetry data source. This component performs the query processing and rewriting functionalities of the data integration approach, allowing to translate NGSI-LD API queries to RPC operations. The implementation of this component depends on the target model-driven network management protocol used to perform the RPC operations. It has been fully validated with NETCONF (see Subsection 4.1.3 from Chapter 4 for more details).
- *NGSI-LD Data Integration:* This is the core component for integrating the NDT data using either a data materialization approach or a data virtualization approach. For management data collected from real networks, after the data is retrieved and translated to NGSI-LD by the YANG to NGSI-LD Data Mapping mechanism, the NGSI-LD Data Integration component validates the translated data against the corresponding

NDT Data Models and forwards the resulting NDT Data to the NGSI-LD Context Broker. Using a data materialization approach, the NDT data is stored in the Context Broker, while using a data virtualization approach, the data is sent through the Context Broker directly to the requesting consumers. For network configuration or monitoring instructions provided by a consumer, once the NGSI-LD data is uploaded to the Context Broker, the NGSI-LD Data Integration component receives and validates the data as NGSI-LD Modeled Data. The NGSI-LD Modeled Data can then be processed by the NGSI-LD to RPC Data Mapping mechanism to apply the requesting action.

Therefore, through this Descriptive NDT, consumers, such as analytics applications from a network management system (i.e., NMS Apps), can interact with the real network and then consume and manipulate the integrated NDT data using the NGSI-LD API.

In Chapter 4, a prototypical implementation of this reference architecture for modeling a Descriptive NDT solution is presented.

3.2.4 Network automation using the Descriptive NDT and the NGSI-LD standard

Once the Descriptive NDT architecture has been described, it is necessary to determine how an external application or data consumer can interact with the Descriptive NDT via the NGSI-LD API to automate network operations and management processes.

In this sense, the following subsections describe in detail how to orchestrate network operations and management tasks specific to model-driven network management protocols using the NGSI-LD standard applied to each supported data integration approach already covered in Subsection 3.2.1.

But first, the procedure for registering compatible network data sources in the Descriptive NDT is determined. As part of the integration of network devices as new data sources within the Decriptive NDT solution, they must be registered to expose their capabilities. The operators of the Descriptive NDT system require these capabilities in order to discover what data are available in the network devices and how these data can be ingested through the NDT. In this regard, the NDT collects context information that describes the capabilities of devices with support for YANG-based model-driven network management protocols that are present in the real network. To model this context information, the Descriptive NDT again leverages the NGSI-LD standard.

To this end, in the work in [172] we proposed a semantic network inventory system to integrate and facilitate the access to context information associated with the network data sources that support YANG data modeling. The purpose of a network inventory is to allow Network Operations (NetOps) engineers to discover the available YANG data (i.e., *what*) and the network management protocols that can stream such data (i.e., *how*) from a given platform or device. This NetOps engineer is a new role that, by applying DevOps techniques, can develop applications that automate the management of network infrastructures. Then, the context information offered from the network inventory solution is provided by using the NGSI-LD standard and covers two domains: 1) The *Platform Domain*, which contains information

regarding the implementation of YANG modules and model-driven network management protocols on network platforms or devices.; and 2) The *Catalog Domain*, which includes metadata on YANG modules that exist in external YANG catalogs[3.2].

In this sense, based on the insights collected from the network inventory, NetOps engineers can autonomously discover the information needed to manage network data sources supporting model-driven network management protocols based on YANG, without having to interact manually with external resources or other engineers on the team.

Assuming that the YANG-based network data source registration process is covered by the network inventory solution just outlined[3.3], we can now move on to how data integration approaches based on the NGSI-LD standard can be used to automate network operations and management processes.

NGSI-LD for network automation using a data materialization approach

A first option for orchestrating network operations and management processes through the Descriptive NDT solution and the NGSI-LD API is a data materialization approach. For activating this data materialization approach, there is a custom NGSI-LD model for agnostically defining the client-side key parameters of a model-driven network management protocol, such as NETCONF or gNMI clients, for network operations and management purposes in a Descriptive NDT system.

This NGSI-LD model allows external NetOps users or applications to trigger network operations and management tasks against reachable data sources supporting MDT mechanisms. These users or applications interact directly with the NGSI-LD Context Broker via the NGSI-LD API before querying the materialized data resulting from the network tasks performed. The implementation details for handling the network operations and management tasks defined by using the aforementioned NGSI-LD model and the NGSI-LD API, and transforming them into RPCs are described in Subsection 4.1.3 of Chapter 4.

Figure 3.10 shows the high-level UML diagram of the NGSI-LD model used to represent client-side data for a model-driven network management protocol. As depicted in Figure 3.10, the NGSI-LD model is composed of three different entities: 1) *Protocol* for indicating the endpoint of a network device that supports a model-driven network management protocol; 2) *Credentials* for indicating the credentials of a network device that supports a model-driven network management protocol; and 3) *RpcOperation* for triggering a RPC operation against a network device that supports a model-driven network management protocol.

[3.2]A YANG catalog can be defined as a solution that extracts the metadata from YANG modules – typically stored in a repository – and exposes these metadata through a comprehensive search system. The YANG Catalog is a project driven by the IETF to implement a centralized catalog that operators can leverage to search for YANG modules and their associated metadata.

[3.3]The details of the network inventory solution can be found in [172], which is a research elaborated within the GIROS research group of DIT (ETSIT-UPM).

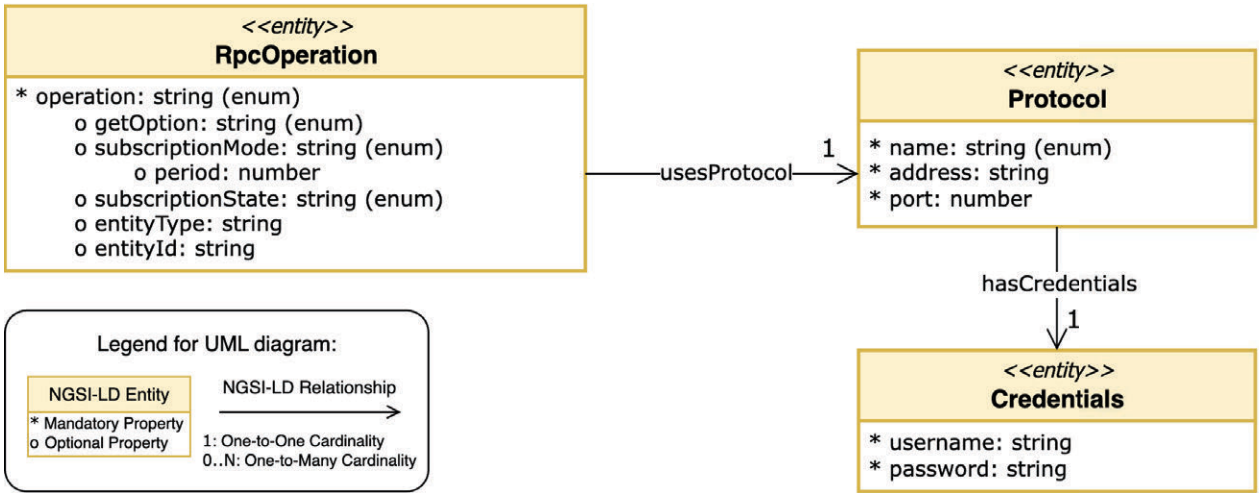


Figure 3.10: UML diagram of NGSI-LD model for network operations and management based on data materialization approach.

The associated NGSI-LD model of a *Protocol* NGSI-LD entity type for connecting to a network device defines the following attributes:

- *name:* An NGSI-LD property to indicate the name of the model-driven network management protocol that is supported by the network device in server mode, such as NETCONF or gNMI.
- *address:* An NGSI-LD property to indicate the hostname or address of the model-driven network management protocol server.
- *port:* An NGSI-LD property to indicate the administration port of the model-driven network management protocol server.
- *hasCredentials:* An NGSI-LD relationship pointing to the *Credentials* NGSI-LD entity type that specifies the credentials of the model-driven network management protocol server.

The corresponding NGSI-LD model of a *Credentials*[3.4] NGSI-LD entity type defines the following attributes:

- *username:* An NGSI-LD property to indicate the administration username of the server that supports a model-driven network management protocol.
- *password:* An NGSI-LD property to indicate the administration password of the server that supports a model-driven network management protocol.

[3.4]The *Credentials* entity described is only a very basic first approximation for considering authentication with the server or device that supports the model-driven network management protocol using a username/password authentication mechanism. This has been the initial approach used to validate the solution, but it can be extended to include additional authentication options supported by client-server communication using model-driven network management protocols, such as the use of SSH keys for the NETCONF protocol or the use of TLS certificates for the gNMI protocol. In addition, a solution must be found to encrypt sensitive data in NGSI-LD, such as passwords specific to the authentication method used.

The corresponding NGSI-LD model of an *RpcOperation* NGSI-LD entity type defines the following attributes:

- *usesProtocol:* An NGSI-LD relationship pointing to the *Protocol* NGSI-LD entity type that specifies the model-driven network management protocol supported by a network device in server mode that triggers the RPC operation.
- *operation:* An NGSI-LD property to indicate the type of RPC operation (i.e., GET, SET, or SUBSCRIBE). Depending on the type of RPC operation, the model includes additional parameters defined as subproperties.
  - *getOption:* The option for the NETCONF RPC GET operation. There are 3 options: 1) *state* for GET RPC operation; 2) *config* for GET-Config RPC operation; and 3) *edit-config* for GET-Config RPC operation triggering a NETCONF SET RPC operation.
  - *subscriptionMode:* The subscription mode for the SUBSCRIBE RPC operation (i.e., on-change or period).
    - ▷ *period:* The period for the SUBSCRIBE RPC operation when there is a periodic subscription.
  - *subscriptionState:* The state for controlling the operation lifecycle for a SUBSCRIBE RPC operation (i.e., ON, OFF, or IDLE).
  - *entityType:* NGSI-LD entity type relative to the original data node specified in a YANG model resulting from the YANG to NGSI-LD mapping and translation processes. Both short-hand string (type name) or URI are allowed.
  - *entityId:* NGSI-LD entity id relative to the original data node specified in a YANG model resulting from the YANG to NGSI-LD mapping and translation processes.

It should be noted that the *RpcOperation* entity is the one that invokes the execution of an intent instruction within the NDT instance to perform a management operation on the real network. In the following, we describe the workflow diagrams that represent how to interact with the Descriptive NDT using the NGSI-LD API to trigger the various RPC operations for managing a real network based on a data materialization approach.

Figure 3.11 shows the workflow to complete a GET RPC operation by using the NGSI-LD API and a data materialization approach. When a Data Consumer creates a *RpcOperation* NGSI-LD Entity within the Context Broker, it triggers the corresponding GET RPC operation. The Context Producer, which is subscribed to *RpcOperation* entities, receives the notification with the necessary parameters to perform the GET RPC operation. Once the GET RPC operation is performed by the Context Producer against the corresponding Data Source (i.e., the YANG-based server), the configuration and operation data retrieved are translated to the NGSI-LD model and is updated within the Context Broker. The data is then materialized and can be queried by the Data Consumer by interacting directly with the Context Broker. The Data Consumer can define the type of GET RPC operation (i.e., GET or GET CONFIG) by using the *getOption* parameter.

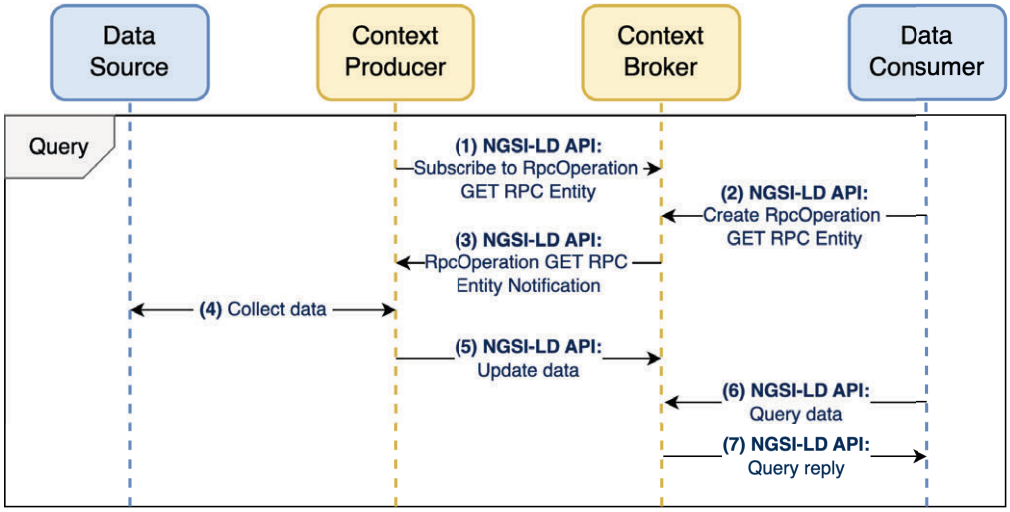


Figure 3.11: Data materialization with GET RPC operation.

Figure 3.12 shows the workflow for completing a SUBSCRIBE RPC operation by using the NGSI-LD API and a data materialization approach. When a Data Consumer creates a *RpcOperation* NGSI-LD Entity within the Context Broker, it triggers the corresponding SUBSCRIBE RPC operation. The Context Producer, which subscribes to RpcOperation entities, receives the notification with the necessary parameters to perform the SUBSCRIBE RPC operation. Once the SUBSCRIBE RPC operation is performed by the Context Producer against the corresponding Data Source (e.g., a router network device), the telemetry data notifications retrieved are translated to the NGSI-LD data model and are updated within the Context Broker. The data is then materialized, and the Data Consumer can subscribe to this data by interacting directly with the Context Broker. The Data Consumer can define the type of SUBSCRIBE RPC operation by using the *subscriptionMode* parameter.

To keep control of the SUBSCRIBE RPC operation lifecycle, the Data Consumer can update the *subscribeState* parameter and the rest of the parameters of the *RpcOperation* NGSI-LD model. To control the lifecycle of the SUBSCRIBE RPC operation, the Data Consumer can set the parameters of the modeled *RpcOperation* NGSI-LD entity. For example, the Data Consumer can set the *subscribeState* property of the *RpcOperation* NGSI-LD entity to stop or cancel the SUBCRIBE RPC operation. Another possibility is that the Data Consumer can set the *subscriptionMode* property to change the type of subscription and reload it. To cancel the SUBCRIBE RPC operation, the Data Consumer can also delete the corresponding *RpcOperation* NGSI-LD entity.

Figure 3.13 shows the workflow to complete a SET RPC using the NGSI-LD API and a data materialization approach. When a Data Consumer creates a *RpcOperation* NGSI-LD entity for the GET CONFIG RPC operation within the Context Broker, including the parameter *getOption* with the value *edit-config*, triggers that the Context Producer subscribes to changes of the corresponding NGSI-LD entity with the configuration data materialized within the Context Broker.

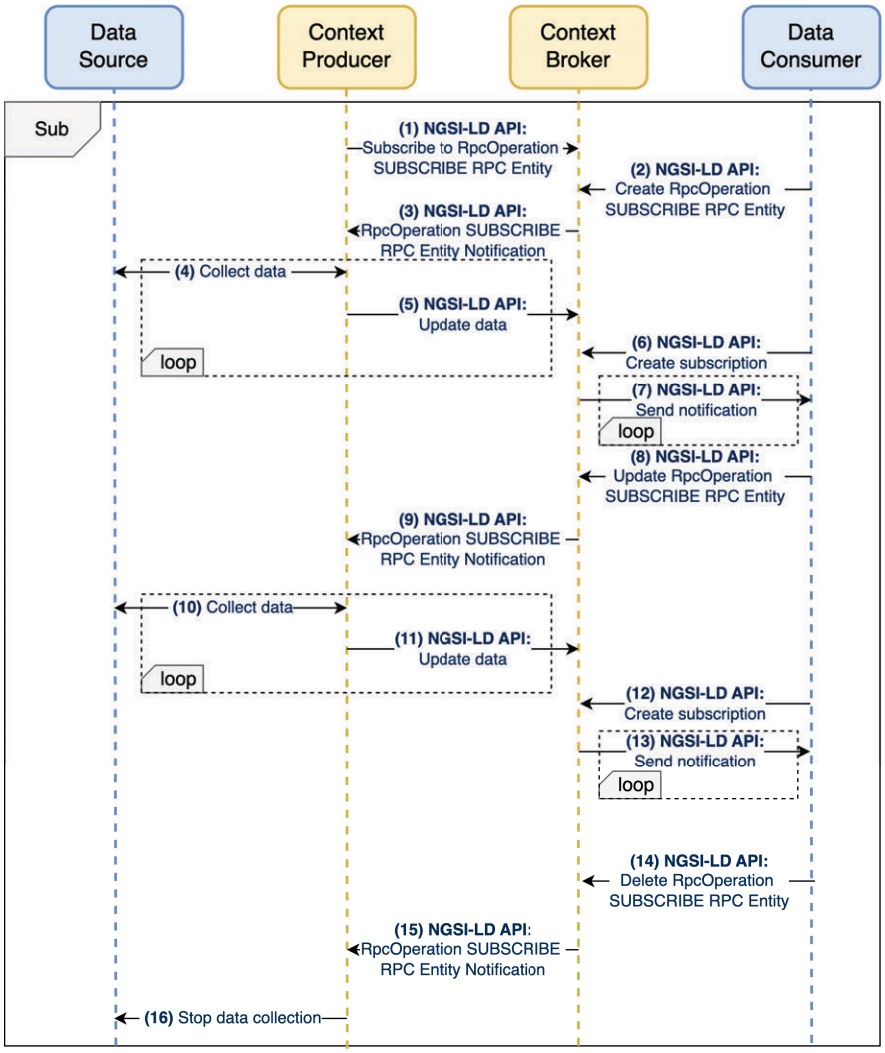


Figure 3.12: Data materialization with SUBSCRIBE RPC operation.

Then, when the Data Consumer performs an update in the retrieved NGSI-LD entity, the Context Broker generates a notification to the Context Producer to configure the SET RPC operation to apply the new configuration within the Data Source accordingly, thus replacing the last one.

When applying this new configuration with the SET RPC operation, as shown in Figure 3.13, the Context Producer must first verify the current relative configuration and delete it accordingly by executing the nested GET RPC and SET RPC operations. Once the SET RPC operation finished, the Context Producer deletes the previous subscription to the corresponding NGSI-LD entity performed within the Context Broker. Note that it is important to perform the GET RPC operation before the SET RPC operation so that the Data Consumer gets the most up-to-date source of truth for the data. If the Data Consumer tries to query only the last instance of configuration data that was previously materialized within the Context Broker, it would not get a current notion of the information, and the Data Consumer would try to update the configuration with respect to a non-truthful data instance.

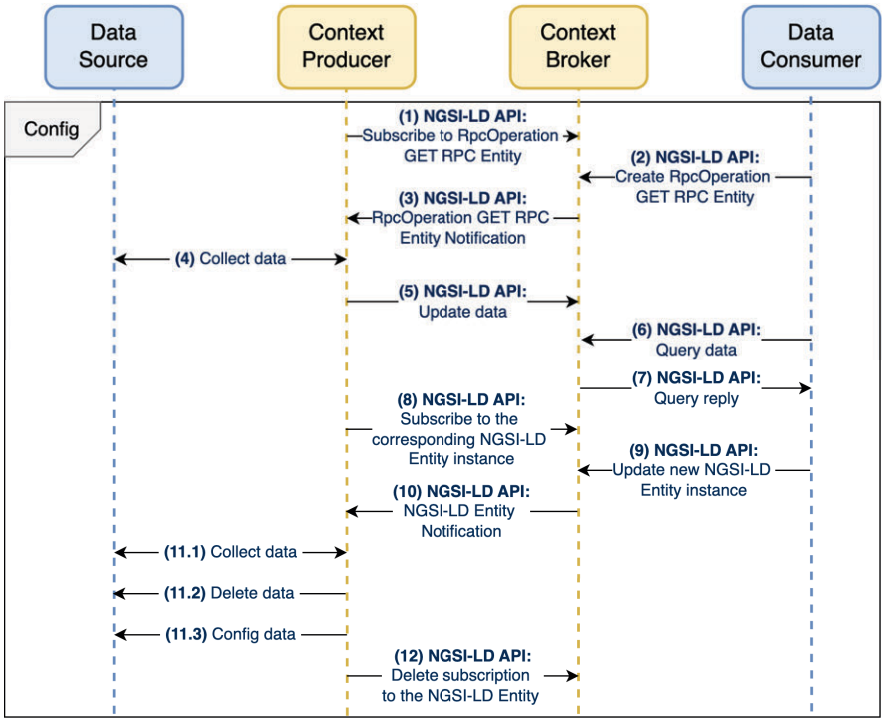


Figure 3.13: Data materialization with SET RPC operation.

NGSI-LD for network automation using a data virtualization approach

A second option for orchestrating network operations and management processes through the Descriptive NDT solution and the NGSI-LD API is a data virtualization approach. For this data virtualization approach, there is also a custom NGSI-LD model for agnostically defining the client-side key parameters of a model-driven network management protocol, such as NETCONF or gNMI clients, for network operations and management purposes in the Descriptive NDT system, which is similar to the one aforementioned for a data materialization approach. This model allows to external NetOps users or applications to define and initialize the minimal authentication and administration parameters required to manage the MDT data sources by interacting directly with the NGSI-LD Context Broker, since the query processing mechanisms for performing the network operations and management tasks are determined on-demand using ordinary NGSI-LD API operations. The implementation details for performing the NGSI-LD API query processing and the query rewriting to RPCs are described in Subsection 4.1.3 of the Chapter 4.

Figure 3.14 shows the high-level UML diagram of the NGSI-LD model used to represent client-side data for a model-driven network management protocol. As depicted in Figure 3.14, the NGSI-LD information model is composed of three different entities: 1) *Protocol* for indicating the endpoint of a network device; 2) *Credentials* for indicating the credentials of a network device; and 3) *SubscribeRpcTemplate* for indicating the needed parameters to trigger a SUBSCRIBE RPC operation.

The associated NGSI-LD models of *Protocol* and *Credentials* NGSI-LD entities are the same as those used with the data materialization approach. Meanwhile, the corresponding NGSI-LD

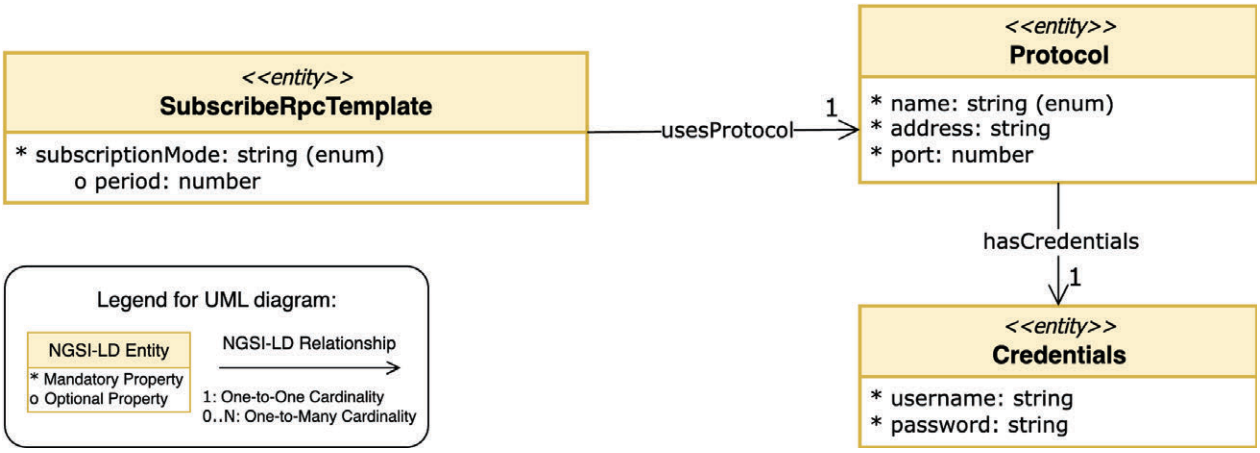


Figure 3.14: UML diagram of NGSI-LD model for network operations and management based on data virtualization approach.

model of an *SubscribeRpcTemplate* NGSI-LD entity type to indicate the parameters needed to trigger a SUBSCRIBE RPC operation defines the following attributes:

- *usesProtocol:* An NGSI-LD relationship pointing to the *Protocol* NGSI-LD entity type that specifies the model-driven network management protocol supported by a network device in server mode that will trigger the SUBSCRIBE RPC operation.
- *subscriptionMode:* An NGSI-LD property to indicate the subscription mode for the SUBSCRIBE RPC operation (i.e., on-change or periodic).
  - *period:* A subproperty of the *subscriptionMode* NGSI-LD property to indicate the period for a SUBSCRIBE RPC operation when there is a periodic subscription.

In the following, we describe the workflow diagrams that represent how to interact with the Descriptive NDT using the NGSI-LD API to trigger the various RPC operations for managing a real network based on a data virtualization approach.

Figure 3.15 shows the workflow to complete a GET RPC operation by using the NGSI-LD API and a data virtualization approach, and interacting with the NGSI-LD Context Broker. When a Data Consumer performs an HTTP query operation to get the entity corresponding to the data to be retrieved from the Data Source, the Context Broker discovers the corresponding Context Source that could provide the information. The query is then forwarded to the Context Source that triggers the GET RPC operation against the corresponding Data Source, and the configuration and operation data retrieved are translated to the NGSI-LD model and forwarded to the Context Broker. The data is then virtualized and redirected to the Data Consumer.

As explained earlier, the YANG servers have configuration and operational state data, and there is a need for additional details about the NGSI-LD API to allow data consumers to define which type of GET RPC operation (i.e., GET or GET CONFIG) they want to perform. In this regard, Annex A.3 provides details on how to handle YANG configuration and operational state data with the NGSI-LD standard.

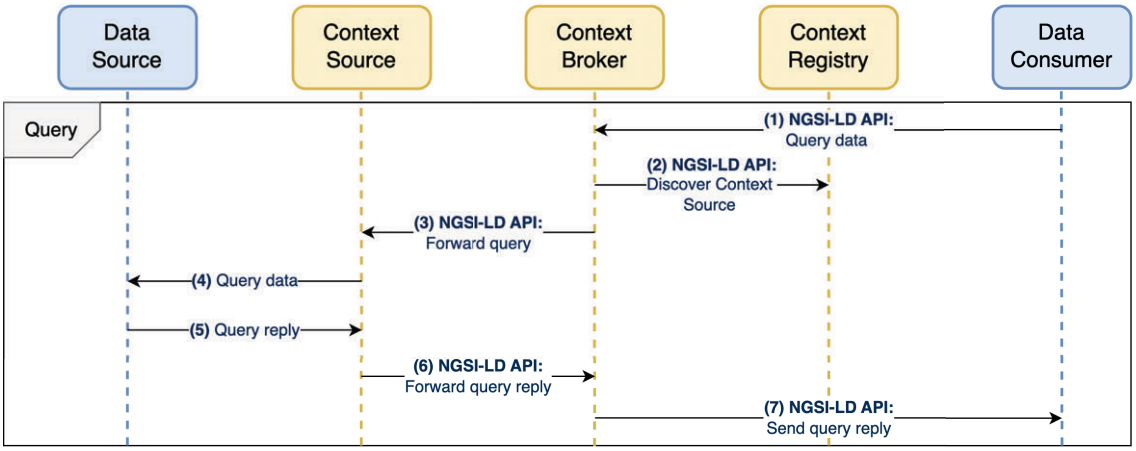


Figure 3.15: Data virtualization with GET RPC operation by interacting with the NGSI-LD Context Broker.

Figure 3.16 shows the workflow to complete a GET RPC operation using the NGSI-LD API and a data virtualization approach, and directly interact with the NGSI-LD Context Source. In this case, the Data Consumer performs an HTTP query operation to get the entity corresponding to the data to be retrieved from the Data Source by interacting with the Context Source that could provide the information. The Context Source then triggers the GET RPC operation against the corresponding Data Source, and the retrieved configuration or operational data is translated to the NGSI-LD model and sent directly to the Data Consumer.

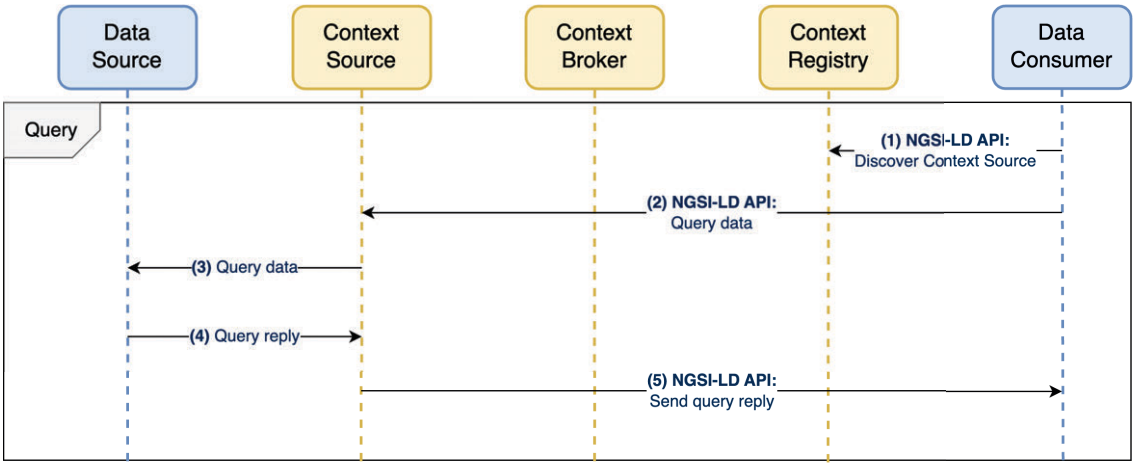


Figure 3.16: Data virtualization with GET RPC operation by interacting with the NGSI-LD Context Source.

To perform a SUBSCRIBE RPC operation, the Data Consumer must first create the *SubscribeRpcTemplate* NGSI-LD entity with the required parameters, as shown in Figure 3.17. Then, the Context Source, which is subscribed to *SubscribeRpcTemplate* entities, receives the notification with the parameters needed to perform the corresponding SUBSCRIBE RPC operation. However, the SUBSCRIBE RPC operation is only triggered when the Data Consumer performs the NGSI-LD subscribe operation via the NGSI-LD API.

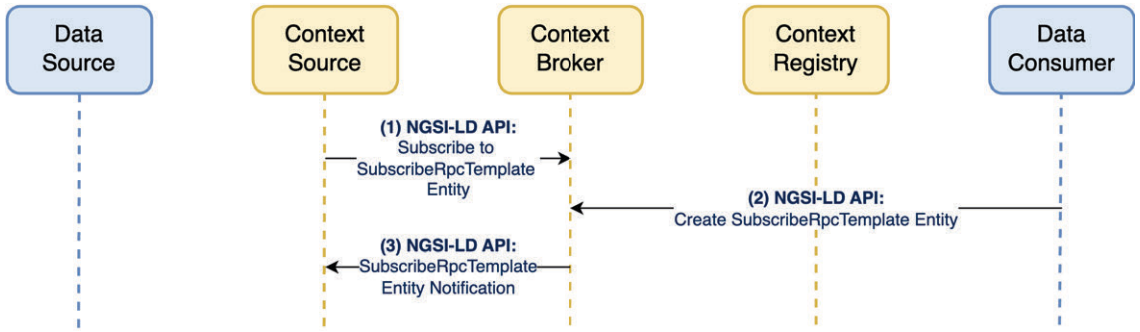


Figure 3.17: Data virtualization with initial step for triggering SUBSCRIBE RPC operations.

Figure 3.18 shows the workflow to complete a SUBSCRIBE RPC operation using the NGSI-LD API and a data virtualization approach and interacting with the NGSI-LD Context Broker. When a Data Consumer performs an HTTP subscription operation to subscribe to an entity for which it wants to receive notifications, the Context Broker discovers the corresponding Context Source that can provide the information. Then the subscription request is forwarded to the Context Source that triggers the SUBSCRIBE RPC operation against the corresponding Data Source, and the telemetry notification data are translated into the NGSI-LD model and forwarded to the Context Broker. The data is then virtualized and redirected to the Data Consumer. The Data Consumer can define the type of SUBSCRIBE RPC operation by using the *subscriptionMode* parameter.

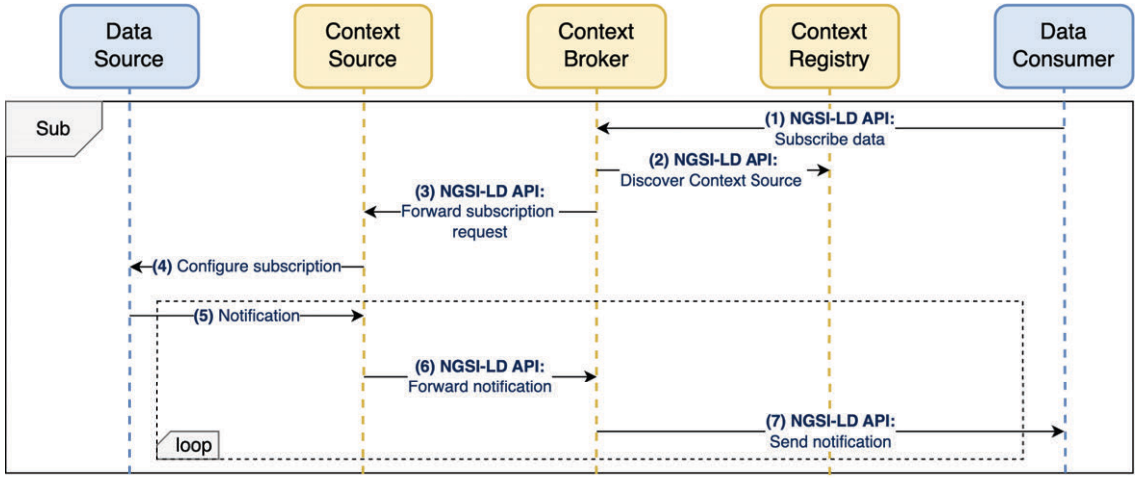


Figure 3.18: Data virtualization with SUBSCRIBE RPC operation by interacting with the NGSI-LD Context Broker.

Figure 3.19 shows an alternative workflow for completing a SUBSCRIBE RPC using the NGSI-LD API and a data virtualization approach. In this case, the Data Consumer performs an HTTP subscription operation to subscribe to an entity for which it wants to be notified by interacting directly with the Context Source that can provide the information. The Context Source then triggers the SUBSCRIBE RPC operation against the corresponding Data Source, and the notification data is translated into the NGSI-LD model and sent directly to the Data Consumer.

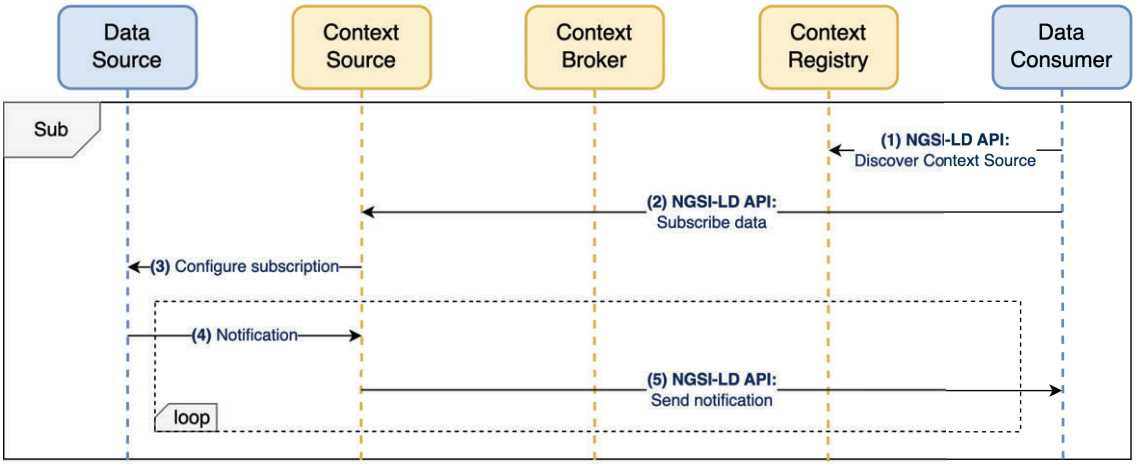


Figure 3.19: Data virtualization with SUBSCRIBE RPC operation by interacting directly with the NGSI-LD Context Source.

Figure 3.20 shows the workflow to complete a SET RPC operation using the NGSI-LD API and a data virtualization approach. The SET RPC operation is commited after a Data Consumer performs an HTTP GET request using the NGSI-LD API to query and retrieve an NGSI-LD entity instance with the latest configuration data from the Data Source by triggering a GET CONFIG RPC operation. Triggering this GET CONFIG RPC operation using the NGSI-LD API allows data consumers to obtain the most up-to-date data instance with the source of truth of the current configuration before it is updated.

The Data Consumer can then perform an HTTP POST request by interacting directly with the Context Source in order to create a new NGSI-LD entity instance with new values. This triggers the Context Source to configure a SET RPC operation to apply the configuration changes within the Data Source accordingly, thus replacing the last one.

Similarly to the data materialization approach, when applying this new configuration with the SET RPC operation, as shown in Figure 3.20, the Context Producer must first verify the current relative configuration and delete it accordingly by executing the nested GET RPC and SET RPC operations [3.5].

[3.5]Depending on the client-side and server-side capabilities of the network management protocol used, the SET RPC operation allows deciding in which server datastore to store the new configurations, either overwriting the current configurations (i.e., *configuration* datastore), setting possible candidate configurations (i.e., *candidate* datastore) or configurations available at the next server restart (i.e., *startup* datastore). In the current implementation for orchestrating SET RPC operations using the NGSI-LD API, whether a data materialization approach or a data virtualization approach is used, there is no control over the target datastore where the configurations are set and the previous configurations are directly overwritten.

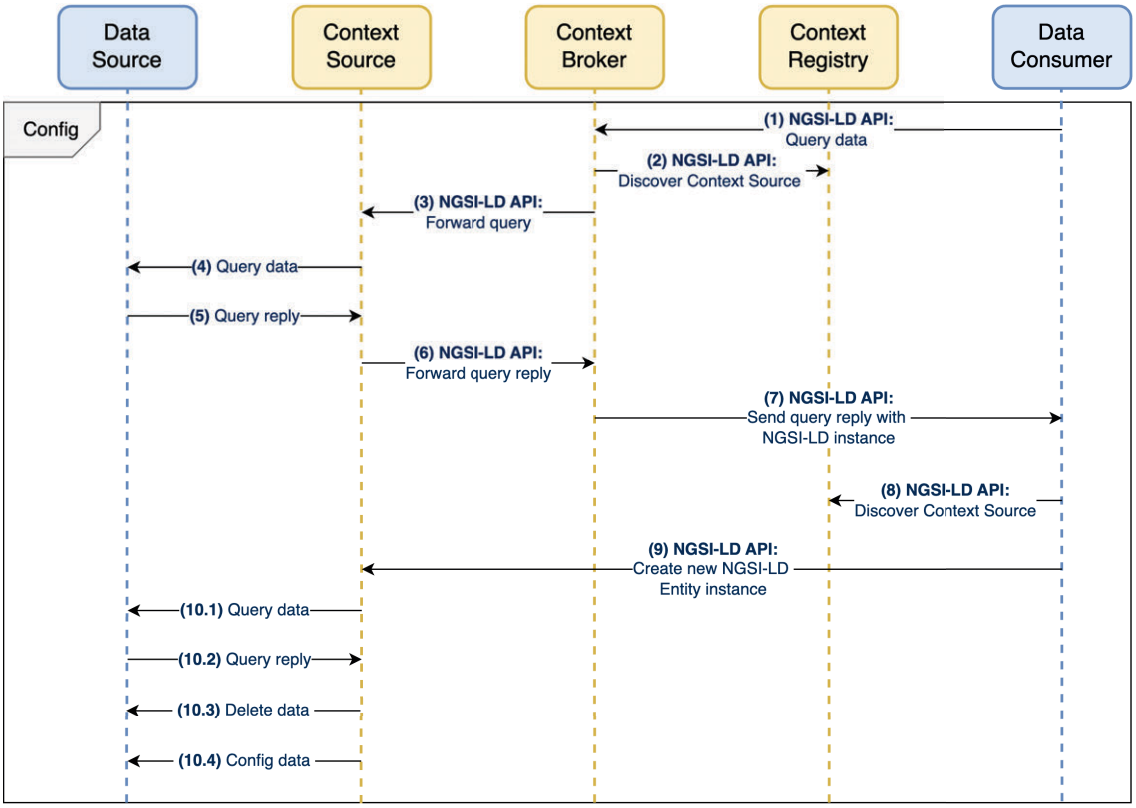


Figure 3.20: Data virtualization with SET RPC operation.

3.3 Overview of the NDT Proposal

This Chapter 3 describes the design and methodology aspects for the modeling of an NDT solution, which is focused on network operations and management purposes. The NDT design and architectural principles are based on the NDT vision provided by the IETF and the application of the NGSI-LD standard, both already introduced in the State of the Art of Chapter 2. The presented solution proposes model-driven network management mechanisms based on the use of the NGSI-LD standard, formalizing standardized interfaces and representation data models that facilitate the adaptation and implementation of an NDT system. The methodology for modeling the proposed NDT solution is based on 4 main principles: 1) addressing data integration mechanisms using the NGSI-LD standard; 2) convention rules for translating data modeled in YANG to data modeled in NGSI-LD; 3) a functional NDT architecture that brings together the design and architectural principles of an NDT for the representation and management of operational data, configuration data, and telemetry statistics from real networks; and 4) determining how to address automatable mechanisms in the proposed NDT solution to cover network operations and management tasks by directly using the NGSI-LD standard. Chapter 4 will present the implementation of an NDT system that meets the requirements to address the proposal just described in Chapter 3.

Chapter 4

Descriptive NDT System for Network Operations and Management

This chapter includes the prototype implementation details of the Descriptive NDT solution proposed in this Doctoral Thesis. The chapter describes the prototype implementation for an NDT solution for network operations and management. Then, a case study is presented to showcase different examples of network operations and management requirements that can be addressed by this NDT solution.

Table of Contents

4.1 A Programmatic Digital Mapping System for Network Operations and Management 98

4.1.1 Descriptive NDT Generator Plane 98

4.1.2 Data Integration Plane Pipeline 103

4.1.3 Query processing and rewriting for model-driven network management purposes 106

4.2 Case Study: Descriptive NDT for Network Operations and Management 111

4.2.1 IETF-based Descriptive NDT model 111

4.2.2 OpenConfig-based Descriptive NDT model 115

4.2.3 Descriptive NDT model for NetFlow v9 protocol 119

4.2.4 Application of the NGSI-LD Descriptive NDT for network operations and management purposes 123

4.3 Overview of the Descriptive NDT System 134

4.1 A Programmatic Digital Mapping System for Network Operations and Management

This section introduces the implementation of a Descriptive NDT solution that is in line with the reference architecture defined in Subsection 3.2.3 of Chapter 3. The Descriptive NDT prototype is based on a programmatic digital mapping solution between real network data and NDT data that consists of a set of developed software components corresponding to the functional building blocks considered in the Descriptive NDT architecture. Figure 4.1 shows the different software components of the Descriptive NDT prototype[4.1] and their interrelationships. This implementation has been validated for network automation purposes, covering network operations and management processes. There is an open-source, Docker microservices-based prototype implementation publicly available on GitHub [173].

The prototype implements actuation mechanisms based on the NGSI-LD API to obtain configuration and state data from real networks, apply configuration changes, and request telemetry data on demand by using data integration approaches, thus facilitating network automation. The validated prototype adopts a hybrid data integration approach based on both data materialization and data virtualization. The methodology is targeted to MDT network assets based on the YANG data modeling language. The system separates its functionality into two main planes: i) Descriptive NDT Generator Plane; and ii) Data Integration Plane.

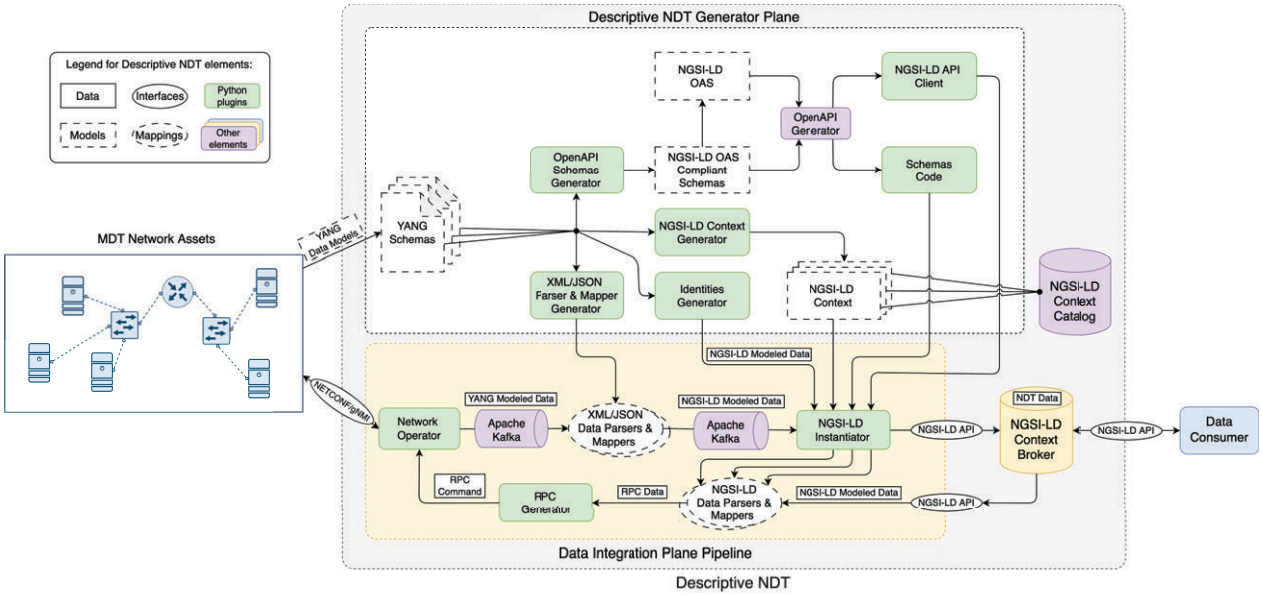


Figure 4.1: A prototypical implementation of the Descriptive NDT architecture. It is based on a programmatic digital mapping system tailored to MDT network assets.

4.1.1 Descriptive NDT Generator Plane

The general functionality of the Descriptive NDT Generator Plane is highlighted in Figure 4.2. It consists of a series of components that process the native data models of the network

[4.1] An extended view of the Descriptive NDT prototype diagram is provided in Annex A.1.

assets (i.e., the YANG Data Models in Figure 4.2) to generate different programming artifacts that enable autonomous YANG to NGSI-LD and NGSI-LD to RPC translations. The Data Integration Plane Pipeline then uses the resulting artifacts to parse the NGSI-LD API operations and generate RPC commands to perform actions on network nodes, but also to retrieve and parse the resulting network monitoring data and integrate them as NDT data.

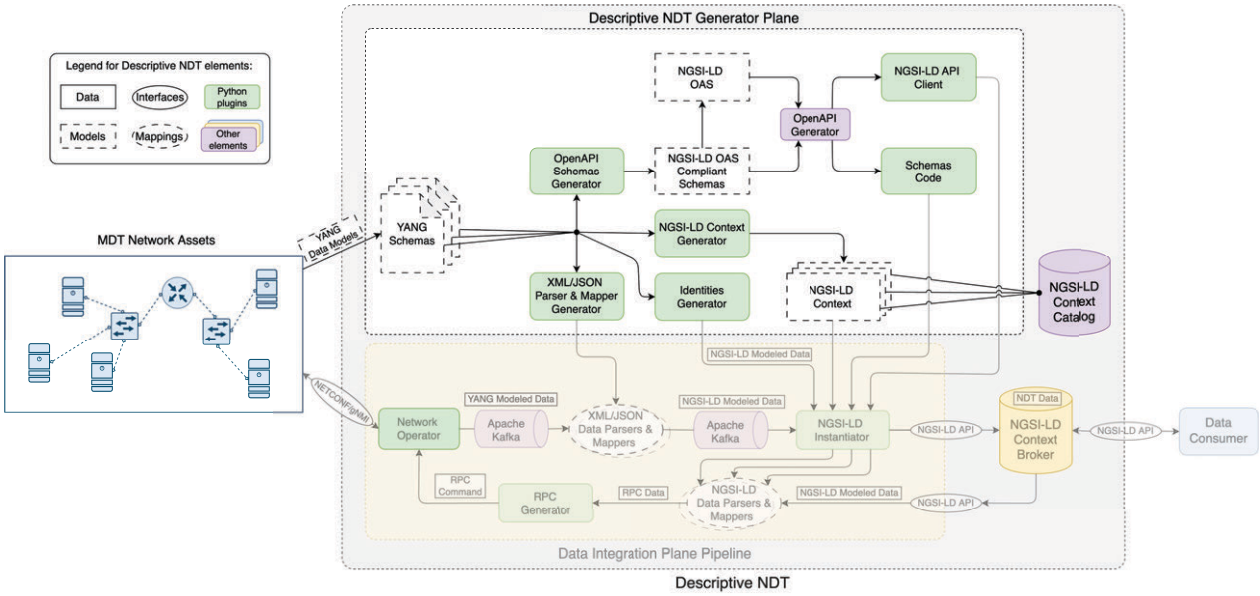


Figure 4.2: Descriptive NDT Generator Plane functionality.

The programming artifacts from the Descriptive NDT Generator Plane are generated automatically by processing YANG Schemas (i.e., the different schemas provided by the YANG Data Models). For this purpose, an open-source tool named *pyang* [174] has been used. *Pyang* is a YANG validator, transformer, and code generator written in the Python programming language, which includes features that make it an ideal solution to code the necessary plugins to generate the programming artifacts. These generator plugins are the following:

- *OpenAPI Schemas Generator:* Given one or more YANG schemas, the OpenAPI Schemas Generator dynamically generates OpenAPI Schemas compliant with the NGSI-LD OpenAPI Specification (NGSI-LD OAS) [175]. The generated schemas represent the knowledge baseline for the resulting NGSI-LD domain model associated with the input YANG schemas. Then, the *OpenAPI Generator* [176] tool is used to generate the stub code to manipulate the information modeled in the schemas in an object-oriented programming form based on the Python language (i.e., Schemas Code in Figure 4.2). The resulting programming code is based on the *pydantic* library [177], which provides Python classes with data parsing and data validation functionalities. In addition, the *OpenAPI Generator* tool allows to automatically generate source code from the NGSI-LD OAS for a Python-based NGSI-LD API client, which provides the different operations and options of the NGSI-LD API in a programmatic way. By using the Python classes for the particular schemas associated with an MDT network asset, we can build data instances that are used as payloads in the operations defined by the

NGSI-LD API client to manage the instantiation of the required NGSI-LD modeled data. In addition, there are OpenAPI schemas dedicated to modeling the orchestration of RPC operations for both data materialization and data virtualization approaches, as described in Subsection 3.2.4 of Chapter 3. Similarly, Python code is generated from these OpenAPI schemas for orchestrating RPCs to automate network operations and management tasks.

Figure 4.3 shows the OAS Schema for modeling the information of network interface statistics according to IETF RFC 8343 [132] as an NGSI-LD entity type named *InterfaceStatistics*. One of their attributes, *inOctets*, is modeled as a particular NGSI-LD property and has its own OAS Schema as represented in Figure 4.4.

```yaml
InterfaceStatistics:
  description: |
    A collection of interface-related statistics objects.

    YANG module: ietf-interfaces.yang
  allOf:
    - $ref: 'https://forge.etsi.org/rep/cim/ngsi-ld-openapi/-/raw/master/ngsi-ld-api.yaml#/components/schemas/Entity'
    - type: object
      properties:
        type:
          description: NGSI-LD Entity identifier. It has to be InterfaceStatistics.
          type: string
          enum:
            - InterfaceStatistics
          default: InterfaceStatistics
        discontinuityTime:
          $ref: '#/components/schemas/DiscontinuityTime'
        inOctets:
          $ref: '#/components/schemas/InOctets'
        inUnicastPkts:
          $ref: '#/components/schemas/InUnicastPkts'
        inBroadcastPkts:
          $ref: '#/components/schemas/InBroadcastPkts'
        inMulticastPkts:
          $ref: '#/components/schemas/InMulticastPkts'
        inDiscards:
          $ref: '#/components/schemas/InDiscards'
        inErrors:
          $ref: '#/components/schemas/InErrors'
        inUnknownProtos:
          $ref: '#/components/schemas/InUnknownProtos'
        outOctets:
          $ref: '#/components/schemas/OutOctets'
        outUnicastPkts:
          $ref: '#/components/schemas/OutUnicastPkts'
        outBroadcastPkts:
          $ref: '#/components/schemas/OutBroadcastPkts'
        outMulticastPkts:
          $ref: '#/components/schemas/OutMulticastPkts'
        outDiscards:
          $ref: '#/components/schemas/OutDiscards'
        outErrors:
          $ref: '#/components/schemas/OutErrors'
        isPartOf:
          $ref: '#/components/schemas/IsPartOf'
          description: isPartOf Relationship with Entity type Interface.
    - required:
      - type
      - isPartOf
      - discontinuityTime
```

Figure 4.3: OAS Schema for *InterfaceStatistics* NGSI-LD entity type.

- *XML/JSON Parser & Mapper Generator:* Given one or more YANG schemas, the XML/JSON Parser & Mapper Generator generates the Python code of the XML and JSON parsers that read the data modeled by these schemas and generate the corresponding NGSI-LD meta-model data structures. Therefore, this plugin addresses the translation conventions described in Subsection 3.2.2 to implement the YANG to NGSI-LD mapping process. For the prototype implementation, we developed generators for data retrieved by query and subscription RPCs using the NETCONF and gNMI network management protocols. Examples of configuration data, operational status

```
InOctets:
  description: |
    The total number of octets received on the interface,
    including framing characters.

    Discontinuities in the value of this counter can occur
    at re-initialization of the management system, and at
    other times as indicated by the value of
    'discontinuity-time'.

    Reference: RFC 2863: The Interfaces Group MIB - ifHCInOctets

    YANG module: ietf-interfaces.yang
  additionalProperties: false
  allOf:
    - $ref: 'https://forge.etsi.org/rep/cim/ngsi-ld-openapi/-/raw/master/ngsi-ld-api.yaml#/components/schemas/Property'
    - type: object
      properties:
        value:
          type: integer
      required:
        - value
```

Figure 4.4: OAS Schema for *inOctets* NGSI-LD property.

data, and telemetry notifications obtained by NETCONF and gNMI, both in raw format (i.e., XML and JSON data encoding formats) and in the NGSI-LD-based target representation format, are provided in Annex A.2.

- *NGSI-LD Context Generator:* NGSI-LD uses context entries to map URIs from expanded full versions to compacted short versions for the NGSI-LD elements represented in particular domain models. Given one or more YANG schemas, the NGSI-LD Context Generator produces the corresponding NGSI-LD context files in JSON-LD format. These context files are structured as key-value dictionaries, where each key is the NGSI-LD element type that matches the corresponding YANG data node, and each value is a URN (Uniform Resource Name) in XPath notation that allows navigation within the YANG data model to locate the associated YANG concept. The NGSI-LD Context Generator provides a context file for each resulting NGSI-LD entity type, where the key-value pairs within the context file represent the resulting NGSI-LD context entries associated with the NGSI-LD entity and its NGSI-LD attributes. Figure 4.5 shows an example about the organization of different NGSI-LD Context URLs, which are accessible within a Context Catalog service, for representing the context related to the *ietf-interfaces* YANG module [132] in its 2014-05-08 revision (see Subsection 4.2.1 from Chapter 4 for getting more details about the NGSI-LD model associated with this YANG module). Figure 4.5 also shows the context file for getting the context entries associated with the *InterfaceConfig* NGSI-LD entity that represents the context for configuration data about network interfaces.

  In addition to maintaining the catalog of NGSI-LD context files, the Context Catalog service also maintains additional NGSI-LD metadata files. Like the NGSI-LD context files, these NGSI-LD metadata files are independent of each resulting NGSI-LD entity and also collect the NGSI-LD context entries associated with each entity. However, these NGSI-LD metadata files contain additional metadata. On the one hand, if the resulting NGSI-LD entity is configurable, the metadata file contains a dictionary of NGSI-LD elements, both the NGSI-LD entity itself and its attributes, which represent configuration data. On the other hand, if the resulting NGSI-LD entity comes from a YANG data node of type list, the metadata file will also include the key of that YANG list, so that it can be inferred in query processing when the resulting RPC operation

is performed (see Subsection 4.1.3 from Chapter 4 for getting more details about the query processing mechanisms implemented by the Descriptive NDT system). Figure 4.6 shows the NGSI-LD metadata file for representing the context entries associated with the *InterfaceConfig* NGSI-LD entity, as well as the dictionary of configurable NGSI-LD elements and the key associated with the original YANG list data node.

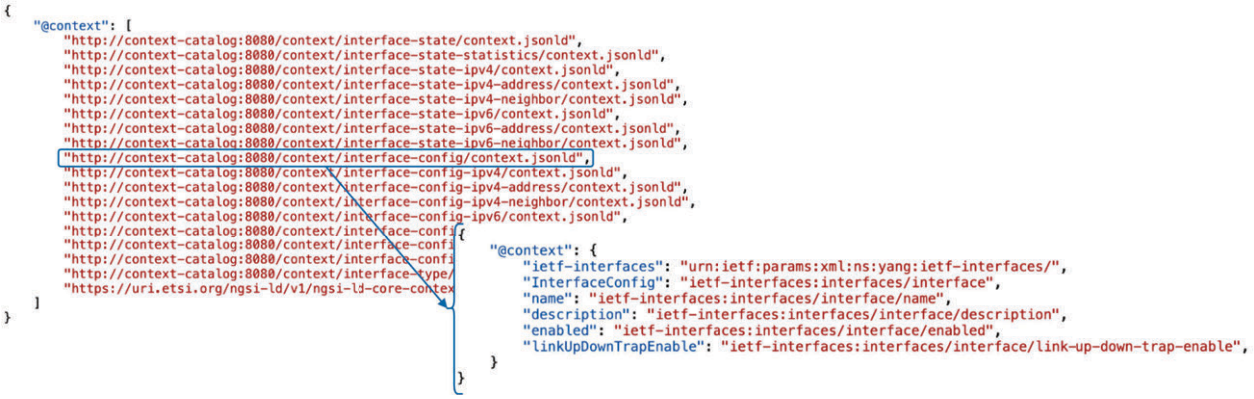


Figure 4.5: NGSI-LD context associated with *InterfaceConfig* entity type.


```
{
    "@context": {
        "ietf-interfaces": "urn:ietf:params:xml:ns:yang:ietf-interfaces/",
        "InterfaceConfig": "ietf-interfaces:interfaces/interface",
        "name": "ietf-interfaces:interfaces/interface/name",
        "description": "ietf-interfaces:interfaces/interface/description",
        "enabled": "ietf-interfaces:interfaces/interface/enabled",
        "linkUpDownTrapEnable": "ietf-interfaces:interfaces/interface/link-up-down-trap-enable"
    },
    "key": "name",
    "config": [
        "InterfaceConfig",
        "name",
        "description",
        "enabled",
        "linkUpDownTrapEnable"
    ]
}
```


Figure 4.6: NGSI-LD metadata associated with *InterfaceConfig* entity type.

- *Identities Generator*: Given one or more YANG modules, the Identities Generator discovers the resulting YANG identities and generates the corresponding NGSI-LD meta-model data structures, which are directly instantiated as NGSI-LD modeled data in the form of NGSI-LD entities. The plugin processes YANG modules that define identities, such as the *iana-if-type* YANG module that contains the YANG definitions of interface types registered by the Internet Assigned Numbers Authority (IANA). In addition, the plugin allows to manage YANG module imports, if any, in order to dynamically discover if there are identities defined in those imported modules. To use this plugin, the network operator must run it before running the XML/JSON Parser & Mapper Generator, so that the NGSI-LD entities for every possible YANG identity are already created, instantiated and uploaded to the Context Broker. Figure 4.7 depicts an

instance of the *InterfaceConfig* NGSI-LD entity type whose *interfaceType* attribute is an NGSI-LD relationship that points to an NGSI-LD entity of type *YANGIdentity* that indicates that the interface is an Ethernet interface. In addition, this *YANGIdentity* entity refers to another *YANGIdentity* entity that represents a broader or global baseline definition for network interfaces as defined by IANA in the corresponding *iana-if-type* YANG module. In addition, the associated *YANGIndentity* entity type has a global context. Figure 4.8 depicts an example about the organization of different NGSI-LD Context URLs accessible for representing the context related to the *ietf-interfaces* YANG model and, in particular, the different context entries to obtain the metadata associated with the *YANGIdentity* NGSI-LD entity.



Figure 4.7: Details of a *YANGIdentity* entity associated with the case of network interface type as specified by the *ietf-interfaces* and *iana-if-type* standardized IETF YANG models.

4.1.2 Data Integration Plane Pipeline

Based on the functionality of the Descriptive NDT Generator Plane components, the Data Integration Plane Pipeline can handle the full data integration process required to complete both the YANG to NGSI-LD data translation and the NGSI-LD to RPC data translation. The NGSI-LD Context Broker is the source and target of the Data Integration Plane Pipeline that handles requests and associated responses formulated to the Descriptive NDT via the NGSI-LD API by external NetOps consumers. Figure 4.9 considers that a Data Consumer can request different operations using the NGSI-LD API for 1) querying operational state and configuration data from MDT network assets; 2) configuring MDT network assets; and 3) subscribing to telemetry data coming from MDT network assets.

```
{
    "@context": [
        "http://context-catalog:8080/context/interface-stat
        "http://context-catalog:8080/context/interface-stat
        "http://context-catalog:8080/context/interface-stat
        "http://context-catalog:8080/context/interface-stat
        "http://context-catalog:8080/context/interface-stat
        "http://context-catalog:8080/context/interface-stat
        "http://context-catalog:8080/context/interface-stat
        "http://context-catalog:8080/context/interface-stat
        "http://context-catalog:8080/context/interface-conf
        "http://context-catalog:8080/context/interface-conf
        "http://context-catalog:8080/context/interface-config-ipv4-address/context.jsonld",
        "http://context-catalog:8080/context/interface-config-ipv4-neighbor/context.jsonld",
        "http://context-catalog:8080/context/interface-config-ipv6/context.jsonld",
        "http://context-catalog:8080/context/interface-config-ipv6-address/context.jsonld",
        "http://context-catalog:8080/context/interface-config-ipv6-neighbor/context.jsonld",
        "http://context-catalog:8080/context/interface-config-ipv6-autoconf/context.jsonld",
        "http://context-catalog:8080/context/interface-type/context.jsonld",
        "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"
    ]
}
```

{
    "@context": {
        "ietf-yang": "http://ietf.yang.org#",
        "YANGIdentity": "ietf-yang:YANGIdentity",
        "description": "ietf-yang:description",
        "identifier": "ietf-yang:identifier",
        "namespace": "ietf-yang:namespace",
        "broader": "http://www.w3.org/2004/02/skos/core#broader"
    }
}

Figure 4.8: NGSI-LD context associated with *YANGIdentity* entity type.

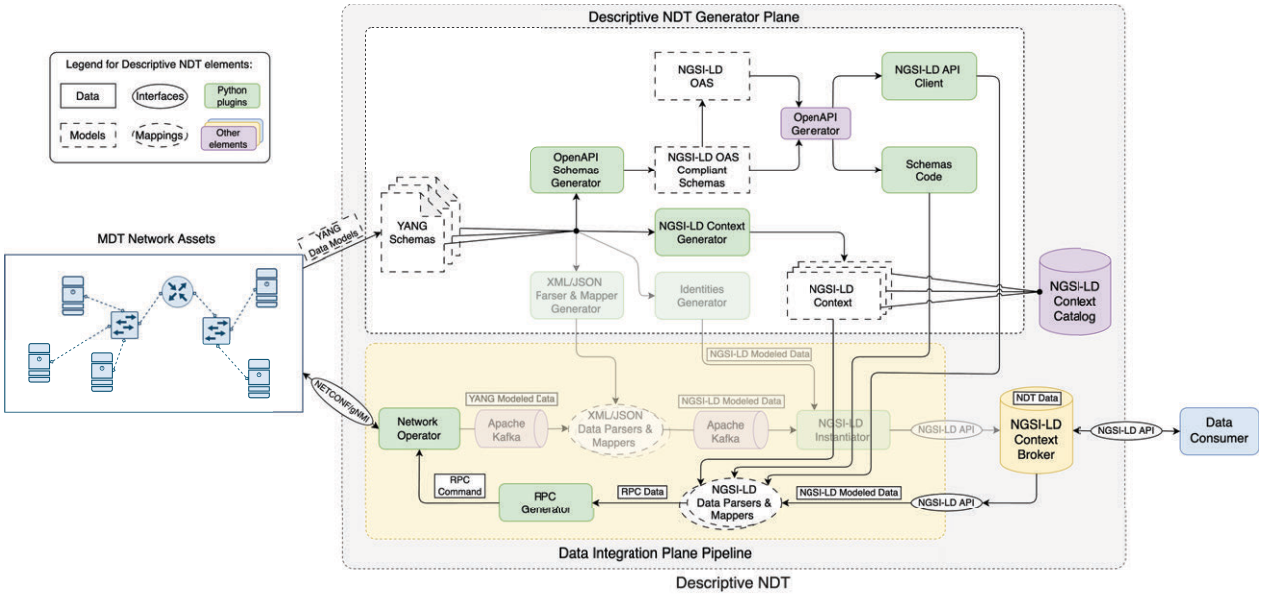


Figure 4.9: Data Integration Plane Pipeline (Phase 1). Network Operator executing RPC commands.

The focus here is on how the NGSI-LD API operations are translated to the corresponding model-driven network management protocol operations. The NGSI-LD Data Parsers & Mappers components address this translation process by converting the NGSI-LD API operation data to corresponding RPC data to fulfill network operations and management tasks. To complete this process, the NGSI-LD Data Parsers & Mappers components are subscribed to the corresponding NGSI-LD API operations by using the NGSI-LD API client and use the Python classes of the source schemas in order to parse the NGSI-LD modeled data accordingly.

In addition, the NGSI-LD Data Parsers & Mappers components process the NGSI-LD Context to discover the NGSI-LD context entries associated with the data requested within the NGSI-LD API operation. For the different RPC cases, these context entries allow determining the hierarchical structure of the XPath or schema tree required to complete the

RPC data structure. The complete process for processing NGSI-LD API queries and rewriting them as RPC queries is explained in Subsection 4.1.3.

Once the RPC data structure is created, the RPC Generator is responsible for receiving the structured RPC data and the network management protocol credentials to be able to apply the RPC commands. Finally, the Network Operator component commits the RPC commands to the corresponding MDT network assets. This component is based on a Python microservice that uses the NETCONF and gNMI protocol clients as network management interfaces.

As a result of the RPC operation executed by the Network Operator, a data collection process can be triggered from the network. Then, this Network Operator component enables the collection of telemetry, operational state, and configuration data from MDT network assets for its integration into the Descriptive NDT as shown in Figure 4.10. The data is modeled according to the YANG models and need to be mapped to the corresponding representation of the NGSI-LD model. To cover this translation process, the XML/JSON Data Parsers & Mappers components perform the parsing of the YANG modeled data encoded in XML or JSON format and the mapping process to the corresponding NGSI-LD modeled data in the form of Python dictionary data structures.

The NGSI-LD Instantiator component is then responsible for receiving the NGSI-LD modeled data and validating it against the Python classes of the source schemas. During this process, the NGSI-LD Instantiator ensures compatibility and consistency of the data with the appropriate NGSI-LD model. The exchange of streaming data between the aforementioned components of the pipeline is carried out by using a message bus system based on Apache Kafka [178]. Finally, the NGSI-LD Instantiator uses instantiated Python objects with the NGSI-LD API client to materialize or virtualize the resulting NGSI-LD modeled data within a compatible NGSI-LD Context Broker. There, interested consumers can access the data.

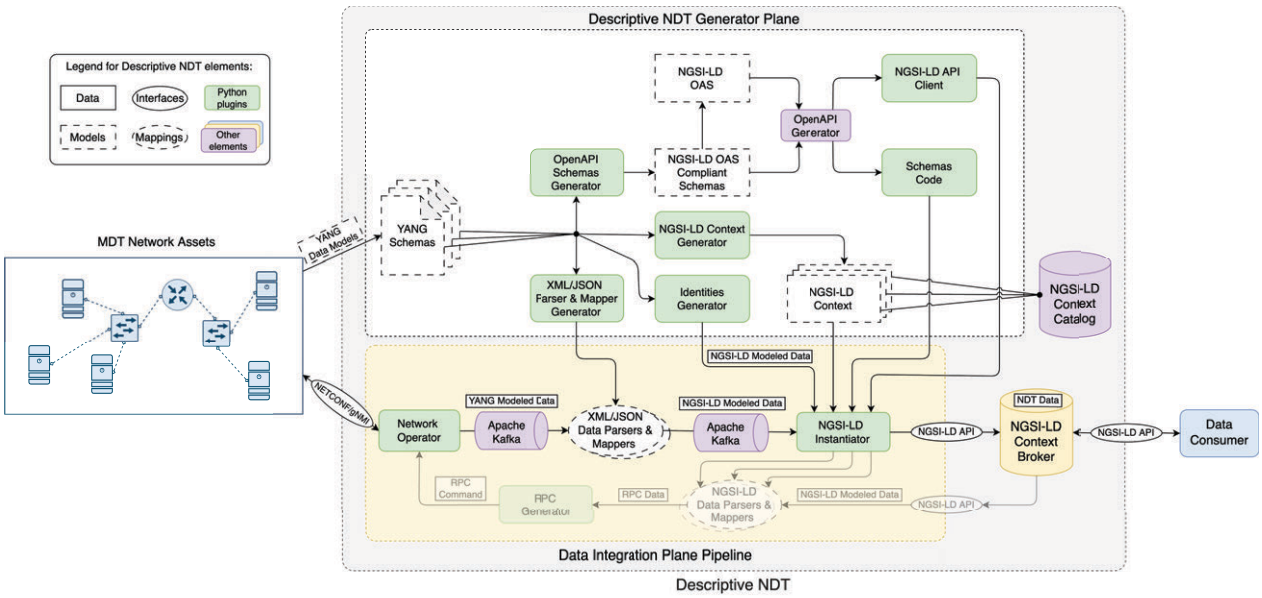


Figure 4.10: The Data Integration Plane Pipeline (Phase 2). Network Operator collecting configuration, operational state, and telemetry data.

4.1.3 Query processing and rewriting for model-driven network management purposes

This subsection describes how our Descriptive NDT prototype solution implements the NGSI-LD Data Parsers & Mappers components for processing the NGSI-LD API queries dedicated to model-driven network management purposes and rewrite them as RPC queries. Although the methodology can be applied to any model-driven network management protocol, it has been fully validated only for NETCONF, which has been used as an example application (see Annex A.4 for more details on complete examples dedicated to network interface management using the NETCONF protocol). The following subsections explain at a high level how to handle each of the basic model-based network management operations: i) querying operational status data from a network device (i.e., GET RPC); ii) subscription to telemetry statistics from a network device (i.e., SUBSCRIBE RPC); iii) querying configuration data from a network device (i.e., GET CONFIG RPC); and iv) updating the configuration of a network device (i.e., SET RPC).

Although query processing and rewriting processes are commonly used in data integration for data virtualization approaches, in our solution, they are also applied to the data materialization approach to trigger network operations and management processes on the data sources and to replicate the data collected from these sources to the NGSI-LD Context Broker. Therefore, the methodology followed in this subsection applies to both data integration approaches.

Querying operational status data from a network device

Figure 4.11 shows the general workflow we have followed in the Descriptive NDT prototype to process a query performed from a data consumer by using the NGSI-LD API and rewrite it to a GET RPC operation in order to retrieve operational status data from a network device.

In the data virtualization approach, the data consumer performs the query NGSI-LD entity operation to query entities by type and also by *id*, if specified. In the data materialization approach, the data consumer creates an *RpcOperation* NGSI-LD entity for GET RPC operations by using the NGSI-LD API, specifying the entity type and optionally the entity *id*. Then, the query processing solution looks for the entity type in the Context Catalog service. As explained before, each context entry has a key with the NGSI-LD element type in short form (i.e., type name) and a value with the NGSI-LD element type in long form (i.e., URI form). The query processing solution allows the data consumer to specify the entity type in either form. If the entity type is found, either by its short or long version, the query processing solution takes the corresponding context entry value (i.e., the URI form of the entity type), which follows a structure closely similar to an XPath notation that allows the network management protocol to navigate the target YANG data model to locate the associated YANG concept accordingly. Finally, the query rewriting process comes into play by building the XPath filter accordingly and generating the corresponding GET RPC operation to request the operational data from the target network device. In case the data consumer specifies an entity *id*, then the XPath filter will add this *id* as an index or key value in the corresponding subpath if possible to indicate that the consumer wants to retrieve just the information corresponding to this particular entity *id*. In this situation, the query processing

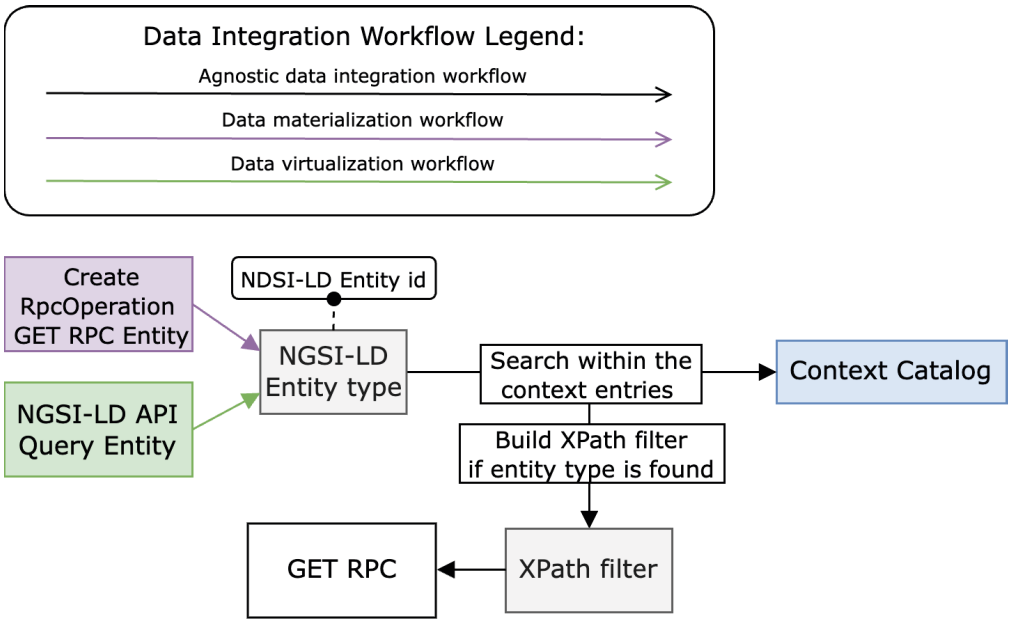


Figure 4.11: Query processing and rewriting general workflow for performing GET RPC model-driven network management operations.

solution infers which target data node types in YANG have indexes or keys (i.e., YANG data nodes of type *list*).

Annex A.4.1 provides examples of the workflow for processing a query entity operation performed by a data consumer using the NGSI-LD API and rewriting it to a GET RPC operation to retrieve operational status data from a network interface using the NETCONF management protocol for both NMDA compliant and non-compliant network devices.

Subscription to telemetry statistics from a network device

Figure 4.12 shows the general workflow followed by the Descriptive NDT prototype to process a subscription performed from a data consumer by using the NGSI-LD API and rewrite it to a SUBSCRIBE RPC operation in order to receive notification about telemetry statistics from a network device.

In a data virtualization approach, the data consumer performs the create subscription operation by using the NGSI-LD API to create a subscription to entities by type and also by *id*, if specified. In a data materialization approach, the data consumer creates an *RpcOperation* NGSI-LD entity for SUBSCRIBE RPC operations by using the NGSI-LD API, specifying the entity type and optionally the entity *id*. Then, the query processing solution looks for the entity type in the Context Catalog service. If the entity type is found, either by its short or long version, the query processing solution takes the corresponding context entry value (i.e., the URI form of the entity type), which follows a structure closely similar to an XPath notation and, as already mentioned, allows the network management protocol to navigate the target YANG data model to find the specific YANG concept. Finally, the query rewriting process comes into play by building the XPath filter accordingly and generating the

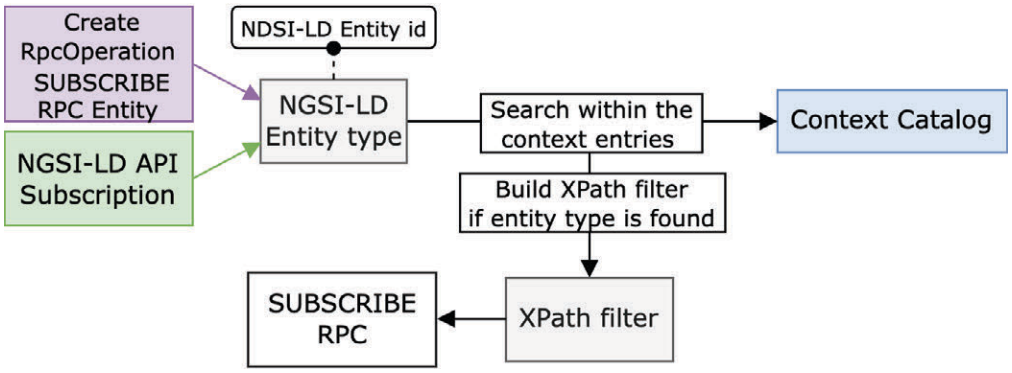


Figure 4.12: Query processing and rewriting general workflow for performing SUBSCRIBE RPC model-driven network management operations.

corresponding SUBSCRIBE RPC operation to subscribe to operational data from the target network device. Depending on the subscription details defined within the NGSI-LD model for triggering RPCs, the subscription can be either periodic, in order to receive notifications about the corresponding operational data periodically, or on-change, to receive notifications when there are changes within the corresponding operational data. In addition, through the subscription creation operation performed via the NGSI-LD API, the data consumer can also indicate from which NGSI-LD properties it needs to receive notifications.

Annex A.4.2 provides examples of the workflow for processing a subscribe entity operation performed from a data consumer by using the NGSI-LD API and rewriting it to a SUBSCRIBE RPC operation in order to receive notifications about telemetry statistics from a network interface using NETCONF for both NMDA compliant and non-compliant network devices.

Querying configuration data from a network device

Figure 4.13 shows the general workflow followed by the Descriptive NDT prototype to process a query performed from a data consumer by using the NGSI-LD API and rewrite it to a GET CONFIG RPC operation in order to retrieve configuration data from a network device.

In a data virtualization approach, the data consumer performs the query NGSI-LD entity operation to query entities by type and also by *id*, if specified. In a data materialization approach, the data consumer creates an *RpcOperation* NGSI-LD entity for GET CONFIG RPC operations by using the NGSI-LD API, specifying the entity type and optionally the entity *id*. Then, the query processing solution looks for the entity type in the Context Catalog service. If the entity type is found, either by its short or long version, the query processing solution takes the corresponding context entry value (i.e., the URI form of the entity type), which follows a structure closely similar to an XPath notation, as explained earlier. After that, the query rewriting process comes into play by building the corresponding XPath filter accordingly. But for this case, we propose to perform the NETCONF RPC operation in a different way. Depending on the capabilities of the NETCONF server, it can support RPCs by using XPath filters or by using subtree type filters. Subtree filters are specified in a hierarchical XML structure that, similarly to XPath, allows the network management protocol to navigate the target YANG data model to find the corresponding YANG concept. Assuming

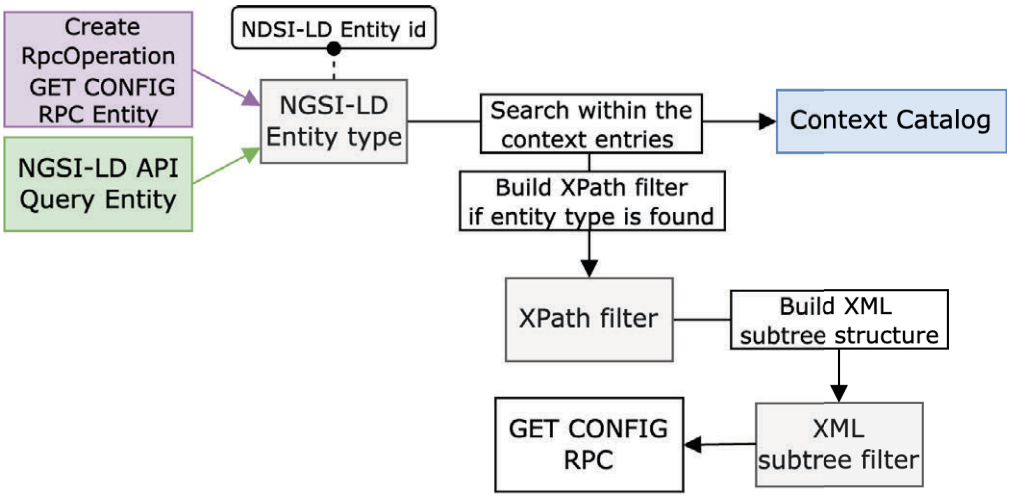


Figure 4.13: Query processing and rewriting general workflow for performing GET CONFIG RPC model-driven network management operations.

then the case that the NETCONF server only supports subtree filters for GET CONFIG RPC operations, the query rewriting process must build the XML subtree structure from the XPath filter. And then the query rewriting process generates the appropriate GET CONFIG RPC operation to request the configuration data from the target network device. In case the data consumer specifies an entity *id*, then the XML subtree filter will add this *id* as an index or key value in the corresponding XML tag if possible for indicating that the consumer wants to retrieve just the information corresponding to this particular entity *id*. For this situation, the query processing solution also infers which target data node types in YANG have keys.

Annex A.4.3 provides examples of the workflow for processing a query entity operation performed from a data consumer by using the NGSI-LD API and rewriting it to a GET CONFIG RPC operation to retrieve configuration data from a network interface using NETCONF for both NMDA compliant and non-compliant network devices.

Updating the configuration of a network device

Figure 4.14 shows the general workflow followed by the Descriptive NDT prototype to process a query performed from a data consumer by using the NGSI-LD API and rewrite it to a SET RPC operation in order to update the configuration of a network device.

In a data virtualization approach, the data consumer performs the NGSI-LD entity creation operation to create a new entity with specific content according to the schema defined for this entity type. In a data materialization approach, the data consumer performs the NGSI-LD entity update operation to update an entity instantiation with specific content according to the schema defined for this entity type. Then, the query processing solution looks for the entity type in the Context Catalog service. If the entity type is found, either by its short or long version, the query processing solution takes the corresponding context entry value (i.e., the URI form of the entity type), which follows a structure closely similar to an XPath notation, as explained earlier. After that, the query rewriting process comes into play by building the corresponding XPath filter accordingly.

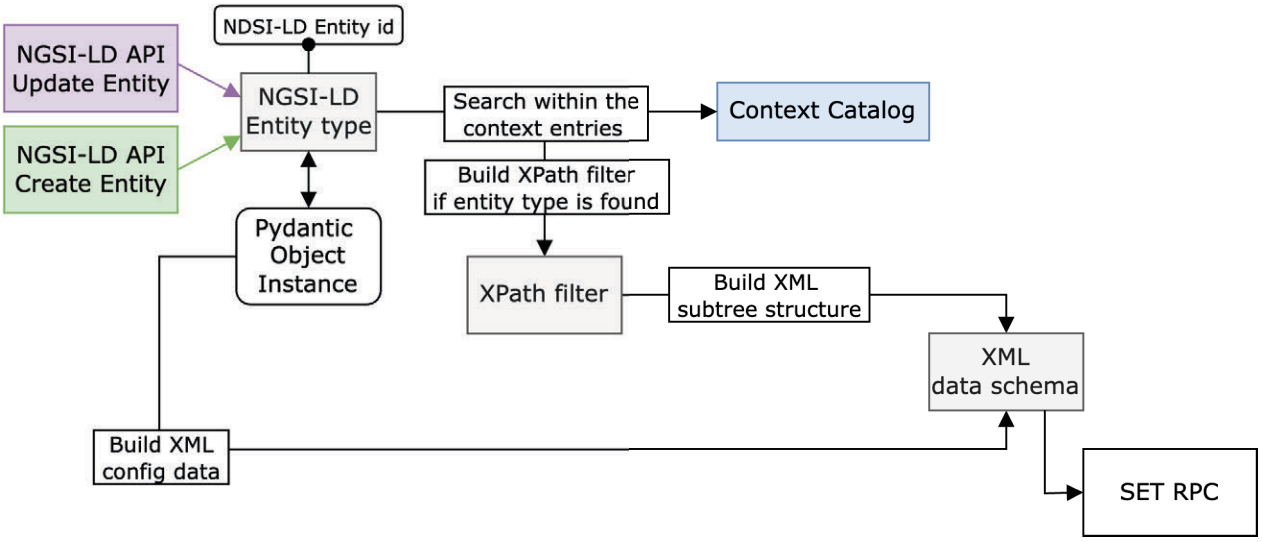


Figure 4.14: Query processing and rewriting general workflow for performing SET RPC model-driven network management operations.

But for this case, as shown in the example with the GET CONFIG RPC operation, we assume that the NETCONF server only supports subtree filters for SET RPC operations. Therefore, the query rewriting process must build the XML subtree structure from the XPath filter. In addition, the query processing process must incorporate the configuration content into the XML data structure. This is done by constructing an XML block with the configuration data and combining it with the XML subtree structure in order to create the XML data schema required to execute the SET RPC query. Finally, the query rewriting process generates the appropriate SET RPC operation to update the configuration of the target network device.

Similarly to the GET CONFIG RPC example, since the creation of the NGSI-LD entity includes its *id*, then the XML data schema filter will add this *id* as a key value in the corresponding XML tag if possible to indicate that the consumer wants to configure the information corresponding to just this particular entity *id*. The query processing solution then infers which target data node types in YANG have keys.

Annex A.4.4 provides examples of the workflow for processing a create entity operation performed from a data consumer by using the NGSI-LD API and rewriting it to a SET RPC operation in order to update the configuration of a network interface using NETCONF for both NMDA compliant and non-compliant network devices.

4.2 Case Study: Descriptive NDT for Network Operations and Management

In this section, we propose a case study for applying the Descriptive NDT solution for network operations and management purposes. The validation use case involves the discovery and exposition of data related to the network topology and node interfaces. Network topology and node interfaces are characterized using YANG models, which are automatically converted to NGSI-LD models for the Descriptive NDT. We analyze the case study with standardized YANG models from both the IETF and the OpenConfig consortium. As an additional proof of concept, we evaluate the application of the Descriptive NDT solution to a traditional network flow monitoring protocol such as NetFlow by defining the corresponding NGSI-LD domain model. Finally, we illustrate the usefulness of the Descriptive NDT solution by showing some specific operations and management actions.

4.2.1 IETF-based Descriptive NDT model

Figure 4.15 shows a high-level UML diagram of the NGSI-LD domain model provided by the Descriptive NDT to represent data from network topologies and node interfaces based on IETF RFCs 8345 [108] and 8343 [132]. The NGSI-LD model supports the latest revision of the *ietf-interfaces* YANG module (i.e., *ietf-interfaces@2018-02-20*) for managing network interfaces, which is available for devices that support the NMDA architecture.

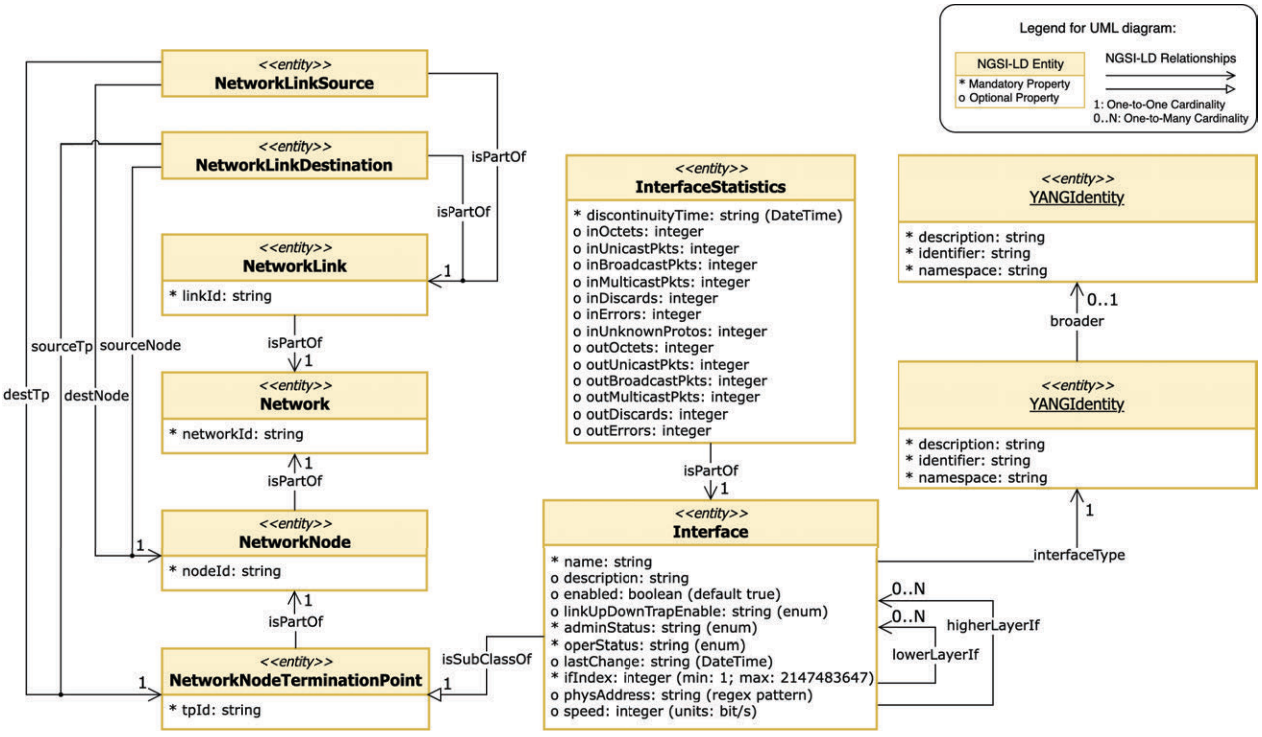


Figure 4.15: UML diagram of NGSI-LD domain model for managing network topology and interfaces based on IETF RFCs 8345 and 8343 (NMDA compliant).

On the left-hand side of the diagram, we can find the different NGSI-LD entities for modeling the network topology. Each of the different entities has a mandatory property to identify the network, node, link, and termination point within the network topology. To determine the source and destination of a network link, there are two separate entities to indicate the corresponding node and the termination points at each end. In addition, we define the *isPartOf* NGSI-LD relationship to represent compositional relationships between entities (e.g., a termination point is part of a network node). This *isPartOf* NGSI-LD relationship type, which is a term standardized by the Dublin Core Metadata Initiative (DCMI)[179][4.2] for linking physical or logical resources, has been leveraged as a *de facto* rule to establish these hierarchical relationships between NGSI-LD entities. These relationships are automatically inferred in the process of parsing the YANG data models.

On the right-hand side of the diagram, there are the two NGSI-LD entities modeling information from node network interfaces. The *Interface* entity has NGSI-LD properties representing the configuration and operational data of a network interface, while the *InterfaceStatistics* entity has NGSI-LD properties representing the interface statistics relative to traffic counters. The *Interface* entity type has two optional NGSI-LD relationships, *lowerLayerIf* and *higherLayerIf*, that target other *Interface* entities to reference interfaces that are either layered below or on top of an interface in the case of underlay or overlay networks. There is also a particular *YANGIdentity* entity type to represent the type of interface (e.g., Ethernet-type interface or Fiber Channel interface), and also a parent *YANGIdentity* entity type for a broader explanation of the interface type concept. As mentioned before, IANA specified the YANG module to define the original YANG identities that record the types of network interface. This YANG module is maintained by IANA, and its initial version is part of RFC 7224 [180].

In addition, there is a special NGSI-LD relationship named *isSubclassOf*, which establishes inheritance relationships between entities by identifying closely related concepts beforehand. This case of relationships is defined in the model to unify the concept of a network interface between *Interface* and *NetworkNodeTerminationPoint* entities. The use of this type of NGSI-LD relationship is inspired by the notion of *rdfs:subClassOf*, defined by the W3C in OWL [77] and RDFS [82] to specify inheritance relationships between classes[4.3].

Figure 4.16 depicts a UML diagram of the NGSI-LD domain model to represent data for the IP management of network node interfaces. The resulting NGSI-LD model is based on the YANG data model defined in the IETF RFC 8344 [181], and is directly related to the NGSI-LD model just described in Figure 4.15 since this YANG model is an extension of the YANG data model defined for RFC 8343 for managing network node interfaces. The model represents information for managing IPv4 and IPv6 addresses, covering configuration and operational parameters. The *InterfaceIpv4* and *InterfaceIpv6* entities define NGSI-LD properties for enabling IPv4 and IPv6 addressing and packet forwarding, in addition to MTU adjustment for packet sizes. Meanwhile, the *InterfaceIpv4Address* and *InterfaceIpv6Address* entities have

[4.2] Dublin Core Metadata Initiative (DCMI) is an organization dedicated to fostering the widespread adoption of interoperable metadata standards and promoting the development of specialized metadata vocabularies for describing resources to enable more intelligent resource discovery systems.

[4.3] W3C defines OWL and RDFS as fundamental languages for modeling ontologies on the Semantic Web.

properties for specifying the IPv4 and IPv6 addresses and the subnet prefix length or subnet netmask of a network interface accordingly. In case of IPv4 address, either the subnet prefix length or the subnet netmask must be defined[4.4]. In addition, the *InterfaceIpv4Neighbor* and *InterfaceIpv6Neighbor* entities have information related to the Address Resolution Protocol (ARP) cache and the Neighbor Discovery Protocol (NDP) cache related to the mappings from IPv4 and IPv6 addresses to Layer 2 addresses discovered from a network node interface. Finally, the *InterfaceIpv6Autconf* entity includes properties related to the available parameters to control the autoconfiguration of IPv6 addresses.

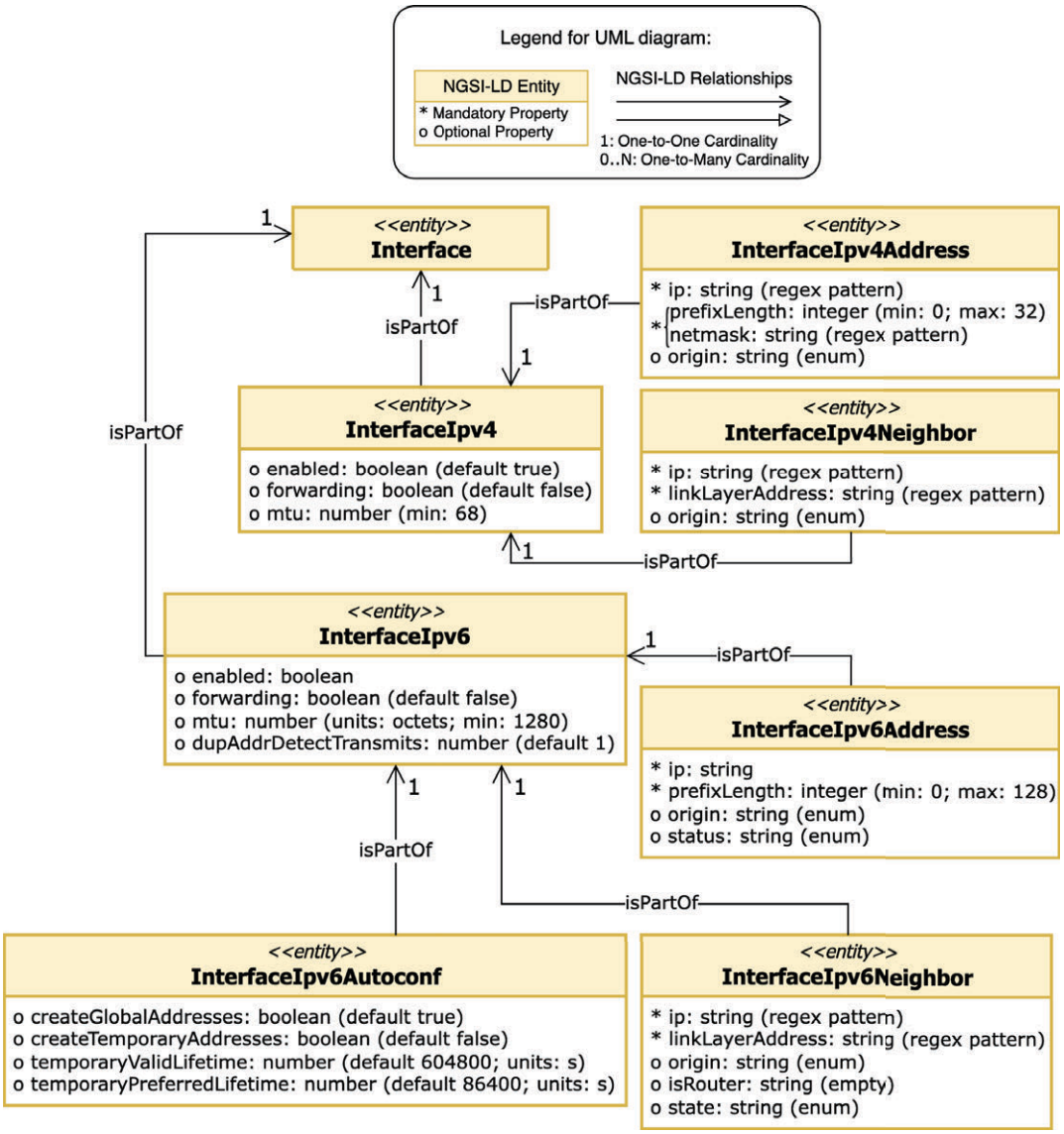


Figure 4.16: UML diagram of NGSI-LD domain model for network interface IP management based on IETF RFC 8344 (NMDA compliant).

[4.4]In the YANG data model defined in RFC 8344, the concept of a subnet is specified as a *choice* YANG schema node where only one of a number of identified alternatives, called cases in the YANG language, is valid (e.g., the *subnet* YANG *choice* has two cases, *netmask* and *prefix-length*).

For network devices that do not support the NMDA architecture, Figure 4.17 represents an alternative NGSI-LD model to represent data from network topologies and node interfaces. The resulting NGSI-LD model differs from the NMDA-compliant IETF model compatible with NMDA in that the configuration data and the operational state data are separated into two different entities, *InterfaceConfig* and *InterfaceState*, respectively. This NGSI-LD model is based on an earlier revision of the *ietf-interfaces* YANG module (i.e., *ietf-interfaces@2014-05-08*) for managing network interfaces as defined by the now obsolete RFC 7223 [182], which is available for devices that do not support the NMDA architecture.

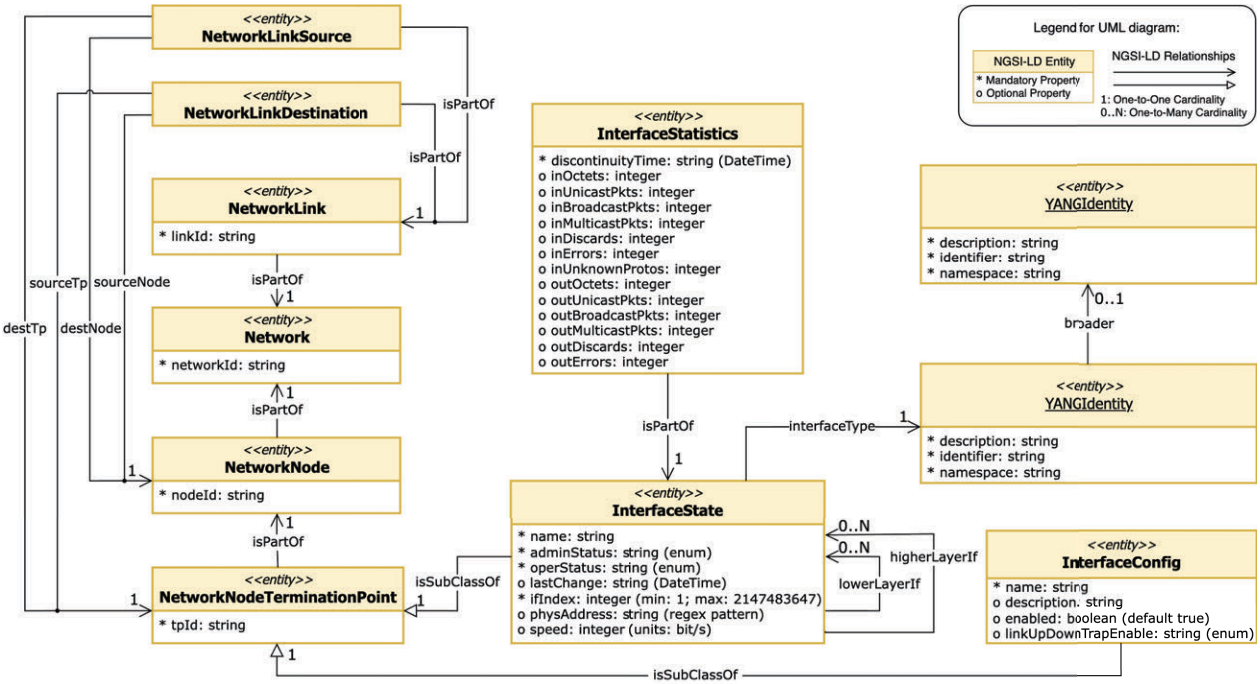


Figure 4.17: UML diagram of NGSI-LD domain model for managing network topology and interfaces based on IETF RFCs 8345 and 7223 (non-NMDA compliant).

There is also a version of the NGSI-LD model to manage IP interfaces addressing data for network devices not compatible with the NMDA architecture, as shown in Figure 4.18. The resulting NGSI-LD model differs from the NMDA-compliant IETF model compatible with NMDA in that the configuration data (i.e., the right side in Figure 4.18) and the operational state data (i.e., the left side in Figure 4.18) are separated into different entities. For example, while the operational model provides operational state information about IP addressing of a network interface, such as the origin of the address (e.g., DHCP or statically assigned address) or the status of the IPv6 address as defined by RFC 4862 [183], the configuration model is the one that allows IPv4 and IPv6 to be enabled on a network interface and gives access to set the autoconfiguration parameters for the IPv6 addresses. This NGSI-LD model is based on an earlier revision of the *ietf-ip* YANG module (i.e., *ietf-ip@2014-06-16*) for IP management as defined by RFC 7277 [184], which is available for devices that do not support the NMDA architecture.

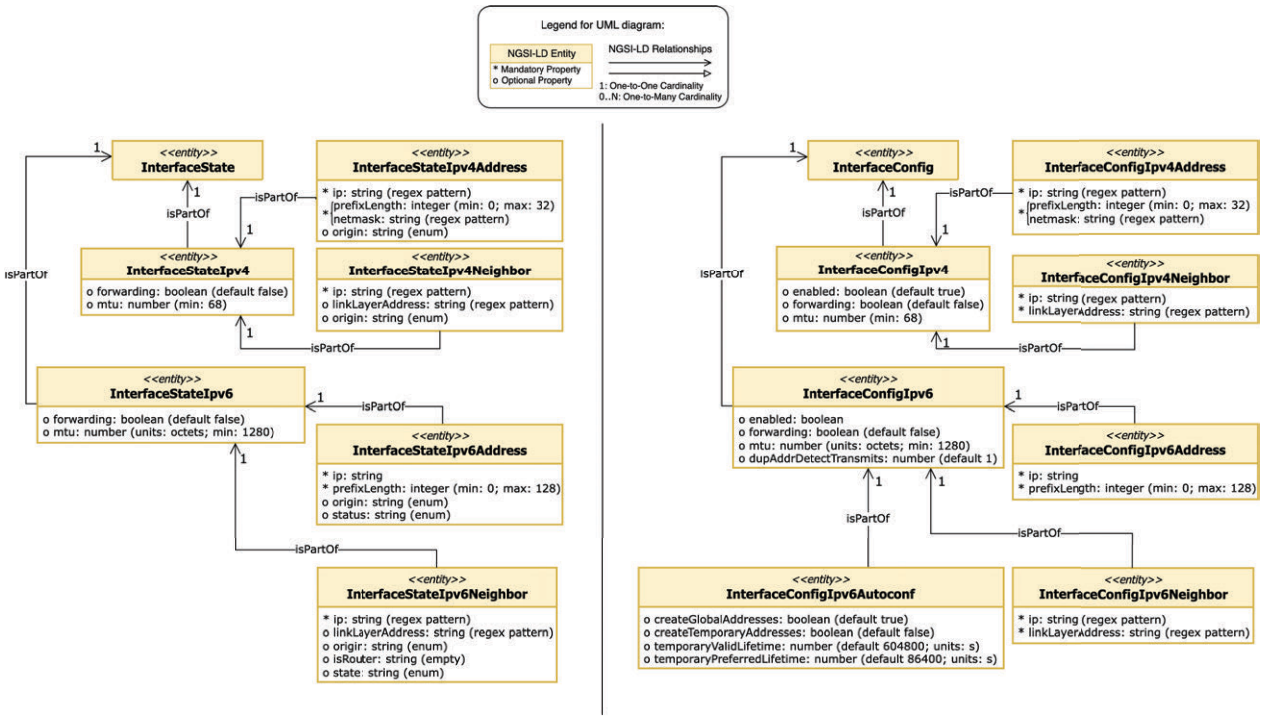


Figure 4.18: UML diagram of NGSI-LD domain model for network interface IP management based on IETF RFC 7277 (non-NMDA compliant).

4.2.2 OpenConfig-based Descriptive NDT model

Figure 4.19 depicts the resulting NGSI-LD domain model provided by the Descriptive NDT to represent data regarding network node interfaces management, but modeled according to the YANG definitions of the OpenConfig consortium [185]. Again, the *isSubclassOf* relationship is used to represent that an *Interface* is a subclass of the entity *NetworkNodeTerminationPoint* used in the network topology model standardized by IETF. And, similar to the model derived from IETF RFC 8343, there is an entity representing interface statistics, *InterfaceStateCounters*, to collect interface traffic counters. However, the resulting NGSI-LD model for OpenConfig differs from the NMDA-compliant IETF model in that configuration data (i.e., *InterfaceConfig*) and operational state data (i.e., *InterfaceState*) are separated into different entities, although certain properties are found in both the configuration and state entities.

Furthermore, this model differentiates the concept of subinterface with its corresponding entities dedicated to configuration, status, and telemetry statistics information. These subinterface entities model a concept equivalent to that of higher and lower-layer interfaces within the IETF model, but with separate entities. As in the case of the IETF NGSI-LD model, we find entities of type *YANGIdentity* to conceptualize the interface type.

In addition, the *InterfaceHoldTimeConfig* and *InterfaceHoldTimeState* entities represent configuration and state information used to adjust hold-time used for dampening advertisements when the interface state changes from up to down and vice versa, with a value of zero meaning that dampening is disabled (i.e., notifications about the state

of the interface have no delay). There are also the *InterfacePenaltyBasedAiedConfig* and *InterfacePenaltyBasedAiedState* entities, which represent the configuration and state of the interface link transition settings.

Similarly to the IETF-based model, OpenConfig provides additional YANG modules that extend the YANG data model to manage network interfaces by including definitions of additional details. The resulting NGSI-LD domain model shown in the UML diagram in Figure 4.20 is based on the YANG data model defined by the OpenConfig consortium for managing the configuration and operational status of IP interfaces [186]. Like the IETF model, this model encompasses information for managing IP addressing, either IPv4 or IPv6, and neighbor discovery, among other aspects, but specifically in the context of network subinterfaces.

Additionally, this model introduces statistical counters for IP transmission and reception, categorized by the address family of the interface, either IPv4-related counters or IPv6-related counters. As in the IETF model, the OpenConfig-based NGSI-LD model for IP management includes Layer 2 neighbor discovery information for both IPv4 and IPv6. Furthermore, the model specifies how a subinterface can be unnumbered, indicating whether an IP address is expected to be assigned to it. The model also includes configuration and operational parameters related to proxy ARP functionality, which allows a system to respond to ARP requests that are not explicitly destined for it.

Finally, this model provides additional information on the configuration and operational details for router advertisements and Virtual Router Redundancy Protocol (VRRP) [187] groups[4.5] handled by the interface for both IPv4 and IPv6 addresses.

[4.5]The information about VRRP groups covered by the OpenConfig YANG model for managing IP interfaces is based on the RFC 5798, which defines version 3 of the VRRP protocol for IPv4 and IPv6. VRRP specifies an election protocol that dynamically assigns virtual router responsibility to one of the VRRP routers on a LAN, which is the router master that handles forwarding for IPv4 and IPv6 addresses.

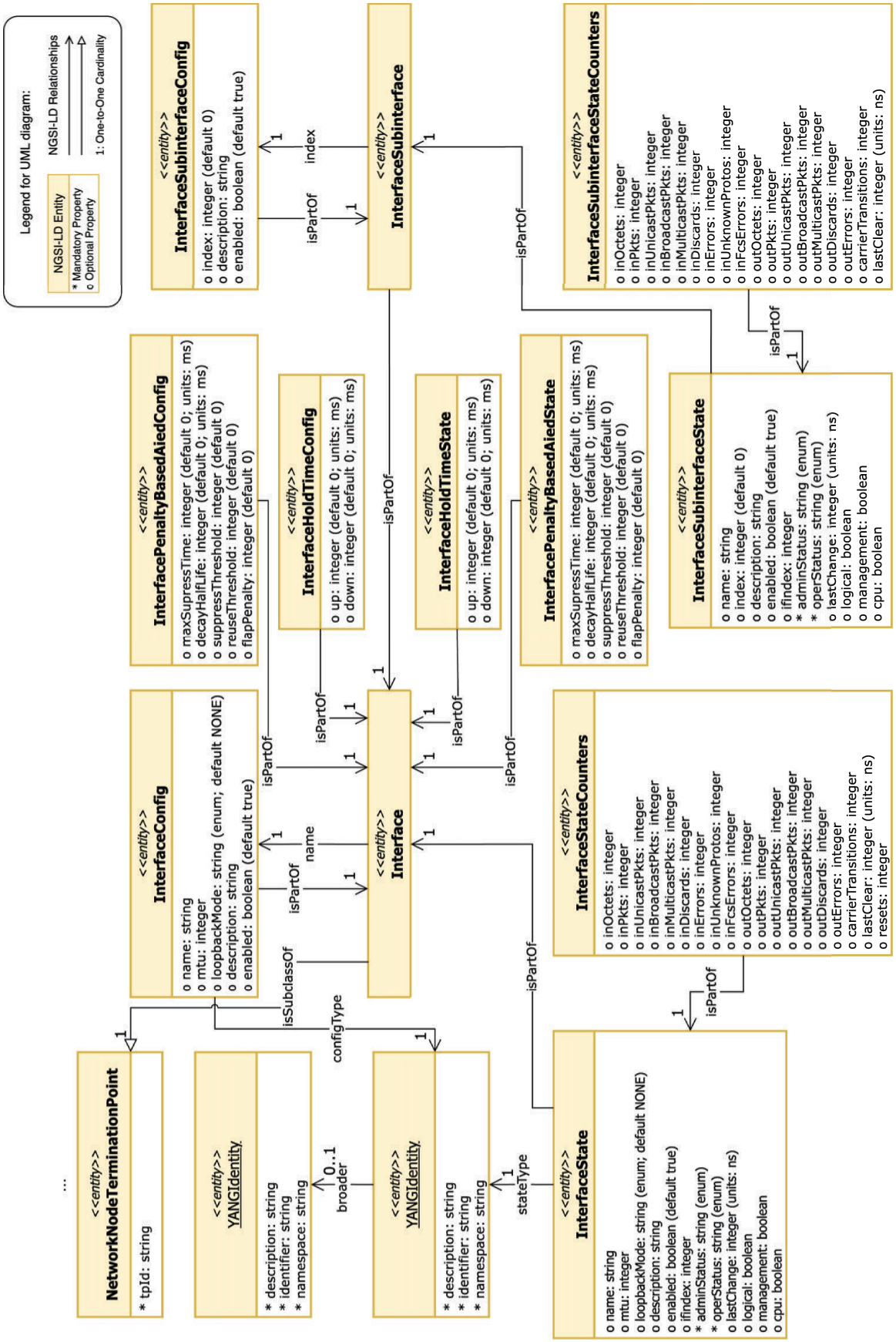


Figure 4.19: UML diagram of NGSI-LD domain model for managing network interfaces according to OpenConfig consortium.

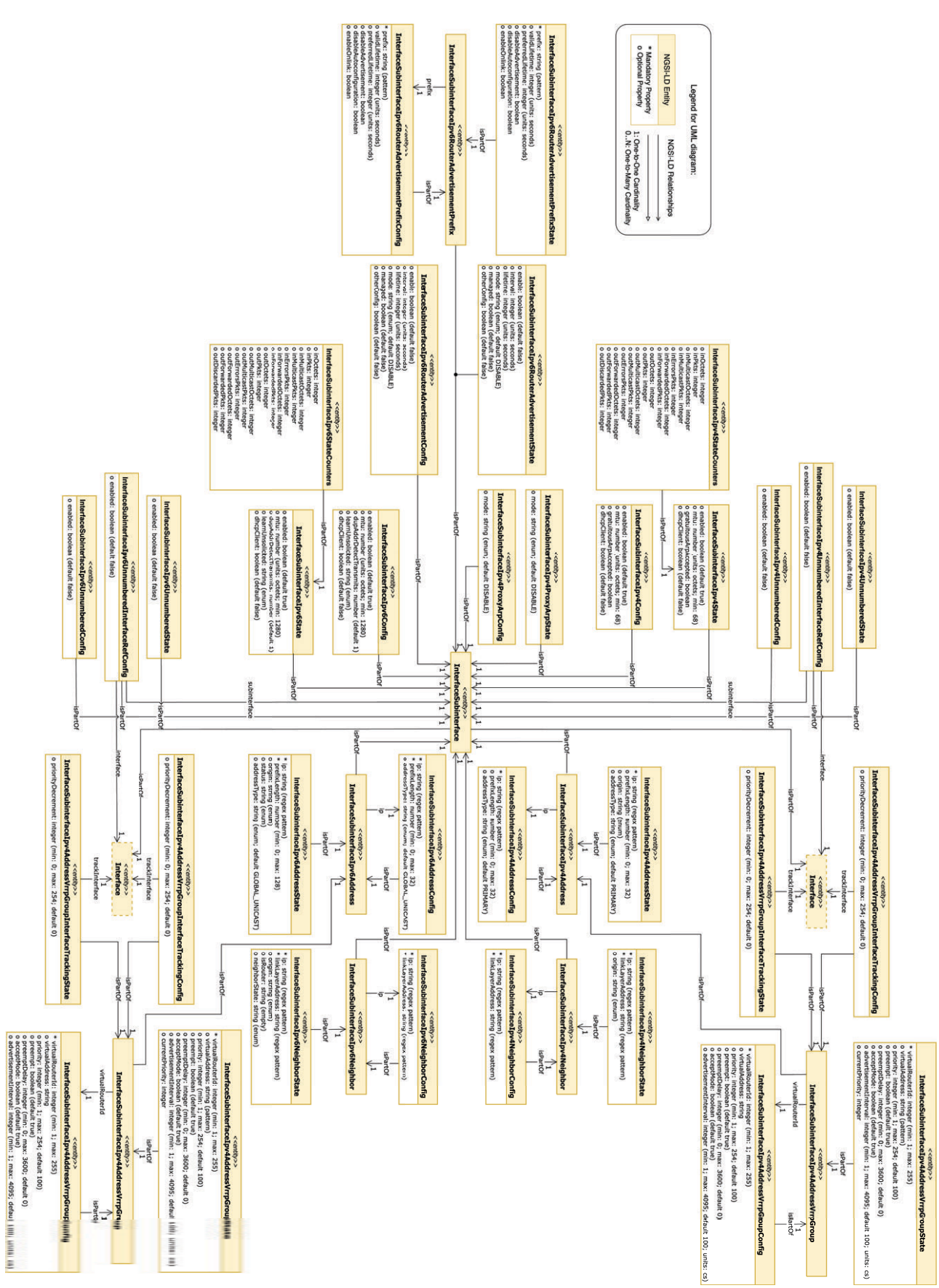


Figure 4.20: UML diagram of NGSI-LD domain model for network interface IP management according to OpenConfig consortium.

4.2.3 Descriptive NDT model for NetFlow v9 protocol

Aside from the NGSI-LD domain models mentioned above related to modeling network topologies and network node interfaces as specified by the IETF organization and the OpenConfig consortium, the digital mapping process of the Descriptive NDT solution has been evaluated with other YANG models, such as a custom YANG data model defined for the NetFlow v9 protocol [128] as a result of previous works [188], [189].

NetFlow is one of the main monitoring protocols for providing flow-level network measurements and is composed of two main roles, the exporter and the collector. The NetFlow exporter is a network device that implements the NetFlow protocol and is in charge of providing flow information and exporting it to a NetFlow collector. Meanwhile, the NetFlow collector is a component that receives the exported flow information. Then, the content of the collected network flows can be processed and analyzed by external systems.

In [188] we evaluated the possibility of applying a data model-driven monitoring approach to network flows by defining a YANG data model for the NetFlow v9 protocol, as no standardized data model was implemented for NetFlow-based monitoring data. In addition, the work presents a reference architecture that enables the collection, processing, and delivery of NetFlow-based monitoring data modeled according to this YANG model.

The NetFlow v9 YANG data model was designed to capture the contents of NetFlow export packets that are relevant for subscribers to NetFlow monitoring data, which are the packet header and data flow sets. The resulting YANG module, as depicted in Figure 4.21, comprises two main YANG containers: *collector* and *export-packet*.

The *collector* YANG container includes metadata related to the reception of an export packet in the NetFlow collector. These metadata range from the reception timestamp to the IP address, either version 4 or version 6, of the NetFlow exporter.

The *export-packet* YANG container incorporates the contents of an export packet. This container draws from the NetFlow Version 9 Flow-Record Format specification [190] which provides a thorough description of the structure of a NetFlow export packet. First, the data in the packet header are modeled using YANG leafs that represent fields such as the sequence number of the packet or the identifier of the NetFlow exporter that sent the packet (i.e., *source-id*). Second, a YANG list named *flow-data-record* models the data pertaining to the different data flow sets found within an export packet. To uniquely identify each flow within the YANG list, the *flow-id* YANG leaf is designated the key of the list. The *flow-id* leaf represents the hash code obtained from the 5-tuple formed by the source IP address, destination IP address, source port, destination port, and protocol number.

Aside from the *flow-id* leaf, the child nodes of the *flow-data-record* container represent all possible fields that can be included in a data flow set. The data fields present in each data flow set will vary depending on the type of network traffic that has been sampled with NetFlow. Thus, the proposed YANG module introduces multiple YANG containers, each enclosing those data fields that are specific to a particular network protocol. For instance, *mpls* contains a set of possible MPLS labels for each position in the stack; *bgp* holds data related to BGP such as next-hop IP address or the autonomous system (AS) numbers of the peers involved

```
module: netflow-v9
 +--ro netflow
  +--ro collector
  | +--ro time-received           yang:timestamp
  | +--ro sampler-address?        inet:ipv4-address
  | +--ro sampler-address-ipv6?   inet:ipv6-address
  +--ro export-packet
   +--ro sequence-number     yang:counter32
   +--ro count?              uint16
   +--ro system-uptime?      yang:timestamp
   +--ro unix-seconds?       yang:timestamp
   +--ro source-id?          uint32
   +--ro flow-data-record* [flow-id]
    +--ro flow-id                int32
    +--ro bytes-in               yang:counter64
    +--ro bytes-out?             yang:counter64
    +--ro pkts-in                yang:counter64
    +--ro pkts-out?              yang:counter64
    +--ro flows?                 yang:counter64
    +--ro protocol               net-v9:protocol-type
    +--ro src-tos?               uint8
    +--ro dst-tos?               uint8
    +--ro tcp-flags?             net-v9:tcp-flags-type
    +--ro src-port               inet:port-number
    +--ro dst-port               inet:port-number
    +--ro snmp-in?               int32
    +--ro snmp-out?              int32
    +--ro bytes-out-mul?         yang:counter64
    +--ro pkts-out-mul?          yang:counter64
    +--ro first-switched         yang:timestamp
    +--ro last-switched          yang:timestamp
    +--ro min-pkt-len?           uint16
    +--ro max-pkt-len?           uint16
    +--ro icmp-type?             uint16
    +--ro igmp-type?             net-v9:igmp-type
    +--ro sampler-name?          string
    +--ro sampling-interval?     uint32
    +--ro sampling-algorithm?    net-v9:sampling-mode-type
    +--ro flow-active-tout?      uint16
    +--ro flow-inactive-tout?    uint16
    +--ro engine-type?           net-v9:engine-type
    +--ro engine-id?             uint8
    +--ro tot-bytes-exp?         yang:counter64
    +--ro tot-pkts-exp?          yang:counter64
    +--ro tot-flows-exp?         yang:counter64
    +--ro flow-sampler-id?       uint8
    +--ro flow-sampler-mode?     net-v9:sampling-mode-type
    +--ro flow-sampler-random?   uint32
    +--ro min-ttl?               uint8
    +--ro max-ttl?               uint8
    +--ro src-mac-in             yang:mac-address
    +--ro dst-mac-in             yang:mac-address
    +--ro src-mac-out?           yang:mac-address
    +--ro dst-mac-out?           yang:mac-address
    +--ro ip-version             net-v9:ip-version-type
    +--ro direction?             net-v9:direction-type
    +--ro if-name?               string
    +--ro if-desc?               string
    +--ro frag-offset?           uint16
    +--ro forwarding-status?     net-v9:forwarding-status-type
    +--ro postip-dscp?           inet:dscp
    +--ro repl-factor-mul?       uint32
    +--ro ipv4
    +--ro ipv6
    +--ro mpls
    +--ro bgp
    +--ro vlan
    +--ro permanent-flow
    +--ro application
    +--ro layer2-pkt-section
```

Figure 4.21: YANG module for NetFlow version 9. The YANG tree representation is displayed in two columns for better readability.

in the BGP session; whereas the *vlan* container groups the source and destination VLAN identifiers that may have been used in the sampled network flow. On the other hand, data related to the IP protocol is also grouped into a separate YANG container: one for IPv4 and another for IPv6. As a result, the contents of the data flow set follow a clear structure within the YANG module, hence, improving its readability and enabling other YANG modules to import these groups.

Overall, the proposed YANG module attempts to exploit the data typing functionality of the YANG language by rigorously defining all nodes in the module. In this sense, fields such as *protocol* or *igmp-type*, which are commonly modeled as integers, are defined in the module as enumerates with their respective semantic explanation. The *tcp-flags* field, which represents a list of flags that are optionally enabled in a TCP flow, is accurately defined using the *bits* type. Other data fields related to IP or MAC addresses, are modeled by importing data types from standardized YANG modules such as *ietf-inet-types* and *ietf-yang-types*.

NetFlow has been widely used as a network monitoring protocol to provide flow-level measurements useful for network security analysis, with the objective of detecting anomalous activities or networking threats [191]. In recent years, different approaches have applied Machine Learning models on NetFlow information to detect vulnerabilities in the network, such as the work in [192] which offers a solution for detecting cryptomining flows in real time. In this sense, in our previous works [188], [189] we defined a mechanism to aggregate additional parameters into each network flow in the form of new KPIs, which can be computed from the

original data flow sets. To this end, we defined an augmentation of the original YANG model for NetFlow v9 which incorporates the semantic information for new KPIs[4.6]. Figure 4.22 shows the augmentation of the original NetFlow v9 YANG data model that extends the list of child nodes from the *flow-data-record* container with new data nodes[4.7]. The new child nodes represent the aggregated KPIs, whose semantics is summarized as follows:

- *flow-duration*: Duration of the network flow. Difference between *last-switched* and *first-switched* data nodes.
- *bytes-in-per-second* and *bytes-out-per-second*: Counters for the number of incoming and outgoing bytes per second associated with an IP flow.
- *pkts-in-per-second* and *pkts-out-per-second*: Counters for the number of incoming and outgoing packets per second associated with an IP flow.
- *bytes-in-per-packet* and *bytes-out-per-packet*: Counters for the number of incoming and outgoing bytes per packet associated with an IP flow.
- *ratio-bytes-in-per-out* and *ratio-pkts-in-per-out*: The ratio of input bytes/packets to output bytes/packets.

```
module: netflow-v9-agg

  augment /net-v9:netflow/net-v9:export-packet/net-v9:flow-data-record:
    +--ro flow-duration?             yang:timestamp
    +--ro bytes-in-per-second?       per-decimal
    +--ro bytes-out-per-second?      per-decimal
    +--ro pkts-in-per-second?        per-decimal
    +--ro pkts-out-per-second?       per-decimal
    +--ro bytes-in-per-packet?       per-decimal
    +--ro bytes-out-per-packet?      per-decimal
    +--ro ratio-bytes-in-per-out?    per-decimal
    +--ro ratio-pkts-in-per-out?     per-decimal
```

Figure 4.22: Augmentation of YANG module for NetFlow version 9 with aggregated KPIs.

The aggregated KPIs provide information related to the volume of data at the flow-level. This information can be useful for a Machine Learning model in order to detect different anomalies in the analyzed network flows.

In the work on [189], we validate this data model-based monitoring methodology for aggregating KPIs on network flows for the analysis of a Machine Learning engine consisting of a cryptomining detection system. As a result, better performance results were obtained for predicting network flows related to cryptomining activities.

[4.6]Both the applications to normalize data according to the predefined YANG data model for NetFlow v9 and to perform aggregation functions to calculate additional KPIs are programmed as Apache Flink [193] data engineering applications in the Java programming language. Specifically, these applications use the YANG Tools [194] library, which allows data to be normalized according to a specific YANG data model and serialized/deserialized to or from a JSON or XML encoding format to perform the appropriate data transformations.

[4.7]The YANG modules defined for both the NetFlow v9 protocol and the extension to include additional information about existing network flow data records in a NetFlow export packet are available in a GitHub repository at the following link: https://github.com/giros-dit/netflow-v9-yang-models/tree/main

We have evaluated the Descriptive NDT methodology using the aforementioned YANG models for NetFlow v9 by applying the digital mapping process to translate the YANG-modeled data into NGSI-LD data. The resulting NGSI-LD domain model, which is depicted in Figure 4.23, represents as entities both the different export packets (i.e., *ExportPacket* entity) and the corresponding flow data records particularized for each export packet (i.e., *ExportPacketFlowDataRecord* entity) with the different properties that characterize the network flows.

The *ExportPacketFlowDataRecord* also includes as properties the additional KPIs resulting from the augmentation of the original NetFlow v9 YANG data model. Furthermore, different entities are derived to include characteristics related to IPv4, IPv6, MPLS, VLAN, and BGP traffic flows.

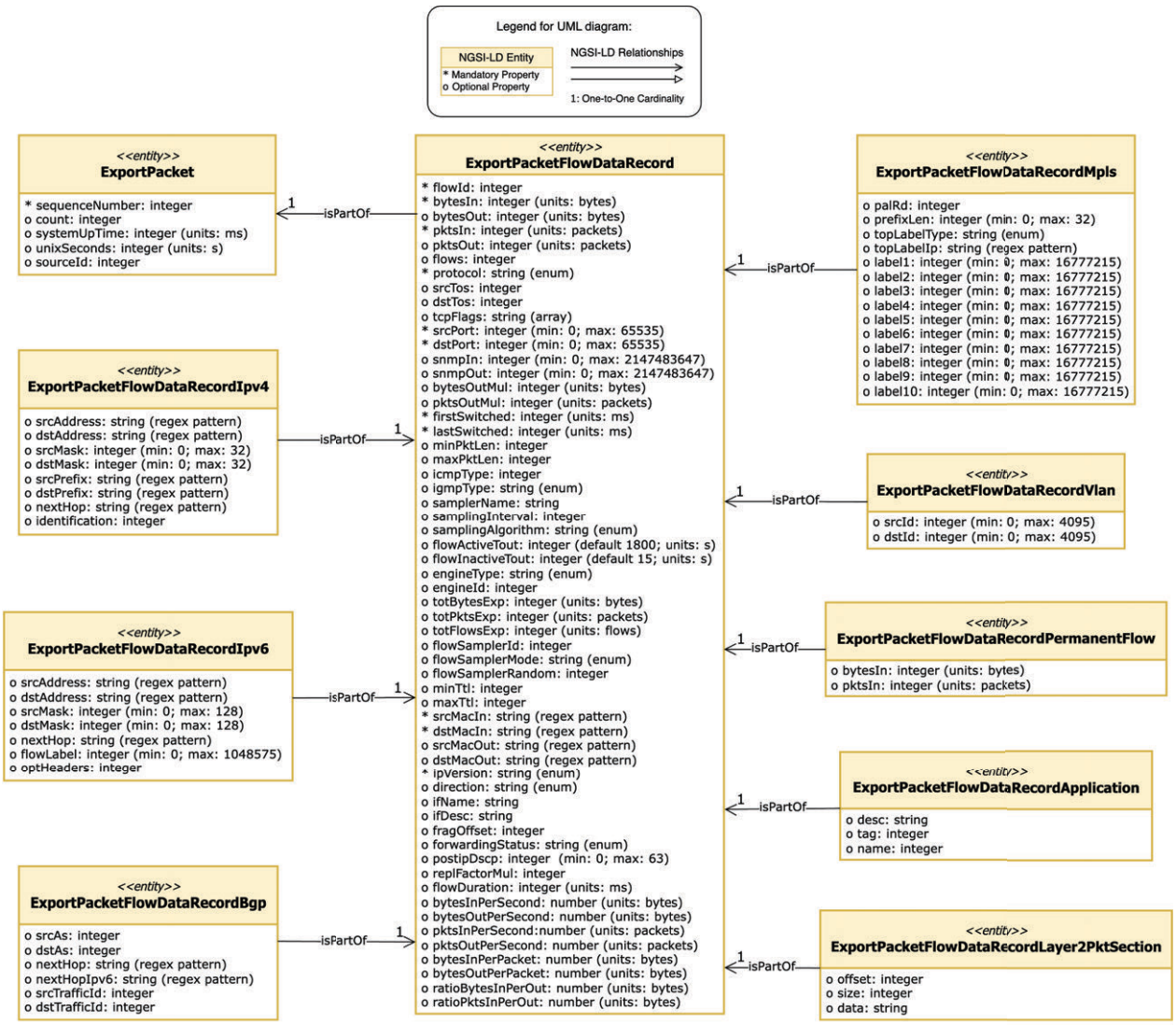


Figure 4.23: NGSI-LD domain model for network flows according to NetFlow v9 protocol.

4.2.4 Application of the NGSI-LD Descriptive NDT for network operations and management purposes

The NGSI-LD domain models mentioned above characterize the NDT models and associated data, allowing their consumption and manipulation. Thanks to the NGSI-LD API, the following use cases demonstrate the usability of the Descriptive NDT for efficient network operations and management purposes. For each use case, the best data integration option is selected between data materialization and data virtualization.

These use cases illustrate functionalities that can help NetOps engineers using network management systems to analyze and estimate network performance, supporting complex network automation tasks such as network planning, detecting anomalies, identifying potential problems or risks, troubleshooting purposes, predicting future conditions, or recommending optimization policies.

Discovering node adjacencies in network topology

The objective of this use case is to determine, given a network device, the devices connected to the other edge of its links in order to discover adjacencies and find neighboring nodes within a network topology. First, the interested NetOps consumer needs to discover the network links where the given node (e.g., an entity of type *NetworkNode* identified by the *id node1*) is the source or destination by using the following request:

```
GET /ngsi-ld/v1/entities/?type=NetworkLinkSource,
NetworkLinkDestination&q=(sourceNode=="node1"|
destNode=="node1") HTTP/1.1
```

The response to the previous query provides a list of entities of type *NetworkLinkSource* and *NetworkLinkDestination* where the *node1* is the source or destination of a link (i.e., the *sourceNode* or the *destNode* relationships point to the name *node1*), and where the *isPartOf* relationship points to object identifiers that reference entities of type *NetworkLink*. In addition, the received *NetworkLinkSource* and *NetworkLinkDestination* entities provide the source or destination interface of the node *node1* which is the termination point of the link (i.e., the *sourceTp* or the *destTp* relationship objects).

Let's imagine that the response provided indicates that *node1* has three links where the node is the source or destination, and that those links have object identifiers *link1*, *link2*, and *link3*. Now, the NetOps consumer needs to identify the nodes adjacent to *node1* on the other side of those links. To do this, the consumer would perform the following query:

```
GET /ngsi-ld/v1/entities/?type=NetworkLinkSource,
NetworkLinkDestination&q=(isPartOf=="link1"|
isPartOf=="link2"|isPartOf=="link3");
(sourceNode!="node1"|destNode!="node1") HTTP/1.1
```

As a result, the query response will provide the list of entities of type *NetworkLinkSource* and *NetworkLinkDestination* with the information of adjacent nodes to *node1* and their

termination point interfaces within the network link that joins them to *node1*. The NetOps consumer could then determine and select network links between specific pair of nodes to measure, for example, their traffic occupancy or throughput.

In this use case, the Descriptive NDT follows a data materialization approach because it needs to access data about the network topology that is replicated within the NGSI-LD Context Broker. This information is usually not very variable, so it is recommended to materialize the data to preserve the history of changes.

Obtaining history about the configuration and status of network interfaces

This use case consists of obtaining the configuration and status of network interfaces along time using the Temporal API offered by the NGSI-LD API. NetOps consumers can obtain different instances of the *Interface*, *InterfaceConfig*, and *InterfaceState* entities along time, depending on whether they follow the IETF or the OpenConfig compliant NGSI-LD domain model. For this example, we have selected the IETF NGSI-LD domain model compatible with the NMDA architecture and the OpenConfig NGSI-LD domain model. The following request is an example of retrieving the instance history of *Interface*, *InterfaceConfig*, and *InterfaceState* entities materialized within the NGSI-LD Context Broker in January 15th using the NGSI-LD Temporal API:

```
GET /ngsi-ld/v1/temporal/entities/?type=Interface,
InterfaceConfig,InterfaceState&timerel=between&timeAt=
2025-01-15T00:00:00Z&endTimeAt=2025-01-15T23:59:59Z&
timeproperty=modifiedAt&options=sysAttrs HTTP/1.1
```

As a result, the query response will retrieve the history of configuration and state changes from the interfaces on that date. In addition, by using the *sysAttrs* query parameter, the NetOps consumer can declare that system-generated temporary properties should be included in the response to indicate the date and time when the changes were established and materialized in the Context Broker.

This test case can be useful for troubleshooting purposes, allowing network administrators to review or revert changes to previous configurations. In such a use case, the Descriptive NDT again takes a data materialization approach because it is the appropriate data integration solution to have a temporal history of the configuration and operational state data from the managed network interfaces. In contrast, in a data virtualization approach, there is no replication of the data, and the Context Source role of the NDT solution is not expected to have native support for the NGSI-LD Temporal API.

Configuring network node interfaces

This use case consists of configuring the interface of a network device by using the NGSI-LD API. To validate this case, we propose that the network device is a YANG server that does not support the NMDA architecture, and therefore the resulting NGSI-LD model differentiates between NGSI-LD entities for configuration and operational state. Depending on the data integration approach, the NetOps consumer needs either to create a new NGSI-LD entity

instance of type *InterfaceConfig* in case of data virtualization or to update an existing instance of the *InterfaceConfig* NGSI-LD entity in case of data materialization.

Let's imagine that the network administrator needs to configure a network interface *GigabitEthernet2* of a network node *node1* to update its description and to enable trap notifications about changes in the operational status of the interface. Figure 4.24 shows an example of the request body required when using a data materialization approach. This request body is used in an HTTP PATCH operation to update the existing *InterfaceConfig* NGSI-LD entity as follows:

```
PATCH /ngsi-ld/v1/urn:ngsi-ld:InterfaceConfig:
node1:GigabitEthernet2/attrs HTTP/1.1
```

```
{
    "description": {
        "type": "Property",
        "value": "GigabitEthernet2 interface"
    },
    "enabled": {
        "type": "Property",
        "value": true
    },
    "linkUpDownTrapEnable": {
        "type": "Property",
        "value": "enabled"
    }
}
```

Figure 4.24: Request body for a NGSI-LD update entity operation to configure network interface using a data materialization approach.

Meanwhile, Figure 4.25 shows an example of the request body required when using a data virtualization approach. This request body is used in an HTTP POST operation to create the new instance of *InterfaceConfig* NGSI-LD entity as follows:

```
POST /ngsi-ld/v1/entities HTTP/1.1
```

Then, let's assume that the network administrator wants to configure a new IPv4 address 10.1.1.11 with netmask 255.255.255.0 for the same network interface *GigabitEthernet2* of the network node *node1*. Figure 4.26 shows an example of the required request body when a data materialization approach is used. This request body is used in an HTTP PATCH operation to update the existing *InterfaceConfigIPv4Address* NGSI-LD entity as follows:

```
PATCH /ngsi-ld/v1/urn:ngsi-ld:InterfaceConfigIPv4
Address:node1:GigabitEthernet2:10.1.1.11/attrs
HTTP/1.1
```

Meanwhile, Figure 4.27 shows an example of the required request body when a data virtualization approach is used. This request body is used in an HTTP POST operation to create the new instance of the *InterfaceConfigIPv4Address* NGSI-LD entity.

```
{
    "id": "urn:ngsi-ld:InterfaceConfig:node1:GigabitEthernet2",
    "type": "InterfaceConfig",
    "name": {
        "type": "Property",
        "value": "GigabitEthernet2"
    },
    "description": {
        "type": "Property",
        "value": "GigabitEthernet2 interface"
    },
    "interfaceType": {
        "type": "Relationship",
        "object": "urn:ngsi-ld:YANGIdentity:ianaift:ethernetCsmacd"
    },
    "enabled": {
        "type": "Property",
        "value": true
    },
    "linkUpDownTrapEnable": {
        "type": "Property",
        "value": "enabled"
    }
}
```


Figure 4.25: Request body for a NGSI-LD update entity operation to configure network interface using a data virtualization approach.


```
{
    "ip": {
        "type": "Property",
        "value": "10.1.1.11"
    },
    "netmask": {
        "type": "Property",
        "value": "255.255.255.0"
    }
}
```


Figure 4.26: Request body for a NGSI-LD update entity operation to configure IPv4 addressing of a network interface using a data materialization approach.

The data virtualization approach is the most suitable and efficient because in a data materialization approach the data consumer must perform a GET RPC operation before the SET RPC operation, as described in the workflows of Subsection 3.2.4 of Chapter 3.

Subscribing to telemetry data and calculating network link performance

In this use case, a network administrator needs to evaluate the performance of network links along a monitored network. To do this, the NetOps consumer could create a NGSI-LD subscription operation to entities of type *InterfaceStatistics* or *InterfaceStateCounters* to receive performance statistics about network link interfaces periodically over time.

In addition, the consumer could rely on a network analytics application to calculate network KPIs on demand by using aggregation functions on the telemetry data. Using particular options of the NGSI-LD Subscription API to receive notifications about the value of particular

```
{
    "id": "urn:ngsi-ld:InterfaceConfigIpv4Address:node1:GigabitEthernet2:10.1.1.11",
    "type": "InterfaceConfigIpv4Address",
    "isPartOf": {
        "type": "Relationship",
        "object": "urn:ngsi-ld:InterfaceConfigIpv4:node1:GigabitEthernet2"
    },
    "netmask": {
        "type": "Property",
        "value": "255.255.255.0"
    },
    "ip": {
        "type": "Property",
        "value": "10.1.1.11"
    }
}
```


Figure 4.27: Request body for a NGSI-LD create entity operation to configure IPv4 addressing of a network interface using a data virtualization approach.

NGSI-LD properties and the date and time on which they were obtained, the application could calculate KPIs, such as throughput, from the materialized telemetry data.

Let's imagine that the network administrator needs to monitor the network link performance between nodes *node1* and *node2*. By subscribing to traffic statistics from the network link interfaces, the NetOps consumer could periodically receive notifications of incoming or outgoing bytes and then calculate the traffic rate. Figure 4.28 shows an example of the required subscription request body, which is used in an HTTP POST operation as follows:

```
POST /ngsi-ld/v1/subscriptions HTTP/1.1
```

In the body of the request, we can see as the target of the subscription the entities of type *InterfaceStatistics* and *InterfaceStateCounters* with identifiers that determine the target node interfaces in a given network to obtain notifications of the *inOctects* and *outOctets* properties. The *attributes* parameter allows consumers to specify the properties and relationships to include in the notifications to be received when the subscription operation is triggered. Once the subscription is completed, the notifications will contain the latest value of the *inOctets* and *outOctets* properties, as updated in the Context Broker with the periodic telemetry data received from the network nodes.

In addition, the data parsing solution implemented in the Descriptive NDT for data retrieved by the NETCONF or gNMI protocols allows including as metadata the timestamp with the instant in which the notifications were received from the network nodes. This is possible thanks to the native *observedAt* temporal property facilitated by the NGSI-LD API.

Then, if the NetOps consumer keeps a record of the value and time of the last measurement that is included in the notifications received, determining the increment between the current and previous measurement values divided by the increment between measurement times, the network analytics application could calculate the traffic rate at each given moment, as illustrated by the following expression for calculating the incoming traffic rate (Expression 4.1):

$$Incoming\ Traffic\ Rate\ (t);\ (Mbps) = \frac{\Delta(inOctets(t)) \times 8}{\Delta t \times 10^6}, \tag{4.1}$$

where t represents the observation time of the counters, recorded by the *observedAt* parameter. The time interval Δt corresponds to the difference between two consecutive observation instants.

```
{
    "id": "urn:ngsi-ld:Subscription:InterfaceTrafficTelemetryStatistics",
    "type": "Subscription",
    "entities": [
        {
            "type": "InterfaceStatistics",
            "id": "urn:ngsi-ld:InterfaceStatistics:network1:node1:eth1"
        },
        {
            "type": "InterfaceStateCounters",
            "id": "urn:ngsi-ld:InterfaceStateCounters:network1:node2:eth1"
        }
    ],
    "description": "Subscription to eth1 interface traffic statistics on node1
                    and node2 network nodes.",
    "notification": {
        "attributes": ["inOctets", "outOctets"],
        "endpoint": {
            "uri": "http://network-analytics-consumer:8080",
            "accept": "application/json"
        }
    }
}
```

Figure 4.28: NGSI-LD subscription to interface traffic telemetry statistics.

In a similar way, the network analytics application can determine the packet rate in the given network link as another network KPI. By subscribing to traffic statistics from the network link interfaces, the NetOps consumer could periodically receive notifications of incoming or outgoing packets, and then calculate the packet rate at any given moment. Figure 4.29 shows an example of the required subscription request body, which is used in an HTTP POST operation as the latest using the NGSI-LD API.

In the body of the request, we again see as the subscription target the entities of type *InterfaceStatistics* and *InterfaceStateCounters* with their identifiers that determine the target nodes and their interfaces within a given network from which to receive notifications. The *attributes* parameter is now omitted because it is necessary to receive notifications from multiple properties according to the IETF and OpenConfig NGSI-LD domain models. Once subscribed, the notifications will contain the latest value of all properties of the textitInterfaceStatistics and *InterfaceStateCounters* entities, updated in the Context Broker with the periodic telemetry data received from the network nodes.

Then, considering the increment of the values of the different counters received between each notification occurrence, the network analytics application could calculate the packet rate within each interface at the edge of the network link at any given time, as shown in the following expression for calculating the incoming packet rate (Expression 4.2):

```
{
    "id": "urn:ngsi-ld:Subscription:InterfaceTelemetryStatistics",
    "type": "Subscription",
    "entities": [
        {
            "type": "InterfaceStatistics",
            "id": "urn:ngsi-ld:InterfaceStatistics:network1:node1:eth1"
        },
        {
            "type": "InterfaceStateCounters",
            "id": "urn:ngsi-ld:InterfaceStateCounters:network1:node2:eth1"
        }
    ],
    "description": "Subscription to eth1 interface statistics on node1 and
                    node2 network nodes.",
    "notification": {
        "endpoint": {
            "uri": "http://network-analytics-consumer:8080",
            "accept": "application/json"
        }
    }
}
```


Figure 4.29: NGSI-LD subscription to interface telemetry statistics.

$$Incoming\ Packet\ Rate\ (t);\ (packets/s) = \frac{\Delta(inPkts(t))}{\Delta t}, \tag{4.2}$$

where *inPkts* represents the total number of packets successfully received within a network interface. It should be noted that this *inPkts* parameter exist in the *InterfaceStateCounters* entity underlying the OpenConfig NGSI-LD model, but not in the *InterfaceStatistics* entity defined by the IETF NGSI-LD model, and in this case should be calculated by the sum of received packets that are unicast, multicast, and broadcast.

Now, let's assume that the network administrator needs to calculate a different KPI that measures the percentage of packet loss on the same network link. By subscribing to traffic statistics from network link interfaces using the same subscription request body as in the packet rate KPI case (see Figure 4.29), the NetOps consumer could periodically receive notifications of incoming or outgoing packets with and without errors, and then calculate packet loss.

Once again, considering the increment of the values of the different counters received between each notification occurrence, the network analytics application could calculate the percentage of packet loss within each interface at the edge of the network link at any given time, as shown in the following expression for calculating the incoming packet loss (Expression 4.3):

$$Incoming\ Packet\ Loss\ (t);\ (\%) = \frac{\Delta(discards(t) + errors(t)) \times 100}{\Delta(inPkts(t) + discards(t) + errors(t))}, \tag{4.3}$$

where *errors* and *discards* represent the total number of erroneous and discarded packets on a network interface, allowing the analytics application to calculate packet loss. To measure these erroneous and discarded packets, the *inDiscards* and *inErrors* parameters provided by

both the IETF and OpenConfig NGSI-LD domain models could be used. On the one hand, the *inDiscards* parameter represents packets discarded at the input of the interface. On the other hand, the *inErrors* parameter represents packets received with errors.

For the different situations raised in this use case, the Descriptive NDT can take a data integration approach based on either data materialization or data virtualization. If keeping a record of the telemetry statistics notifications is not required, data virtualization would be the most efficient solution.

Subscribing to operational status of network interfaces for fault detection

For this use case, a network administrator wants to detect link failures that may affect network operation. To this end, the NetOps consumer could rely on a network analytics application that notifies about the operational status of network interfaces within the whole network topology. In this sense, the network analytics application can detect an unexpected link fault and determine its impact on the network.

Let's imagine that the network administrator needs to monitor failures on a network link between the nodes *node1* and *node2*. For this case, the NetOps consumer needs to subscribe to changes in entities of type *Interface* and *InterfaceState* with identifiers that determine the target linked nodes and their interfaces within a given network to obtain notifications about the operation status of the corresponding network interfaces.

Figure 4.30 shows an example of the required subscription request body. The *watchedAttributes* parameter allows consumers to subscribe to particular properties such as *operStatus* and *lastChange*, which will allow them to get notifications about the change of the operational status of the interfaces, as well as the date and time when this change occurs.

This example demonstrates that the Descriptive NDT could provide information about the operational status of a monitored network, allowing the network analytics application to identify possible link failures.

As in the previous use case, the Descriptive NDT can follow a data integration approach based on either data materialization or data virtualization. However, if keeping a record of the network interface operational status notifications is not required, data virtualization would be the most efficient solution.

Dynamic discovery of new link-layer neighbors

For this use case, a network administrator wants to detect new Level 2 neighbors in a specific network node interface. To this end, the NetOps consumer could rely on a network analytics application that notifies about new entries in the ARP or neighbor cache within a network interface. In this sense, the network analytics application can dynamically discover new neighbors.

```
{
    "id": "urn:ngsi-ld:Subscription:InterfaceOperationalStatus",
    "type": "Subscription",
    "entities": [
        {
            "type": "Interface",
            "id": "urn:ngsi-ld:Interface:network1:node1:eth1"
        },
        {
            "type": "InterfaceState",
            "id": "urn:ngsi-ld:InterfaceState:network1:node2:eth1"
        }
    ],
    "description": "Subscription to the status of the network link
                    interfaces between nodes node1 and node2.",
    "watchedAttributes": ["operStatus", "lastChange"],
    "notification": {
        "attributes": ["operStatus", "lastChange"],
        "endpoint": {
            "uri": "http://network-analytics-consumer:8080",
            "accept": "application/json"
        }
    }
}
```

Figure 4.30: NGSI-LD subscription to the operational status of network interfaces.

Let's imagine that the network administrator needs to monitor neighbor entries over IPv4 and IPv6 in a network interface of a network node *node1*. For this case, and following the non-NMDA IETF NGSI-LD model for IP management, the NetOps consumer needs to subscribe to changes in *InterfaceStateIpv4Neighbor* and *InterfaceStateIpv6Neighbor* entities to receive notifications about new ARP or NDP entries, depending on the IP address type. Figure 4.31 shows an example of the required subscription request body.

Similarly to the previous use cases, in this use case, the Descriptive NDT can take a data integration approach based on either data materialization or data virtualization. However, if keeping a record of the new dynamically discovered neighbors is not required, data virtualization would be the most efficient solution.

Obtaining network flows matching with particular criteria

Regarding the case of monitored network flows, using the NGSI-LD model described in Subsection 4.2.3, a NetOps can search for network flows that match particular conditions, such as source and destination IP addresses, source and destination Ethernet addresses, source and destination network interfaces, TCP or UDP transport protocol, IPv4 or IPv6 traffic, etc. For example, the following query allows getting the entities of type *ExportPacketFlowDataRecordIPv4* with particularized source and destination IP addresses:

```
GET /ngsi-ld/v1/entities/?type=ExportPacket-
FlowDataRecordIpv4&q=srcAddress==192.168.1.10;
dstAddress==192.168.1.1 HTTP/1.1
```

```
{
    "id": "urn:ngsi-ld:Subscription:NeighborDiscovery",
    "type": "Subscription",
    "entities": [
        {
            "type": "InterfaceStateIpv4Neighbor",
            "id": "urn:ngsi-ld:InterfaceStateIpv4Neighbor:network1:node1:eth1"
        },
        {
            "type": "InterfaceStateIpv6Neighbor",
            "id": "urn:ngsi-ld:InterfaceStateIpv6Neighbor:network1:node1:eth1"
        }
    ],
    "description": "Subscription to new ARP and Neighbor Discovery Protocol
                    entries dynamically discovered on the eth1 interface of
                    network node node1.",
    "notification": {
        "endpoint": {
            "uri": "http://network-analytics-consumer:8080",
            "accept": "application/json"
        }
    }
}
```


Figure 4.31: NGSI-LD subscription to receive notifications of new Level 2 neighbors discovered on a network interface.

The response of the above query provides the list of *ExportPacketFlowDataRecordIPv4* entities that match with the specified IP address criteria. In addition, these entities have relationships of type *isPartOf* referring to entities of type *ExportPacketFlowDataRecord* with the detailed information of the corresponding network flows.

An interested NetOps consumer can also query particular traffic statistics from network flows, such as searching the information of network flows with incoming packets or incoming bytes above a certain threshold and counting how many of these flows are found. For example, the following query searches for *ExportPacketFlowDataRecord* entities with the *bytesIn* property with a value equal to or greater than 1,000 bytes that are TCP flows for IPv6 traffic:

```
GET /ngsi-ld/v1/entities/?type=ExportPacket-
FlowDataRecord&q=bytesIn>=1000;protocol==tcp;
ipVersion==ipv6&count=true&limit=50 HTTP/1.1
```

The query response provides the list of *ExportPacketFlowDataRecord* entities that match the specified criteria. In addition, thanks to the count number of results functionality of NGSI-LD API using the *count* parameter, a special HTTP header named *NGSILD-Results-Count* is also included within the response, determining the number of entities that match those criteria. Apart from that, due to the pagination behavior functionality of NGSI-LD API, by using the *limit* parameter, the consumer can specify the maximum number of matching NGSI-LD elements to be retrieved in the query.

Similarly, when the monitoring data from the network flow data records is aggregated to calculate the additional KPIs described in Subsection 4.2.3, the NetOps consumer can combine counting and searching for network flows based on the KPIs. For example, the following query searches for *ExportPacketFlowDataRecord* entities with the *flowDuration* property with

a value equal to or greater than 1,000 milliseconds and the *pktsInPerSecond* property with a value equal to or greater than 100 packets, as well as matching UDP flows for IPv4 traffic:

```
GET /ngsi-ld/v1/entities/?type=ExportPacket-
FlowDataRecord&q=flowDuration>=1000;
pktsInPerSecond>=100;protocol==udp;ipVersion==ipv4
&count=true HTTP/1.1
```

This use case for NetFlow is intended for a data materialization approach, as the NetOps consumer communicates with the Descriptive NDT to access historical information about network flows stored in the NGSI-LD Context Broker. The materialization approach would require the NDT data integration system to be able to store a large amounts of information about the monitored flows in the network, which seems unfeasible without a back-end system capable of handling huge volumes of data.

Alternatively, the Descriptive NDT could offer a solution based on data virtualization. In this alternative, the NetOps consumer could subscribe to the NGSI-LD context information related to the NetFlow v9 protocol to trigger the monitoring of network flows and extract accurate information. For instance, similar to the second example of the data materialization case, the NetOps consumer can subscribe to *ExportPacketFlowDataRecord* entities with the *bytesIn* property with a value equal to or greater than 1,000 bytes in order to receive notifications about network flows matching this criteria. Figure 4.32 shows an example of the required subscription request body:

```
{
    "id": "urn:ngsi-ld:Subscription:ExportPacketFlowDataRecord",
    "type": "Subscription",
    "entities": [
        {
            "type": "ExportPacketFlowDataRecord"
        }
    ],
    "description": "Subscription to IP network flows with incoming octets equals
                    or greater that 1000 bytes.",
    "q": "bytesIn>=1000",
    "notification": {
        "endpoint": {
            "uri": "http://network-analytics-consumer:8080",
            "accept": "application/json"
        }
    }
}
```

Figure 4.32: NGSI-LD subscription to network flows.

According to this latter case of subscribing to network flows under a virtualization approach, the NetOps consumer will receive notifications about the network flows that meet the specified requirements as they are being monitored from the network. In this case, the data is not stored in the NDT and is consumed on-demand, leaving the management of the data in the hands of NetOps. A problem with the virtualization approach is that additional network flow information associated with other NGSI-LD entities is lost, such as the source and destination IP addresses of the flow and the associated export packet. One

possible solution is for the data consumer to create a subscription to multiple entities, such as *ExportPacket*, *ExportPacketFlowDataRecord*, and *ExportPacketFlowDataRecordIpv4*, and manage the resulting notifications.

4.3 Overview of the Descriptive NDT System

This Chapter 4 describes the implementation details of a prototypical Descriptive NDT solution that follows the architectural and design principles of the NDT proposal defined in Chapter 3. The NDT solution implements a digital mapping system to convert YANG modeled data to NGSI-LD, allowing the automation of the translation process from real network data to NDT data. To complete this translation process, this programmatic methodological approach enables the automatic generation of data parsers to process YANG compliant data, ensuring seamless data integration. The validated prototype employs a hybrid data integration approach, implementing both data materialization and data virtualization options. Chapter 5 will evaluate the performance of both data integration approaches to realize the data mapping mechanism implemented in the Descriptive NDT proposal. In addition, Chapter 4 presented a case study to validate the Descriptive NDT approach. The study included examples covering several network operations and management use cases to demonstrate the feasibility and usefulness of the proposed solution.

Chapter 5

Experimental Results and Evaluation

This chapter presents the experimental performance results and evaluation of the prototype implemented for the NDT solution proposed in this Doctoral Thesis. First, the experimental testbed used to carry out the performance experiments is described. Then, we evaluate the performance results of the NDT data integration pipeline using a data materialization approach for the network data acquisition process. Finally, data materialization and data virtualization approaches are compared in the context of network management by analyzing the query processing and data integration steps of the proposed NDT solution.

Table of Contents

5.1 Experimental Testbed 136

5.2 Performance of the Descriptive NDT Prototype for Network Data Acquisition 139

5.2.1 Interface telemetry performance experiments 140

5.2.2 NetFlow monitoring performance experiments 144

5.2.3 Discussion on experimental results 146

5.3 Performance of the Descriptive NDT Prototype for Completing Network Operations 148

5.3.1 Latency performance results for NETCONF operations 151

5.3.2 Latency performance results for gNMI operations 154

5.3.3 Latency performance results for NetFlow monitoring 156

5.3.4 Discussion on experimental results 158

5.1 Experimental Testbed

This section describes the experimental testbed used to validate the functionality of the Descriptive NDT solution and measure the experimental performance results. Initially, the testbed designed for the functional evaluation of the Descriptive NDT solution consists of virtual network emulation scenarios composed of router devices supporting NETCONF and gNMI. For NETCONF, a router model supporting basic RPC operations and telemetry subscriptions based on YANG Push is required. Similarly, for gNMI, a router model capable of handling basic RPC operations and gNMI subscriptions is required.

Although network automation based on YANG data modeling is widespread in the industry, in the context of academic research, it is difficult to find network devices that fully implement the capabilities offered by model-driven network management protocols such as NETCONF or gNMI. However, network manufacturers or vendors such as Cisco, Nokia, Juniper, or Arista Networks provide access to the network operating system of their own network device models that support YANG capabilities, either as open-source or through a purchased software license. The software distribution of these network operating systems enables installing and virtualizing the functionality of these network devices in virtual machines or using lightweight container-based virtualization running on top of general-purpose hardware devices based on Linux servers.

However, there are some difficulties in finding network devices that support all the capabilities offered by the NETCONF and gNMI network management protocols. On the one hand, since gNMI is a fairly recent management protocol, only the most recent models of network device manufacturers support data management and monitoring mechanisms using this protocol. On the other hand, for the NETCONF management protocol, only certain models of network device manufacturers support dynamic subscription operations using the YANG Push mechanism.

Table 5.1 provides a comparative analysis of NETCONF and gNMI capabilities for router models from different vendors. As can be seen in Table 5.1, among the router models analyzed, only the Cisco IOS XE router model supports dynamic subscriptions using YANG Push with the NETCONF protocol and XPath filtering for performing the different RPC operations. It is also the only one that provides compatibility with the YANG modules standardized by the IETF and support for the YANG Library module [195]. This YANG Library module is important in this context because it provides additional information about YANG modules and datastores supported by a network management server, and it also adds support for network platforms compatible with the NMDA architecture. Additionally, none of the router models analyzed supports the NMDA architecture. Therefore, they do not implement differentiated datastores for configuration and operational status data. Instead, they implement YANG modules to fork the data for this purpose. Meanwhile, among the router models evaluated with gNMI support, Cisco IOS XRv is the most compatible with the YANG data models defined by OpenConfig and is the most suitable for extracting network telemetry data, since in most of the other network operating systems evaluated, telemetry event notifications usually arrive out of sequence, out of time, and with ungrouped data. In addition, Table 5.1 shows that all router models analyzed support NetFlow v9.

Table 5.1: The classification of NETCONF, gNMI, and NetFlow v9 protocol capabilities in router models from different manufacturers.

Capabilities	**Arista cEOS**	**Cisco IOS XE**	**Cisco IOS XRv**	**Juniper vJunos**	**Nokia SR Linux**
NETCONF basic RPC operations (RFC 6241)	X	X	X	X	X
NETCONF Subtree filter	X	X[a]	X	X	X
NETCONF XPath filter	-	X	-	-	-
YANG Push subscriptions (RFCs 8639 and 8641)	-	X	-[b]	-	-
Native IETF YANG modules support	-	X	-	-	-
YANG Library module support (RFC 8525)	-	X	-	-	X
NMDA support (RFCs 8526 and 8342)	-	-	-	-	-
gNMI version	0.7.0	-	0.7.0	-	0.10.0
gNMI supported encodings	JSON, JSON_IETF, and ASCII	-	ASCII, JSON_IETF, and PROTOBUF	-	JSON, JSON_IETF, and ASCII
gNMI basic RPC operations	X	-	X	-	X
gNMI telemetry notification capable	X[c]	-	X	-	X[c]
Native OpenConfig YANG modules support	X	-	X	-	X[d]
NetFlow v9 support	X	X	X	X	X

[a]For NETCONF YANG Push subcriptions, Cisco IOS XE model only support XPath filtering. Subtree filtering is not available.
[b]Only some series of Cisco IOS XRv model supports YANG Push subscriptions, as well as requires additional licensing and specific configurations.
[c]Telemetry events may be delivered out of sequence, out of time, and separately due to internal buffering and processing issues.
[d]OpenConfig support in gNMI requires the activation of additional features, advanced configuration, and in some cases, specific licenses.

After analyzing the capabilities offered by these different router device models, a Cisco IOS XE router model CSR1000V series was selected to evaluate NETCONF, while a Cisco IOS XRv router model 9000 series was selected to evaluate gNMI[5.1]. For a NETCONF client working as a data collector component, the *ncclient* [196] Python library is used, which supports all RPC operations and capabilities defined by the NETCONF management protocol. Meanwhile, for gNMI, a CLI client called *gNMIc* [197] is used, since it provides full support for all RPC operations and capabilities defined by the gNMI management protocol. In addition, both router models support the NetFlow v9 exporter role, enabling to validate the integration of NetFlow monitoring data within the Descriptive NDT prototype and evaluate the performance as well. To collect network flows, the collector implementation for NetFlow v9 called *GoFlow2* [198] has been used. *GoFlow2* can publish the collected network flow data in a semi-structured JSON format using a message bus system based on Apache Kafka. However, before integrating NetFlow monitoring data into the Descriptive NDT, it must first be translated from the native *GoFlow2* data format to a normalized JSON data format according to the YANG model defined for NetFlow v9. This process was described in Subsection 4.2.3 of Chapter 4.

The virtualized scenarios used to validate the Descriptive NDT performance consist of the deployment of simple point-to-point network topologies with the aforementioned router models by using the Containerlab [199] virtualization tool. Containerlab facilitates the creation of container-based networking scenarios consisting of network devices with different network operating systems[5.2].

However, since the containerized network devices used in these scenarios consume a lot of computational resources, the experimental testbed was adapted to generate and collect synthetic data. This is used in the initial experimental phase described in Section 5.2, which focuses on measuring the performance of network data acquisition. This was done to avoid limiting the performance of the machine on which the experiments are conducted. These synthetic data resemble the data collected from functional tests using NETCONF, gNMI, and NetFlow v9 within Containerlab-based networking scenarios. Thus, the experimental performance of the Descriptive NDT framework can be analyzed under better and more flexible conditions because samples of synthetic data can be replicated for specific tests. Note that these synthetic data were also used to perform unit tests during the initial validation phases of the proposed framework for testing purposes. Meanwhile, the experimental phase conducted in Section 5.3 considers the virtual environment in Containerlab as part of the Descriptive NDT experimental testbed. Rather than focusing solely on data acquisition, the resulting tests in Section 5.3 measure the performance of querying network operational data, subscribing to network interface telemetry data, and subscribing to monitored network flows.

[5.1] Access to the network operating system of these router models was obtained through a subscription license to the Cisco Modeling Labs network simulation platform, which provides disk image files in QEMU Copy On Write (QCOW) format for the software of the various router models. More information about Cisco Modeling Labs is available at: https://developer.cisco.com/modeling-labs/

[5.2] In order to be able to use the QCOW2 disk images with the network operating system of the selected router models in Containerlab scenarios, the open-source tool called *vrnetlab* was used, as it allows to configure router images prepared to boot as KVM (Kernel-based Virtual Machine) virtual machines to run as Docker containers. More information about *vrnetlab* is available at: https://github.com/hellt/vrnetlab

Figure 5.1 depicts the complete environment for validating the Descriptive NDT capabilities and performing the experimental analysis.

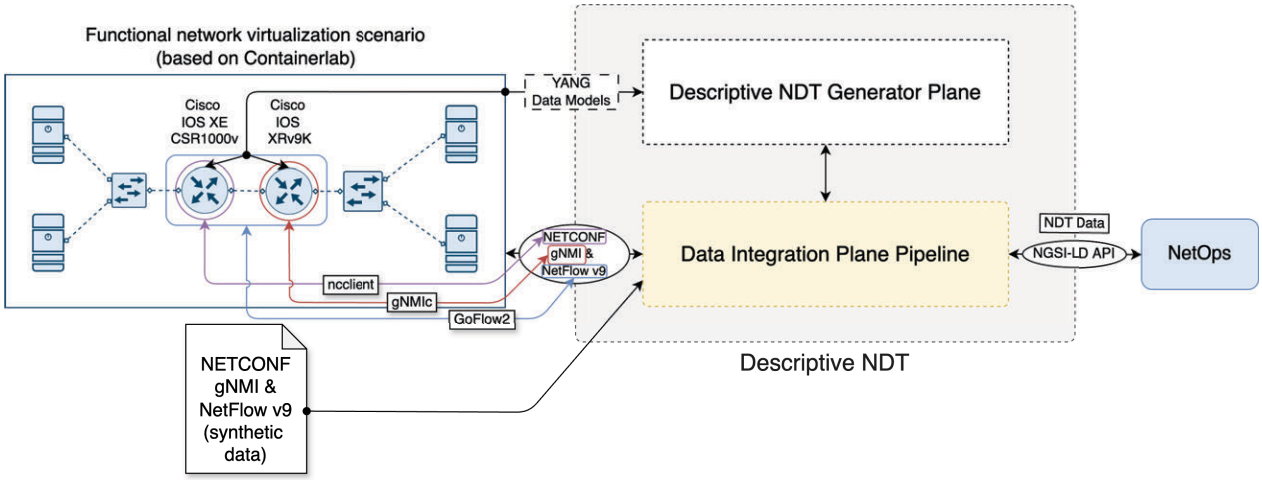


Figure 5.1: Experimental testbed for validating the Descriptive NDT capabilities and performing the performance analysis.

In addition, both Orion-LD [200] and Scorpio [201] were used as NGSI-LD Context Brokers during the functional and experimental phases of the Descriptive NDT framework. The testbed used for conducting these performance experiments and functional evaluation has been deployed on a single machine running Ubuntu 20.04.4 LTS, with 32 GB RAM and 8 CPU cores. The results of these experiments and the definition of the functional virtualization scenarios mentioned above are available on GitHub along with the code of the Descriptive NDT prototype [173].

5.2 Performance of the Descriptive NDT Prototype for Network Data Acquisition

This section describes the initial performance experiments conducted to validate the Descriptive NDT prototype solution approach presented in this dissertation, focusing on the acquisition and replication of network data on the NDT. The experiments described in this section include performance measurements to cover: i) data translation process from YANG to NGSI-LD; and ii) NDT data instantiation process following a data materialization approach. The validation has been tailored to the YANG models for network interface management and network flows, which are covered in the case study presented in Section 4.2 of Chapter 4.

The experiments were designed to analyze the performance of the Descriptive NDT prototype in terms of latency and throughput. Three experiments were defined. The first experiment measures the latency of the XML and JSON Data Parsers & Mappers components used to translate from YANG modeled data to NGSI-LD modeled data. The second experiment measures the combined latency of the YANG to NGSI-LD translation process plus the data materialization process within the NGSI-LD Context Broker using the corresponding NGSI-LD Instantiator functionality.

The NGSI-LD Instantiator implements the required data validation process to achieve compatibility with the target NGSI-LD model. This process occurs after the YANG to NGSI-LD translation and before data materialization. The NGSI-LD Instantiator component combines the data validation and data materialization stages, which are referred to as data instantiation. Both stages are included in the latency measurements performed for the second experiment.

Figure 5.2 summarizes the procedure for measuring the latency of the network data acquisition process from the time the XML and JSON Data Parsers & Mappers components receive the network monitoring data until the resulting NDT data are integrated within the NGSI-LD Context Broker, where they can be consumed by interested NetOps. Finally, there is a third type of experiment for measuring the throughput of both the YANG to NGSI-LD translation phase and the data instantiation phase. The experiments are based on collecting telemetry statistics from network interfaces via notifications that are received either by the NETCONF or by the gNMI management protocols, as well as collecting NetFlow monitoring data.

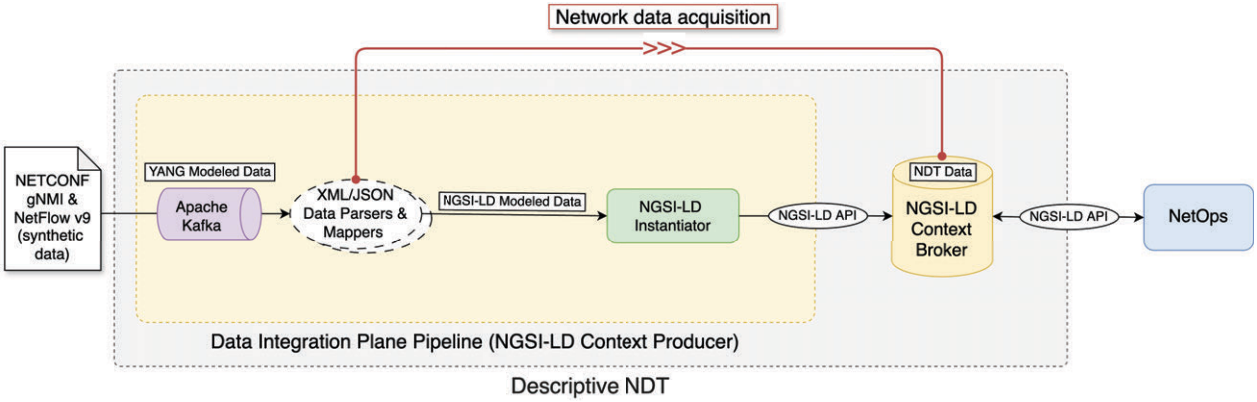


Figure 5.2: Measuring the latency performance for network data acquisition.

To visualize the results of the experiments, boxplot charts are used, implemented through the Python *matplotlib* [202] library. The boxplot is a graphical representation that summarizes the distribution of a dataset using its quartiles. It displays a box spanning from the first to the third quartile, which is referred to as the interquartile range (IQR). The box includes a line indicating the median, also known as the second quartile. Data points within 1.5 times the IQR are covered by whiskers, while values beyond the whiskers are considered outliers. Boxplots allow for quick identification of the symmetry and dispersion in a distribution, as well as the presence of extreme values or anomalies, making them especially useful for comparing series of measurements such as latency or processing time.

5.2.1 Interface telemetry performance experiments

In latency experiments involving interface telemetry data received from the NETCONF and gNMI management protocols, the number of network interfaces of a router device from which telemetry statistics notifications arrive was considered as a variable of interest. The experiments are carried out according to the number of network interfaces that are common for routers, considering sets of 12, 24, 36, and 48 interfaces. The time between notifications

is configured to 5 seconds. In each type of latency experiment, the average delay time of a total of 1,000 notifications measured based on the number of network interfaces is obtained. It should be noted that in NETCONF, notifications from all interfaces come wrapped in a single notification event, while in gNMI each event from each interface comes in a separate notification. This means, for example, that in the case of obtaining notifications from 48 interfaces, 1,000 notifications will come for NETCONF and 48,000 for gNMI. Therefore, in the case of gNMI, the sum of the delay times of the sequential notification events associated with each set of interfaces needs to be combined until the total number of notification events is reached, to obtain the overall average delay time for each category of interface sets.

Figure 5.3 depicts the latency results of the first experiment, representing the distribution of average processing latency as a function of the number of interfaces. It can be seen that a linear function with a smooth slope is obtained and that the time to perform the data parsing and data mapping processes is minimal. We also observe that the average latency with gNMI notifications for each interface set is higher than with NETCONF, due to the aggregation of notification events considered in gNMI. In any case, the average delay is always below 10 milliseconds.

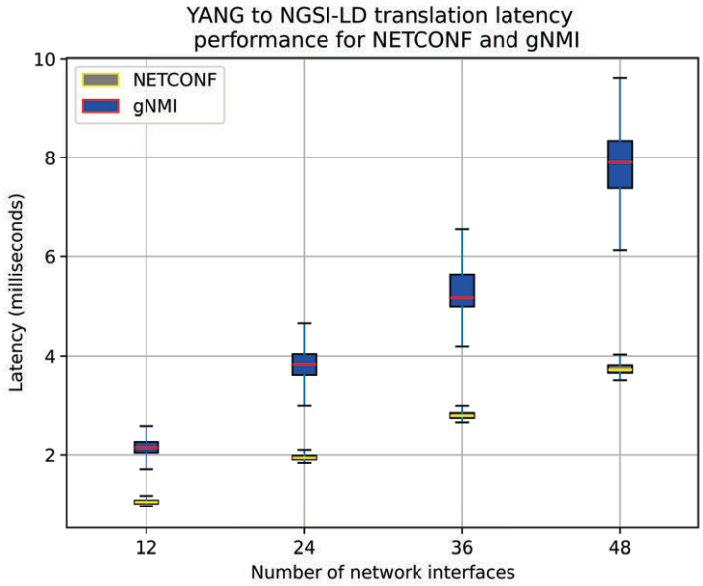


Figure 5.3: YANG to NGSI-LD translation latency.

Figure 5.4 depicts the latency results of the second experiment, again representing the distribution of the average processing latency as a function of the number of interfaces. It is notable how the latency time is more affected by the number of interfaces in this experiment. This is due to the materialization stage using the NGSI-LD API, which requires either more requests in the case of gNMI, or requests with higher payload in the case of NETCONF. This results in an increased processing time in the NGSI-LD Context Broker service. In this case, the average delay does not exceed 300 milliseconds. It should be noted that the processing time for instantiating the data also depends on the implementation of the chosen NGSI-LD Context Broker, with Orion-LD providing the best performance. This can be

seen in Figure 5.5, where we represent the latency comparing the performance of Orion-LD and Scorpio NGSI-LD Context Brokers. Note that in the case of Scorpio, unlike Orion-LD, the YANG to NGSI-LD mapping process works better with a larger number of requests, as required for gNMI, than with higher payload requests, as required for NETCONF.

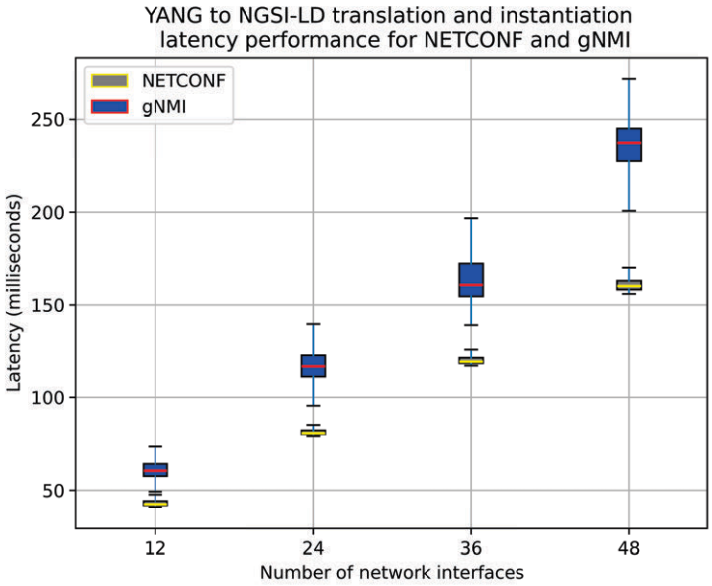


Figure 5.4: YANG to NGSI-LD translation and instantiation latency.

For the throughput experiments involving interface telemetry data, we compared the throughput of the YANG to NGSI-LD translation and data instantiation phases, taking into account the maximum throughput rate of incoming notifications on the Kafka message bus during the data extraction phase. These experiments consider notification events using NETCONF and gNMI with telemetry statistics from 48 interfaces for a period of 600 seconds. For these throughput experiments, the NGSI-LD Context Broker implementation of Orion-LD is used because of its better latency performance.

Figure 5.6 depicts the throughput results for NETCONF, representing the average throughput resulting from the translation of YANG to NGSI-LD and the data instantiation phases, considering the maximum event rate for the extraction of telemetry notifications. Notice that the maximum event rate for notification extraction is approximately 111 events per second, and the translation phase supports this service rate. However, the data instantiation phase limits the integration of NDT data to fewer than 6 events per second, meaning that it can service telemetry notifications from fewer than 288 network interfaces per second.

Figure 5.7 depicts the throughput results for gNMI. In this case, the maximum event rate for notification extraction is on the order of 1,240 events per second, and the YANG to NGSI-LD translation phase also supports this service rate. However, once again, the data instantiation phase limits the integration of the NDT data to a maximum of 250 events per second. Since each notification event in gNMI has data from only one network interface, this would mean servicing telemetry notifications from approximately 250 interfaces per second.

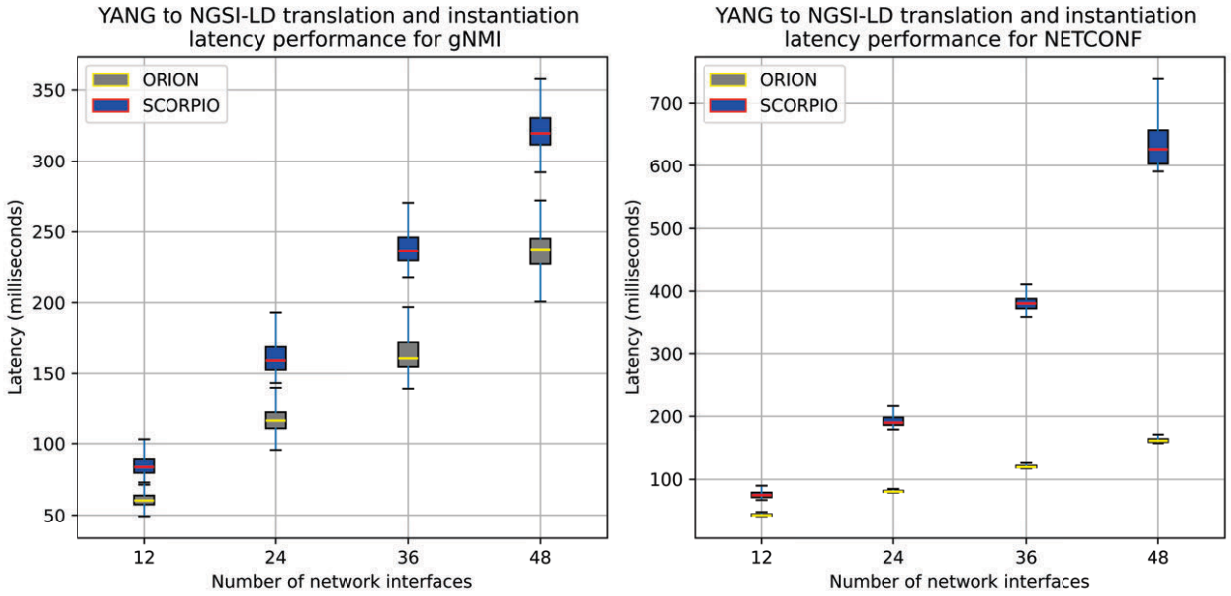


Figure 5.5: YANG to NGSI-LD translation and instantiation latency for NECONF and gNMI comparing performance of the Orion-LD and Scorpio NGSI-LD Context Brokers.

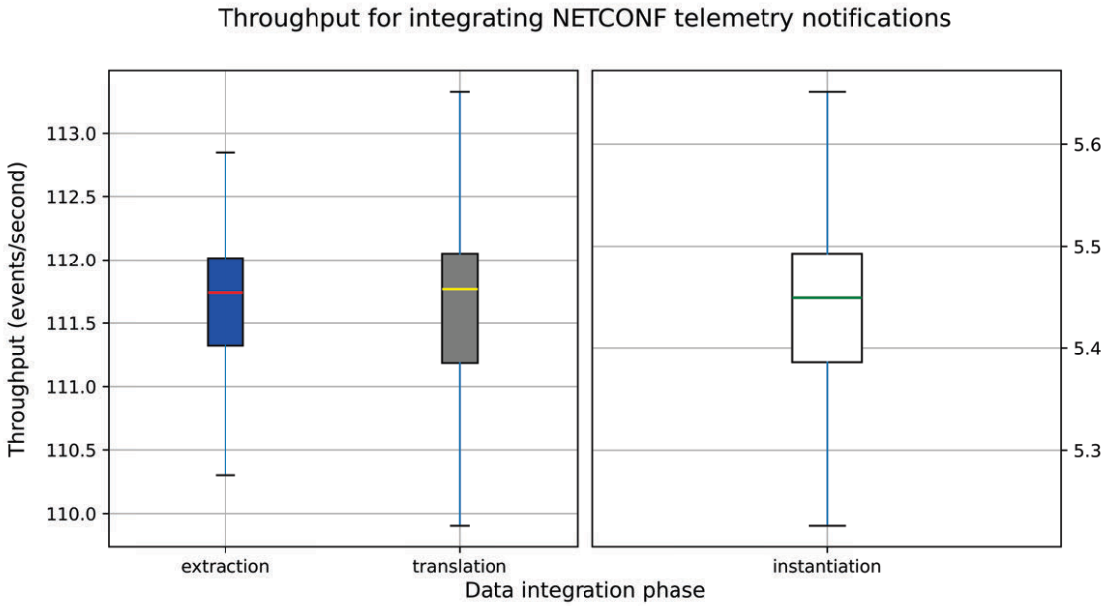


Figure 5.6: Throughput experiments for NETCONF notifications.

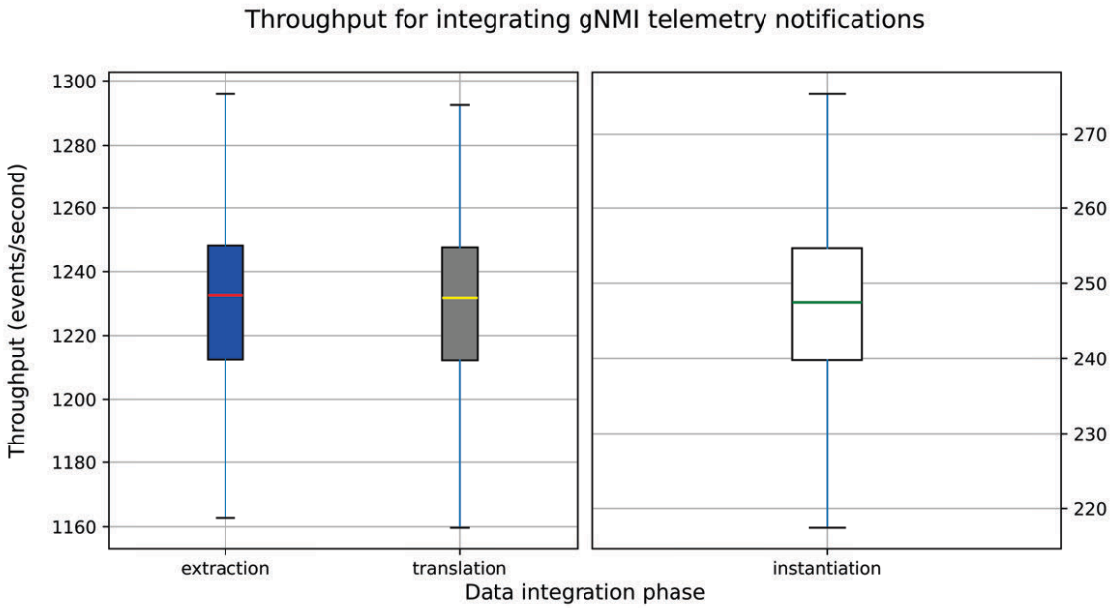


Figure 5.7: Throughput experiments for gNMI notifications.

5.2.2 NetFlow monitoring performance experiments

Additional performance experiments have been conducted to measure the latency of the YANG to NGSI-LD translation and data materialization processes according to NetFlow monitoring data. The initial functional testbed that has been used to validate the integration of NetFlow monitoring data consisted of a virtual network emulation scenario with *Cisco IOS XE CSR1000V* routers that support the NetFlow v9 protocol and can be configured as NetFlow exporters. However, as with the experiments involving telemetry data obtained via NETCONF and gNMI, the experimental measurements were performed using synthetic data resembling that obtained by a NetFlow collector instead of using a virtualized network environment from which to retrieve network flows.

In the experiments, the number of network flows per export packet that arrive as monitoring data was taken into account as a variable of interest. The experimentation is carried out according to an increasing number of network flows 10, 20, and 50 within an export packet. The time between export packet arrivals is considered to be 5 seconds. As a result, in each type of experiment, the average delay time of a total of 1,000 export packets measured based on the number of network flows is obtained. It should be noted that in the reference NetFlow collector implementation used in functional tests, which is called GoFlow2 [198], each received telemetry event consists of a single network flow data record related to an export packet. This means that the set of network flows in an export packet come in separate monitoring events and that for each event only a single network flow arrives. The same behavior has been simulated in the synthetic data generated. In the case of 1,000 export packets, each containing 50 network flows, a total of 50,000 events will be generated and processed. Then, the delay times of the monitoring events of each network flow within an export packet are averaged to determine the global average delay time of all the network flows per export packet.

The synthetic data generated from NetFlow monitoring data relate to normal IPv4 flows. Therefore, the data translation and instantiation processes account for the materialization of NGSI-LD entities that correspond to network flows (i.e., *ExportPacketFlowDataRecord* entity), as well as the export packets (i.e., *ExportPacket* entity), and IPv4 information (i.e., *ExportPacketFlowDataRecordIpv4* entity) relative to them.

Figure 5.8 depicts the latency results for NetFlow experiments, representing the average processing latency as a function of the number of network flows per export packet. In the experiment for measuring only the translation performance from YANG to NGSI-LD, it can be seen that the time to perform the data parsing and data mapping processes is minimal. The resulting average delay is of the order of units of milliseconds.

In the experiment that includes the data materialization process, we can see how latency is affected by the number of network flows. Again, this is due to the materialization stage, due to the larger number of requests that must be sent to the NGSI-LD Context Broker to upload and replicate the resulting entities, which leads to a longer processing time within the NGSI-LD Context Broker service. Even so, the average delay does not exceed one second. As in previous experiments, better results are obtained with Orion-LD than with Scorpio.

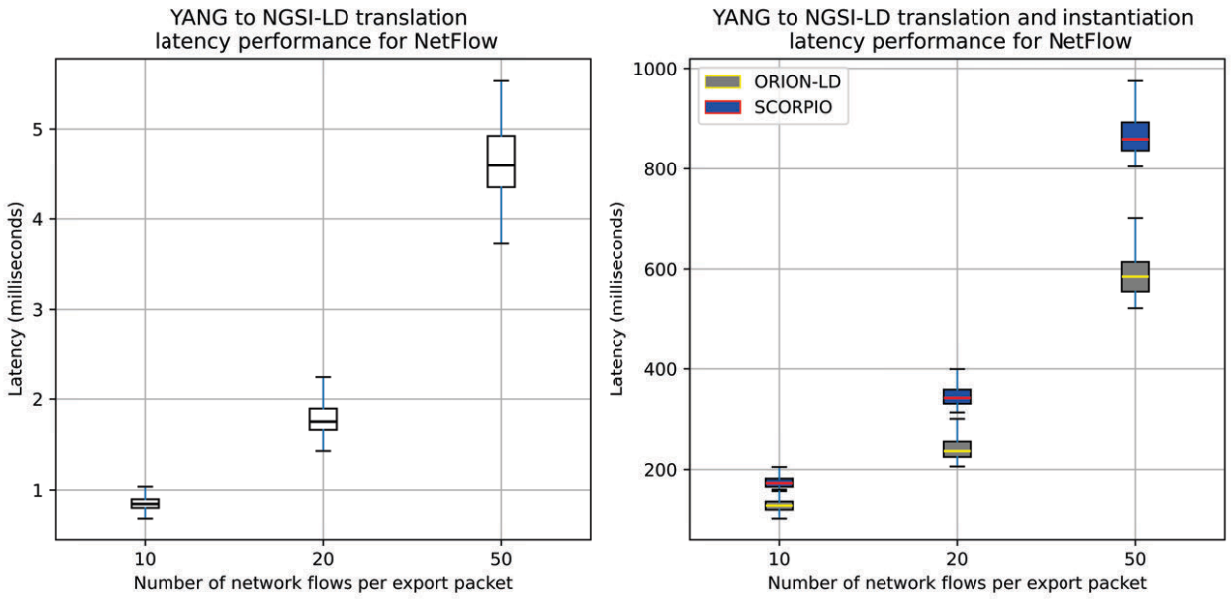


Figure 5.8: YANG to NGSI-LD translation and instantiation latency performance for NetFlow v9.

We have also included the throughput experiments for the NetFlow case. Similarly as with NETCONF and gNMI, we compare the throughput of the YANG to NGSI-LD translation and data instantiation phases considering the maximum throughput rate of incoming notifications on the Kafka message bus during the data extraction phase. These experiments consider notification events, each with only one network flow data record, for a period of 600 seconds. Again, for the throughput experiments, the NGSI-LD Context Broker implementation of Orion-LD is used given its better latency performance. Figure 5.9 depicts the throughput results for NetFlow. In such a case, the maximum event rate for notification extraction is on

the order of 1,140 events per second, and the YANG to NGSI-LD translation phase supports that service rate. However, once again, the data instantiation phase limits the integration of the NDT data to a maximum of 82 events per second, which means that only 82 network flow data records per second can be handled.

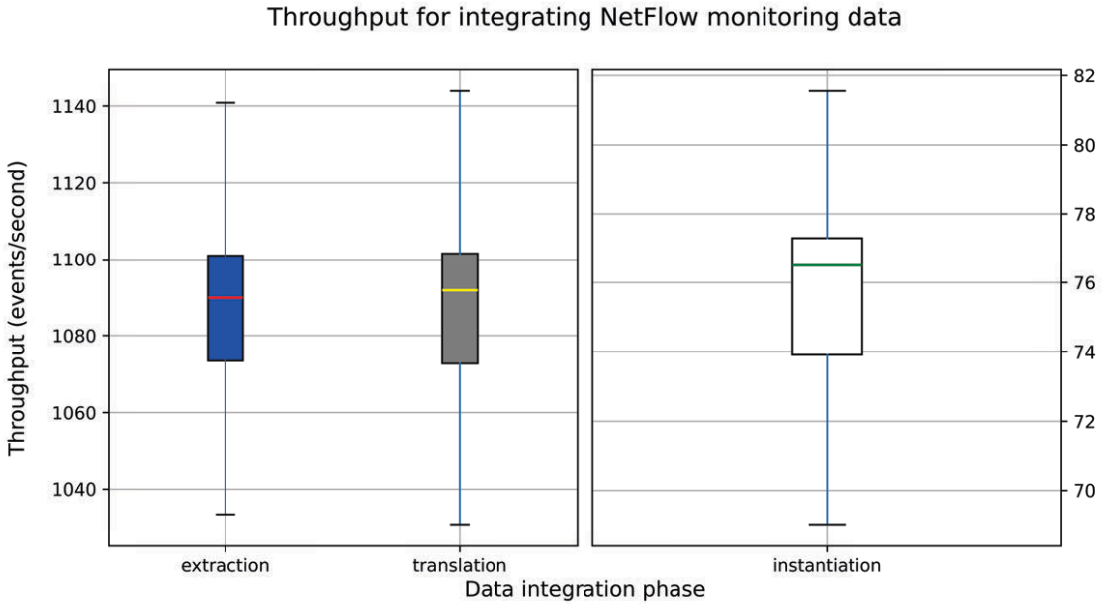


Figure 5.9: Throughput experiments for NetFlow notifications.

5.2.3 Discussion on experimental results

During this experimental phase, we validated the performance of the NDT solution for integrating telemetry data acquired from network devices. However, we found that the data instantiation phase introduces significant delays and becomes a bottleneck in the NDT data integration pipeline. This is due to the high volume of requests made during the data materialization process, which the NGSI-LD Context Broker cannot process in a timely manner. These limitations may result from the working environment, which consists of a single machine with limited computing resources running the Descriptive NDT prototype solution based on Docker microservices. We expect that the results will be significantly better in a production environment with servers that have greater computing capabilities and a prototype solution based on Kubernetes.

There are also some additional proposals that could be explored in the future to overcome these limitations. One of them is to take advantage of the distributed or federated architectures proposed by the NGSI-LD standard to improve the scalability and performance of the solution. Deploying different instances of Context Brokers and segmenting the scope of the Descriptive NDT solution across the target managed network would allow balancing the demands of requests so that they can be served efficiently. For this purpose, the implementation of Scorpio's NGSI-LD Context Broker needs to be analyzed more in depth due to its support of distributed NGSI-LD operations. Another proposal is to work on a variant of the Descriptive

NDT prototype solution where the data integration mechanism is based on a data virtualization approach. As previously analyzed throughout this document, this alternative data integration approach can be more efficient by eliminating the need to replicate and store the data in the NGSI-LD Context Broker and sending it directly to the interested consumers, thus improving the performance. Precisely, this Descriptive NDT solution based on the data virtualization approach will be evaluated in the following Section 5.3 to analyze its performance in terms of latency and compare the results with the data materialization implementation case.

Another challenge is how to ensure data replication and storage in an NDT solution based on a data materialization approach (i.e., a process also known as data warehousing), while taking into account the large volume of data resulting from network operations and management demands. As a first example, considering the experimental study realized above, for a telemetry notification event received from 48 interfaces in one instance, the size of the resulting materialized NGSI-LD data in the Descriptive NDT was 145,696 bytes for NETCONF and 93,840 bytes for gNMI. As a second example, for a NetFlow export packet with 50 flow data records monitored from a network device interface, the NGSI-LD data size was 150,650 bytes. Therefore, if we were to monitor the same number of network flows from 48 interfaces simultaneously in a single instant, the resulting NGSI-LD data size would be 7,231.2 kilobytes. Assuming that events of this size were continuously received from the 48 network interfaces every five seconds, including both telemetry statistics notifications received from NETCONF and gNMI and monitored network flows, the total volume of data would reach 129 gigabytes over the course of a full day. This highlights the difficulty of managing such massive amounts of data in an NDT solution without a robust data warehouse system.

Current NGSI-LD Context Broker implementations have a back-end based on SQL (e.g., Scorpio with PostgreSQL) or NoSQL (e.g., Orion-LD with MongoDB) database management system (DBMS) to store and maintain consistency of NGSI-LD data. Therefore, there is a growing need for high-performance DBMS solutions that can adapt to the NGSI-LD ecosystem, enabling distributed storage and processing of large amounts of data. One such solution is the Hadoop framework, which is commonly used for Big Data purposes in Data Lake or Data Warehouse solutions. This would enable a data repository structured according to NGSI-LD with high availability, fault tolerance, and scalability requirements.

In addition, complementary mechanisms can be sought to allow for data storage expiration control in the NDT data integration solution. The NGSI-LD API implements deletion operations on managed data entities, which would allow scheduling periodic or on-demand deletion of information. This would allow us to delete the resulting NGSI-LD data associated with the telemetry statistics extracted from network interfaces or network flow data records considering different criteria. For instance, the timestamp of observation of the resulting NGSI-LD entities (i.e., the *observedAt* metadata) could be used to implement an expiration mechanism. However, the Temporal API of NGSI-LD does not support deletion of specific temporal snapshots or instances of an NGSI-LD entity. This would be particularly useful, for example, in the case of materializing the configuration and operational status data from network interfaces, where the data of the resulting NGSI-LD entities are updated over time. Therefore, if the use of deletion operations were considered in the NGSI-LD Temporal API, an expiration control could be included for certain temporal conditions of these data.

5.3 Performance of the Descriptive NDT Prototype for Completing Network Operations

This section describes the performance experimentation conducted on the Descriptive NDT prototype solution based on the hybrid data integration approach presented in this work, comparing the results of the data materialization approach with the data virtualization approach. The experiments examine the query processing and data integration phases to evaluate the performance in terms of latency for both data integration approaches to complete the automation of network operations and management tasks. The experiments were carried out using the NETCONF and gNMI network management protocols and the NetFlow monitoring protocol.

The first experiment measures the latency of translating the operational status of the network interfaces from YANG to NGSI-LD, according to the query operations performed by NETCONF and gNMI. Total latency was measured from the time a data consumer originally requested a query entity operation to retrieve *InterfaceState* entities using the NGSI-LD API until the corresponding response with the result of the related GET RPC operation performed on the network device arrived back to the consumer. This measurement takes into account both the query processing and data integration phases. Additionally, latency was measured from the time the NETCONF and gNMI clients received the query responses until the data consumer received the responses in the NGSI-LD data format. This measurement only considers the delay in the data integration phase.

The second experiment measures the latency involved in translating from YANG modeled data to NGSI-LD modeled data for telemetry statistics extracted from network interfaces. These statistics are based on notifications received via the NETCONF and gNMI management protocols when a subscription operation is created for an entity of type *InterfaceStatistics* via the NGSI-LD API. In such a case, the latency is measured from when the NETCONF and gNMI clients collect telemetry notifications until the corresponding NetOps consumer receives the notifications in the NGSI-LD data format.

These two experiments were conducted using NETCONF and gNMI for both the data materialization and data virtualization approaches. The experiments related to NETCONF are described in Subsection 5.3.1, and the experiments related to gNMI are described in Subsection 5.3.2.

In the first experiment, two alternatives were analyzed to measure the total delay in performing network request operations in the case of data virtualization. These alternatives enable the Data Consumer to communicate with the Descriptive NDT system via the NGSI-LD API in two different ways: i) direct communication with the Data Integration Pipeline Plane of the Descriptive NDT system, which works as an NGSI-LD Context Source; and ii) communication with the NGSI-LD Context Broker, which works as an intermediary between the Data Consumer and the Data Integration Pipeline Plane. Figure 5.10 shows the procedure to measure end-to-end latency to complete network request operations by interacting with the NGSI-LD Context Broker using either a data materialization approach or a virtualization approach. In the case of data materialization, due to implementation details, a Kafka message

bus is not used between the XML/JSON Data Parsers and Mappers components and the NGSI-LD Instantiator components because they are part of the same service. Conversely, a Kafka message bus is used in the case of data virtualization because these components are decoupled into different services.

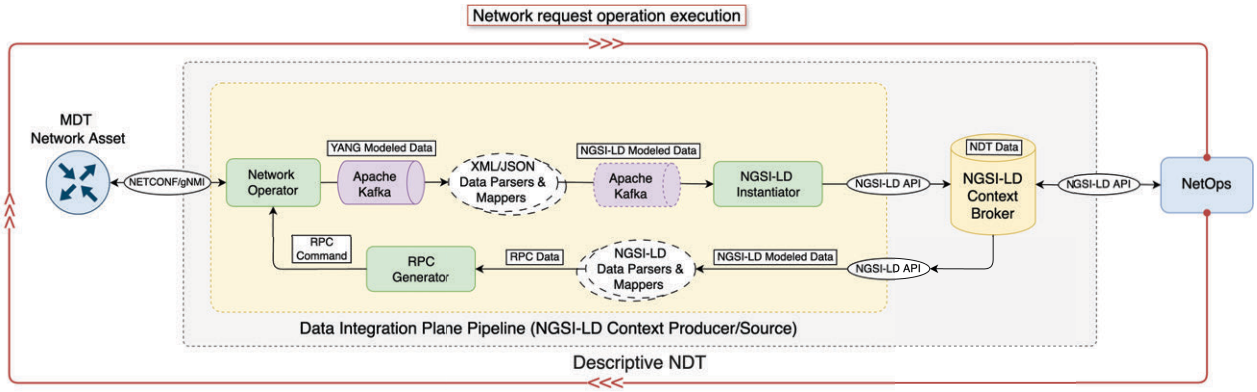


Figure 5.10: Measuring the latency for network request operations when interacting with the NGSI-LD Context Broker.

Meanwhile, Figure 5.11 shows the procedure for measuring end-to-end latency to complete network request operations, but by interacting with the NGSI-LD Context Source and using the data virtualization approach.

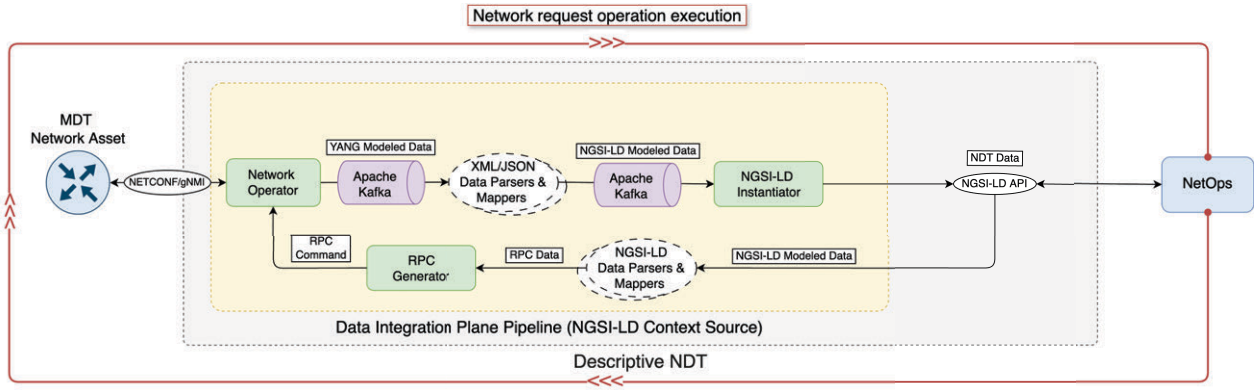


Figure 5.11: Measuring the latency for network request operations when interacting with the NGSI-LD Context Source.

For measuring the delay in receiving the notifications for network subscription operations realized in the second experiment, in the case of data virtualization, there is direct communication between the Data Consumer and the Context Source, since there is currently no Context Broker implementation that allows subscription-type operations in NGSI-LD distributed architectures. Figure 5.12 shows the procedure to measure the latency performance of network data reception using the data virtualization approach.

Meanwhile, Figure 5.13 shows the procedure for measuring the latency performance of network data reception using the data materialization approach. Both procedures have also been considered to measure the delay in receiving responses for the network request operations

realized in the first experiment. In this case, the virtualization approach for retrieving data from queries can follow the procedure depicted in either Figure 5.12, which sends the resulting data directly to the NetOps consumer, or Figure 5.13, which redirects the data through the NGSI-LD Context Broker.

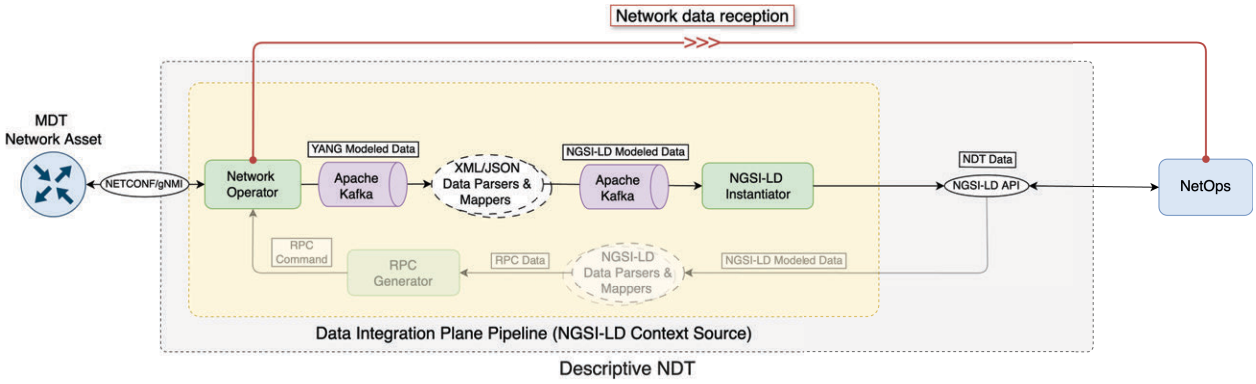


Figure 5.12: Measuring the latency of network data reception when interacting with the NGSI-LD Context Source.

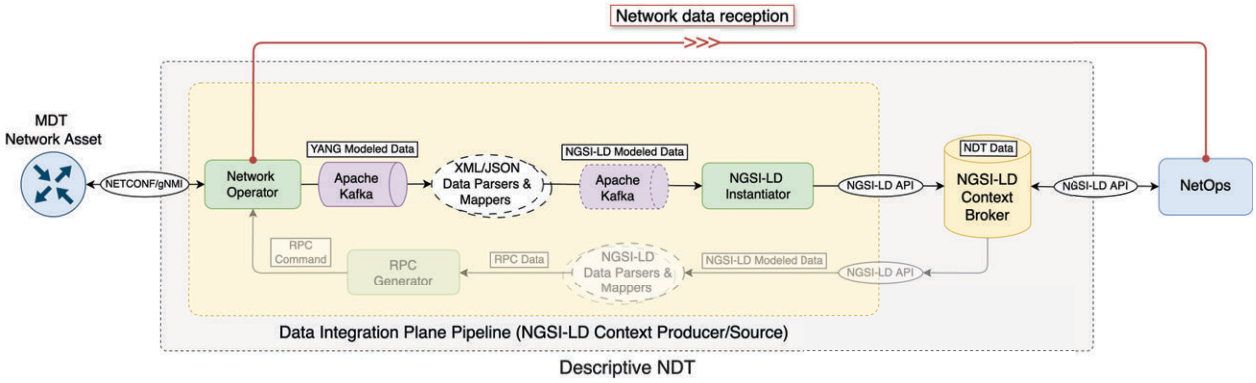


Figure 5.13: Measuring the latency of network data reception when interacting with the NGSI-LD Context Broker.

Finally, an additional experiment is described in Subsection 5.3.3 to measure latency performance when integrating NetFlow monitoring data. The NetFlow monitoring data are based on notifications received from a NetFlow collector for network flow data records when a subscription operation is created for an entity of type *ExportPacketFlowDataRecord* via the NGSI-LD API. Then, latency is measured from the time the NetFlow collector retrieves network flows until the corresponding NetOps consumer receives the notifications in the NGSI-LD data format. This experiment again compares the performance of the data materialization and data virtualization approaches.

The testbed that has been used for experimentation consists of a virtual network validation scenario with routers that support NETCONF, gNMI, and NetFlow v9. Therefore, as in the experimental and functional environment described in Section 5.1, the *Cisco IOS XE CSR1000V* and *Cisco IOS XRv9K* router models have been tested. In addition, Scorpio was selected as the implementation of the NGSI-LD Context Broker because it partially supports

the distribution operations of the NGSI-LD API and the registration of NGSI-LD Context Sources. Unlike the experiments in Section 5.2, these experiments used real data rather than synthetic data. The goal was to add an actuation component to network devices to trigger RPC operations or trigger the monitoring of network flows, and then measure the associated performance. Consequently, the amount and scale of data evaluated in the performance measurements is smaller. Nevertheless, it allows for a comparative analysis of the performance of the solution.

In each type of experiment for the NETCONF and gNMI case, the average delay time of a total of 1,000 events, either telemetry notifications or query responses, was measured. In addition, each experiment is repeated three times, and the average of the measurements is calculated. For subscription operations, we considered receiving telemetry notification for a single network interface of the target network device. Meanwhile, for query operation, we considered receiving responses with operational status from all network interfaces of the target network device. The *Cisco IOS XE CSR1000V* router, which supports the NETCONF protocol and is used in the network virtualization scenario, has three network interfaces: one management interface and two LAN interfaces. On the other hand, the *Cisco IOS XRv9K* router, which supports the gNMI protocol and is also used in the network virtualization scenario, has four network interfaces: one management interface, two LAN interfaces, and one black hole interface for traffic discarding (i.e., *Null0* interface). The time elapsed between the occurrence of notifications or responses to queries is considered to be 5 seconds.

In the NetFlow v9 experimental case, the average delay time of a total of 5,000 events was measured. This experiment is also repeated three times and the average of the measurements is calculated. The time elapsed between notification event occurrences is sporadic while network flows are being monitored. For the experiment, a virtualized scenario was used in which a router *Cisco IOS XE CSR1000V* works as a NetFlow exporter. To produce NetFlow monitoring data, traffic related to point-to-point connectivity tests between end devices was generated. Considering this work environment, 5,000 network flow events are received in about 10 minutes during each repetition of the experiment.

As in Section 5.2, boxplot charts are used to represent the experimental results. In this case, the charts include the mean and standard deviation of the distribution of values represented in each boxplot chart.

5.3.1 Latency performance results for NETCONF operations

In this subsection, we analyze the latency associated with performing query operations to retrieve the operational status of network interfaces, as well as the latency associated with network interface telemetry notifications using NETCONF as the network management interface between the Descriptive NDT and the managed network scenario.

Figure 5.14 depicts the latency results of the first experiment, representing the average processing latency on each data integration approach for completing the NETCONF RPC query operations, considering both query processing and data integration phases. It is notable how this latency time is longer in data virtualization when the operation is forwarded by means of the Context Broker due to the additional number of hops and message exchanges

in the distributed data virtualization architecture (i.e., *virtualization_CB* in Figure 5.14). However, in a data virtualization case where the Data Consumer interacts directly with the Context Source (i.e., *virtualization_CS* in Figure 5.14), the latency results are lower because the number of communication interactions to perform the query operation in this case is fewer and therefore less time consuming.

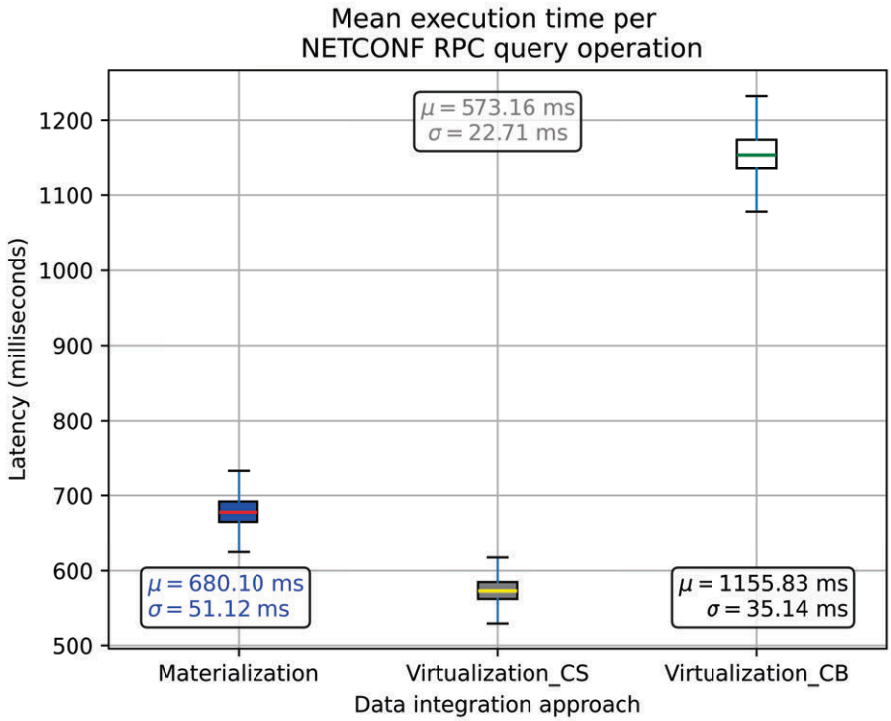


Figure 5.14: Mean execution time for completing NETCONF RPC query operations.

Figure 5.15 depicts the latency results for each data integration approach but only for integrating the resulting response data from NETCONF RPC queries without considering the query processing phase. In this case, we can clearly observe how the latency results for both cases of the data virtualization approach are lower than in the data materialization approach. In particular, *Virtualization_CS* is the best. This improvement in virtualization is the result of avoiding data replication in the data instantiation phase versus materialization, where the Context Broker ensures that the resulting data are stored in its back-end.

Figure 5.16 depicts the latency results of the second experiment, again representing the latency elapsed in each data integration approach but for notifications produced by a NETCONF YANG-Push RPC subscription operation. It can be seen that the average latency in the data virtualization approach is lower than that obtained with the data materialization approach, providing better performance results in terms of latency. As in the analysis of the experiments for responses received from RPC query operations, the improvement in integrating data for subscription notifications in data virtualization is the result of avoiding data replication in the data instantiation phase.

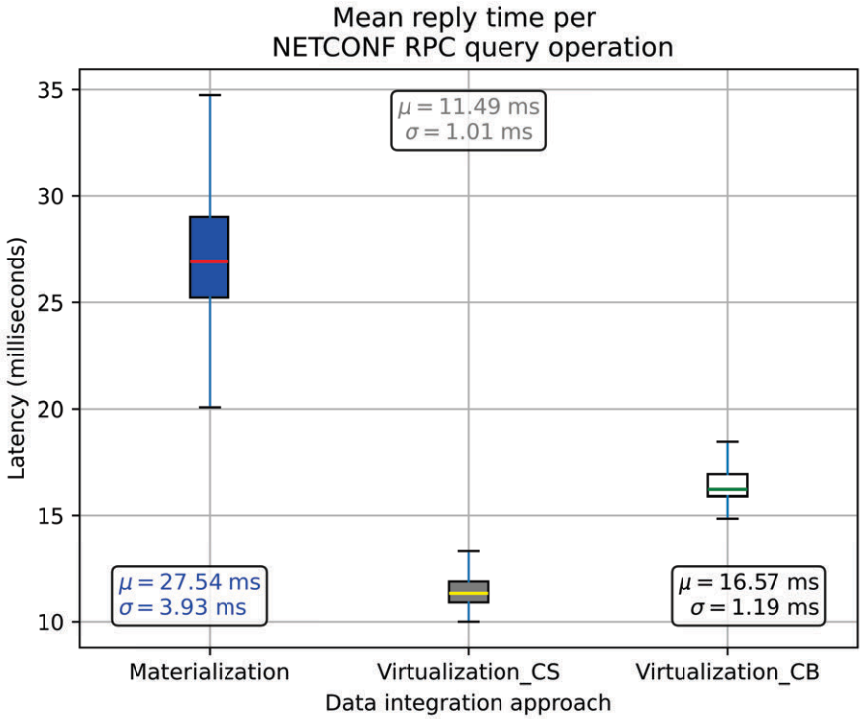


Figure 5.15: Mean execution time for replies generated from NETCONF RPC query operations.

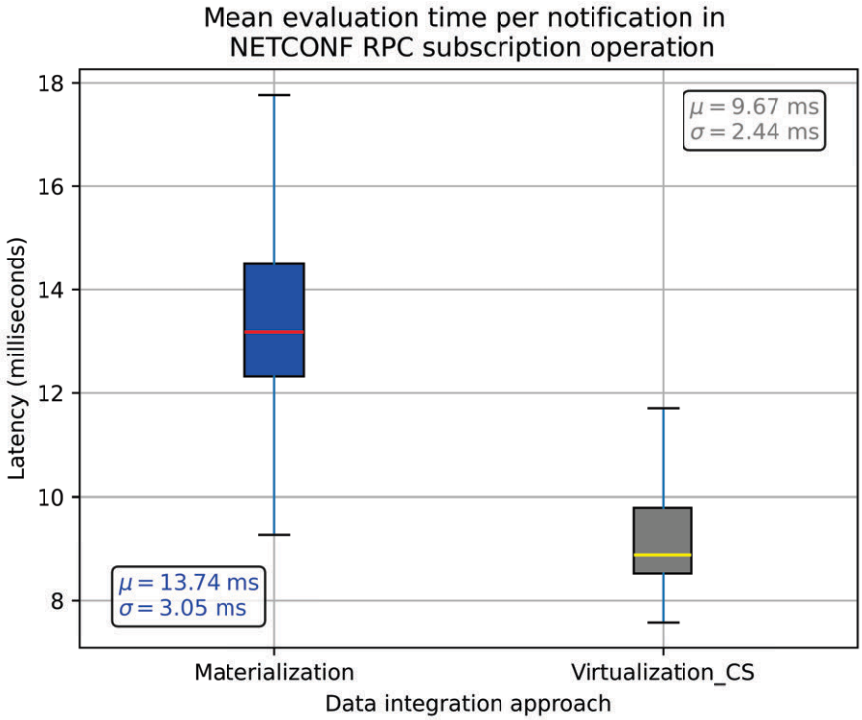


Figure 5.16: Mean evaluation time for notifications generated from NETCONF RPC subscribe operations.

5.3.2 Latency performance results for gNMI operations

In this subsection, we analyze the latency associated with performing query operations to retrieve the operational status of network interfaces, as well as the latency associated with network interface telemetry notifications using gNMI as the network management interface between the Descriptive NDT and the managed network scenario.

Figure 5.17 shows the latency results of the first experiment, representing the average processing latency of each data integration approach to complete the gNMI RPC query operations, considering both query processing and data integration phases. As in the experiments with NETCONF, it is notable how this latency time is longer in data virtualization when the operation is forwarded by the Context Broker due to the additional number of hops and message exchanges in the distributed data virtualization architecture (i.e., *Virtualization_CB* in Figure 5.17). However, in a data virtualization case where the Data Consumer interacts directly with the Context Source (i.e., *Virtualization_CS* in Figure 5.17), the latency results are lower because the number of communication interactions to perform the query operation in this case is fewer and therefore less time consuming.

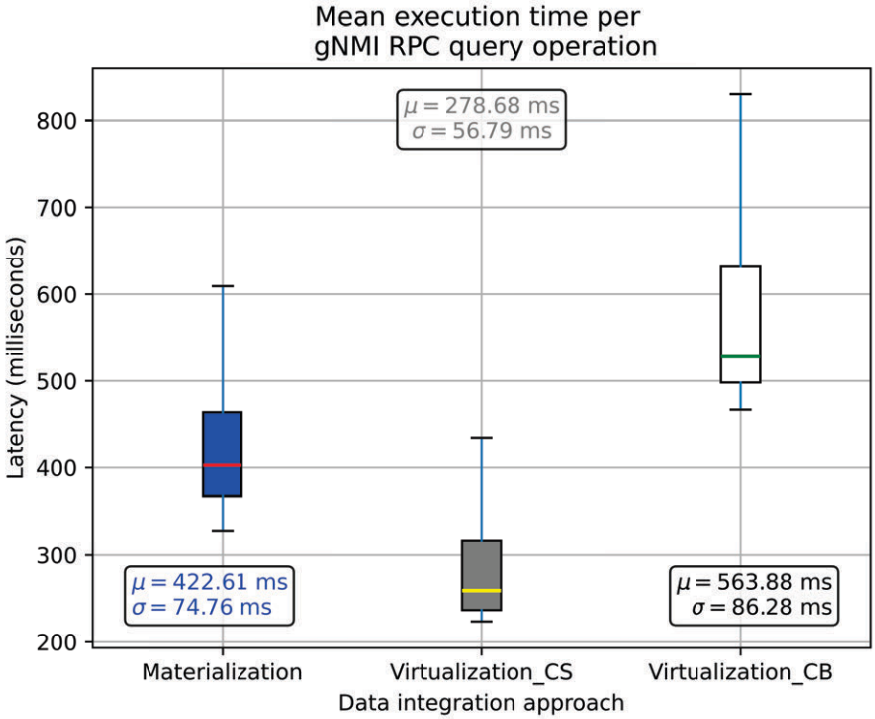


Figure 5.17: Mean execution time for completing gNMI RPC query operations.

Figure 5.18 shows the latency results for each data integration approach but only for integrating the resulting response data from gNMI RPC queries without considering the query processing phase. In this case, we can clearly observe how the latency results for both cases of the data virtualization approach are lower than in the data materialization approach. In particular, *Virtualization_CS* is the best.

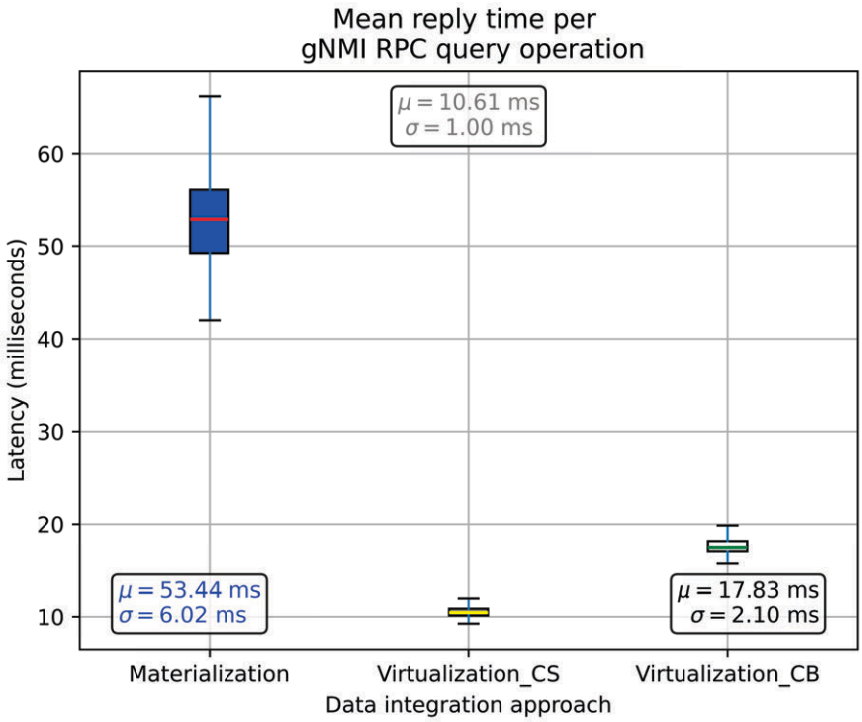


Figure 5.18: Mean execution time for replies generated from gNMI RPC query operations.

By comparing the latency results for executing and receiving queries in NETCONF and gNMI, we can see first that query execution is significantly slower in NETCONF. This may be due to the higher performance in the execution of GET RPC operations by the gNMI client compared to the NETCONF client. However, the dispersion of values in the gNMI results is greater. On the other hand, the latency results for query reception show better performance in NETCONF for data materialization. However, this is primarily due to the fact that NETCONF obtains operational status data from three network interfaces, whereas gNMI obtains it from four. Given the difference in the number of integrated interfaces, data virtualization performs better in gNMI than in NETCONF. This suggests that the data integration pipeline of the NDT solution performs better when managing responses from RPC query operations performed via gNMI.

Figure 5.19 depicts the latency results of the second experiment, again representing the latency elapsed in each data integration approach for notifications produced by a gNMI RPC subscription operation. It can be seen that the average latency in the data virtualization approach is lower than that obtained with the data materialization approach, providing better performance results in terms of latency.

When we compare the experimental results for the latency associated with receiving subscription notifications in NETCONF (i.e., Figure 5.16) and gNMI (i.e., Figure 5.19), we observe better performance in NETCONF for data materialization. However, the results for data virtualization are quite similar for NETCONF and gNMI, though gNMI has a lower mean and lower dispersion of values.

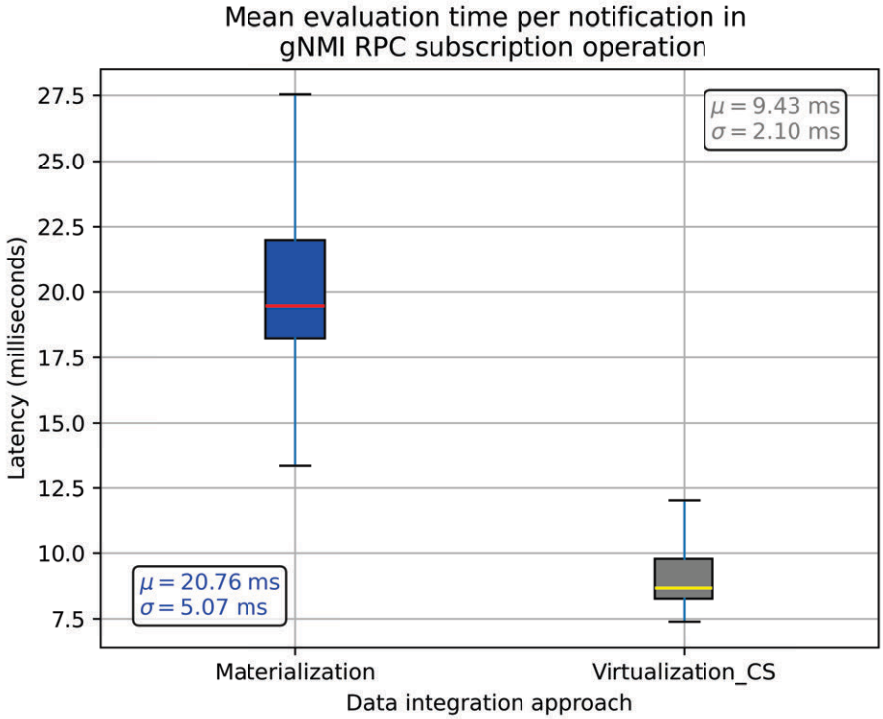


Figure 5.19: Mean evaluation time for notifications generated from gNMI RPC subscribe operations.

5.3.3 Latency performance results for NetFlow monitoring

In this subsection, we analyze the latency for notifications received about network flows using NetFlow v9 as the monitoring protocol between the Descriptive NDT and the managed network scenario.

Figure 5.20 shows the particular procedure for measuring the latency performance of the reception of network flows by interacting with the NGSI-LD Context Broker using a data materialization approach. The NetFlow data collector used enables the direct publication of network flows into a Kafka message bus, which initiates the data translation and instantiation processes to integrate the relevant NDT data within the NGSI-LD Context Broker. Then, the NetOps consumer can subscribe to the resulting NGSI-LD data by interacting with the NGSI-LD Context Broker in order to receive notifications about network flows accordingly.

Meanwhile, Figure 5.21 shows the procedure for measuring the latency performance of the reception of network flows by interacting with the NGSI-LD Context Source using a data virtualization approach. In this case, as in the data materialization approach, the NetFlow collector first collects the network flows and publishes them in Kafka. Then, the NetOps consumer can subscribe to the corresponding NGSI-LD data, which triggers the data translation and instantiation processes. This integrates the NetFlow monitoring data into the Descriptive NDT. Subsequently, the NetOps consumer receives notifications about network flows from the Descriptive NDT.

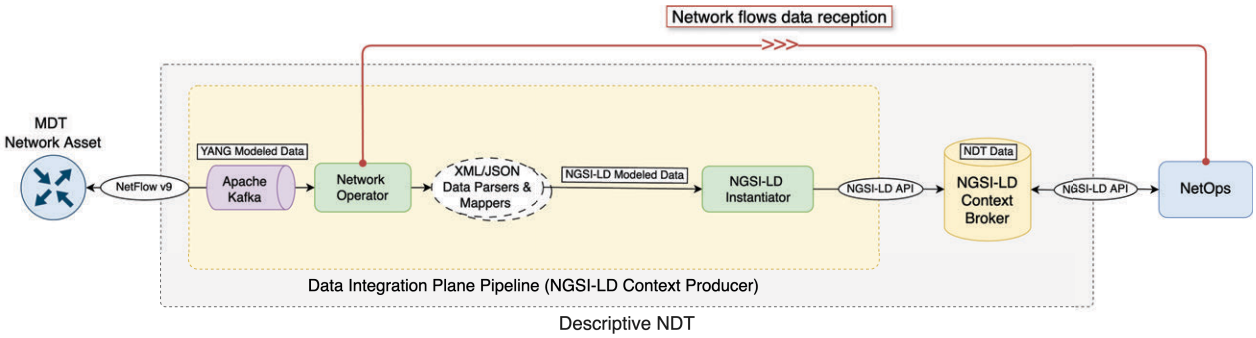


Figure 5.20: Measuring the latency for the reception of network flow data when interacting with the NGSI-LD Context Broker.

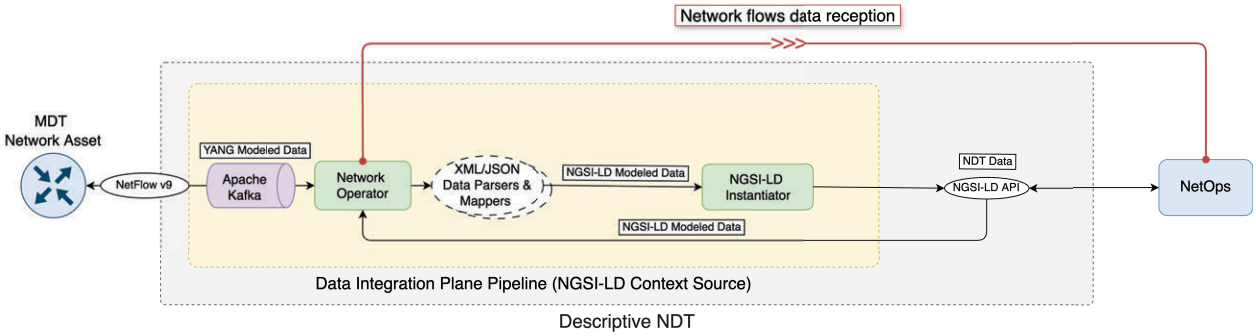


Figure 5.21: Measuring the latency for the reception of network flow data when interacting with the NGSI-LD Context Source.

Figure 5.22 depicts the latency results for NetFlow monitoring, representing the latency elapsed in each data integration approach for notifications received about network flows. As can be seen, the average latency of the data virtualization approach is lower than that obtained by the data materialization approach, and the standard deviation is lower as well.

These results show that the data virtualization approach performs better in terms of latency. It needs to be considered that the measured latency applies to each notification of a single network flow data record. Because a single export packet monitored by NetFlow can encompass many network flows, processing all the flows in an export packet takes multiple times longer than processing a single network flow.

The number of monitored network flows can be highly variable depending on the amount of network traffic at any given moment. Consequently, the performance of the data integration process is further limited by the maximum rate at which network flows can be published in Kafka after being collected by the NetFlow collector.

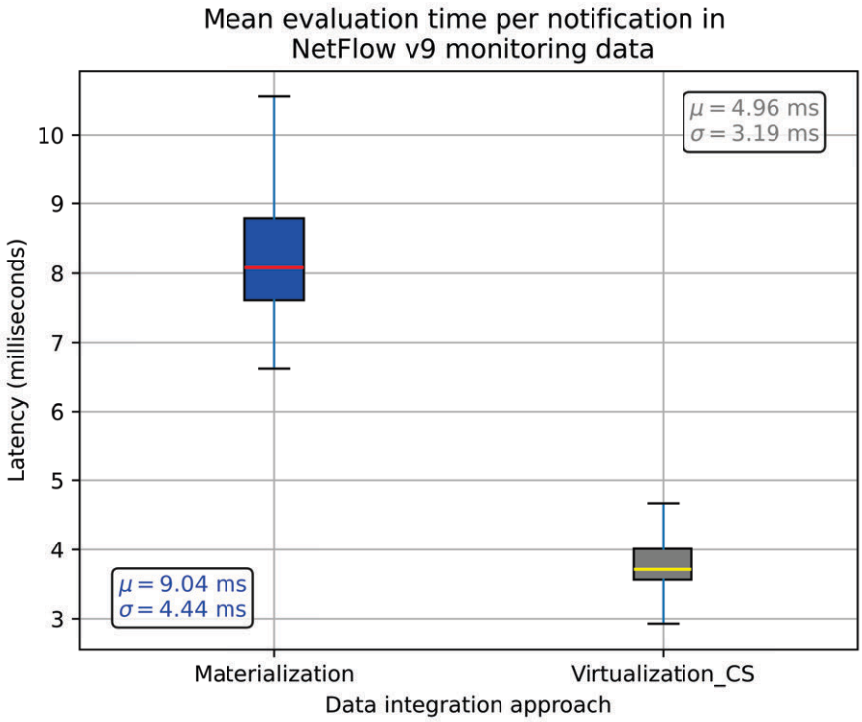


Figure 5.22: Mean evaluation time for integrating notifications about NetFlow monitoring data.

5.3.4 Discussion on experimental results

In this second experimental phase, we have successfully evaluated the performance of the hybrid data integration mechanism implemented by the NDT solution for querying network operational data and subscribing to network telemetry notifications. In general, we can see that the NDT solution based on a virtualized data integration mechanism performs better in terms of latency than the NDT solution based on a data materialization approach.

However, in the case of the NETCONF and gNMI experiments, if in addition to the data integration, we consider the process of rewriting the query executed with the NGSI-LD API to trigger the necessary RPC operation, the data virtualization approach can be more time consuming when the consumer needs to continuously interact directly with the Context Broker. Operation activation is not a consideration in RPC subscription operations, in which the operation is initially activated, and continuous notifications are then received. However, operation activation is important for RPC query operations, which must always be restarted from scratch. In these cases, we observe a significant latency difference when executing RPC query operations versus processing only RPC query responses. This discrepancy is evident in both the NETCONF and gNMI protocols. It suggests that executing the query consumes most of the processing time. Although a significant portion of this time is attributed to operations performed with NETCONF and gNMI protocol clients, further analysis would be needed to determine the processing time involved in rewriting NGSI-LD API operations as RPC operations.

When performing end-to-end performance tests considering the actuation on the real network to trigger network operations and management processes, no overhead tests were performed to measure the throughput of virtualized data. The reason is that the implemented prototype only allows virtualization of data resulting from the asynchronous activation of network operations and management processes via the NGSI-LD API. Currently, the prototype does not support synchronous load testing with synthetic data, which would allow us to conduct throughput tests similar to those analyzed in Section 5.2.

While the main focus of this dissertation has been to adapt the NGSI-LD API for network automation mechanisms based on the use of model-driven network management protocols, the adaptation for monitoring network flows using the NetFlow v9 protocol has also been validated. Experiments with NetFlow demonstrate that the performance of the NDT solution may be impacted by a high rate of flows collected from the network. These experiments served as preliminary results for the integration of data from other monitoring protocols into the NDT solution, but further analysis is required. Developing network automation solutions that are also compatible with other model-driven network management protocols (e.g., RESTCONF) and other traditional network monitoring protocols (e.g., IPFIX and SNMP) would allow us to better determine the performance, usefulness, and adaptability of this new model-driven network management approach based on the NGSI-LD standard for NDT applications.

Chapter 6

Conclusions and Future Work

This chapter presents the conclusions and future work derived from the research presented in this Doctoral Thesis.

Table of Contents

6.1 Conclusions 162
6.2 Future Work 165

6.1 Conclusions

This work introduces a new knowledge-based data integration system for modeling Network Digital Twin (NDT) solutions. Relying on automated mechanisms, standard interfaces, and standard data models allows network management to benefit from a comprehensive NDT platform for network data integration. An NDT platform facilitates uniform access to network management data while abstracting from the heterogeneity and complexity of the underlying real-world networks.

The Descriptive NDT solution proposed in this work is a model-driven network management system that can mirror the operational status and configuration of real networks using both a data materialization approach and a data virtualization approach. In addition, the proposed Descriptive NDT solution supports actuation mechanisms for network operations and management purposes, enabling the configuration of the real network and obtaining configuration, operational, and telemetry data on demand by using data integration approaches.

Our experimental prototype allows to expose the capabilities of real networks, but also to manipulate them by defining representations of monitoring and configuration data in a formal and interoperable model based on the NGSI-LD standard, demonstrating how an NDT can solve an important challenge in telecommunications network management to properly mirror and handle network behavior.

The resolution of this challenge motivated the completion of this Doctoral Thesis and served as the starting point for answering the three main questions raised in the introduction of this report, which constitute the core of the research:

- *What should be the design principles of a Digital Twin architecture that facilitate its application in managing, operating, and automating telecommunications networks?*

  To answer this question, it is necessary to know the architectural requirements defined for the construction of Digital Twins in the field of telecommunications network management. In this context, Section 2.2 of Chapter 2 reviewed the state of the art of the Network Digital Twin paradigm, analyzing the efforts and activities of the different standardization organizations for the conceptual specification of the term and its reference architecture. Specifically, the IETF organization has attempted to conceptualize the core elements and a reference architecture for the definition of an NDT in the network management domain, including the enabling technologies, use cases, functional requirements and challenges to be addressed.

  To accomplish the design principles of an NDT paradigm, Chapter 3 proposes guidelines and a methodology to build an NDT system for network operations and management purposes. The novelty of this work is threefold: 1) a new Digital Twin solution based on a data integration approach for mirroring and handling accordingly the real network behavior; 2) an original proposal to address a Network Digital Twin solution focused on network management domain using the NGSI-LD standard; and 3) a new reference implementation for model-driven network management based on the NGSI-LD standard.

The solution proposed in this work, in addition to being aligned with the core elements and reference architecture proposed for NDT according to the IETF organization, also adopts the Descriptive Twin vision proposed by the ETSI Industry Specification Group on Context Information Management (ETSI ISG CIM), focusing on the network management domain. The functional implementation of the NDT solution has been described in Chapter 4.

- *How can the harmonization of heterogeneous network data sources be guaranteed in the design of Digital Twins covering the formalization and use of standardized data representations and communication interfaces for integrating real network data?*

  This question arises from the need to integrate data from real heterogeneous data sources in the network and to homogenize the representation of their information, which is one of the major barriers to implementing a Digital Twin solution. This challenge is even greater in the field of network management, where there is a multitude of monitoring mechanisms and protocols that implement different interfaces and specify different data models and formats. Among the core components of an NDT highlighted by the IETF, as described in Subsection 2.2.3 of Chapter 2, the search for formal models that represent with high fidelity the information extracted from reality stands out. In addition, there is a need to find standard interfaces or APIs that allow bidirectional communication between the NDT instance and the real network (i.e., NDT southbound interface), as well as between external services or applications and the NDT instance itself (i.e., NDT northbound interface). These APIs will allow external systems to maintain the lifecycle control of the NDT, either to collect operational, configuration, or telemetry data from the network, or to control actions on the network for reconfiguration or decision-making, which is a key enabler for achieving network automation. In order to achieve both the formal, high-fidelity modeling of the data and the dual north-south bidirectional NDT interfaces, the NDT solution presented in this dissertation proposes the use of the NGSI-LD standard applied over model-driven network management protocols. To this end, the NGSI-LD standard is used to define domain-specific network management models for data originally modeled in YANG. In turn, the NGSI-LD API is used as the NDT data management interface and as the northbound interface for the interaction between the NDT and external systems, abstracting from the southbound interface managed by model-based network management protocols such as NETCONF or gNMI.

  The NDT solution focuses on network devices that implement model-driven network management protocols based on the use of standard YANG data models, which, as reviewed in Section 2.3 of Chapter 2 of this document, is one of the current trends in network management and automation. To accomplish the integration of real network data, the NDT solution implements a data mapping mechanism to translate from YANG-modeled data to data modeled according to the NGSI-LD standard, which is formalized according to translation conventions, as described in detail in Subsection 3.2.2 of Chapter 3. In addition, as reflected in Subsection 3.2.1 of Chapter 3, the NDT solution implements a hybrid data integration mechanism that supports both a data materialization approach and a data virtualization approach for integrating real network data into the NDT by leveraging the capabilities offered by the NGSI-LD standard.

One of the main advantages of the NDT solution proposed in this research is its ability to implement a programmable and automatic digital mapping system that converts YANG-modeled data to NGSI-LD. This minimizes manual intervention and improves interoperability between the two corresponding data formats. To complete this translation process, our programmatic approach enables the automatic generation of data parsers to process YANG-compliant data, ensuring seamless integration. Another key benefit of the solution is the automatic generation of OpenAPI Schemas that are fully compliant with the NGSI-LD OpenAPI Specification, providing a solid knowledge foundation for adapting YANG models to NGSI-LD meta-models. The proposed solution generally delivers good performance, making it a viable approach for various applications. Furthermore, the programmatic digital mapping solution is based on already standardized translation conventions, leveraging Python as the core programming language. The approach of this NDT solution can be seen as a "Bottom-Up Programming-based Methodology", which is a data integration methodology based directly on the format and schema that follow the data exposed by the network data source, where the knowledge representation is particularized and implicit.

However, the approach also presents some drawbacks. One significant challenge is the difficulty of determining or inferring the identifier of entities in the resulting NGSI-LD model, which can lead to inconsistencies. Performance can also be affected when the generated data parser needs to analyze data from multiple YANG models, particularly in cases involving highly extensible YANG models. Another major issue is that similar or identical concepts across different models may collide or overlap, complicating data integration. This problem might be addressed by using ontologies, although there are cases where YANG modules modeling the same or similar aspects contain a considerable amount of their own information, which makes generalizing the ontologies difficult. Finally, the programmatic methodology of the proposed solution presents a semantic limitation, since it lacks an additional layer with an intrinsic mechanism to expose the knowledge baseline of the NDT data and represent it in a semantic way, which hinders the effective generalization of the NDT solution to the network operations and management domain. This could be solved by applying a "Top-Down Ontology-based Methodology", which would be a data integration methodology based on knowledge acquired from data sources in a given domain, where the knowledge representation is generalized and explicit by adding a semantic layer using ontologies. However, a semantic data integration approach based on ontologies consists of manual procedures, especially in data mapping and data modeling processes, such as semantic annotation and semantic interoperability, which hinder the automation of the solution [40].

Therefore, an ideal solution would be to see how the "Bottom-Up Programming-based Methodology" followed in the NDT solution proposed in this Doctoral Thesis can be combined with the aforementioned "Top-Down Ontology-based Methodology" to take advantage of both the automation processes and the semantic annotations of the data represented in the NDT. This is one of the main areas of work that will be explored in the future.

- *What could be the interoperable mechanism of the Digital Twin solution that would allow external systems to operate on the real network and have control over the status and configuration of the network itself?*

  To answer this question, as determined in the state of the art on Digital Twins in Chapter 2, one of the key features of the Digital Twin is the automation of data flows between the associated physical asset and the Digital Twin instance in both directions.

  In this way, the Digital Twin can both collect data and act as a control instance for the physical asset. Applying this vision of bidirectional or two-way data flow to the NDT paradigm, the NDT instance serves as an interoperable platform for external network management systems to collect configuration, operational and telemetry data from the real network, as well as to apply new changes or actions in the real network. These features make the NDT a key enabler of network automation through the closed-loop paradigm.

  For the sake of achieving this network automation functionality in the NDT solution proposed in this Doctoral Thesis, this document describes in Subsection 3.2.4 of Chapter 3 a methodology for using the NGSI-LD standard and its API to automate network operations and management processes. This methodology proposes a solution to orchestrate the NGSI-LD standard to perform network operations and management tasks specific to model-driven network management protocols and for each data integration approach supported by the NDT solution. In this regard, Subsection 4.1.3 of Chapter 4 describes the implementation details of the NDT prototype solution for processing NGSI-LD API operations and rewriting them as operations on the real network. Although the methodology can be applied to any model-driven network management protocol, it has been implemented and validated for the NETCONF and gNMI protocols, which serve as examples of application. The methodology has also been adapted to integrate network flows monitored using the NetFlow v9 protocol. In addition, at the end of Chapter 4, several application use cases are presented to demonstrate the usability of the NDT prototype solution for efficient network operations and management purposes. This NDT prototype could be adapted accordingly to run on top of different network infrastructures once the data of interest modeled in YANG is identified, thus allowing it to be the basis for implementing a Network Digital Twin as a Service (NDTaaS) concept.

6.2 Future Work

During this Doctoral Thesis, different challenges have been identified, which can become the starting point for future lines of work and research. These include analyzing how to efficiently handle the request overhead to integrate data from large-scale networks into the NDT solution, which addresses the requirements for massive data storage and high availability, as well as scalability and performance improvements. As suggested in Chapter 5, a complementary mechanism can be sought to control the expiration of data storage in the NDT data integration solution. Traditional network monitoring solutions, such as MRTG [203], Cacti [204], and Nagios [205], have long used RRDtool [206] to efficiently store and visualize time-series

data, leveraging its temporal data management mechanisms. RRDtool uses a round-robin database structure that enables long-term monitoring without requiring unlimited storage. This approach optimizes resource usage and provides a strategy for controlling data retention and expiration, ensuring the efficient handling of historical data. These benefits underscore the importance of incorporating similar mechanisms into modern, data-intensive network monitoring systems that require scalability and long-term efficiency. One such system is the network management mechanism leveraged by the NDT solution proposed in this dissertation, based on the NGSI-LD standard. The NGSI-LD API implements deletion operations on managed data entities, allowing the scheduling of periodic or on-demand deletion of information. However, the Temporal API of NGSI-LD does not support deletion operation for specific temporal instances of an NGSI-LD entity. Therefore, if deletion operations were considered for use in the NGSI-LD Temporal API, an expiration control could be included for certain temporal conditions of these data (e.g., retain only the last week's data instances of the NGSI-LD entity). In this sense, it would be appropriate to point out to the ETSI ISG CIM committee that these deletion operations should be considered in the Temporal API of the NGSI-LD standard.

Another area of interest would be to apply and analyze the proposed digital mapping process methodology to translate real network data into its NDT representation taking into account other network management protocols and the diversity of monitoring data. Network management protocols based on the YANG model, such as NETCONF and gNMI, were the basis for the solution provided for NDT. Other network management protocols based on YANG data modeling such as RESTCONF or CORECONF [207] could be analyzed and adapted in the future within the NDT proposal. On the one hand, RESTCONF is an HTTP-based protocol widely used to manage network devices through operations that follow the REST principles. Consequently, it is well suited to operations performed with the NGSI-LD API, which is also based on the HTTP and REST principles. Furthermore, RESTCONF supports both the XML and JSON data formats that are already used in the data parsing and mapping solution for integrating data into the NDT. On the other hand, CORECONF is a network management interface for the Constrained Application Protocol (CoAP) [208], which extends the set of YANG-based protocols to include the capability for managing constrained devices and networks, where efficiency in resource management is crucial. Like RESTCONF, CORECONF is based on the HTTP and REST principles, but uses CBOR (Concise Binary Object Representation) [209], a binary encoding format that transmits data more efficiently. This makes CORECONF ideal for managing environments with limitations in terms of data transfer and resources, such as devices used in IoT or Machine-to-Machine (M2M) networks. CORECONF has already been identified as a promising solution for managing networks in pursuit of Time-Sensitive Networking (TSN) [210]. TSN is a set of technologies and standards, chosen as one of the main pillars of Industry 4.0 for enabling highly reliable, deterministic, and low-latency communications over standard Ethernet networks. Efficient management of resource-constrained devices is crucial in these networks, and CORECONF could offer significant benefits in time, power, and memory [211], [212]. Due to the widespread use of these protocols in recent trends related to network operations and management, adapting them to the proposed NDT solution may be highly relevant.

Additionally, a proof of concept was performed to adapt other traditional network monitoring protocols, such as NetFlow, into the NDT solution. In this case, the option of defining a YANG model compatible with the protocol was raised as an additional pre-adaptation step before applying the data transformation methodology required to represent the data in the NDT solution. While this is possible, it adds an additional requirement and difficulty to the data mapping process, which can be costly in terms of additional delays and inefficiencies. YANG is a popular specification and data model in the telecommunications network industry that is promoted by international standards organizations. However, additional data mapping mechanisms must be developed for the NDT to be compatible with other data models used by traditional and alternative network monitoring and management solutions. Thus, a significant contribution would be to evolve the NDT solution to adopt those network monitoring and management protocols. For instance, the solution could adopt IPFIX, a standardized protocol for monitoring network flow statistics that provides more advanced and complete flow-level information than NetFlow v9. Alternatively, the solution could adopt traditional network management protocols, such as SNMP, which is still widely used by network administrators. However, these protocols have their own data models and encoding formats, which makes coupling them with the NDT solution a major challenge.

In addition, as mentioned in the conclusions, we are investigating how to align the data integration approach based on a "Bottom-Up Programming-based Methodology" with a "Top-Down Ontology-based Methodology" to correlate the network domain information at the conceptual level. Using an ontology-based data integration mechanism would facilitate the establishment of links between similar concepts considered in data models from different network data sources, either YANG data models or others, and prevent collisions or overlaps in the resulting NDT solution. For example, the ontology-based approach would facilitate automating the resolution of the *isSubClassOf* relationship type defined in the NDT models of the NDT solution, as described in Section 4.2. This approach would enable the establishment and inference of inheritance relationships between closely related concepts in NDT models through semantic annotations. Therefore, combining programming-based and ontology-based approaches would enable the development of an automatic, comprehensive Knowledge Graph solution for network operations and management.

We are also investigating improvements related to software, tools, or libraries that will be used in the implementation of the NDT prototype to improve its performance and efficiency in the future. Implementation improvements could be fourfold: 1) finding client libraries for the model-driven network management protocols that are more efficient in terms of computational cost, latency, and handling of asynchronous and parallelizable communication sessions (e.g., *scrapli_netconf* [213] library in Python is a NETCONF client that supports asyncronous communication); 2) using libraries for more efficient data encoding/decoding and validation in programming languages like Python (e.g., the *msgspec* [214] Python library promises to be 12 times faster than *pydantic* [177] library and is natively supported by OpenAPI stub code generator tools such as *datamodel-code-generator* [215]); 3) migrating the prototype implementation to a production distributed cluster environment based on a Kubernetes infrastructure for better scalability and performance results, evaluating NGSI-LD federated architectures and finding efficient streaming data management solutions, such as the Apache Kafka's native solution for distributed Kubernetes environments called *Strimzi* [216];

and 4) defining and deploying realistic and large-scale network virtualization scenarios in distributed Kubernetes environments where to further test the NDT proposal by leveraging network virtualization tools oriented to Kubernetes cluster infrastructures, such as Kubernetes Network Emulation (KNE) [217] from OpenConfig and Clabernetes [218] from Nokia[6.1]

Finally, the ultimate goal would be to address complex network automation tasks, such as automating the configuration and delivery of network services according to on-demand intents requested from networking customers (e.g., auto-configuring of Ethernet Virtual Private Networks or BGP routing services in Data Center networks), predicting future conditions and potential issues, or recommending optimization policies, by leveraging the capabilities offered by the NDT proposal and applying Machine Learning models or Artificial Intelligence techniques to infer certain outcomes and behaviors in the network.

To automate the delivery of network services, the NDT instance would act as a controller system capable of providing an intended service configuration by applying the Intent-Based Networking paradigm and autonomously controlling and managing the real network data plane as an SDN controller functionality, similar to the TeraFlowSDN[6.2] solution focused on the continuous delivery of agile and programmable NetOps processes. Although TeraFlowSDN offers a programmable SDN-based automation framework for NetOps functions, it lacks the intrinsic data modeling and management capabilities inherent to an NDT architecture. This data-driven layer is essential for achieving closed-loop automation and proactive decision-making, both of which are fundamental to Intent-Based Networking and advanced NDT implementations. The NDT proposal could also provide benefits for network operations and management intelligence solutions based on Artificial Intelligence systems in terms of automating the network data collection process required to infer network behavior and the actuation process to resolve network issues or provision new configurations. In this regard, depending on the required network application environment, our NDT proposal could offer different network automation services to enhance and support network operations and management processes, understanding each NDT instance as a particularized Network Digital Twin as a Service (NDTaaS).

In line with the utility expectations for the NDT solution proposed in this Doctoral Thesis, we are currently working on a use case for an NDT solution for smart energy-aware zero-touch traffic engineering within the European ACROSS project. This use case aims to apply an NDT solution alongside with Machine Learning techniques to transform network telemetry events into intelligent network orchestration decisions while considering energy consumption. In this context, employing the NDT system proposed in this dissertation would automate the process of collecting network data on demand and applying dynamic reconfigurations to make energy-aware routing decisions across the network.

[6.1] Clabernetes is the Containerlab version for Kubernetes.

[6.2] The official TeraFlowSDN website is available at the following link: https://www.teraflow-h2020.eu

References

[1] Grieves, Michael and Vickers, John, "Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems", in *Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches*, F.-J. Kahlen and A. Flumerfelt Shannon nd Alves, Eds., Cham: Springer International Publishing, 2017, pp. 85–113. DOI: 10.1007/978-3-319-38756-7_4.

[2] A. M. Madni, C. C. Madni, and S. D. Lucero, "Leveraging Digital Twin Technology in Model-Based Systems Engineering", *Systems*, vol. 7, no. 1, 2019. DOI: 10.3390/systems7010007.

[3] N. Crespi, A. T. Drobot, and R. Minerva, "The Digital Twin: What and Why?", in *The Digital Twin*, N. Crespi, A. T. Drobot, and R. Minerva, Eds. Cham: Springer International Publishing, 2023, pp. 3–20. DOI: 10.1007/978-3-031-21343-4_1.

[4] F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee, "Digital twin in industry: State-of-the-art", *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2405–2415, 2019. DOI: 10.1109/TII.2018.2873186.

[5] P. Almasan, M. Ferriol-Galmés, J. Paillisse, *et al.*, "Network Digital Twin: Context, Enabling Technologies, and Opportunities", *IEEE Communications Magazine*, vol. 60, no. 11, pp. 22–27, 2022. DOI: 10.1109/MCOM.001.2200012.

[6] Y. Wu, K. Zhang, and Y. Zhang, "Digital Twin Networks: A Survey", *IEEE Internet of Things Journal*, vol. 8, no. 18, pp. 13 789–13 804, 2021. DOI: 10.1109/JIOT.2021.3079510.

[7] C. Zhou, H. Yang, X. Duan, *et al.*, "Network Digital Twin: Concepts and Reference Architecture", Internet Engineering Task Force, Internet-Draft draft-irtf-nmrg-network-digital-twin-arch-10, Feb. 2025, Work in Progress, 36 pp. [Online]. Available: https://datatracker.ietf.org/doc/draft-irtf-nmrg-network-digital-twin-arch/10/ (visited on 09/05/2025).

[8] ITU-T, "Digital twin network — Requirements and architecture", Telecommunication Standardization Sector of ITU (ITU-T), Recommendation Y.3090, Feb. 2022.

[9] L. Hui, M. Wang, L. Zhang, L. Lu, and Y. Cui, "Digital Twin for Networking: A Data-Driven Performance Modeling Perspective", *IEEE Network*, vol. 37, no. 3, pp. 202–209, 2023. DOI: 10.1109/MNET.119.2200080.

[10] H. Shin, S. Oh, A. Isah, I. Aliyu, J. Park, and J. Kim, "Network Traffic Prediction Model in a Data-Driven Digital Twin Network Architecture", *Electronics*, vol. 12, no. 18, 2023. DOI: 10.3390/electronics12183957.

[11] ETSI, "Context Information Management (CIM); Feasibility of NGSI-LD for Digital Twins", ETSI, GR CIM 017 v1.1.1, Dec. 2022.

[12] ETSI, "Zero-Touch Service and Managment; Network Digital Twin", ETSI, GR CIM 015 v1.1.1, Feb. 2024.

[13] ETSI, "Zero-touch network and Service Management (ZSM); Network Digital Twin for enhanced zero-touch network and service management", ETSI, GS ZSM 018 v1.1.1, Dec. 2024.

[14] O. Havel, B. Claise, O. G. de Dios, A. Elhassany, and T. Graf, "Modeling the Digital Map based on RFC 8345: Sharing Experience and Perspectives", Internet Engineering Task Force, Internet-Draft draft-havel-nmop-digital-map-02, Oct. 2024, Work in Progress, 35 pp. [Online]. Available: https://datatracker.ietf.org/doc/draft-havel-nmop-digital-map/02/ (visited on 09/05/2025).

[15] O. Havel, B. Claise, O. G. de Dios, and T. Graf, "Digital Map: Concept, Requirements, and Use Cases", Internet Engineering Task Force, Internet-Draft draft-ietf-nmop-digital-map-concept-02, Oct. 2024, Work in Progress, 15 pp. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-nmop-digital-map-concept/02/ (visited on 09/05/2025).

[16] H. Hong, Q. Wu, F. Dong, *et al.*, "NetGraph: An Intelligent Operated Digital Twin Platform for Data Center Networks", in *Proceedings of the ACM SIGCOMM 2021 Workshop on Network-Application Integration*, ser. NAI'21, Virtual Event, USA: Association for Computing Machinery, 2021, pp. 26–32. DOI: 10.1145/3472727.3472802.

[17] D. Gelernter, *Mirror Worlds: Or the Day Software Puts the Universe in a Shoebox... How It Will Happen and What It Will Mean*. Oxford University Press, 1993.

[18] Stefan Sonvilla-Weiss, "Mirror Worlds: The Universe in a Box?", in *(IN)VISIBLE: Learning to Act in the Metaverse*. Vienna: Springer Vienna, 2009, pp. 36–72. DOI: 10.1007/978-3-211-78539-3_3.

[19] M. W. Grieves, "Product lifecycle management: the new paradigm for enterprises", *International Journal of Product Development*, vol. 2, no. 1-2, pp. 71–84, 2005.

[20] M. Grieves, "Digital Twin: Manufacturing Excellence Through Virtual Factory Replication", *White Paper*, vol. 1, pp. 1–7, 2014.

[21] A. Barbie and W. Hasselbring, "From Digital Twins to Digital Twin Prototypes: Concepts, Formalization, and Applications", *IEEE Access*, vol. 12, pp. 75 337–75 365, 2024. DOI: 10.1109/ACCESS.2024.3406510.

[22] M. Grieves and J. Vickers, "Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems", in *Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches*, F.-J. Kahlen, S. Flumerfelt, and A. Alves, Eds. Cham: Springer International Publishing, 2017, pp. 85–113. DOI: 10.1007/978-3-319-38756-7_4.

[23] J. Trauer, S. Schweigert-Recksiek, C. Engel, K. Spreitzer, and M. Zimmermann, "What is a Digital Twin? – Definitions and Insights from an Industrial Case Study in Technical Product Development", *Proceedings of the Design Society: DESIGN Conference*, vol. 1, pp. 757–766, 2020. DOI: 10.1017/dsd.2020.15.

[24] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital Twin in manufacturing: A categorical literature review and classification", *IFAC-PapersOnLine*,

vol. 51, no. 11, pp. 1016–1022, 2018, 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018. DOI: https://doi.org/10.1016/j.ifaco l.2018.08.474.

[25] ISO, "Automation systems and integration — Digital twin framework for manufacturing — Part 2: Reference architecture", International Organization for Standardization (ISO), International Standard 23247-2, Oct. 2021.

[26] ISO/IEC, "Digital twin — Concepts and terminology", International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), International Standard 30173, Nov. 2023.

[27] Digital Twin Consortium, *Definition of Digital Twin*. [Online]. Available: https://ww w.digitaltwinconsortium.org/initiatives/the-definition-of-a-digital-tw in/ (visited on 09/05/2025).

[28] IBM, *What is a digital twin?* [Online]. Available: https://www.ibm.com/think/topi cs/what-is-a-digital-twin (visited on 09/05/2025).

[29] Michael Grieves, *Don't 'Twin' Digital Twins and Simulations*. [Online]. Available: h ttps://www.eetimes.com/dont-twin-digital-twins-and-simulations/ (visited on 09/05/2025).

[30] M. Attaran and B. G. Celik, "Digital Twin: Benefits, use cases, challenges, and opportunities", *Decision Analytics Journal*, vol. 6, p. 100 165, 2023. DOI: https://doi .org/10.1016/j.dajour.2023.100165.

[31] A. Fuller, Z. Fan, C. Day, and C. Barlow, "Digital Twin: Enabling Technologies, Challenges and Open Research", *IEEE Access*, vol. 8, pp. 108 952–108 971, 2020. DOI: 10.1109/ACCESS.2020.2998358.

[32] W. Hu, T. Zhang, X. Deng, Z. Liu, and J. Tan, "Digital twin: A state-of-the-art review of its enabling technologies, applications and challenges", *Journal of Intelligent Manufacturing and Special Equipment*, vol. 2, no. 1, pp. 1–34, 2021.

[33] M. Perno, L. Hvam, and A. Haug, "Implementation of digital twins in the process industry: A systematic literature review of enablers and barriers", *Computers in Industry*, vol. 134, p. 103 558, 2022. DOI: https://doi.org/10.1016/j.compind.202 1.103558.

[34] J. Conde Díaz, "Contribución al desarrollo de arquitecturas y soluciones para la implementación de gemelos digitales basados en datos abiertos", Tesis (Doctoral), May 2023. DOI: 10.20868/UPM.thesis.74176.

[35] C. Zhuang, J. Gong, and J. Liu, "Digital twin-based assembly data management and process traceability for complex products", *Journal of Manufacturing Systems*, vol. 58, pp. 118–131, 2021, Digital Twin towards Smart Manufacturing and Industry 4.0. DOI: https://doi.org/10.1016/j.jmsy.2020.05.011.

[36] C. Schranz, F. Strohmeier, and V. Damjanovic-Behrendt, "A digital twin prototype for product lifecycle data management", in *2020 IEEE/ACS 17th International Conference on Computer Systems and Applications (AICCSA)*, 2020, pp. 1–6. DOI: 10.1109 /AICCSA50499.2020.9316506.

[37] M. Malaibari, M. Siddiqui, S. Xie, B. Bahramimianrood, S. Abdoli, and L. Djukic, "Digital Twin as A New Approach to Data Management: A Review", *Procedia CIRP*, vol. 128, pp. 375–380, 2024, 34th CIRP Design Conference. DOI: https://doi.org/1 0.1016/j.procir.2024.06.028.

[38] M. Lenzerini, "Data integration: A theoretical perspective", in *Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, ser. PODS '02, Madison, Wisconsin: Association for Computing Machinery, 2002, pp. 233–246. DOI: 10.1145/543613.543644.

[39] M. Cheatham and C. Pesquita, "Semantic Data Integration", in *Handbook of Big Data Technologies*, A. Y. Zomaya and S. Sakr, Eds. Cham: Springer International Publishing, 2017, pp. 263–305. DOI: 10.1007/978-3-319-49340-4_8.

[40] M. Masmoudi, S. Ben Abdallah Ben Lamine, M. H. Karray, B. Archimede, and H. Baazaoui Zghal, "Semantic Data Integration and Querying: A Survey and Challenges", *ACM Comput. Surv.*, vol. 56, no. 8, Apr. 2024. DOI: 10.1145/3653317.

[41] Levy, Alon Y., "Logic-Based Techniques in Data Integration", in *Logic-Based Artificial Intelligence*, J. Minker, Ed. Boston, MA: Springer US, 2000, pp. 575–595. DOI: 10.1007/978-1-4615-1567-8_24.

[42] J. Hossain, N. F. M. Sani, L. S. Affendey, I. Ishak, and K. A. Kasmiran, "Semantic Schema Matching Approaches: A Review.", *Journal of Theoretical & Applied Information Technology*, vol. 62, no. 1, pp. 139–147, 2014.

[43] L. Liu and M. T. Özsu, *Encyclopedia of Database Systems*. Springer New York, NY, 2009. DOI: https://doi.org/10.1007/978-0-387-39940-9.

[44] S. Chawathe, H. Garcia-Molina, J. Hammer, *et al.*, "The TSIMMIS Project: Integration of Heterogenous Information Sources", in *Information Processing Society of Japan (IPSJ 1994)*, 1994. [Online]. Available: http://ilpubs.stanford.edu:8090/66/ (visited on 09/05/2025).

[45] O. M. Duschka, M. R. Genesereth, and A. Y. Levy, "Recursive query plans for data integration", *The Journal of Logic Programming*, vol. 43, no. 1, pp. 49–73, 2000. DOI: https://doi.org/10.1016/S0743-1066(99)00025-4.

[46] M. Friedman, A. Levy, and T. Millstein, "Navigational plans for data integration", in *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence*, ser. AAAI '99/IAAI '99, Orlando, Florida, USA: American Association for Artificial Intelligence, 1999, pp. 67–73.

[47] P. McBrien and A. Poulovassilis, "Data integration by bi-directional schema transformation rules", in *Proceedings 19th International Conference on Data Engineering (Cat. No.03CH37405)*, 2003, pp. 227–238. DOI: 10.1109/ICDE.2003.1260795.

[48] P. Vassiliadis, "A Survey of Extract—Transform—Load Technology", *International Journal of Data Warehousing and Mining (IJDWM)*, vol. 5, no. 3, pp. 1–27, 2009. DOI: 10.4018/jdwm.2009070101.

[49] S. Chen, B. Liu, and E. A. Rundensteiner, "Multiversion-based view maintenance over distributed data sources", *ACM Trans. Database Syst.*, vol. 29, no. 4, pp. 675–709, Dec. 2004. DOI: 10.1145/1042046.1042049.

[50] G. Wiederhold, "Mediators in the architecture of future information systems", *Computer*, vol. 25, no. 3, pp. 38–49, 1992. DOI: 10.1109/2.121508.

[51] S. Tamtam, A. Laguidi, and A. Elkalay, "Data integration systems and bibliometrics", in *ITM Web of Conferences*, EDP Sciences, vol. 52, 2023. DOI: https://doi.org/10.1051/itmconf/20235201008.

[52] M. T. Özsu, P. Valduriez, *et al.*, *Principles of Distributed Database Systems*. Springer Cham, 1999. DOI: https://doi.org/10.1007/978-3-030-26253-2.

[53] M. Jacoby and T. Usländer, "Digital Twin and Internet of Things—Current Standards Landscape", *Applied Sciences*, vol. 10, no. 18, 2020. DOI: 10.3390/app10186519.

[54] Azure Contributors, *Digital Twins Definition Language (DTDL)*. [Online]. Available: https://github.com/Azure/opendigitaltwins-dtdl (visited on 09/05/2025).

[55] Platform Industrie 4.0, *The Asset Administration Shell: Implementing digital twins for use in Industrie 4.0*, 2022.

[56] ETSI, "Context Information Management (CIM); NGSI-LD API", ETSI, GS CIM 009 V1.8.1, Mar. 2024.

[57] Open Geospatial Consortium (OGC), *OCG SensorThings API*, 2019. [Online]. Available: https://www.opengeospatial.org/standards/sensorthings (visited on 09/05/2025).

[58] World Wide Web Consortium (W3C), *Web of Things*, 2023. [Online]. Available: https://www.w3.org/WoT/ (visited on 09/05/2025).

[59] Innovative Optical Wireless Network Global Forum, "Network Digital Twin Use Case", IOWN GLOBAL FORUM, Approved Reference Document 2.0, Jul. 2024.

[60] A. Pras and J. Schönwälder, *On the Difference between Information Models and Data Models*, RFC 3444, Jan. 2003. DOI: 10.17487/RFC3444.

[61] B. Claise, J. Clarke, and J. Lindblad, *Network Programmability with YANG: The Structure of Network Automation with YANG, NETCONF, RESTCONF, and gNMI*. Addison-Wesley Professional, 2019.

[62] Object Management Group (OMG), "Unified Modeling Language (UML), Version 2.5.1", OMG Standards Development Organization, Specification formal/17-12-05, Dec. 2017.

[63] R. Presuhn, *Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)*, RFC 3418, Dec. 2002. DOI: 10.17487/RFC3418.

[64] J. Schönwälder, D. T. Perkins, and K. McCloghrie, *Structure of Management Information Version 2 (SMIv2)*, RFC 2578, Apr. 1999. DOI: 10.17487/RFC2578.

[65] M. Fedor, M. L. Schoffstall, J. R. Davin, and D. J. D. Case, *Simple Network Management Protocol (SNMP)*, RFC 1157, May 1990. DOI: 10.17487/RFC1157.

[66] J. Stegeman, *What is a Knowledge Graph?*, Jul. 2024. [Online]. Available: https://neo4j.com/blog/genai/what-is-knowledge-graph/ (visited on 09/05/2025).

[67] C. Gutierrez and J. F. Sequeda, "Knowledge Graphs", *Communications of the ACM*, vol. 64, no. 3, pp. 96–104, Mar. 2021. DOI: 10.1145/3418294.

[68] Ontotext, *RDF Levels the Advantages of Labeled Property Graphs & Keeps Three Key Benefits: Standards, Semantics & Interoperability*, Jan. 2025. [Online]. Available: https://www.ontotext.com/knowledgehub/webinars/rdf-three-key-benefits/ (visited on 09/05/2025).

[69] Google, *Introducing the Knowledge Graph: Things, not strings*, May 2012. [Online]. Available: https://blog.google/products/search/introducing-knowledge-graph-things-not/ (visited on 09/05/2025).

[70] Ontotext, *What are Ontologies?* [Online]. Available: https://www.ontotext.com/knowledgehub/fundamentals/what-are-ontologies/ (visited on 09/05/2025).

[71] V. Ryen, A. Soylu, and D. Roman, "Building Semantic Knowledge Graphs from (Semi-)Structured Data: A Review", *Future Internet*, vol. 14, no. 5, p. 129, Apr. 2022. DOI: 10.3390/fi14050129.

[72] Ontotext, *What is Data Fabric?* [Online]. Available: https://www.ontotext.com/knowledgehub/fundamentals/what-is-data-fabric/ (visited on 09/05/2025).

[73] E. Hechler, M. Weihrauch, and Y. C. Wu, "Data Fabric Architecture Patterns", in *Data Fabric and Data Mesh Approaches with AI: A Guide to AI-based Data Cataloging, Governance, Integration, Orchestration, and Consumption*. Berkeley, CA: Apress, 2023, pp. 231–255. DOI: 10.1007/978-1-4842-9253-2_10.

[74] A.-M. Ghiran and R. A. Buchmann, "The Model-Driven Enterprise Data Fabric: A Proposal Based on Conceptual Modelling and Knowledge Graphs", in *Knowledge Science, Engineering and Management*, C. Douligeris, D. Karagiannis, and D. Apostolou, Eds., Cham: Springer International Publishing, 2019, pp. 572–583.

[75] S. Martin, B. Szekely, and D. Allemang, *The Rise of the Knowledge Graph*. O'Reilly Media, Inc., 2021. [Online]. Available: https://learning.oreilly.com/library/view/the-rise-of/9781098100407/ (visited on 09/05/2025).

[76] World Wide Web Consortium (W3C), *Resource Description Framework (RDF)*, 2014. [Online]. Available: https://www.w3.org/RDF/ (visited on 09/05/2025).

[77] World Wide Web Consortium (W3C), *OWL 2 Web Ontology Language: Document Overview*, 2012. [Online]. Available: https://www.w3.org/TR/owl2-overview/ (visited on 09/05/2025).

[78] Neo4j, *Neo4j Graph Database Platform*. [Online]. Available: https://neo4j.com/ (visited on 09/05/2025).

[79] J. Barrasa and J. Webber, *Building Knowledge Graphs: A Practitioner's Guide*. O'Reilly Media, Inc., Jun. 2023.

[80] W3C, *RDF-star Working Group*, 2025. [Online]. Available: https://www.w3.org/groups/wg/rdf-star/ (visited on 09/05/2025).

[81] ETSI, "Context Information Management (CIM); NGSI-LD Information Model", ETSI, GS CIM 006 V1.3.1, Mar. 2024.

[82] World Wide Web Consortium (W3C), *RDF Schema 1.1*, 2014. [Online]. Available: https://www.w3.org/TR/rdf-schema/ (visited on 09/05/2025).

[83] World Wide Web Consortium (W3C), *SKOS Simple Knowledge Organization System*, 2009. [Online]. Available: https://www.w3.org/TR/skos-reference/ (visited on 09/05/2025).

[84] I. D. Martinez-Casanueva, L. Bellido, D. González-Sánchez, and D. Lopez, "CANDIL: A federated data fabric for network analytics", *Future Generation Computer Systems*, vol. 158, pp. 98–109, 2024. DOI: https://doi.org/10.1016/j.future.2024.04.013.

[85] D. R. R. RAJ, T. A. Shaik, A. Hirwe, P. Tammana, and K. Kataoka, "Building a Digital Twin Network of SDN Using Knowledge Graphs", *IEEE Access*, vol. 11, pp. 63 092–63 106, 2023. DOI: 10.1109/ACCESS.2023.3288813.

[86] O. Havel, B. Claise, O. G. de Dios, and T. Graf, "SIMAP: Concept, Requirements, and Use Cases", Internet Engineering Task Force, Internet-Draft draft-ietf-nmop-simap-concept-03, Mar. 2025, Work in Progress, 33 pp. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-nmop-simap-concept/03/ (visited on 09/05/2025).

[87] ITU-T, "Signalling and protocol for Digital Twin Network supported for IMT-2020 network and beyond", Telecommunication Standardization Sector of ITU (ITU-T), Recommendation Q.SDTN, Jun. 2023.

[88] ITU-T, "Reference Architecture of digital twin federation in smart cities and communities", Telecommunication Standardization Sector of ITU (ITU-T), Recommendation Y.4489, Nov. 2023.

[89] ITU-T, "Requirements for digital twin federation in smart cities and communities", Telecommunication Standardization Sector of ITU (ITU-T), Recommendation Y.4224, Nov. 2023.

[90] ITU-T, "Information exchange model for digital twin federation in smart cities and community", Telecommunication Standardization Sector of ITU (ITU-T), Recommendation Y.4605, Nov. 2023.

[91] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Management and orchestration; Study on management aspect of Network Digital Twin", 3GPP, TR 28.915 v19.0.0, Dec. 2024.

[92] T. Forum, "Digital Twin and Closed Loop Automation", TeleManagement Forum, TR 284E v1.0.0, Jul. 2023.

[93] T. Forum, "Digital Twin and Closed Loop Automation for Autonomous Network Operations", TeleManagement Forum, TR 284G v1.0.0, Dec. 2024.

[94] T. Forum, "Digital Twin for Decision Intelligence (DT4DI)", TeleManagement Forum, IG 1307 v2.2.0, Aug. 2024.

[95] T. Forum, "Digital Twin for Decision Intelligence (DT4DI) Reference Architecture", TeleManagement Forum, IG 1310A v2.2.0, Feb. 2024.

[96] T. Forum, "Digital twin network application in 5G network operation", TeleManagement Forum, Catalyst Program C22.0.386, 2025.

[97] R. Poorzare, D. N. Kanellopoulos, V. K. Sharma, P. Dalapati, and O. P. Waldhorst, "Network Digital Twin Toward Networking, Telecommunications, and Traffic Engineering: A Survey", *IEEE Access*, vol. 13, pp. 16 489–16 538, 2025. DOI: 10.1109/ACCESS.2025.3531947.

[98] R. Enns, M. Björklund, A. Bierman, and J. Schönwälder, *Network Configuration Protocol (NETCONF)*, RFC 6241, Jun. 2011. DOI: 10.17487/RFC6241.

[99] P. Aitken, B. Claise, and B. Trammell, *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information*, RFC 7011, Sep. 2013. DOI: 10.17487/RFC7011.

[100] H. Song, F. Qin, P. Martinez-Julia, L. Ciavaglia, and A. Wang, *Network Telemetry Framework*, RFC 9232, May 2022. DOI: 10.17487/RFC9232.

[101] M. Björklund, *The YANG 1.1 Data Modeling Language*, RFC 7950, Aug. 2016. DOI: 10.17487/RFC7950.

[102] NS-3 Consortium, *NS-3 Network Simulator*, 2025. [Online]. Available: https://www.nsnam.org/ (visited on 09/05/2025).

[103] Bob Lantz and the Mininet Contributors, *Mininet: Rapid Prototyping for Software Defined Networks*, 2021. [Online]. Available: https://mininet.org/ (visited on 09/05/2025).

[104] GNS-3, *GNS3 - Graphical Network Simulator-3*, 2025. [Online]. Available: https://www.gns3.com/ (visited on 09/05/2025).

[105] EVE-NG Ltd., *EVE-NG: Emulated Virtual Environment Next Generation*, 2025. [Online]. Available: https://www.eve-ng.net/ (visited on 09/05/2025).

[106] J. Ros-Giralt, N. Amsel, S. Yellamraju, *et al.*, "Designing data center networks using bottleneck structures", in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, ser. SIGCOMM '21, Virtual Event, USA: Association for Computing Machinery, 2021, pp. 319–348. DOI: 10.1145/3452296.3472898.

[107] K. Rusek, J. Suárez-Varela, P. Almasan, P. Barlet-Ros, and A. Cabellos-Aparicio, "RouteNet: Leveraging Graph Neural Networks for Network Modeling and Optimization in SDN", *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2260–2270, 2020. DOI: 10.1109/JSAC.2020.3000405.

[108] A. Clemm, J. Medved, R. Varga, N. Bahadur, H. Ananthakrishnan, and X. Liu, *A YANG Data Model for Network Topologies*, RFC 8345, Mar. 2018. DOI: 10.17487/RFC8345.

[109] A. Clemm, J. Medved, R. Varga, X. Liu, H. Ananthakrishnan, and N. Bahadur, *A YANG Data Model for Layer 3 Topologies*, RFC 8346, Mar. 2018. DOI: 10.17487/RFC8346.

[110] J. Dong, X. Wei, Q. Wu, M. Boucadair, and A. Liu, *A YANG Data Model for Layer 2 Network Topologies*, RFC 8944, Nov. 2020. DOI: 10.17487/RFC8944.

[111] P. Saint-Andre, *Extensible Messaging and Presence Protocol (XMPP): Address Format*, RFC 7622, Sep. 2015. DOI: 10.17487/RFC7622.

[112] M. Bishop, *HTTP/3*, RFC 9114, Jun. 2022. DOI: 10.17487/RFC9114.

[113] M. Sanz Rodrigo, D. Rivera, J. I. Moreno, M. Álvarez-Campana, and D. R. López, "Digital Twins for 5G Networks: A Modeling and Deployment Methodology", *IEEE Access*, vol. 11, pp. 38 112–38 126, 2023. DOI: 10.1109/ACCESS.2023.3267548.

[114] L. Bariah and M. Debbah, "The Interplay of AI and Digital Twin: Bridging the Gap between Data-Driven and Model-Driven Approaches", *IEEE Wireless Communications*, vol. 31, no. 3, pp. 219–225, 2024. DOI: 10.1109/MWC.133.2200447.

[115] F. Javier Zorzano Mier and C. A. Iglesias, "Applications of Knowledge Graphs in Telecommunication Systems Management", *IEEE Internet Computing*, vol. 27, no. 3, pp. 29–34, 2023. DOI: 10.1109/MIC.2023.3253305.

[116] Z. Guo, F. Li, J. Shen, T. Xie, S. Jiang, and X. Wang, "ConfigReco: Network Configuration Recommendation With Graph Neural Networks", *IEEE Network*, vol. 38, no. 1, pp. 7–14, 2024. DOI: 10.1109/MNET.2023.3336239.

[117] Y. Zhu, D. Chen, C. Zhou, L. Lu, and X. Duan, "A knowledge graph based construction method for Digital Twin Network", in *2021 IEEE 1st International Conference on Digital Twins and Parallel Intelligence (DTPI)*, 2021, pp. 362–365. DOI: 10.1109/DTPI52967.2021.9540177.

[118] R. Van Der Lans, *Data Virtualization for Business Intelligence Systems: Revolutionizing Data Integration for Data Warehouses*. Elsevier, 2012. DOI: https://doi.org/10.1016/C2011-0-07129-6.

[119] I. M. Putrama and P. Martinek, "Heterogeneous data integration: Challenges and opportunities", *Data in Brief*, vol. 56, p. 110 853, 2024. DOI: https://doi.org/10.1016/j.dib.2024.110853.

[120] S. Rieger, L.-N. Lux, J. Schmitt, and M. Stiemerling, "DigSiNet: Using Multiple Digital Twins to Provide Rhythmic Network Consistency", in *NOMS 2024-2024 IEEE Network*

Operations and Management Symposium, 2024, pp. 1–5. DOI: 10.1109/NOMS59830.2024.10575632.

[121] S. Rieger, L.-N. Lux, S. Schickentanz, D. Hermann, T. Mott, and M. Freund, "Challenges of Event-based Streaming and Queuing as Data Exchange for Network Digital Twins", in *2024 20th International Conference on Network and Service Management (CNSM)*, 2024, pp. 1–7. DOI: 10.23919/CNSM62983.2024.10814474.

[122] NetBox Community, *NetBox: The premier source of truth powering network automation*, 2025. [Online]. Available: https://github.com/netbox-community/netbox (visited on 09/05/2025).

[123] Nautobot Community, *Nautobot: Network Source of Truth and Network Automation Platform*, 2025. [Online]. Available: https://github.com/nautobot/nautobot (visited on 09/05/2025).

[124] OpsMill, *Infrahub: Simplify Infrastructure Automation*, 2025. [Online]. Available: https://github.com/opsmill/infrahub (visited on 09/05/2025).

[125] Michael DeHaan, *Ansible*, 2025. [Online]. Available: https://github.com/ansible/ansible (visited on 09/05/2025).

[126] HashiCorp, Inc, *Terraform*, 2025. [Online]. Available: https://github.com/hashicorp/terraform (visited on 09/05/2025).

[127] D. González-Sánchez, L. Bellido, I. D. Martinez-Casanueva, D. Martínez-García, D. Fernández, and D. R. Lopez, "Toward Building a Digital Twin for Network Operations and Management", *IEEE Open Journal of the Communications Society*, vol. 6, pp. 2583–2598, 2025. DOI: 10.1109/OJCOMS.2025.3549873.

[128] B. Claise, *Cisco Systems NetFlow Services Export Version 9*, RFC 3954, Oct. 2004. DOI: 10.17487/RFC3954.

[129] A. Pastor, D. R. López, J. Ordonez-Lucena, S. Fernández, and J. Folgueira, "A Model-Based Approach to Multi-Domain Monitoring Data Aggregation", *Journal of ICT Standardization*, vol. 9, no. 2, pp. 291–310, 2021. DOI: 10.13052/jicts2245-800X.9210.

[130] M. Björklund, *YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)*, RFC 6020, Oct. 2010. DOI: 10.17487/RFC6020.

[131] P. A. Shafer, *An Architecture for Network Management Using NETCONF and YANG*, RFC 6244, Jun. 2011. DOI: 10.17487/RFC6244.

[132] M. Björklund, *A YANG Data Model for Interface Management*, RFC 8343, Mar. 2018. DOI: 10.17487/RFC8343.

[133] E. Voit, R. Rahman, E. Nilsen-Nygaard, A. Clemm, and A. Bierman, *Dynamic Subscription to YANG Events and Datastores over RESTCONF*, RFC 8650, Nov. 2019. DOI: 10.17487/RFC8650.

[134] M. Björklund, J. Schönwälder, P. A. Shafer, K. Watsen, and R. Wilton, *NETCONF Extensions to Support the Network Management Datastore Architecture*, RFC 8526, Mar. 2019. DOI: 10.17487/RFC8526.

[135] M. Björklund, J. Schönwälder, P. A. Shafer, K. Watsen, and R. Wilton, *Network Management Datastore Architecture (NMDA)*, RFC 8342, Mar. 2018. DOI: 10.17487/RFC8342.

[136] E. Voit, A. Clemm, A. G. Prieto, E. Nilsen-Nygaard, and A. Tripathy, *Subscription to YANG Notifications*, RFC 8639, Sep. 2019. DOI: 10.17487/RFC8639.

[137] A. Clemm and E. Voit, *Subscription to YANG Notifications for Datastore Updates*, RFC 8641, Sep. 2019. DOI: 10.17487/RFC8641.

[138] R. Wilton, H. Keller, B. Claise, E. Aries, J. Cumming, and T. Graf, "YANG Datastore Telemetry (YANG Push Lite)", Internet Engineering Task Force, Internet-Draft draft-wilton-netconf-yang-push-lite-00, Mar. 2025, Work in Progress, 129 pp. [Online]. Available: https://datatracker.ietf.org/doc/draft-wilton-netconf-yang-push-lite/00/ (visited on 09/05/2025).

[139] OpenConfig Community, *Grpc network management interface (gnmi)*. [Online]. Available: https://github.com/openconfig/reference/blob/master/rpc/gnmi/gnmi-specification.md (visited on 09/05/2025).

[140] OpenConfig Community, *Openconfig*. [Online]. Available: https://www.openconfig.net/ (visited on 09/05/2025).

[141] A. Bierman, M. Björklund, and K. Watsen, *RESTCONF Protocol*, RFC 8040, Jan. 2017. DOI: 10.17487/RFC8040.

[142] M. Björklund, J. Schönwälder, P. A. Shafer, K. Watsen, and R. Wilton, *RESTCONF Extensions to Support the Network Management Datastore Architecture*, RFC 8527, Mar. 2019. DOI: 10.17487/RFC8527.

[143] M. Oswalt, C. Adell, S. S. Lowe, and J. Edelman, *Network Programmability and Automation: Skills for the Next-generation Network Engineer*. O'Reilly Media, Incorporated, 2023.

[144] Q. Wu, M. Boucadair, D. Lopez, C. Xie, and L. Geng, *A Framework for Automating Service and Network Management with YANG*, RFC 8969, Jan. 2021. DOI: 10.17487/RFC8969.

[145] A. Clemm, L. Ciavaglia, L. Z. Granville, and J. Tantsura, *Intent-Based Networking - Concepts and Definitions*, RFC 9315, Oct. 2022. DOI: 10.17487/RFC9315.

[146] Q. Wu, S. Litkowski, L. Tomotaki, and K. Ogaki, *YANG Data Model for L3VPN Service Delivery*, RFC 8299, Jan. 2018. DOI: 10.17487/RFC8299.

[147] ETSI, "Zero-Touch Service and Managment; Reference Architecture", ETSI, GS ZSM 002 v1.1.1, Aug. 2019.

[148] ETSI, "Zero-Touch Service and Managment; Intent-driven autonomous networks; Generic aspects", ETSI, GR ZSM 011 v2.1.1, Sep. 2024.

[149] Benoit Claise, *The Maslow's Pyramid of Needs of Intent-based Networking*, Jan. 2021. [Online]. Available: https://www.claise.be/the-maslows-pyramid-of-needs-for-intent-based-networking/ (visited on 09/05/2025).

[150] B. Claise, J. Quilbeuf, D. Lopez, D. Voyer, and T. Arumugam, *Service Assurance for Intent-Based Networking Architecture*, RFC 9417, Jul. 2023. DOI: 10.17487/RFC9417.

[151] D. Brodimas, K. Trantzas, B. Agko, *et al.*, "Towards Intent-based Network Management for the 6G System adopting Multimodal Generative AI", in *2024 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*, 2024, pp. 848–853. DOI: 10.1109/EuCNC/6GSummit60053.2024.10597022.

[152] D. M. Manias, A. Chouman, and A. Shami, "Towards Intent-Based Network Management: Large Language Models for Intent Extraction in 5G Core Networks", in *2024 20th International Conference on the Design of Reliable Communication Networks (DRCN)*, 2024, pp. 1–6. DOI: 10.1109/DRCN60692.2024.10539172.

[153] A. Abane, A. Battou, and M. Merzouki, "An Adaptable AI Assistant for Network Management", in *NOMS 2024-2024 IEEE Network Operations and Management Symposium*, 2024, pp. 1–3. DOI: 10.1109/NOMS59830.2024.10574957.

[154] U. M. Natarajan, R. B. Diddigi, and J. Bapat, "RAG-inspired Intent-Based Solution for Intelligent Autonomous Networks", in *2025 17th International Conference on COMmunication Systems and NETworks (COMSNETS)*, 2025, pp. 413–421. DOI: 10.1109/COMSNETS63942.2025.10885706.

[155] E. H. Zach Blumenfeld, *What is Retrieval-Augmented Generation (rag)?*, Apr. 2024. [Online]. Available: https://neo4j.com/blog/genai/what-is-retrieval-augmented-generation-rag/ (visited on 09/05/2025).

[156] ETSI, "Zero-Touch Service and Managment; Enablers for Artificial Intelligence-based Network and Service Automation", ETSI, GS ZSM 012 v1.1.1, Dec. 2022.

[157] C.-N. Hang, P.-D. Yu, R. Morabito, and C.-W. Tan, "Large Language Models Meet Next-Generation Networking Technologies: A Review", *Future Internet*, vol. 16, no. 10, 2024. DOI: 10.3390/fi16100365.

[158] G. Hollósi, D. Ficzere, and P. Varga, "Generative AI for Low-Level NETCONF Configuration in Network Management Based on YANG Models", in *2024 20th International Conference on Network and Service Management (CNSM)*, 2024, pp. 1–7. DOI: 10.23919/CNSM62983.2024.10814410.

[159] E. James, *SR Linux and ChatGPT combine for network AIOps*, Oct. 2023. [Online]. Available: https://www.nokia.com/blog/sr-linux-and-chatgpt-combine-for-network-aiops/ (visited on 09/05/2025).

[160] E. James, *SR Linux GPT*, Oct. 2023. [Online]. Available: https://learn.srlinux.dev/blog/2023/sr-linux-gpt/ (visited on 09/05/2025).

[161] Nokia, *Service Router Linux*. [Online]. Available: https://www.nokia.com/networks/ip-networks/service-router-linux-NOS/ (visited on 09/05/2025).

[162] A. Bandi, J. Babu, R. Zeng, and S. R. Muthyala, "Enhancing Generative AI Chatbot Accuracy Using Knowledge Graph", in *Software and Data Engineering*, W. Feng, N. Rahimi, and V. Margapuri, Eds., Cham: Springer Nature Switzerland, 2025, pp. 157–167. DOI: https://doi.org/10.1007/978-3-031-75201-8_11.

[163] K. H. Tomaž Bratanič, *Knowledge Graphs and LLMs: Fine-Tuning vs. Retrieval-Augmented Generation*, Sep. 2024. [Online]. Available: https://neo4j.com/blog/developer/fine-tuning-vs-rag/ (visited on 09/05/2025).

[164] Y. Liu, Y. Dong, J. Shen, and C. Feng, "MAYA: Exploring multiform attributes of node to align YANG data models", *Computer Networks*, vol. 222, p. 109 538, 2023. DOI: https://doi.org/10.1016/j.comnet.2022.109538.

[165] D. Yongqiang, W. Xin, L. Yongbo, and Y. Wang, "Building Network Domain Knowledge Graph from Heterogeneous YANG Model", *Journal of Computer Research and Development*, vol. 57, no. 04, pp. 699–708, 2020. DOI: https://dx.doi.org/10.7544/issn1000-1239.2020.20190882.

[166] Benoit Claise, *Knowledge Graph as Preparation for LLMs, the Logical Next Step in Networking Data Modeling!*, Feb. 2025. [Online]. Available: https://www.claise.be/knowledge-graph-as-preparation-for-llms-the-logical-next-step-in-networking-data-modeling/ (visited on 09/05/2025).

[167] M. Mackey, B. Claise, T. Graf, H. Keller, D. Voyer, and P. Lucente, "Knowledge Graph Framework for Network Operations", Internet Engineering Task Force, Internet-Draft draft-mackey-nmop-kg-for-netops-01, Oct. 2024, Work in Progress, 28 pp. [Online]. Available: https://datatracker.ietf.org/doc/draft-mackey-nmop-kg-for-netops/01/ (visited on 09/05/2025).

[168] I. D. Martinez-Casanueva, L. C. Rodríguez, and P. Martinez-Julia, "Knowledge Graph Construction from Network Data Sources", Internet Engineering Task Force, Internet-Draft draft-marcas-nmop-kg-construct-00, Feb. 2025, Work in Progress, 20 pp. [Online]. Available: https://datatracker.ietf.org/doc/draft-marcas-nmop-kg-construct/00/ (visited on 09/05/2025).

[169] Lionel Tailhardat, Yoan Chabot, and Raphaël Troncy, "NORIA-O: an Ontology for Anomaly Detection and Incident Management in ICT Systems", in *Semantic Web - 21st International Conference, ESWC 2024, Hersonissos, Crete, Greece, May 26 - 30, 2024, Proceedings*, 2024. DOI: 10.1007/978-3-031-60635-9_2.

[170] M. Poveda-Villalón, A. Fernández-Izquierdo, M. Fernández-López, and R. García-Castro, "LOT: An industrial oriented ontology engineering framework", *Engineering Applications of Artificial Intelligence*, vol. 111, p. 104 755, 2022. DOI: https://doi.org/10.1016/j.engappai.2022.104755.

[171] ETSI, "Context Information Management (CIM); YANG and NGSI-LD interoperability", ETSI, GR CIM 038 v1.1.1, Nov. 2024.

[172] I. D. Martinez-Casanueva, D. Gonzalez-Sanchez, L. Bellido, D. Fernandez, and D. R. Lopez, "Toward Building a Semantic Network Inventory for Model-Driven Telemetry", *IEEE Communications Magazine*, vol. 61, no. 3, pp. 60–66, 2023. DOI: 10.1109/MCOM.001.2200222.

[173] GIROS DIT UPM, *YANG to NGSI-LD for Descriptive NDT*, 2025. [Online]. Available: https://github.com/giros-dit/yang-to-ngsi-ld (visited on 09/05/2025).

[174] M. Bjorklund, *Pyang: An extensible YANG validator and converter in Python*, 2025. [Online]. Available: https://github.com/mbj4668/pyang (visited on 09/05/2025).

[175] ETSI, "OpenAPI Specification for NGSI-LD API", ETSI, GS CIM 047 v1.1.2, Dec. 2024.

[176] OpenAPI Generator Contributors, *OpenAPI Generator*, 2025. [Online]. Available: https://openapi-generator.tech/ (visited on 09/05/2025).

[177] Pydantic Contributors, *Pydantic: Data validation using Python type hints*, 2025. [Online]. Available: https://github.com/pydantic/pydantic (visited on 09/05/2025).

[178] The Apache Software Foundation (ASF), *Apache Kafka*, 2025. [Online]. Available: https://kafka.apache.org/ (visited on 09/05/2025).

[179] DCMI (Dublin Core Metadata Initiative), *DCMI Metadata Terms*, Jan. 2020. [Online]. Available: https://www.dublincore.org/specifications/dublin-core/dcmi-terms/ (visited on 09/05/2025).

[180] M. Björklund, *IANA Interface Type YANG Module*, RFC 7224, May 2014. DOI: 10.17487/RFC7224.

[181] M. Björklund, *A YANG Data Model for IP Management*, RFC 8344, Mar. 2018. DOI: 10.17487/RFC8344.

[182] M. Björklund, *A YANG Data Model for Interface Management*, RFC 7223, May 2014. DOI: 10.17487/RFC7223.

[183] D. T. Narten, T. Jinmei, and D. S. Thomson, *IPv6 Stateless Address Autoconfiguration*, RFC 4862, Sep. 2007. DOI: 10.17487/RFC4862.

[184] M. Björklund, *A YANG Data Model for IP Management*, RFC 7277, Jun. 2014. DOI: 10.17487/RFC7277.

[185] OpenConfig working group, *OpenConfig YANG data model for managing network interfaces and subinterfaces*, Dec. 2024. [Online]. Available: https://openconfig.net/projects/models/schemadocs/yangdoc/openconfig-interfaces.html (visited on 09/05/2025).

[186] OpenConfig working group, *OpenConfig YANG data model for managing configuration and operational state on IP (IPv4 and IPv6) interfaces*, May 2024. [Online]. Available: https://github.com/openconfig/public/blob/master/release/models/interfaces/openconfig-if-ip.yang (visited on 09/05/2025).

[187] S. Nadas, *Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6*, RFC 5798, Mar. 2010. DOI: 10.17487/RFC5798.

[188] D. González-Sánchez, I. D. Martinez-Casanueva, A. Pastor, *et al.*, "Model-Driven Network Monitoring Using NetFlow Applied to Threat Detection", in *2022 IEEE 8th International Conference on Network Softwarization (NetSoft)*, 2022, pp. 450–455. DOI: 10.1109/NetSoft54395.2022.9844107.

[189] A. A. Moreno-Sancho, A. Pastor, I. D. Martinez-Casanueva, G.-S. Daniel, and L. B. Triana, "A data infrastructure for heterogeneous telemetry adaptation: application to Netflow-based cryptojacking detection", *Annals of Telecommunications*, vol. 79, pp. 241–256, Oct. 2023. DOI: https://doi.org/10.1007/s12243-023-00991-6.

[190] Cisco Systems, "Cisco IOS NetFlow Version 9 Flow-Record Format", *White Paper*, pp. 1–14, May 2011. [Online]. Available: http://www.cisco.com/en/US/technologies/tk648/tk362/technologies_white_paper09186a00800a3db9.pdf (visited on 09/05/2025).

[191] B. Li, J. Springer, G. Bebis, and M. Hadi Gunes, "A survey of network flow applications", *Journal of Network and Computer Applications*, vol. 36, no. 2, pp. 567–581, 2013. DOI: https://doi.org/10.1016/j.jnca.2012.12.020.

[192] A. Pastor, A. Mozo, S. Vakaruk, *et al.*, "Detection of Encrypted Cryptomining Malware Connections With Machine and Deep Learning", *IEEE Access*, vol. 8, pp. 158 036–158 055, 2020. DOI: 10.1109/ACCESS.2020.3019658.

[193] The Apache Software Foundation (ASF), *Apache Flink: Stateful Computations over Data Streams*, 2025. [Online]. Available: https://flink.apache.org/ (visited on 09/05/2025).

[194] OpenDayLight Project, *YANG Tools*, 2025. [Online]. Available: https://github.com/opendaylight/yangtools (visited on 09/05/2025).

[195] A. Bierman, M. Björklund, J. Schönwälder, K. Watsen, and R. Wilton, *YANG Library*, RFC 8525, Mar. 2019. DOI: 10.17487/RFC8525.

[196] ncclient Contributors, *Ncclient: Python library for NETCONF clients*, 2025. [Online]. Available: https://github.com/ncclient/ncclient (visited on 09/05/2025).

[197] OpenConfig Community, *Gnmic*, 2025. [Online]. Available: https://github.com/openconfig/gnmic (visited on 09/05/2025).

[198] NetSampler Contributors, *goflow2: High performance sFlow/IPFIX/NetFlow Collector*, 2025. [Online]. Available: https://github.com/netsampler/goflow2 (visited on 09/05/2025).

[199] SRL Labs, *Containerlab*, 2025. [Online]. Available: https://github.com/srl-labs/containerlab (visited on 09/05/2025).

[200] FIWARE Contributors, *Orion Context Broker (with Linked Data Extensions)*, 2025. [Online]. Available: https://github.com/FIWARE/context.Orion-LD (visited on 09/05/2025).

[201] ScorpioBroker Contributors, *Scorpio NGSI-LD Broker*, 2025. [Online]. Available: https://github.com/ScorpioBroker/ScorpioBroker (visited on 09/05/2025).

[202] Matplotlib Contributors, *Matplotlib: Visualization with Python*, 2025. [Online]. Available: https://matplotlib.org/ (visited on 09/05/2025).

[203] Tobi Oetiker, *MRTG - The Multi Router Traffic Grapher*. [Online]. Available: https://oss.oetiker.ch/mrtg/ (visited on 09/05/2025).

[204] The Cacti Group, *Cacti - The Complete RRDTool-based Graphing Solution*. [Online]. Available: https://www.cacti.net/ (visited on 09/05/2025).

[205] Nagios Enterprises, *Nagios - The Industry Standard in IT Infrastructure Monitoring*. [Online]. Available: https://www.nagios.org/ (visited on 09/05/2025).

[206] Tobi Oetiker, *RRDtool - Round Robin Database Tool*. [Online]. Available: https://oss.oetiker.ch/rrdtool/ (visited on 09/05/2025).

[207] M. Veillette, P. V. der Stok, A. Pelov, A. Bierman, and C. Bormann, "CoAP Management Interface (CORECONF)", Internet Engineering Task Force, Internet-Draft draft-ietf-core-comi-19, Nov. 2024, Work in Progress, 48 pp. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-core-comi/19/ (visited on 09/05/2025).

[208] Z. Shelby, K. Hartke, and C. Bormann, *The Constrained Application Protocol (CoAP)*, RFC 7252, Jun. 2014. DOI: 10.17487/RFC7252.

[209] C. Bormann and P. E. Hoffman, *Concise Binary Object Representation (CBOR)*, RFC 8949, Dec. 2020. DOI: 10.17487/RFC8949.

[210] IEEE 802.1 Time-Sensitive Networking Task Group, *Time-Sensitive Networking*. [Online]. Available: https://1.ieee802.org/tsn/ (visited on 09/05/2025).

[211] M. G. Bhat, S. Bhattacharjee, C. Gündoğan, K. Alexandris, and A. Gogolev, "CORECONF, NETCONF, and RESTCONF: Benchmarking Network Orchestration in Constrained IIoT Devices", *IEEE Internet of Things Journal*, vol. 11, no. 7, pp. 13 082–13 090, 2024. DOI: 10.1109/JIOT.2023.3338470.

[212] S. Bhattacharjee, K. Alexandris, and T. Bauschert, "Multi-Domain TSN Orchestration and Management for Large-Scale Industrial Networks", *IEEE Transactions on Network and Service Management*, vol. 21, no. 6, pp. 6480–6492, 2024. DOI: 10.1109/TNSM.2024.3447789.

[213] Scrapli Contributors, *scrapli_netconf: NETCONF support for Scrapli*, 2025. [Online]. Available: https://github.com/scrapli/scrapli_netconf (visited on 09/05/2025).

[214] J. Crist-Harif, *msgspec: A fast serialization and validation library, with builtin support for JSON, MessagePack, YAML, and TOML*, 2025. [Online]. Available: https://github.com/jcrist/msgspec (visited on 09/05/2025).

[215] Koudai Aono, *Data Model Code Generator*, 2025. [Online]. Available: https://koxudaxi.github.io/datamodel-code-generator/ (visited on 09/05/2025).

[216] Strimzi Project, *Strimzi - Apache Kafka on Kubernetes*, 2025. [Online]. Available: https://strimzi.io/ (visited on 09/05/2025).

[217] OpenConfig Community, *Kubernetes Network Emulation (KNE)*, 2025. [Online]. Available: https://github.com/openconfig/kne (visited on 09/05/2025).

[218] SRL Labs, *Clabernetes (c9s)*, 2025. [Online]. Available: https://github.com/srl-labs/clabernetes (visited on 09/05/2025).

[219] M. Björklund and L. Berger, *YANG Tree Diagrams*, RFC 8340, Mar. 2018. DOI: 10.17487/RFC8340.

Annexes

This chapter presents the annexes of this Doctoral Thesis. The annexes include additional design and implementation details of the Network Digital Twin solution presented in this document. It also includes a summary of the software development activities addressed during the thesis period.

Table of Contents

A.1 Descriptive NDT Architecture and Prototype Design 186
A.2 Data Formats within the NDT Data Integration Pipeline 188
A.2.1 Data formats for NETCONF case 188
A.2.2 Data formats for gNMI case 197
A.3 Handling YANG Configuration and Operational Data in Descriptive NDT solution 202
A.3.1 Handling of NMDA compliant IETF models 202
A.3.2 Handling of non-NMDA compliant IETF models 203
A.3.3 Handling of OpenConfig models 204
A.4 Examples of Query Processing Data Integration Workflows . . . 206
A.4.1 Querying operational status data from a network device interface . 206
A.4.2 Subscription to telemetry statistics from a network device interface 206
A.4.3 Querying configuration data from a network device 209
A.4.4 Updating the configuration of a network device interface 211
A.5 Open-Source Software Development Activity 216
A.5.1 Prototype implementation of Descriptive NDT 216
A.5.2 Semantic Data Aggregator and CANDIL Data Fabric 216
A.5.3 OpenAPI Specification for NGSI-LD and artifacts 217
A.5.4 Containerlab testbeds for analyzing network telemetry and management mechanisms based on YANG 217

A.1 Descriptive NDT Architecture and Prototype Design

This annex provides an expanded view of the diagrams of both the Descriptive NDT architecture (see Figure A.1) and the Descriptive NDT prototype design (see Figure A.2).

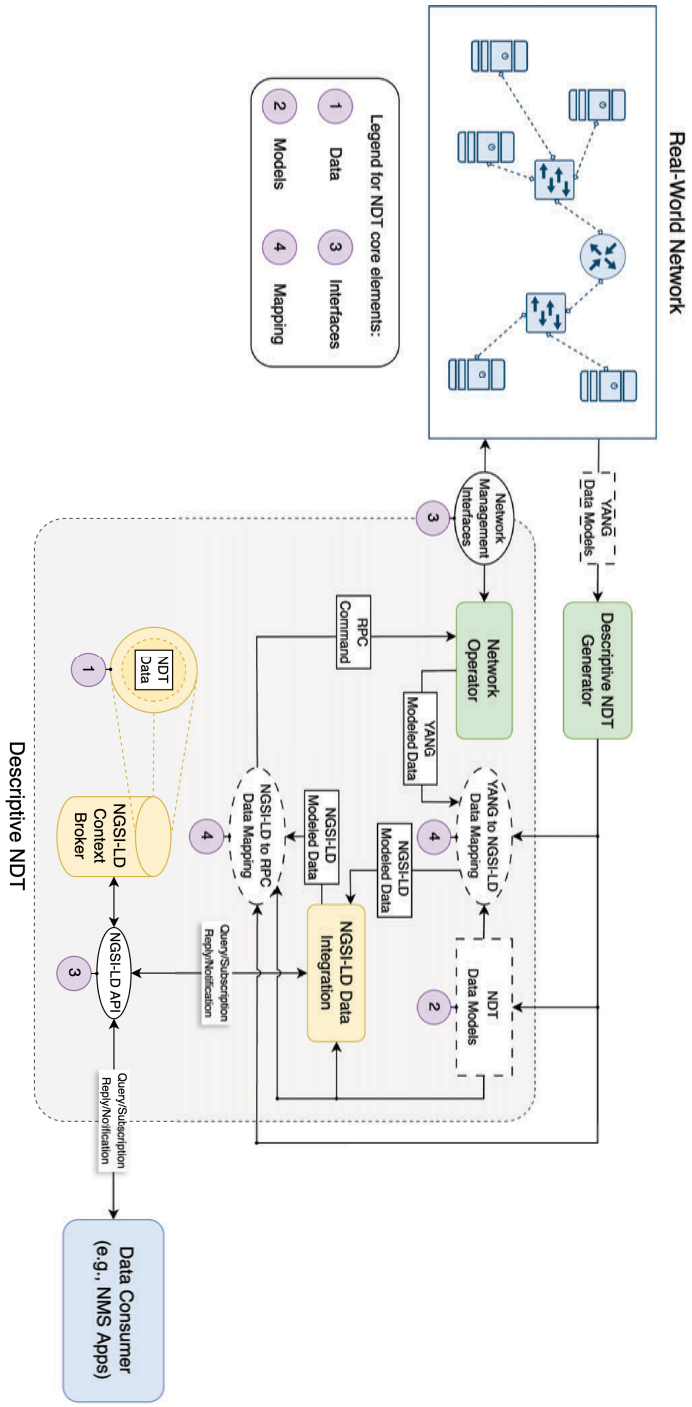


Figure A.1: Descriptive NDT architecture.

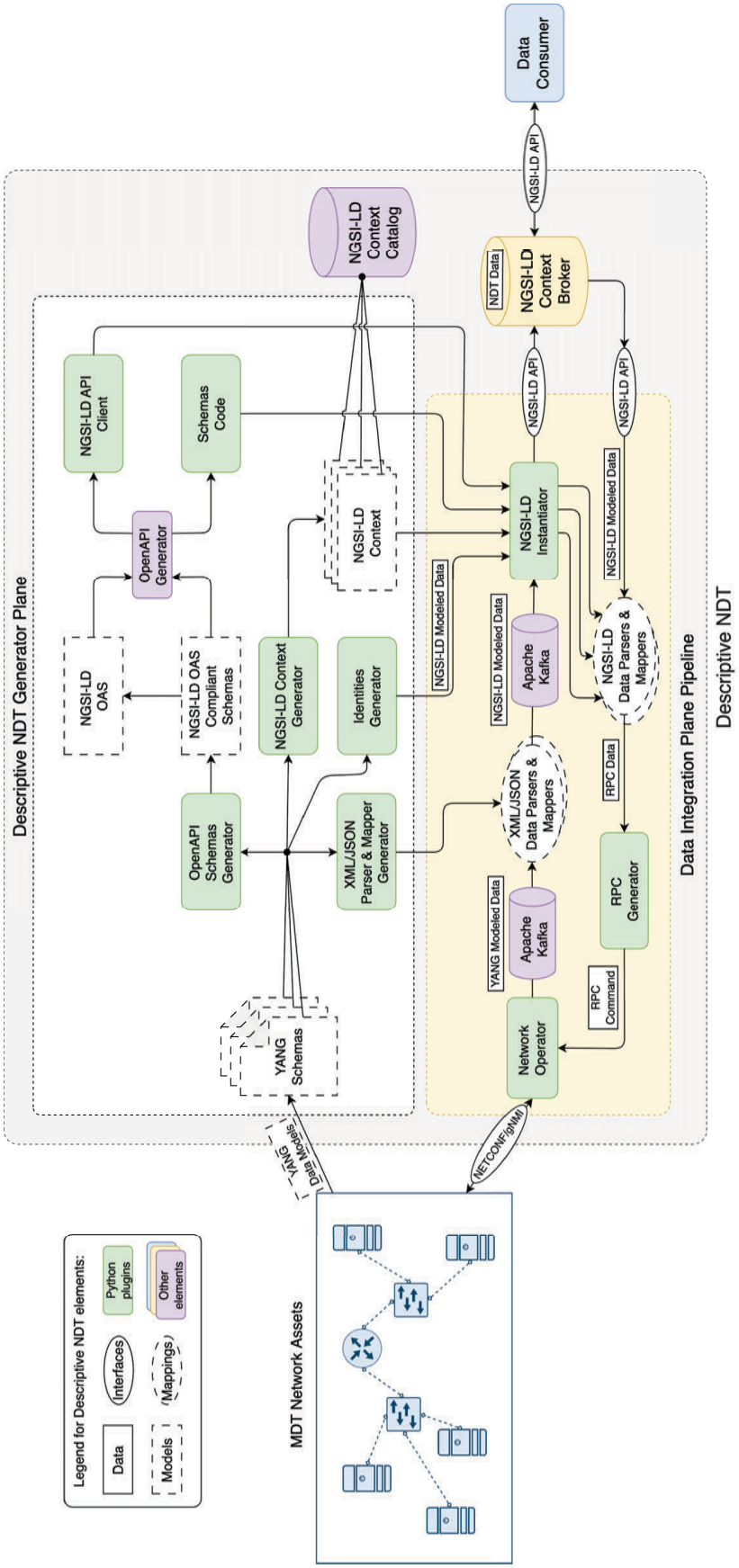


Figure A.2: Descriptive NDT prototype design.

A.2 Data Formats within the NDT Data Integration Pipeline

This annex provides details on the input and output data formats involved in the YANG to NGSI-LD translation process of the data integration pipeline within the Descriptive NDT solution presented in this work. As an example, we have taken into account the data pipeline for the data collection process of operational data, configuration data, and telemetry statistics from network interfaces extracted from servers supporting YANG data modeling language and network management protocols such as NETCONF or gNMI.

A.2.1 Data formats for NETCONF case

On the one hand, Figure A.3 and Figure A.4 depict samples of raw network interface operational and configuration data retrieved by the NETCONF protocol in XML encoding format from a NETCONF sever that does not support the NMDA architecture.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:761b06e4-de81-4c02-a73e-fa77d9e9286f"
    xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <interfaces-state
            xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
            <interface>
                <name>GigabitEthernet2</name>
                <type
                    xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">ianaift:ethernetCsmacd
                </type>
                <admin-status>up</admin-status>
                <oper-status>up</oper-status>
                <last-change>2025-03-25T20:20:58.937+00:00</last-change>
                <if-index>2</if-index>
                <phys-address>0c:00:87:be:c0:01</phys-address>
                <speed>1000000000</speed>
                <statistics>
                    <discontinuity-time>2025-03-25T20:19:28+00:00</discontinuity-time>
                    <in-octets>1486</in-octets>
                    <in-unicast-pkts>19</in-unicast-pkts>
                    <in-broadcast-pkts>0</in-broadcast-pkts>
                    <in-multicast-pkts>0</in-multicast-pkts>
                    <in-discards>0</in-discards>
                    <in-errors>0</in-errors>
                    <in-unknown-protos>0</in-unknown-protos>
                    <out-octets>0</out-octets>
                    <out-unicast-pkts>0</out-unicast-pkts>
                    <out-broadcast-pkts>0</out-broadcast-pkts>
                    <out-multicast-pkts>0</out-multicast-pkts>
                    <out-discards>0</out-discards>
                    <out-errors>0</out-errors>
                </statistics>
            </interface>
        </interfaces-state>
    </data>
</rpc-reply>
```

Figure A.3: Operational data about network interface retrieved by NETCONF protocol in XML encoding format (non-NMDA architecture).

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:8521b5a8-9109-4e25-8f51-c491ea8a8a8e"
    xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <interfaces
            xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
            <interface>
                <name>GigabitEthernet2</name>
                <description>r1-GigabitEthernet2</description>
                <type
                    xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">ianaift:ethernetCsmacd
                </type>
                <enabled>true</enabled>
                <ipv4
                    xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
                    <address>
                        <ip>10.0.1.1</ip>
                        <netmask>255.255.255.0</netmask>
                    </address>
                </ipv4>
                <ipv6
                    xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
                </ipv6>
            </interface>
        </interfaces>
    </data>
</rpc-reply>
```

Figure A.4: Configuration data about network interface retrieved by NETCONF protocol in XML encoding format (non-NMDA architecture).

Meanwhile, Figure A.5 and Figure A.6 show the corresponding representations in the NGSI-LD data format.

The operational data related to the interface statistic counters retrieved by NETCONF, as shown in Figure A.3, would result in an NGSI-LD entity of type *InterfaceStatistics*, which will be discussed later, but with a focus on the telemetry data. The *datasetId* parameter within each property of the resulting NGSI-LD entities allows the data to be tagged as operational or configuration.

The details of how to distinguish operational data from configuration data in the formal NGSI-LD representation used in Descriptive NDT are discussed in Annex A.3.

In addition, the configuration data retrieved by NETCONF includes information about the IPv4 addressing of the interface, as shown in Figure A.4. The resulting representation in NGSI-LD data format is shown in Figure A.7.

```
{
    "id": "urn:ngsi-ld:InterfaceState:clab-telemetry-ixiac-lab:r1:GigabitEthernet2",
    "type": "InterfaceState",
    "interfaceType": {
        "type": "Relationship",
        "datasetId": "urn:ngsi-ld:operational",
        "object": "urn:ngsi-ld:YANGIdentity:ianaift:ethernetCsmacd",
        "observedAt": "2025-03-25T22:48:08.334930Z"
    },
    "adminStatus": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": "up",
        "observedAt": "2025-03-25T22:48:08.334930Z"
    },
    "ifIndex": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": 2,
        "observedAt": "2025-03-25T22:48:08.334930Z"
    },
    "lastChange": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": "2025-03-25T20:20:58.937+00:00",
        "observedAt": "2025-03-25T22:48:08.334930Z"
    },
    "name": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": "GigabitEthernet2",
        "observedAt": "2025-03-25T22:48:08.334930Z"
    },
    "operStatus": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": "up",
        "observedAt": "2025-03-25T22:48:08.334930Z"
    },
    "physAddress": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": "0c:00:87:be:c0:01",
        "observedAt": "2025-03-25T22:48:08.334930Z"
    },
    "speed": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": 1000000000,
        "observedAt": "2025-03-25T22:48:08.334930Z"
    }
}
```


Figure A.5: Operational data about network interface retrieved by NETCONF protocol in NGSI-LD data format (non-NMDA architecture).

```
{
    "id": "urn:ngsi-ld:InterfaceConfig:clab-telemetry-ixiac-lab:r1:GigabitEthernet2",
    "type": "InterfaceConfig",
    "description": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:configuration",
        "value": "r1-GigabitEthernet2",
        "observedAt": "2025-03-25T22:54:52.809574Z"
    },
    "interfaceType": {
        "type": "Relationship",
        "datasetId": "urn:ngsi-ld:configuration",
        "object": "urn:ngsi-ld:YANGIdentity:ianaift:ethernetCsmacd",
        "observedAt": "2025-03-25T22:54:52.809574Z"
    },
    "name": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:configuration",
        "value": "GigabitEthernet2",
        "observedAt": "2025-03-25T22:54:52.809574Z"
    },
    "enabled": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:configuration",
        "value": true,
        "observedAt": "2025-03-25T22:54:52.809574Z"
    }
}
```


Figure A.6: Configuration data about network interface retrieved by NETCONF protocol in NGSI-LD data format (non-NMDA architecture).


```
{
    "id": "urn:ngsi-ld:InterfaceConfigIpv4Address:clab-telemetry-ixiac-lab:r1:GigabitEthernet2:10.0.1.1",
    "type": "InterfaceConfigIpv4Address",
    "isPartOf": {
        "type": "Relationship",
        "object": "urn:ngsi-ld:InterfaceConfigIpv4:clab-telemetry-ixiac-lab:r1:GigabitEthernet2",
        "observedAt": "2025-03-25T22:54:52.809574Z"
    },
    "netmask": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:configuration",
        "value": "255.255.255.0",
        "observedAt": "2025-03-25T22:54:52.809574Z"
    },
    "ip": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:configuration",
        "value": "10.0.1.1",
        "observedAt": "2025-03-25T22:54:52.809574Z"
    }
}
```


Figure A.7: Configuration data about IPv4 addressing of network interface retrieved by NETCONF protocol in NGSI-LD data format (non-NMDA architecture).

On the other hand, Figure A.8 and Figure A.9 depict samples of raw network interface operational and configuration data retrieved by the NETCONF protocol in XML encoding format from a NETCONF sever supporting the NMDA architecture.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:761b06e4-de81-4c02-a73e-fa77d9e9286f"
    xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <interfaces
            xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
            <interface>
                <name>GigabitEthernet2</name>
                <type
                    xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">ianaift:ethernetCsmacd
                </type>
                <description>r1-GigabitEthernet2</description>
                <enabled>true</enabled>
                <admin-status>up</admin-status>
                <oper-status>up</oper-status>
                <last-change>2025-03-25T20:20:58.937+00:00</last-change>
                <if-index>2</if-index>
                <phys-address>0c:00:87:be:c0:01</phys-address>
                <speed>1000000000</speed>
                <statistics>
                    <discontinuity-time>2025-03-25T20:19:28+00:00</discontinuity-time>
                    <in-octets>1486</in-octets>
                    <in-unicast-pkts>19</in-unicast-pkts>
                    <in-broadcast-pkts>0</in-broadcast-pkts>
                    <in-multicast-pkts>0</in-multicast-pkts>
                    <in-discards>0</in-discards>
                    <in-errors>0</in-errors>
                    <in-unknown-protos>0</in-unknown-protos>
                    <out-octets>0</out-octets>
                    <out-unicast-pkts>0</out-unicast-pkts>
                    <out-broadcast-pkts>0</out-broadcast-pkts>
                    <out-multicast-pkts>0</out-multicast-pkts>
                    <out-discards>0</out-discards>
                    <out-errors>0</out-errors>
                </statistics>
                <ipv4
                    xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
                    <address>
                        <ip>10.0.1.1</ip>
                        <netmask>255.255.255.0</netmask>
                    </address>
                </ipv4>
                <ipv6
                    xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
                </ipv6>
            </interface>
        </interfaces>
    </data>
</rpc-reply>
```

Figure A.8: Operational data about network interface retrieved by NETCONF protocol in XML encoding format (NMDA architecture).

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:8521b5a8-9109-4e25-8f51-c491ea8a8a8e"
    xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <interfaces
            xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
            <interface>
                <name>GigabitEthernet2</name>
                <description>r1-GigabitEthernet2</description>
                <type
                    xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">ianaift:ethernetCsmacd
                </type>
                <enabled>true</enabled>
                <ipv4
                    xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
                    <address>
                        <ip>10.0.1.1</ip>
                        <netmask>255.255.255.0</netmask>
                    </address>
                </ipv4>
                <ipv6
                    xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
                </ipv6>
            </interface>
        </interfaces>
    </data>
</rpc-reply>
```

Figure A.9: Configuration data about network interface retrieved by NETCONF protocol in XML encoding format (NMDA architecture).

Meanwhile, Figure A.10 shows the corresponding representation in the NGSI-LD data format.


```
{
    "id": "urn:ngsi-ld:Interface:clab-telemetry-ixiac-lab:r1:GigabitEthernet2",
    "type": "Interface",
    "interfaceType": {
        "type": "Relationship",
        "datasetId": "urn:ngsi-ld:operational",
        "object": "urn:ngsi-ld:YANGIdentity:ianaift:ethernetCsmacd",
        "observedAt": "2025-03-25T22:54:52.809574Z"
    },
    "description": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:configuration",
        "value": "r1-GigabitEthernet2",
        "observedAt": "2025-03-25T22:54:52.809574Z"
    },
    "enabled": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:configuration",
        "value": true,
        "observedAt": "2025-03-25T22:54:52.809574Z"
    },
    "adminStatus": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": "up",
        "observedAt": "2025-03-25T22:48:08.334930Z"
    },
    "ifIndex": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": 2,
        "observedAt": "2025-03-25T22:48:08.334930Z"
    },
    "lastChange": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": "2025-03-25T20:20:58.937+00:00",
        "observedAt": "2025-03-25T22:48:08.334930Z"
    },
    "name": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": "GigabitEthernet2",
        "observedAt": "2025-03-25T22:54:52.809574Z"
    },
    "operStatus": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": "up",
        "observedAt": "2025-03-25T22:48:08.334930Z"
    },
    "physAddress": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": "0c:00:87:be:c0:01",
        "observedAt": "2025-03-25T22:48:08.334930Z"
    },
    "speed": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": 1000000000,
        "observedAt": "2025-03-25T22:48:08.334930Z"
    }
}
```


Figure A.10: Operational and configuration data about network interface retrieved by NETCONF protocol in NGSI-LD data format (NMDA architecture).

Besides, Figure A.11 depicts a sample of a raw telemetry notification about network interface statistics counters retrieved by the NETCONF protocol in XML encoding format and Figure A.12 depicts the corresponding representation in the NGSI-LD data format.

```
<?xml version="1.0" encoding="UTF-8"?>
<notification
    xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <eventTime>2025-03-25T20:34:14.343212Z</eventTime>
    <push-update
        xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
        <subscription-id>2147483650</subscription-id>
        <datastore-contents-xml>
            <interfaces-state
                xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
                <interface>
                    <name>GigabitEthernet2</name>
                    <statistics>
                        <discontinuity-time>2025-03-25T20:19:28+00:00</discontinuity-time>
                        <in-octets>1486</in-octets>
                        <in-unicast-pkts>19</in-unicast-pkts>
                        <in-broadcast-pkts>0</in-broadcast-pkts>
                        <in-multicast-pkts>0</in-multicast-pkts>
                        <in-discards>0</in-discards>
                        <in-errors>0</in-errors>
                        <in-unknown-protos>0</in-unknown-protos>
                        <out-octets>0</out-octets>
                        <out-unicast-pkts>0</out-unicast-pkts>
                        <out-broadcast-pkts>0</out-broadcast-pkts>
                        <out-multicast-pkts>0</out-multicast-pkts>
                        <out-discards>0</out-discards>
                        <out-errors>0</out-errors>
                    </statistics>
                </interface>
            </interfaces-state>
        </datastore-contents-xml>
    </push-update>
</notification>
```

Figure A.11: Telemetry notification about network interface statistics counters retrieved by NETCONF protocol in XML encoding format.

```
{
    "id": "urn:ngsi-ld:InterfaceStatistics:clab-telemetry-ixiac-lab:rl:GigabitEthernet2",
    "type": "InterfaceStatistics",
    "isPartOf": {
        "type": "Relationship",
        "object": "urn:ngsi-ld:Interface:clab-telemetry-ixiac-lab:rl:GigabitEthernet2",
        "observedAt": "2025-03-25T20:34:14.343212Z"
    },
    "discontinuityTime": {
        "type": "Property",
        "value": "2025-03-25T20:19:28+00:00",
        "datasetId": "urn:ngsi-ld:operational",
        "observedAt": "2025-03-25T20:34:14.343212Z"
    },
    "inOctets": {
        "type": "Property",
        "value": 1486,
        "datasetId": "urn:ngsi-ld:operational",
        "observedAt": "2025-03-25T20:34:14.343212Z"
    },
    "inUnicastPkts": {
        "type": "Property",
        "value": 19,
        "datasetId": "urn:ngsi-ld:operational",
        "observedAt": "2025-03-25T20:34:14.343212Z"
    },
    "inBroadcastPkts": {
        "type": "Property",
        "value": 0,
        "datasetId": "urn:ngsi-ld:operational",
        "observedAt": "2025-03-25T20:34:14.343212Z"
    },
    "inMulticastPkts": {
        "type": "Property",
        "value": 0,
        "datasetId": "urn:ngsi-ld:operational",
        "observedAt": "2025-03-25T20:34:14.343212Z"
    },
    "inDiscards": {
        "type": "Property",
        "value": 0,
        "datasetId": "urn:ngsi-ld:operational",
        "observedAt": "2025-03-25T20:34:14.343212Z"
    },
    "inErrors": {
        "type": "Property",
        "value": 0,
        "datasetId": "urn:ngsi-ld:operational",
        "observedAt": "2025-03-25T20:34:14.343212Z"
    },
    "inUnknownProtos": {
        "type": "Property",
        "value": 0,
        "datasetId": "urn:ngsi-ld:operational",
        "observedAt": "2025-03-25T20:34:14.343212Z"
    },
    "outOctets": {
        "type": "Property",
        "value": 0,
        "datasetId": "urn:ngsi-ld:operational",
        "observedAt": "2025-03-25T20:34:14.343212Z"
    },
    "outUnicastPkts": {
        "type": "Property",
        "value": 0,
        "datasetId": "urn:ngsi-ld:operational",
        "observedAt": "2025-03-25T20:34:14.343212Z"
    },
    "outBroadcastPkts": {
        "type": "Property",
        "value": 0,
        "datasetId": "urn:ngsi-ld:operational",
        "observedAt": "2025-03-25T20:34:14.343212Z"
    },
    "outMulticastPkts": {
        "type": "Property",
        "value": 0,
        "datasetId": "urn:ngsi-ld:operational",
        "observedAt": "2025-03-25T20:34:14.343212Z"
    },
    "outDiscards": {
        "type": "Property",
        "value": 0,
        "datasetId": "urn:ngsi-ld:operational",
        "observedAt": "2025-03-25T20:34:14.343212Z"
    },
    "outErrors": {
        "type": "Property",
        "value": 0,
        "datasetId": "urn:ngsi-ld:operational",
        "observedAt": "2025-03-25T20:34:14.343212Z"
    }
}
```


Figure A.12: Telemetry notification about network interface statistics counters retrieved by NETCONF protocol in NGSI-LD data format.

A.2.2 Data formats for gNMI case

On the one hand, Figure A.13 and Figure A.14 depict samples of raw network interface operational and configuration data retrieved by the gNMI protocol in JSON encoding format from a gNMI server.


```
[
    {
        "source": "clab-telemetry-testbed-xrv9k-ceos-4hosts-r1:57400",
        "timestamp": 1742915466720497965,
        "time": "2025-03-25T16:11:06.720497965+01:00",
        "updates": [
            {
                "Path": "openconfig-interfaces:interfaces/interface[name=\"GigabitEthernet0/0/0/0\"]/state",
                "values": {
                    "openconfig-interfaces:interfaces/interface/state": {
                        "admin-status": "UP",
                        "counters": {
                            "carrier-transitions": 0,
                            "in-broadcast-pkts": 54,
                            "in-discards": 6,
                            "in-errors": 0,
                            "in-fcs-errors": 0,
                            "in-multicast-pkts": 0,
                            "in-octets": 91420,
                            "in-pkts": 458,
                            "in-unicast-pkts": 404,
                            "in-unknown-protos": 0,
                            "out-broadcast-pkts": 1,
                            "out-discards": 0,
                            "out-errors": 0,
                            "out-multicast-pkts": 0,
                            "out-octets": 2352,
                            "out-pkts": 56,
                            "out-unicast-pkts": 55
                        },
                        "description": "Interface GigabitEthernet 0",
                        "enabled": true,
                        "ifindex": 6,
                        "last-change": 1742905693176368000,
                        "logical": false,
                        "loopback-mode": "NONE",
                        "mtu": 1512,
                        "name": "GigabitEthernet0/0/0/0",
                        "oper-status": "UP",
                        "type": "iana-if-type:ethernetCsmacd"
                    }
                }
            }
        ]
    }
]
```


Figure A.13: Operational data about network interface retrieved by gNMI protocol in JSON encoding format.


```
[
  {
    "source": "clab-telemetry-testbed-xrv9k-ceos-4hosts-r1:57400",
    "timestamp": 1742915527949665542,
    "time": "2025-03-25T16:12:07.949665542+01:00",
    "updates": [
      {
        "Path": "openconfig-interfaces:interfaces/interface[name=\"GigabitEthernet0/0/0/0\"]/config",
        "values": {
          "openconfig-interfaces:interfaces/interface/config": {
            "description": "Interface GigabitEthernet 0",
            "mtu": 1512,
            "name": "GigabitEthernet0/0/0/0",
            "type": "iana-if-type:ethernetCsmacd"
          }
        }
      }
    ]
  }
]
```


Figure A.14: Configuration data about network interface retrieved by gNMI protocol in JSON encoding format.

Meanwhile, Figure A.15 and Figure A.16 show the corresponding representations in the NGSI-LD data format.


```
{
    "id": "urn:ngsi-ld:InterfaceState:telemetry-testbed-xrv9k-ceos-4hosts:r1:GigabitEthernet0/0/0/0",
    "type": "InterfaceState",
    "isPartOf": {
        "type": "Relationship",
        "object": "urn:ngsi-ld:Interface:telemetry-testbed-xrv9k-ceos-4hosts:r1:GigabitEthernet0/0/0/0",
        "observedAt": "2025-03-25T15:11:06.720497Z"
    },
    "name": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": "GigabitEthernet0/0/0/0",
        "observedAt": "2025-03-25T15:11:06.720497Z"
    },
    "stateType": {
        "type": "Relationship",
        "datasetId": "urn:ngsi-ld:operational",
        "object": "urn:ngsi-ld:YANGIdentity:iana-if-type:ethernetCsmacd",
        "observedAt": "2025-03-25T15:11:06.720497Z"
    },
    "mtu": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": 1500,
        "observedAt": "2025-03-25T15:11:06.720497Z"
    },
    "loopbackMode": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": false,
        "observedAt": "2025-03-25T15:11:06.720497Z"
    },
    "description": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": "Interface GigabitEthernet 0",
        "observedAt": "2025-03-25T15:11:06.720497Z"
    },
    "enabled": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": true,
        "observedAt": "2025-03-25T15:11:06.720497Z"
    },
    "ifindex": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": 6,
        "observedAt": "2025-03-25T15:11:06.720497Z"
    },
    "adminStatus": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": "UP",
        "observedAt": "2025-03-25T15:11:06.7204927Z"
    },
    "operStatus": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": "UP",
        "observedAt": "2025-03-25T15:11:06.720497Z"
    },
    "lastChange": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": 1742905693176368000,
        "observedAt": "2025-03-25T15:11:06.720497Z"
    }
}
```


Figure A.15: Operational data about network interface retrieved by gNMI protocol in NGSI-LD data format.

```
{
    "id": "urn:ngsi-ld:InterfaceConfig:telemetry-testbed-xrv9k-ceos-4hosts:r1:GigabitEthernet0/0/0/0",
    "type": "InterfaceConfig",
    "isPartOf": {
        "type": "Relationship",
        "object": "urn:ngsi-ld:Interface:telemetry-testbed-xrv9k-ceos-4hosts:r1:GigabitEthernet0/0/0/0",
        "observedAt": "2025-03-25T16:12:07.949665Z"
    },
    "name": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": "GigabitEthernet0/0/0/0",
        "observedAt": "2025-03-25T16:12:07.949665Z"
    },
    "configType": {
        "type": "Relationship",
        "datasetId": "urn:ngsi-ld:operational",
        "object": "urn:ngsi-ld:YANGIdentity:iana-if-type:ethernetCsmacd",
        "observedAt": "2025-03-25T16:12:07.949665Z"
    },
    "mtu": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": 1500,
        "observedAt": "2025-03-25T16:12:07.949665Z"
    },
    "description": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": "Interface GigabitEthernet 0",
        "observedAt": "2025-03-25T16:12:07.949665Z"
    }
}
```


Figure A.16: Configuration data about network interface retrieved by gNMI protocol in NGSI-LD data format.

On the other hand, Figure A.17 depicts a sample of a raw telemetry notification about network interface statistics counters retrieved by the gNMI protocol in JSON encoding format. Meanwhile, Figure A.18 depicts the corresponding representation in the NGSI-LD data format.


```
[
    {
        "name": "xrv9k_traffic_stats",
        "tags": {
            "interface_name": "GigabitEthernet0/0/0/0",
            "source": "clab-telemetry-testbed-xrv9k-ceos-4hosts-r1",
            "subscription-name": "xrv9k_traffic_stats"
        },
        "timestamp": 1709208539332762572,
        "values": {
            "openconfig:/interfaces/interface/state/counters/carrier-transitions": 0,
            "openconfig:/interfaces/interface/state/counters/in-broadcast-pkts": "82",
            "openconfig:/interfaces/interface/state/counters/in-discards": "9",
            "openconfig:/interfaces/interface/state/counters/in-errors": 0,
            "openconfig:/interfaces/interface/state/counters/in-fcs-errors": 0,
            "openconfig:/interfaces/interface/state/counters/in-multicast-pkts": "0",
            "openconfig:/interfaces/interface/state/counters/in-octets": "74779",
            "openconfig:/interfaces/interface/state/counters/in-pkts": "434",
            "openconfig:/interfaces/interface/state/counters/in-unicast-pkts": "352",
            "openconfig:/interfaces/interface/state/counters/in-unknown-protos": 0,
            "openconfig:/interfaces/interface/state/counters/out-broadcast-pkts": "1",
            "openconfig:/interfaces/interface/state/counters/out-discards": "0",
            "openconfig:/interfaces/interface/state/counters/out-errors": 0,
            "openconfig:/interfaces/interface/state/counters/out-multicast-pkts": "0",
            "openconfig:/interfaces/interface/state/counters/out-octets": "3402",
            "openconfig:/interfaces/interface/state/counters/out-pkts": "81",
            "openconfig:/interfaces/interface/state/counters/out-unicast-pkts": "80"
        }
    }
]
```


Figure A.17: Telemetry notification about network interface statistics counters retrieved by gNMI protocol in JSON encoding format.

```
{
    "id": "urn:ngsi-ld:InterfaceStateCounters:telemetry-testbed-xrv9k-ceos-4hosts:r1:GigabitEthernet0/0/0/0",
    "type": "InterfaceStateCounters",
    "isPartOf": {
        "type": "Relationship",
        "datasetId": "urn:ngsi-ld:operational",
        "object": "urn:ngsi-ld:InterfaceState:telemetry-testbed-xrv9k-ceos-4hosts:r1:GigabitEthernet0/0/0/0",
        "observedAt": "2024-02-29T12:08:59.332762"
    },
    "inOctets": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": 74779,
        "observedAt": "2024-02-29T12:08:59.332762"
    },
    "inUnicastPkts": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": 352,
        "observedAt": "2024-02-29T12:08:59.332762"
    },
    "inBroadcastPkts": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": 82,
        "observedAt": "2024-02-29T12:08:59.332762"
    },
    "inMulticastPkts": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": 0,
        "observedAt": "2024-02-29T12:08:59.332762"
    },
    "inDiscards": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": 9,
        "observedAt": "2024-02-29T12:08:59.332762"
    },
    "inErrors": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": 0,
        "observedAt": "2024-02-29T12:08:59.332762"
    },
    "inUnknownProtos": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": 0,
        "observedAt": "2024-02-29T12:08:59.332762"
    },
    "inFcsErrors": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": 0,
        "observedAt": "2024-02-29T12:08:59.332762"
    },
    "outOctets": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": 3402,
        "observedAt": "2024-02-29T12:08:59.332762"
    },
    "outUnicastPkts": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": 80,
        "observedAt": "2024-02-29T12:08:59.332762"
    },
    "outBroadcastPkts": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": 1,
        "observedAt": "2024-02-29T12:08:59.332762"
    },
    "outMulticastPkts": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": 0,
        "observedAt": "2024-02-29T12:08:59.332762"
    },
    "outDiscards": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": 0,
        "observedAt": "2024-02-29T12:08:59.332762"
    },
    "outErrors": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": 0,
        "observedAt": "2024-02-29T12:08:59.332762Z"
    },
    "carrierTransitions": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:operational",
        "value": 0,
        "observedAt": "22024-02-29T12:08:59.332762"
    }
}
```


Figure A.18: Telemetry notification about network interface statistics counters retrieved by gNMI protocol in NGSI-LD data format.

A.3 Handling YANG Configuration and Operational Data in Descriptive NDT solution

In a YANG data model, the YANG language specifies which data are meant for configuration and which data represent the operational state in a YANG server (e.g., a network device). To store these different types of data, YANG servers implement databases called YANG datastores. This annex provides an overview of how the data integration solution supported by the Descriptive NDT solution presented in this thesis allows handling the differentiation of configuration and operational data using the NGSI-LD standard. For this purpose, we distinguish between three types of YANG models: i) NMDA compliant IETF models; ii) non-NMDA compliant IETF models; and iii) OpenConfig models. The implementation of this feature is based on the details provided for mapping YANG configuration and operational state data to NGSI-LD in the ETSI GR CIM 038 document entitled *"Context Information Management (CIM); YANG and NGSI-LD interoperability"* [171].

A.3.1 Handling of NMDA compliant IETF models

As described in Chapter 2, Network Management Datastore Architecture (NMDA) [135] defines a unified architecture for defining YANG data models on servers, where YANG data models can have multiple instantiations in different datastores depending on the purpose of the data being either configuration or operational, and allowing network management protocols such as NETCONF to select the appropriate datastore.

Following the NMDA architecture, NGSI-LD can accommodate the representation of YANG data pertaining to different datastores for the same YANG data model. The NGSI-LD standard [81] specifies the *datasetId* subattribute as a mechanism to identify different values for the same NGSI-LD attribute depending on the data source. In this sense, the *datasetId* subattribute can be leveraged to identify the source datastore of the mapped YANG data node in its NGSI-LD representation. Thus, a specific *datasetId* can be used to differentiate the YANG configuration data from the YANG operational state data.

The mapping process between a YANG data model compatible with NMDA and NGSI-LD generates a single NGSI-LD model with the corresponding attributes. However, the NGSI-LD model may contain multiple instances of the same attribute with different *datasetId* values that identify the source YANG datastores. Figure A.19 shows a snapshot of the latest release of the *ietf-interfaces* YANG module [132], where the *enabled* YANG leaf maps to the *enabled* NGSI-LD Property. When interacting with this YANG leaf in the configuration datastore, there will be an instance of the *enabled* NGSI-LD property whose *datasetId* would point to "configuration", whereas when the interaction occurs in the YANG operational datastore, the *datasetId* of the *enabled* NGSI-LD Property instance would point to "operational". The left part of Figure A.19 shows the YANG tree representation [219] of the *ietf-interfaces* YANG module[A.1], while the right part depicts the *enabled* NGSI-LD Property with different values per *datasetId*.

[A.1] The YANG tree representation of the YANG module is generated by a plugin of the *pyang* [174] tool which provides quick visualization of the structure of YANG data models.

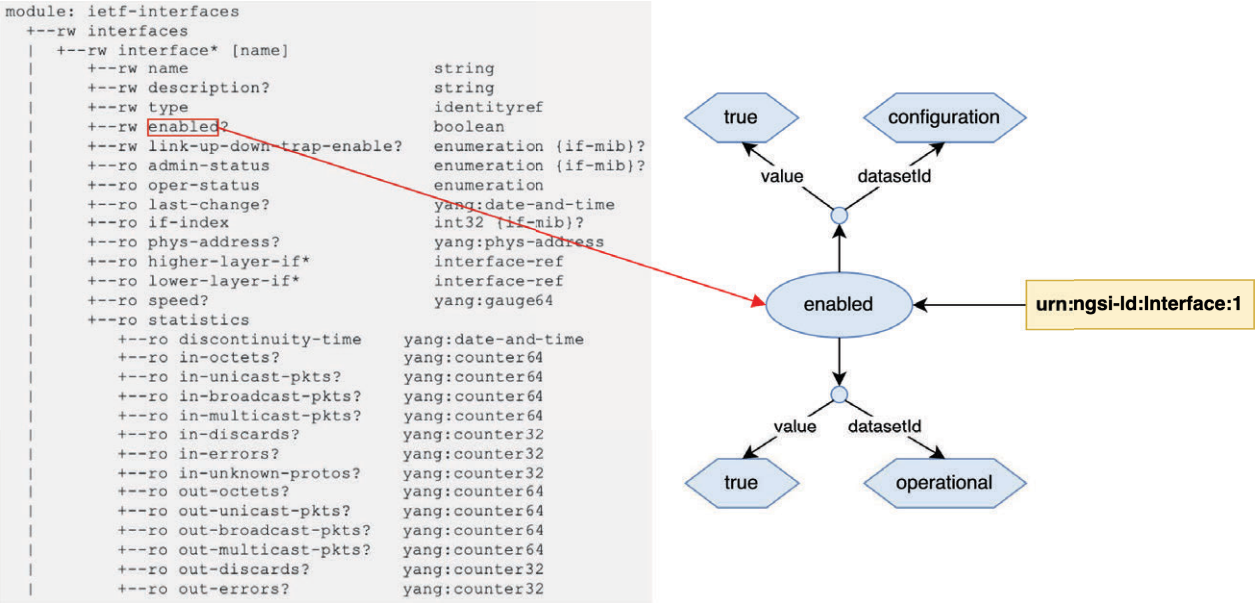


Figure A.19: Example of mapping NMDA-based IETF YANG data model to NGSI-LD.

A.3.2 Handling of non-NMDA compliant IETF models

Before NMDA came to the scene, network management protocols could not select a particular YANG datastore when interacting with network devices. To overcome this limitation, YANG data models would follow a split layout to identify the datastore containing the data. These data models are divided into two branches: one containing configuration data and the other containing operational data. Each branch begins with a YANG container, where the second is differentiated by the suffix "-state".

The process for mapping these YANG data models to NGSI-LD is similar to that described in Subsection A.3.1, but requires additional logic for parsing the split layout. The mapping process then generates two independent NGSI-LD entity types in the resulting NGSI-LD model, one for each branch, with their corresponding NGSI-LD attributes. In addition, the translation of each NGSI-LD attribute considers which branch of the YANG data model is used in order to identify the source YANG datastore, and therefore, the corresponding *datasetId*.

Figure A.20 depicts a fragment of the first release of the ietf-interfaces YANG module (i.e., revision 2014-05-08), which in this case follows a split layout wherein the *interfaces* and *interfaces-state* containers are placed at the root of the data model. The mapping to NGSI-LD will result into one *InterfaceConfig* NGSI-LD entity type for configuration and one *InterfaceState* NGSI-LD entity type for operational state. The right part of Figure A.20 depicts the *enabled* NGSI-LD Property related to an instance of a NGSI-LD entity of type typetextitInterfaceConfig where the *datasetId* "configuration" maps to */interfaces/interface/enabled* YANG leaf, and a *operStatus* NGSI-LD Property related to an instance of a NGSI-LD entity of type *InterfaceState* where the *datasetId* "operational" maps to */interfaces-state/interface/oper-status* YANG leaf.

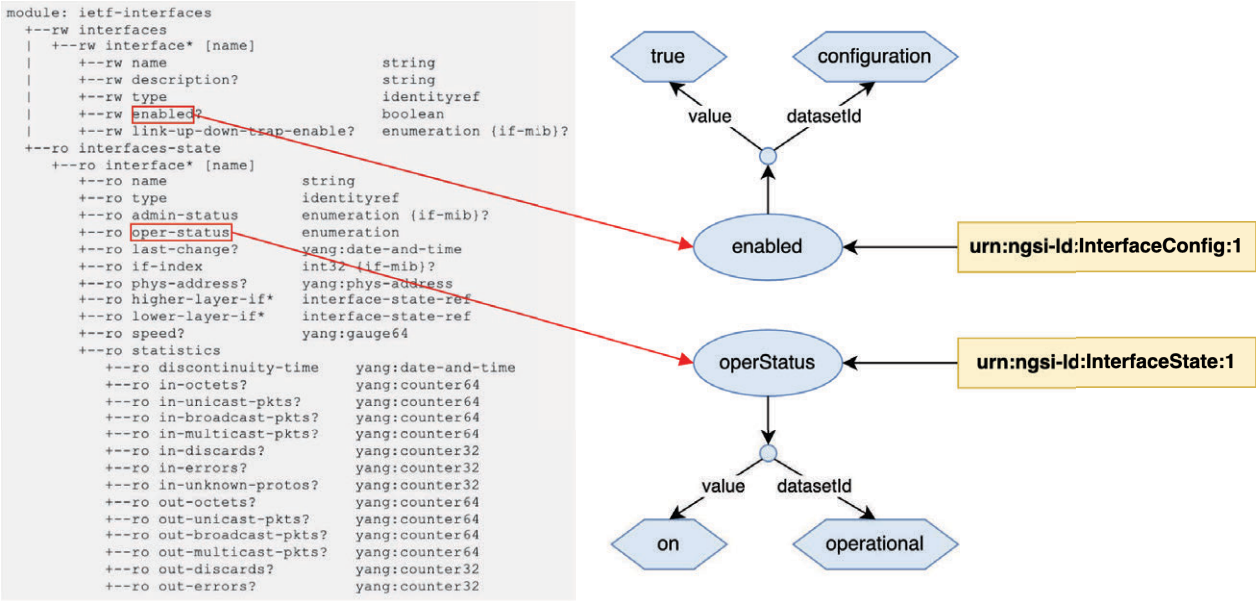


Figure A.20: Example of mapping non-NMDA IETF YANG data model to NGSI-LD.

The details for handling split YANG data models implemented in this solution differ from the definitions in the ETSI GR CIM 038 document in that the latter planned to have a unified target NGSI-LD model, where both the configuration branch and the operational branch would be merged into one NGSI-LD entity type. The resulting NGSI-LD type would then include all the resulting NGSI-LD attributes that are either for configuration or for operational state.

The implemented option of having NGSI-LD models with separate NGSI-LD entity types for configuration and operational state data was preferred here because it is aligned with the structure of the original YANG models, which makes the conversion between models more straightforward. In this case, the *datasetId* is a simple annotation indicating which data are configuration data and which data are operational. Although it may be redundant, this approach allows maintaining a stable implementation with respect to the solution proposed in Subsection A.3.1 for handling NMDA compliant IETF models.

A.3.3 Handling of OpenConfig models

OpenConfig data models follow a different structure from the structures followed in IETF data models. The main difference in OpenConfig YANG data models is that the YANG leaf and leaf-lists are mirrored in *config* and *state* containers. This approach is similar to the IETF's split-based structure, but in this case the configuration and operational state branches (i.e., *config* and *state* containers respectively) are modeled at every level of the data model. Additionally, when modeling YANG lists (e.g., the *interface* YANG list in the example shown in Figure A.21), OpenConfig data models include a reference to the YANG leaf that represents the key of YANG list in its configuration branch (e.g., in the example shown in Figure A.21 the *name* YANG leaf of the *interface* YANG list references the *name* YANG leaf of the *config* container that is the associated key of the *interface* YANG list).

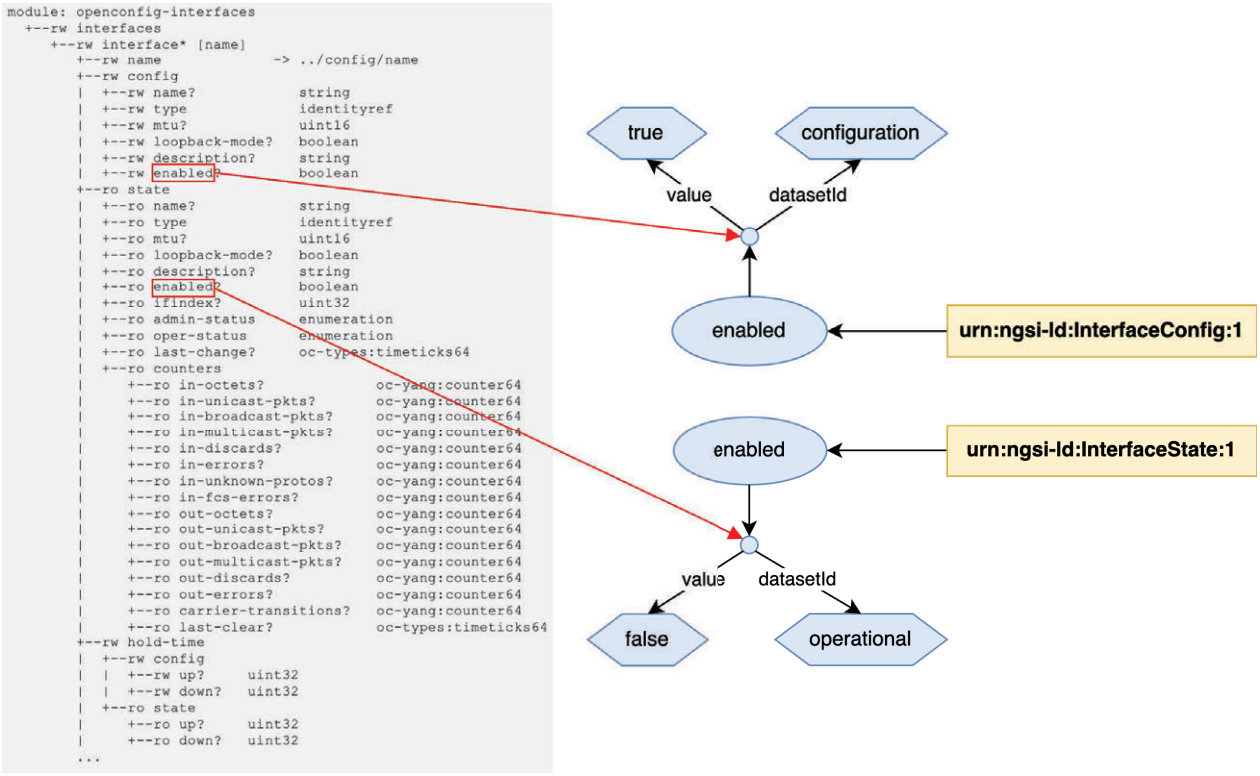


Figure A.21: Example of mapping OpenConfig YANG data model to NGSI-LD.

Figure A.21 depicts a fragment of the *openconfig-interfaces* YANG module [185] in the form of a YANG tree representation and an example mapping to NGSI-LD. The *config* and *state* containers are merged into two different *InterfaceConfig* and *InterfaceState* NGSI-LD entities with their particular instances of the *enabled* NGSI-LD property. The mapping process generates two independent NGSI-LD entity types in the resulting NGSI-LD model, one for each branch. However, in this particular case, because both the YANG *state* and *config* containers have an *enabled* YANG leaf, the mapped NGSI-LD property *enabled* is modeled twice. Therefore, the resulting *InterfaceConfig* NGSI-LD entity will include an *enabled* NGSI-LD property instance with datasetId "configuration" mapping to */interfaces/interface/config/enabled* YANG leaf, whereas the *InterfaceState* NGSI-LD entity will include an *enabled* NGSI-LD property instance with *datasetId* "operational" mapping to */interfaces/interface/state/enabled* YANG leaf.

As outlined for the case described in Subsection A.3.2, the details for handling split YANG data models implemented in this solution differ from the definitions in the ETSI GR CIM 038 document in that the latter planned to have a unified target NGSI-LD model, where both the configuration branch and the operational branch would be merged into one NGSI-LD entity type. Again, the implemented option of having NGSI-LD models with separate NGSI-LD entity types for configuration and operational state data was here preferred because it is aligned with the original YANG models, making the conversion between models more straightforward.

A.4 Examples of Query Processing Data Integration Workflows

This annex provides examples of how to process NGSI-LD API queries that are intended for model-driven network management purposes and to rewrite them as RPC queries. Although the methodology can be applied to any model-driven network protocol, it has been fully validated only for NETCONF, which will be used as an example application. The particular use case is for the management of network interfaces according to the *ietf-interfaces* YANG data model [132]. The following subsections include examples on how to handle each of the basic model-based network management operations: i) querying operational status data from a network device interface; ii) subscription to telemetry statistics from a network device interface; iii) querying configuration data from a network device interface; and iv) updating the configuration of a network device interface

These examples relate to the different clauses presented in Subsection 4.1.3 of Chapter 4.

A.4.1 Querying operational status data from a network device interface

Figure A.22 shows a complete example of the workflow used to process a query entity operation performed by a data consumer using the NGSI-LD API and rewrite it to a GET RPC operation. In this case, the operation is used to retrieve operational status data from a network interface using the NETCONF management protocol for network devices that do not support the NMDA architecture. When the data consumer specifies a query NGSI-LD entity operation with the entity type *Interface* and the entity *id* including *GigabitEthernet2*, the query processing solution finds that the YANG data node corresponding to the entity type *Interface* is a *list* a it has a key called *name*, resulting in an XPath */interfaces-state/interface[name='GigabitEthernet2']* as shown in the example in Figure A.22.

Meanwhile, Figure A.23 shows a complete example of the workflow but for network devices that support the NMDA architecture. In this case, the query NGSI-LD entity operation specifies the *datasetId* parameter to retrieve the operational state data of the network interface from the correct datastore of the target YANG server (see Subsection A.3.1 to get more details about how to manage NMDA compliant IETF models with the NGSI-LD standard).

A.4.2 Subscription to telemetry statistics from a network device interface

Figure A.24 shows a complete example of the workflow used to process a subscribe entity operation performed by a data consumer using the NGSI-LD API and to rewrite it to a SUBSCRIBE RPC operation. In such a case, the operation is used to receive notifications about telemetry statistics from a network interface using the NETCONF management protocol for network devices that do not support the NMDA architecture.

Meanwhile, Figure A.25 shows a complete example of the workflow, but for network devices that support the NMDA architecture.

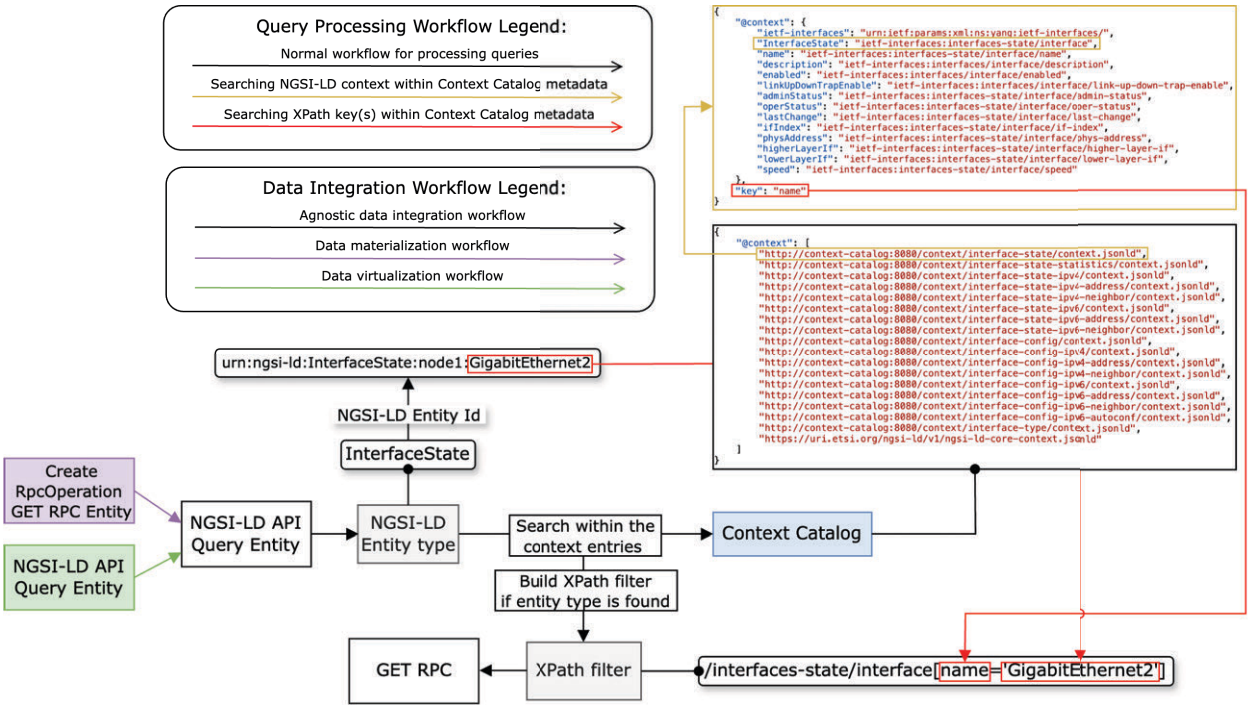


Figure A.22: Query processing and rewriting workflow for retrieving operational status data from a network interface using NETCONF for non-NMDA compliant network devices.

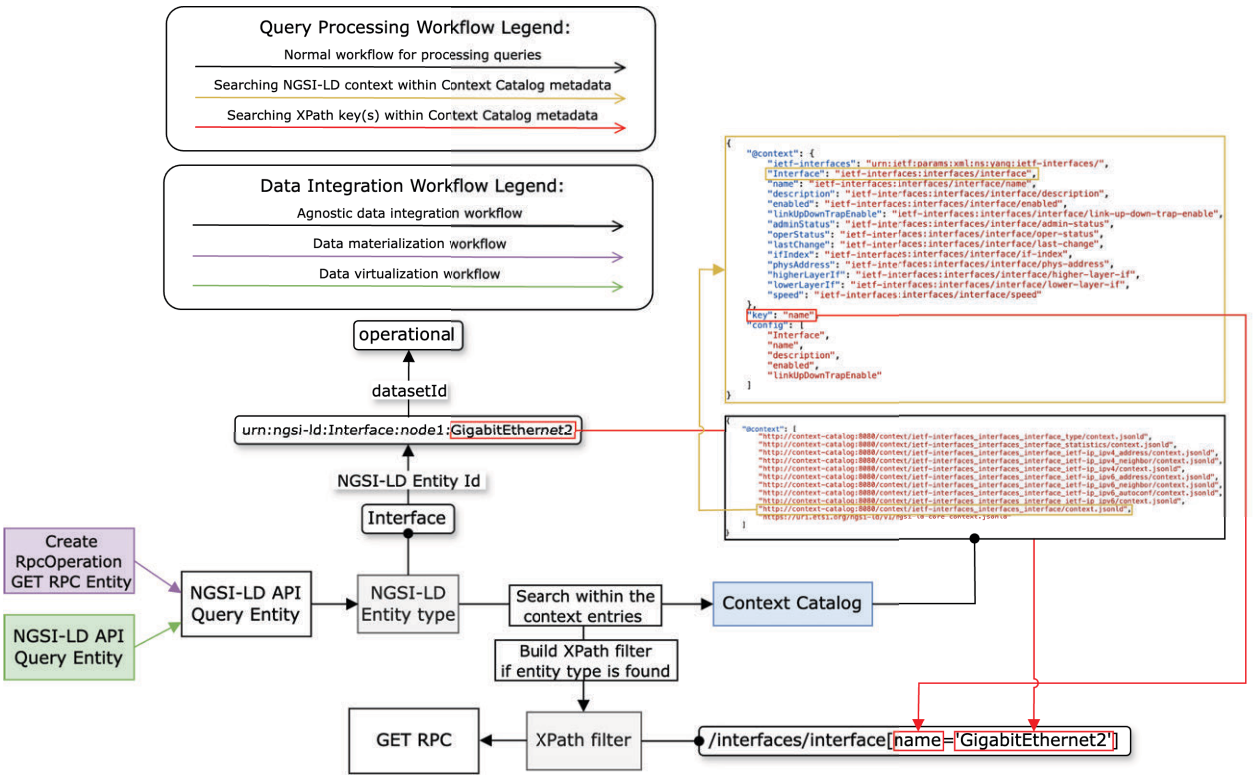


Figure A.23: Query processing and rewriting workflow for retrieving operational status data from a network interface using NETCONF for NMDA compliant network devices.

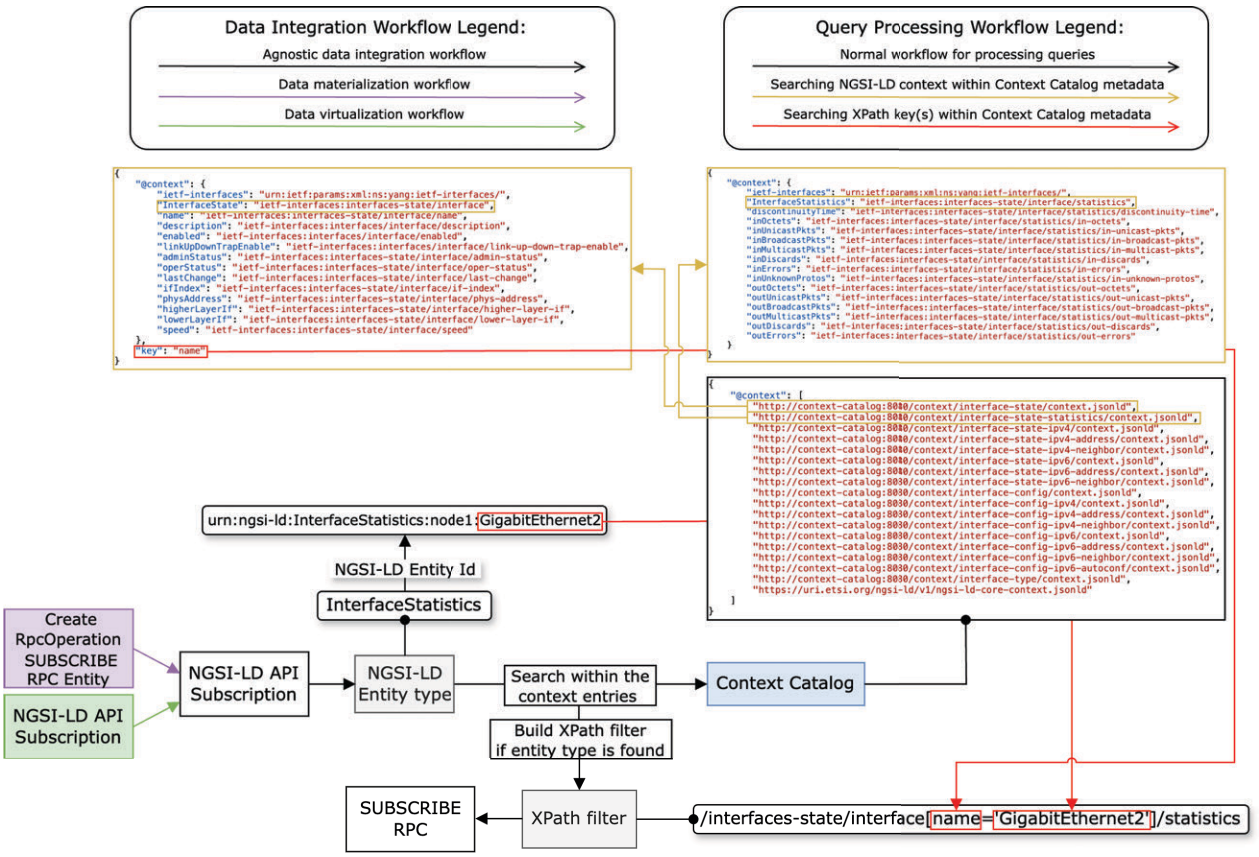


Figure A.24: Query processing and rewriting workflow for receiving notifications about telemetry statistics from a network interface using NETCONF for non-NMDA compliant network devices.

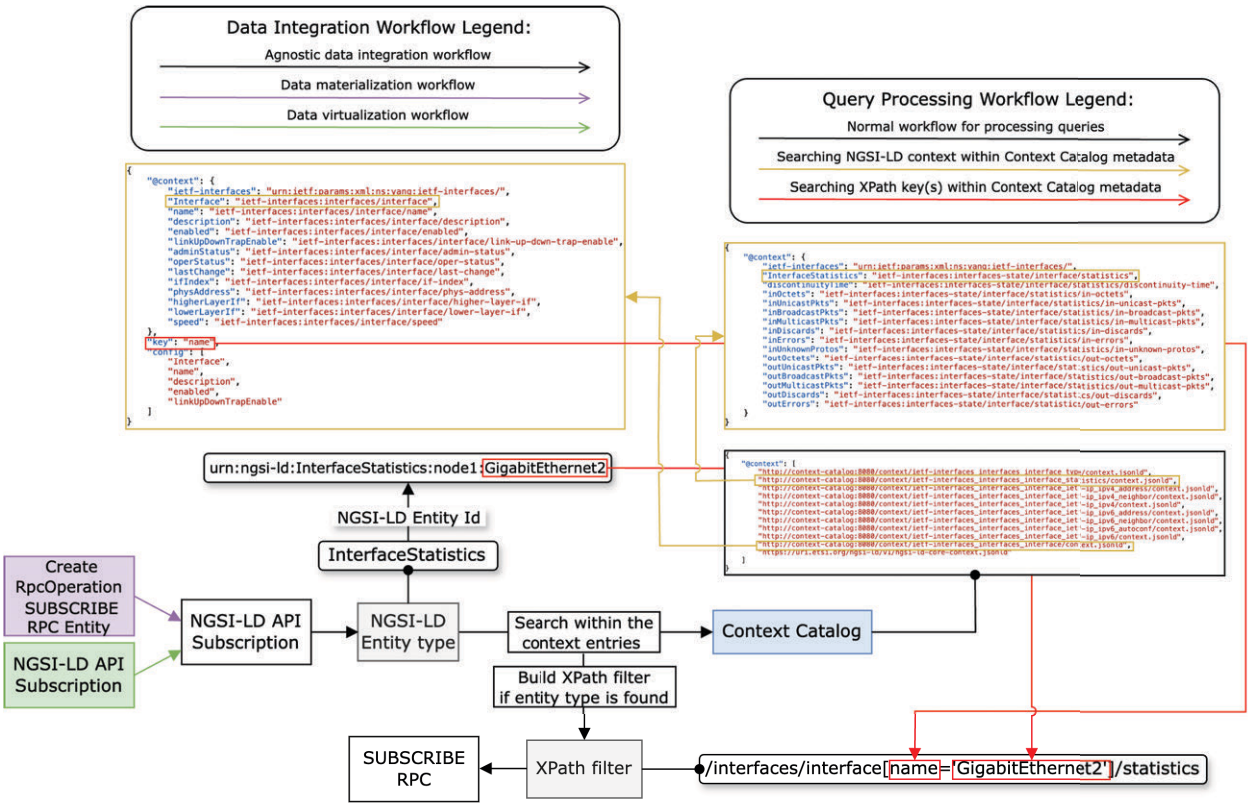


Figure A.25: Query processing and rewriting workflow for receiving notifications about telemetry statistics from a network interface using NETCONF for NMDA compliant network devices.

A.4.3 Querying configuration data from a network device

Figure A.26 shows a complete example of the workflow used to process a query entity operation performed by a data consumer using the NGSI-LD API and to rewrite it as a GET CONFIG RPC operation. In such a case, the operation is used to retrieve configuration data from a network interface using the NETCONF management protocol for network devices that do not support the NMDA architecture. When the data consumer specifies a query NGSI-LD entity operation with the entity type *InterfaceConfig* and the entity *id* including *GigabitEthernet2*, the query processing solution finds that the YANG data node corresponding to the entity type *Interface* is a *list* a it has a key called *name*, resulting in an XML subtree filter like the one shown in Figure A.26.

Meanwhile, Figure A.27 shows a complete example of the workflow but for network devices that support the NMDA architecture. In this case, the query NGSI-LD entity operation specifies the *datasetId* parameter to retrieve the configuration data of the network interface from the correct datastore of the target YANG server (see Subsection A.3.1 to get more details about how to manage NMDA compliant IETF models with the NGSI-LD standard).

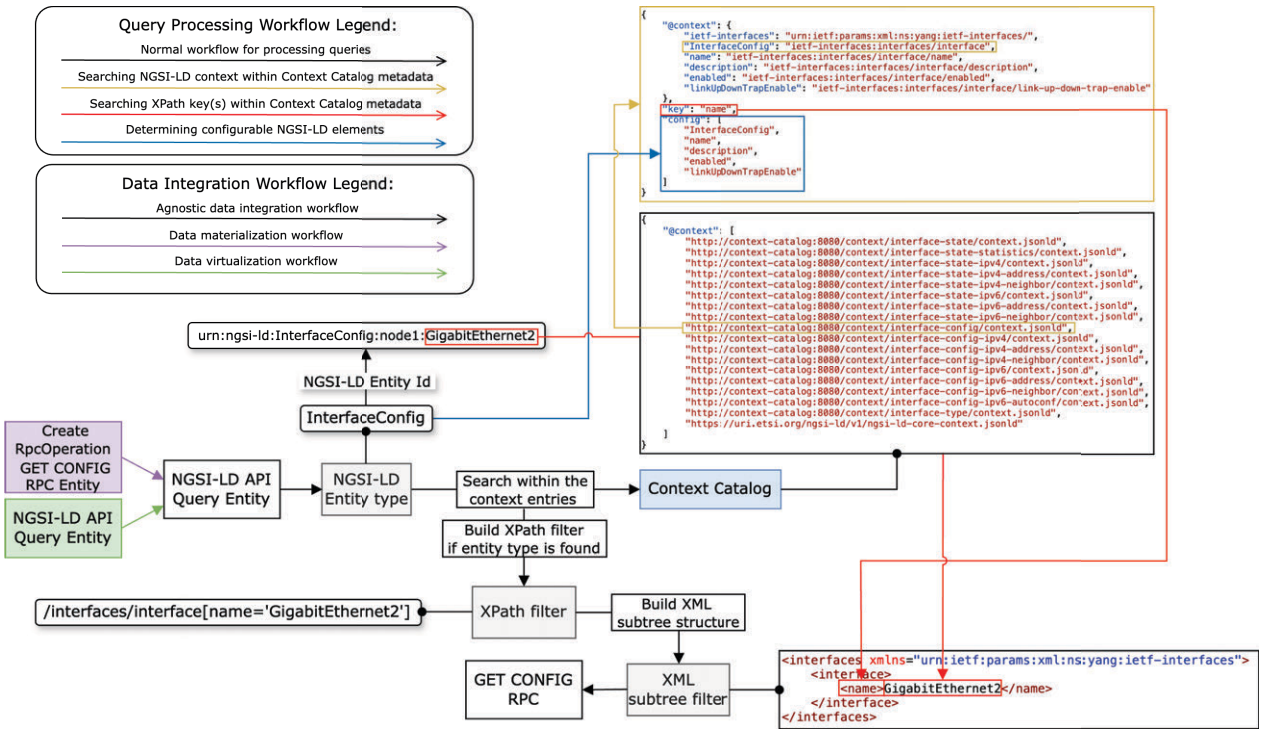


Figure A.26: Query processing and rewriting workflow for retrieving configuration data from a network interface using NETCONF for non-NMDA compliant network devices.

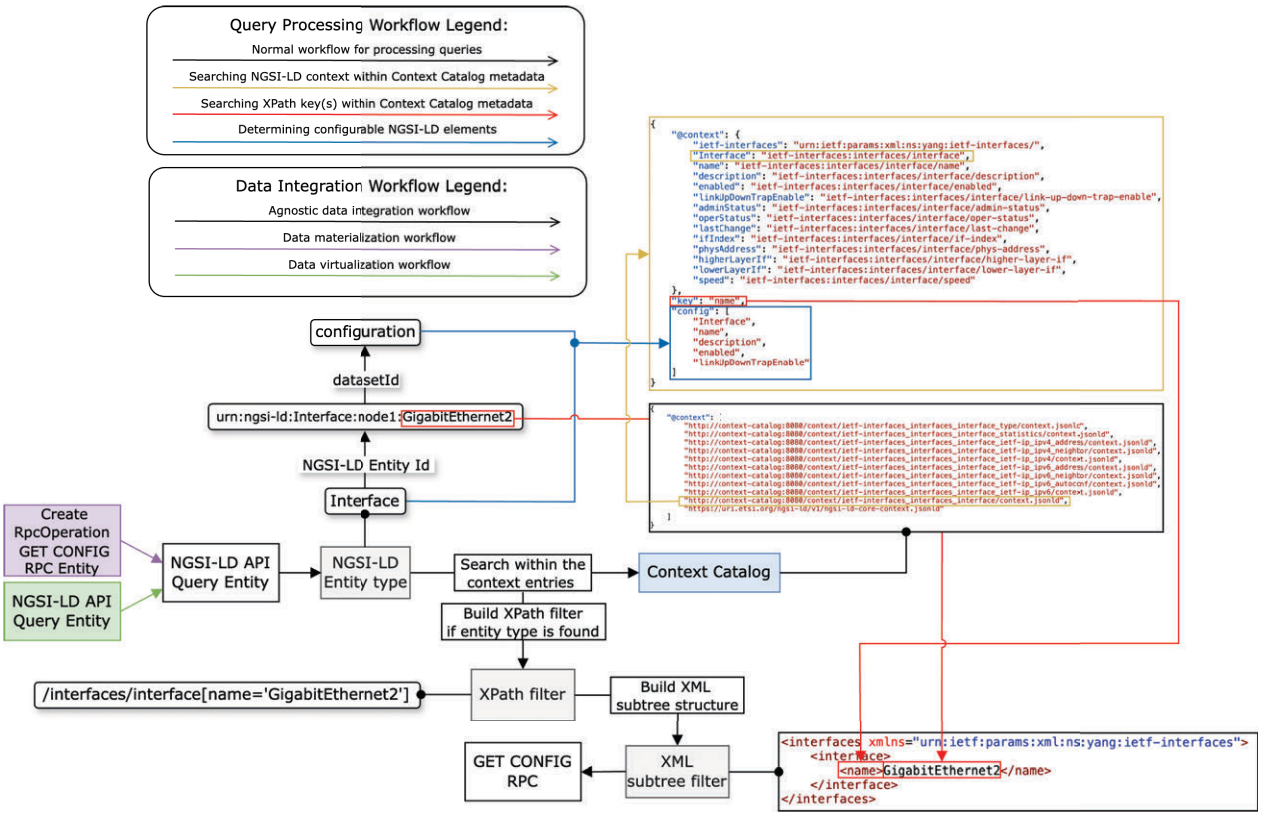


Figure A.27: Query processing and rewriting workflow for retrieving configuration data from a network interface using NETCONF for NMDA compliant network devices.

A.4.4 Updating the configuration of a network device interface

Using a data materialization approach, Figure A.28 shows a complete example of the workflow used to process an update entity operation performed by a data consumer using the NGSI-LD API and to rewrite it to a SET RPC operation. In this case, the operation is used to update the configuration of a network interface using the NETCONF management protocol for network devices that do not support the NMDA architecture. In the example shown in Figure A.28, an NGSI-LD entity of type *InterfaceConfig* is updated to configure a *GigabitEthernet2* interface within the network device that is the target YANG server. Meanwhile, also using a data materialization approach, Figure A.29 shows a complete example of the workflow, but for network devices that support the NMDA architecture.

Using a data virtualization approach, Figure A.30 shows a complete example of the workflow used to process a create entity operation performed by a data consumer using the NGSI-LD API and to rewrite it as a SET RPC operation. In such a case, the operation is used to update the configuration of a network interface using the NETCONF management protocol for network devices that do not support the NMDA architecture. In the example shown in Figure A.30, an NGSI-LD entity of type *Interface* is updated to configure a *GigabitEthernet2* interface within the network device that is the target YANG server. Meanwhile, also using a data virtualization approach, Figure A.31 shows a complete example of the workflow, but for network devices that support the NMDA architecture.

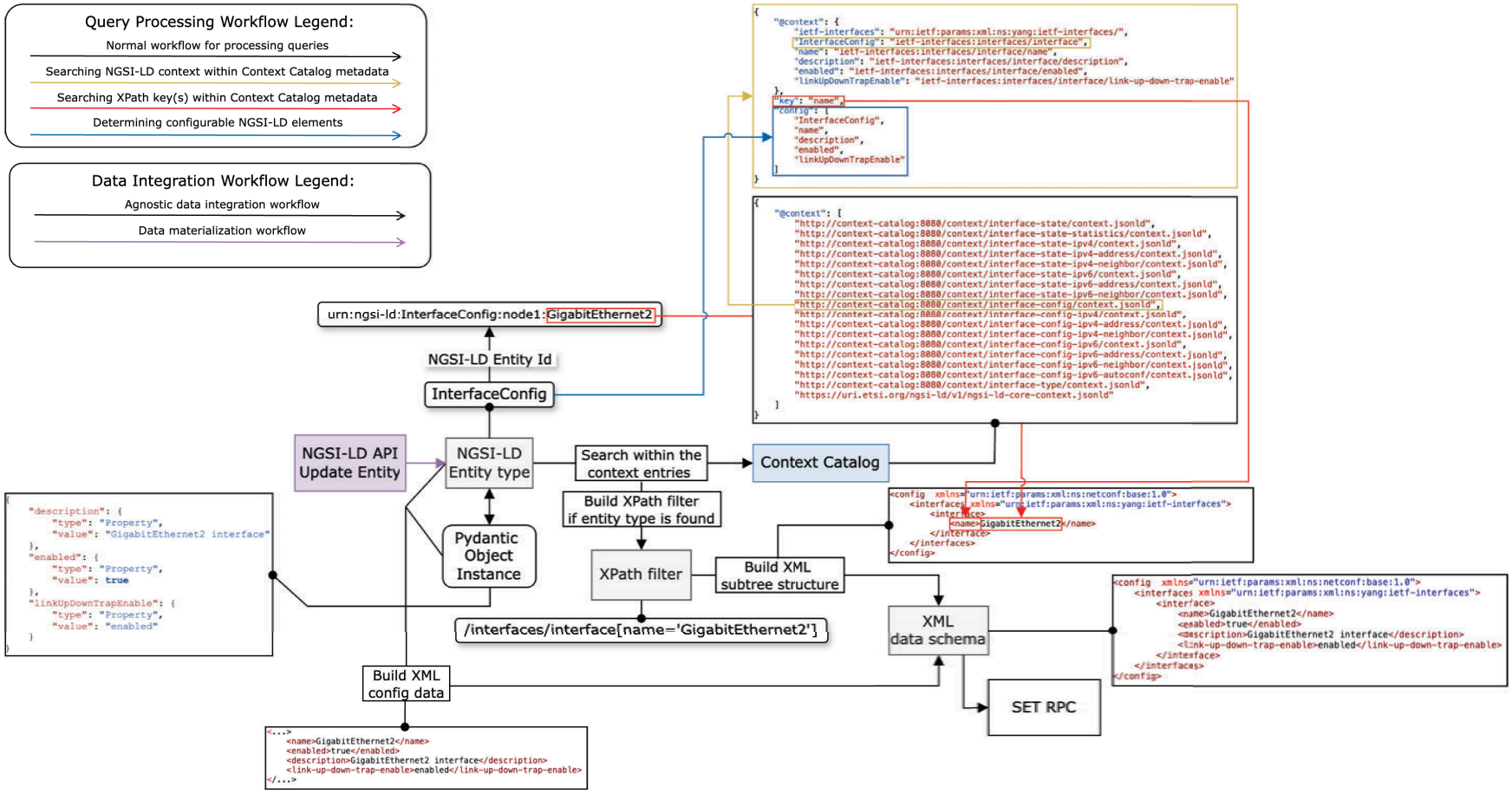


Figure A.28: Query processing and rewriting workflow for updating the configuration of a network interface using NETCONF for non-NMDA compliant network devices (data materialization).

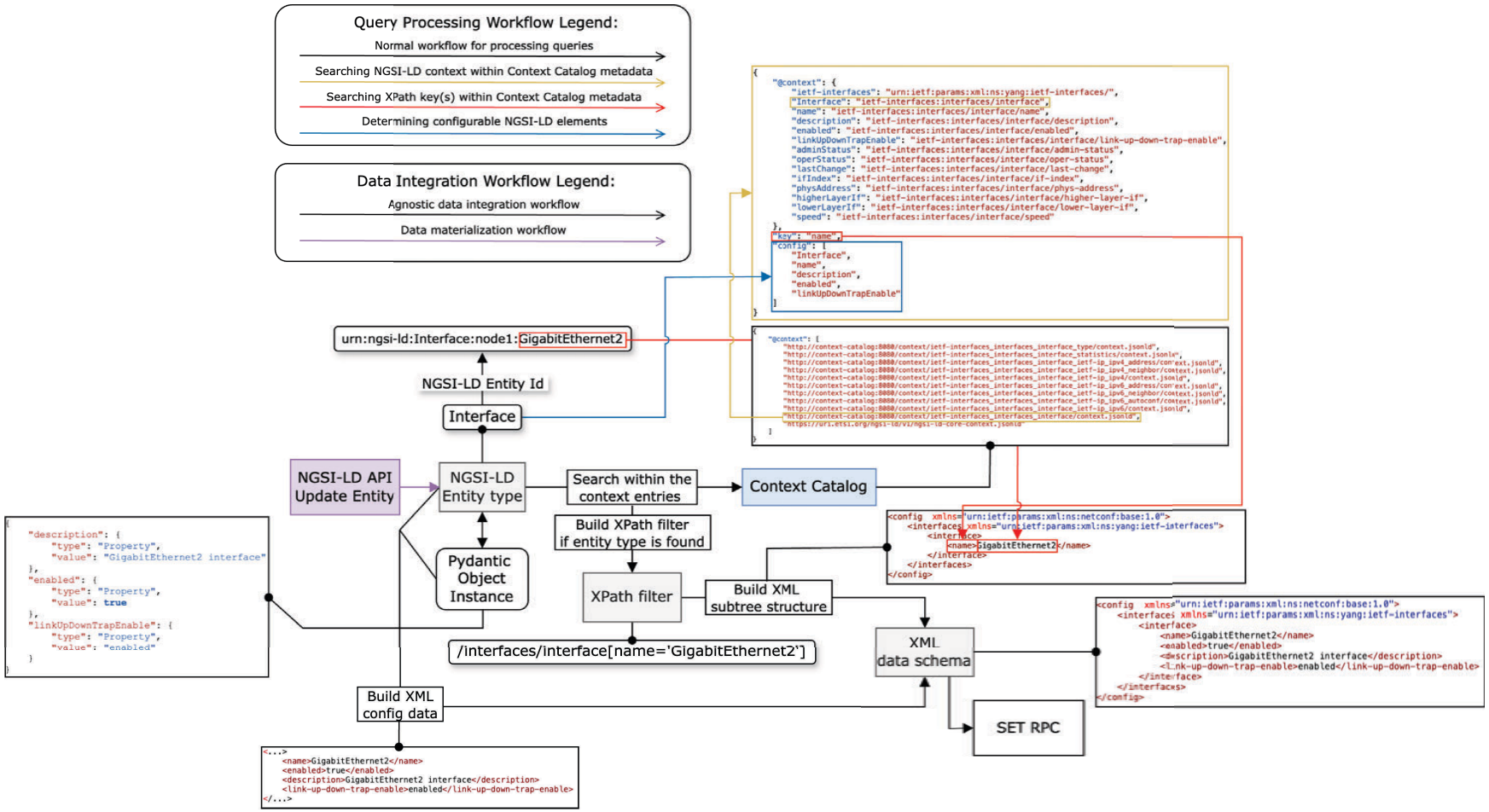


Figure A.29: Query processing and rewriting workflow for updating the configuration of a network interface using NETCONF for NMDA compliant network devices (data materialization).

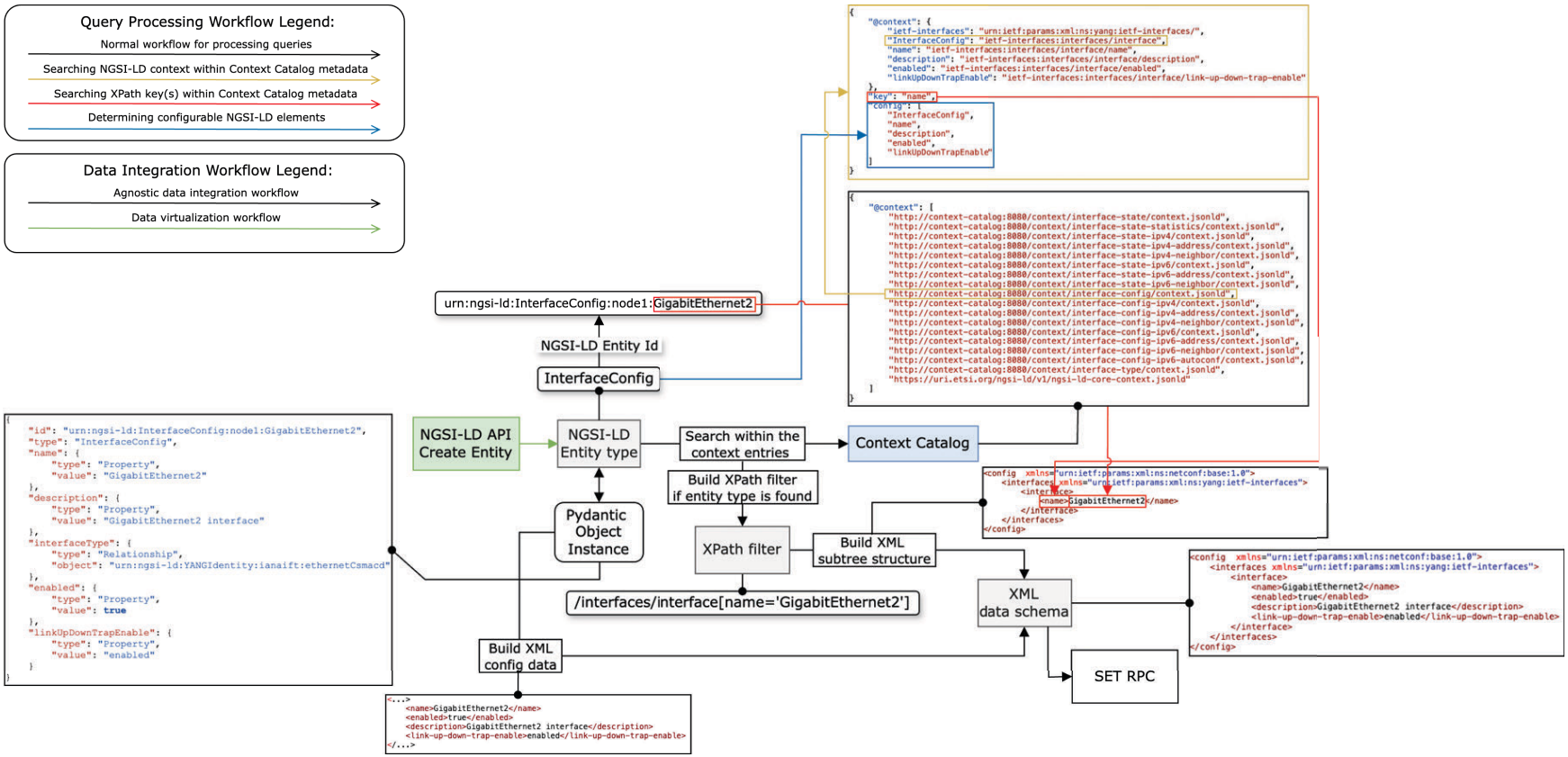


Figure A.30: Query processing and rewriting workflow for updating the configuration of a network interface using NETCONF for non-NMDA compliant network devices (data virtualization).

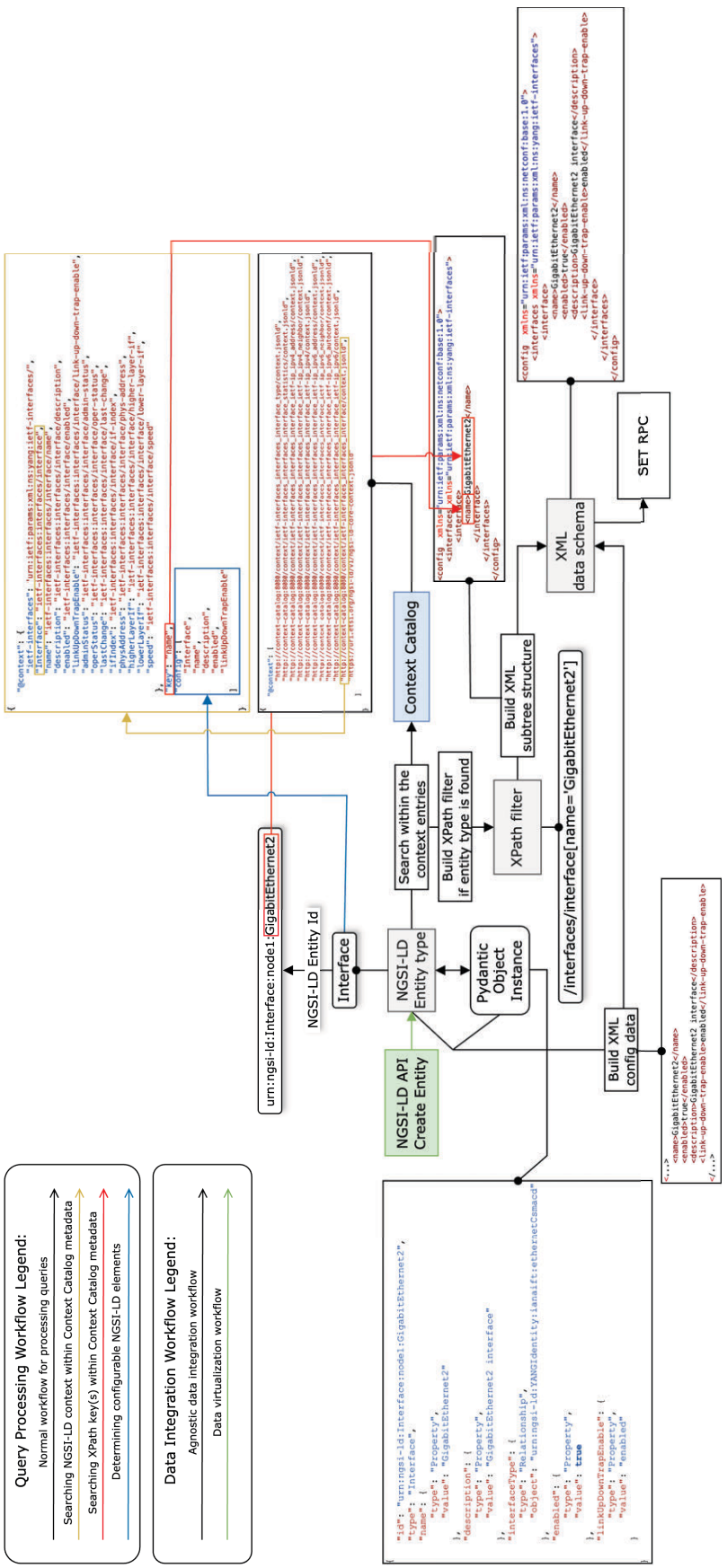


Figure A.31: Query processing and rewriting workflow for updating the configuration of a network interface using NETCONF for NMDA compliant network devices (data virtualization).

A.5 Open-Source Software Development Activity

This section highlights some of the software development activities that are related to the context of the work done during this Doctoral Thesis. All of this software is publicly available on the Internet, which allows its use by the open-source community.

A.5.1 Prototype implementation of Descriptive NDT

There is an open-source contribution with source code, artifacts, and documentation of the Descriptive NDT solution proposed in this Doctoral Thesis[A.2]. The repository contains the prototype implementation details of the Descriptive NDT system to model the management and operation of networks. The current solution allows the characterization of network capabilities in terms of configuration, operational status, and telemetry data using the NGSI-LD standard of ETSI ISG CIM, and focuses on real network devices that support management protocols based on the YANG data modeling language.

A.5.2 Semantic Data Aggregator and CANDIL Data Fabric

Within the context of the work in different European research projects, the monitoring data aggregation framework named Semantic Data Aggregator (SDA) was developed. Its main objective has been the integration of heterogeneous data sources and consumers considered within the network scope and the coordination of data exchange between them. The SDA solution implements a monitoring system based on the NGSI-LD standard defined by the ETSI ISG CIM to model the information that characterizes the different data sources and consumers, as well as the orchestration of services to preprocess and apply transformations to the data collected from the sources and for the exchange of network monitoring data between sources and consumers. Furthermore, this tool was designed primarily to be applied to data sources with support for telemetry mechanisms based on YANG data models, to facilitate efficient and automated network management. The SDA is publicly available as an open-source contribution on GitHub[A.3]. The proposal is evolving towards a new version called CANDIL (Context-Aware Network Data Integration Loom)[A.4], which is also being used in current European research projects. CANDIL covers the standard integration and characterization of heterogeneous data sources, also related to the network management domain, where the management of context information based on a Data Fabric paradigm has a greater weight, as well as the use of ontologies and the application of Knowledge Graphs.

The work developed in these software initiatives has served as a fundamental path and for the conceptualization of ideas in the scope of automation for managing and operating networks, on which this Doctoral Thesis is based.

[A.2] YANG to NGSI-LD for Descriptive NDT. Available at: https://github.com/giros-dit/yang-to-ngsi-ld

[A.3] Semantic Data Aggregator: A semantic monitoring framework for aggregating data from heterogeneous sources. Available at: https://github.com/giros-dit/semantic-data-aggregator

[A.4] CANDIL Data Fabric. Available at: https://github.com/candil-data-fabric

A.5.3 OpenAPI Specification for NGSI-LD and artifacts

We worked on the development of the OpenAPI Specification for the NGSI-LD standard in the ETSI Specialist Task Force (STF) 676 project on Usage Domains Mapping: "OpenAPI Specification for NGSI-LD API". This OpenAPI Specification is available in a repository of the ETSI ISG CIM GitLab organization within the official ETSI Forge[A.5].

This OpenAPI Specification is one of the key components of the NDT solution proposed in this Doctoral Thesis, allowing the generation of stub programming code to model and orchestrate the NGSI-LD API. Specifically, there is a Python client library for the NGSI-LD API, which is generated from the OpenAPI Specification and is publicly available on GitHub[A.6]. In addition, a functional and unit debugging test bench has been developed for this Python library client, which is also publicly available on GitHub[A.7].

A.5.4 Containerlab testbeds for analyzing network telemetry and management mechanisms based on YANG

An important challenge in validating the Descriptive NDT solution proposed in this Doctoral Thesis is to find a realistic network scenario that fits the operational and management requirements defined in the NDT architecture.

Although network automation based on YANG data modeling is widespread in the industry, it is difficult to find network devices that fully implement the capabilities offered by model-driven network management protocols such as NETCONF or gNMI. However, network vendors such as Cisco, Nokia, Juniper, or Arista Networks provide access to the network operating system of their own network device models that support YANG capabilities, either as open-source or through a purchased software license. The software distribution of these network operating systems enables installing and virtualizing the functionality of these network devices in virtual machines or using lightweight container-based virtualization running on top of general-purpose hardware devices based on Linux servers.

In this sense, there are network virtualization environments that facilitate the configuration and deployment of these virtualized network devices to define networking scenarios. For example, Containerlab [199] is a network virtualization tool that enables the specification of container-based networking scenarios consisting of such network devices with network operating systems of different vendors and models. In fact, Containerlab is the network virtualization tool leveraged to configure and deploy networking scenarios for analyzing network telemetry and management mechanisms based on the YANG data modeling and the NETCONF and gNMI protocols. In addition, Containerlab was used on the experimental testbed to evaluate the performance of the NDT solution proposed in this thesis, as detailed in Section 5.1 of Chapter 5.

[A.5] NGSI-LD OpenAPI Specification (NGSI-LD OAS). Available at: https://forge.etsi.org/rep/cim/ngsi-ld-openapi

[A.6] Python-based NGSI-LD API client. Available at: https://github.com/giros-dit/python-ngsi-ld-client

[A.7] Python-based NGSI-LD API client tester. Available at: https://github.com/giros-dit/ngsi-ld-client-tester

To validate the Descriptive NDT system for network operations and management purposes, several proof-of-concept Containerlab scenarios have been developed using network devices from different vendors. These are publicly available on GitHub[A.8,A.9].

[A.8] Containerlab testbeds for studying and analyzing telemetry services over YANG. Available at: https://github.com/giros-dit/containerlab-telemetry-testbed

[A.9] The "YANG to NGSI-LD for Descriptive NDT" repository contains the definition of different Containerlab scenarios for validating the Descriptive NDT functionality. Available at: https://github.com/giros-dit/yang-to-ngsi-ld