SEASON

Self-Managed Sustainable High-Capacity Optical Networks

This project is supported by the SNS Joint Undertaken through the European Union's Horizon RIA research and innovation programme under grant agreement No. 101096120

Deliverable D4.1

First design and implementation of control plane infrastructure

Editor Filippo Cugini (CNIT), Vignesh Karunakaran (Adtran), Achim Autenrieth (Adtran)

Contributors ADTRAN, CTTC, CNIT, UPC, WINGS, TID, TIM, WEST, ACC

Version 1.0

Date March 06, 2024

Distribution PU

DISCLAIMER

REVISION HISTORY

Revision	Date	Responsible	Comment
0.1	*December, 20, 2023*	*CNIT, Adtran*	*Initial ToC version.*
0.2	*January, 14, 2024*	*CNIT, Adtran*	*Full version for review.*
0.3	*January, 06, 2024*	*CNIT, Adtran*	*Review version.*
1.0	*March, 06, 2024*	*CNIT, Adtran*	*Final version for submission.*

LIST OF AUTHORS

Partner	Name Surname
Adtran (ADVA)	*Achim Autenrieth*
Adtran (ADVA)	*Vignesh Karunakaran*
CNIT	*Filippo Cugini*
CNIT	*Kyriakos Vlachos*
CNIT	*Andrea Sgambelluri*
CNIT	*Mohammed Ismaeel*
CTTC	*Ramon Casellas*
CTTC	*Ricardo Martínez*
CTTC	*Carlos Efrén Hernández Chulde*
CTTC	*Josep Maria Fàbrega*
UPC	*Luis Velasco*
UPC	*Marc Ruiz*
UPC	*Jaume Comellas*
UPC	*Sima Barzegar*
UPC	*Josep Prat*
UPC	*Josep Vidal*
UPC	*Sadegh Ghasrizadeh*
WINGS	*Marios Lagos*
WINGS	*Vasilis Tsekenis*
WINGS	*Sokratis Barmpounakis*
ACC	*Revaz Berozashvili*
TID	*Oscar Gonzalez*
TID	*Pablo Armingol*
WEST	*Carlo Centofanti*
WEST	*Andrea Marotta*
WEST	*Stefano Tennina*
TIM	*Roberto Morro*

EXECUTIVE SUMMARY

This deliverable reports the first design and implementation of the SEASON control plane infrastructure. In particular, it contains the first results of all SEASON WP4 tasks, reporting on monitoring/telemetry, single and multi-domain control solutions, and artificial intelligence / machine learning (AI/ML)-based self-management. This report has been organized as follows.

Section 2 provides an overview of WP4 objectives and relate KPIs, while Section 3 an overview of SEASON control plane architectures. In particular, Section 3 portrays the key network components in each segment of the SEASON reference transport network from the radio access network (RAN) to the core segment and, on top of it, the associated control plane technologies.

Section 4 details the monitoring, streaming telemetry and intelligent data aggregation in SEASON. In essence, a comprehensive monitoring infrastructure able to retrieve and intelligently aggregate telemetry data/metadata has been designed and preliminary implemented. Section 4 presents telemetry network analytics and network elements as telemetry data producers. The network elements in the data plane possess the capability to expose the configuration and operational data to reveal the current state of the device and the provisioned channels. Finally, Section 4 presents the algorithms for intelligent data aggregation. The main target was to design a solution that is able to deal with the 5 V's (volume, velocity, variety, veracity, and value) of telemetry data, while providing scalability, efficiency, flexibility, easy integration of different data sources with a variety of measurements and events, and facility for turning data into useful insight that can be used for network automation.

Section 5 presents the control architecture of access infrastructures. SEASON targets the software defining networking (SDN) control of Passive Optical Networks (PON) combining space division multiplexing (SDM) switching, i.e., SDM-PON. The management of SEASON PON architecture involves the control of optical devices like the Optical Line Terminal (OLT) and spatial devices such as spatial aggregation/disaggregation elements. Towards this, SDM is exploited through the aggregation/disaggregation of spatial channels which may be deployed as single core fibers in multi-fiber scenarios of cores in multi-core fibers.

Section 6 presents the innovative SEASON solution for transport network management and control. The transport network management involves the control of optical and packet/optical devices in the underlying data plane. Work focuses on model-driven approach, and this depends on the YANG model adapted by the optical or packet/optical entities. Among the optical domain components, OpenConfig is generally used for transceivers and packet/optical devices while OpenROADM YANG models for ROADMs. However, the scope of the work also considers evaluation of OpenROADM to control pluggables in packet/optical devices.

Section 7 presents the innovative digital twin concepts for optical transport networking. This requires the creation of a virtual representation of optical network elements (NEs) that can configure, monitor, and replicate the real equipment behavior. Section 7 presents a solution for the digital twin to accurately represent the behavior of the network, allowing for dynamic service configuration and management of the network. Its evaluation is performed between the following platforms: (a) an OpenROADM-based network simulator, (b) a vendor-based network

emulator, and (c) an OpenROADM or a vendor-based optical network digital twin. In addition, Section 7 also describes an Optical Time Domain Digital Twin for network automation applications, such as quality of transmission (QoT) estimation and failure management. ML can be employed to facilitate their operation. For implementing the Digital Twin, models for QoT estimation and algorithms for failure management are provided.

Finally, Section 8 discusses the use of novel AI/ML algorithms in support of service management and orchestration. AI/ML Service Orchestration is the process of integrating AI/ML capabilities into the orchestration layer of network management to automate and optimize service delivery across different network domains. By leveraging such techniques, namely Support Vector Machines (SVM), Deep Neural Networks (DNN), etc., AI/ML models ingest operational data, understand complex patterns, make predictions, and take actions in near real-time.

AI/ML Service Orchestration in the SEASON project represents a transformative approach in network management, integrating AI/ML capabilities across RAN, transport, optical, and core networks to automate and optimize service delivery. Section 8 presents AI/ML employment algorithms in B5G networks AI/ML for service orchestration and application placement, in optical network for energy control and for orchestrating computing, and network resources across multiple technology layers for online connectivity service provisioning.

TABLE OF CONTENTS

1 INTRODUCTION 8
2 WP4 scope and objectives 9
2.1 WP4 Scope 9
2.2 Relevant Project Objectives 9
2.3 WP4 Objectives 10
2.3.1 OBJ 5 - Develop a pervasive monitoring infrastructure for secure and truly self-managed networking 11
2.3.2 OBJ 7 - Control plane, Monitoring and streaming telemetry 11
3 Overview of SEASON Control plane architecture for self-managed and autonomous networking 13
3.1 Overall SEASON Solution – Control Plane 13
3.2 SDN based infrastructure configuration and control Architecture 15
3.2.1 Teraflow SDN and North Bound Interface 17
4 Monitoring, Streaming Telemetry and Intelligent Data Aggregation 19
4.1 Monitoring and Streaming Telemetry 19
4.2 Network analytics pipeline 21
4.3 Optical Network Element Telemetry Data Producer 23
4.3.1 SmartNIC/DPU as both Telemetry Data Producer and Consumer 26
4.4 SDN Controller as Telemetry Data Producer 28
4.4.1 TR-548 TAPI streaming 30
4.4.2 RAN / RIC as Telemetry Data Produce 31
4.4.3 RIC as Telemetry Data Consumer 32
4.5 Intelligent Data Aggregation 32
5 Control of Access infrastructures 38
5.1 Control of SDM PON 38
5.1.1 PON Infrastructure Control 38
5.1.2 Design 38
5.1.3 Data Plane capabilities 39
5.1.4 Implementation 39
5.2 RAN Intelligent Controller (RIC) 41
5.2.1 Architecture 41
5.2.2 ORAN Evolution 42
6 Transport Network Control 44
6.1 Infrastructure Control for MBoSDM optical network 44
6.1.1 Design 44
6.1.2 Device Model 45

6.1.3 Data Plane node capabilities 46
6.1.4 Implementation 47
6.1.5 Simple 4-node scenario 47
6.2 Packet / IP/optical control (smartNICs, white box switch) 49
6.3 Infrastructure Control for Transceivers 50
6.4 NetDevOps and Continuous Integration / Continuous development 51
6.5 TeraFlow SDN Controller and Orchestration 53
6.5.1 TFS Controller for IPoWDM 54
6.5.2 TFS Orchestrator 56
6.6 Centralized and Distributed Approaches to Control Optical Point-to-Multipoint Systems Near-Real-Time 58
6.7 Control of DSCM Systems 64
6.8 Integration of Computing and Networking 65
7 Digital Twin 67
7.1 Optical Network Digital Twin 67
7.2 Optical Time Domain Digital Twin and Applications 71
7.3 Extending the OCATA Digital Twin for Digital Subcarrier Multiplexing 80
8 AI/ML in support of service management and orchestration 82
8.1 Coordination between RAN slices and Network Transport 83
8.2 AI/ML Service Orchestration 89
8.2.1 AI/ML-assisted Control for service orchestration and application placement 91
8.3 AI/ML-assisted Control for energy efficient Optical Networks 93
8.4 GNN and DRL for Online Connectivity Services 100
8.5 Privacy Preserving Digital Twin Knowledge Sharing for Multi-domain Networks 104
9 GLOSSARY 107
10 REFERENCES 110

1 INTRODUCTION

The document provides an initial report on the overall SEASON control plane architecture, proposed solutions, and use case scenarios. The WP4 in SEASON steers control of transport network end-to-end from RAN till core, defining a hierarchical control plane architecture. This first report covers the design and development of a distributed software system for the monitoring, streaming telemetry, control and orchestration of MB-over-SDM network; and the overall management of operational and user services covering PON, RIC and transport.

Advanced monitoring and streaming telemetry provide the data for local and centralized control actions and operations. The control system enables configuration of network infrastructure, closed loop domain control and automation, integrated end-to-end network operation, and AI/ML based service orchestration and network simulation based on digital twin. Centralized SDN control is e augmented by novel Continuous Integration and Continuous Delivery (CD/CI) paradigms and tools for networks infrastructure configuration and control to allow incremental network change management with configuration change management, automated testing and rollback in case of errors.

In essence, this deliverable details the following:

- the monitoring, streaming telemetry solution for SEASON (data producers, collectors and network analytics);
- algorithms for intelligent data aggregation for telemetry applications;
- the control architecture of access infrastructures through SDN for the management of optical devices like the Optical Line Terminal (OLT) as well as spatial devices such as spatial aggregation/disaggregation elements;
- solution for transport network management and control of the optical and packet/optical devices in the underlying data plane;
- Digital twinning applications for:
    - optical transport networking that can configure, monitor, and replicate the real equipment behavior allowing for its dynamic service configuration and management;
    - network automation applications such as, for example quality of transmission (QoT) estimation and failure management.
- AI/ML algorithms in support of service management and orchestration in:
    - B5G networks AI/ML for service orchestration and application placement;
    - optical network for energy control;
    - different technology layers for online connectivity service provisioning.

2 WP4 SCOPE AND OBJECTIVES

2.1 WP4 SCOPE

WP4 covers the design and development of a distributed software system for the monitoring, streaming telemetry, control and orchestration of MB-over-SDM network and the overall management of operational and user services covering PON, RIC and transport. Advanced monitoring and streaming telemetry provide the data for local and centralized control actions and operations. The control system enables configuration of network infrastructure, closed loop domain control and automation, integrated end-to-end network operation, and AI/ML based service orchestration and network simulation based on digital twin.

Centralized SDN control is augmented by novel Continuous Integration and Continuous Delivery (CD/CI) paradigms and tools for networks infrastructure configuration and control. This allows incremental network change management with automated testing and rollback in case of errors. The system is conceived as a modular control platform supporting the Dynamic deployment and life-time management of cloud and edge services. AI/ML based service orchestration and network simulation based on digital twin is proposed to exhibit use cases on network configuration, automation and management. Development of Physical/Optical layer digital twins and applying AI/ML solutions on top is also proposed to mimic the real network. This allows testing use cases with simulating network topologies to study different scenarios and forecast the network behavior.

2.2 RELEVANT PROJECT OBJECTIVES

The project's objective is to design and validate an end-to-end transport network to support 5G and emerging services enforcing efficiency in latency, resource usage and energy consumption. The transport network spans across multiple network segments (RAN, metro/access, and core network) with different network components. The project KPIs target 5G and emerging services, which are mapped with relevant solutions and use cases are proposed to verify in practical scenarios.

The SEASON solution includes the assessment of Muti-Band over Space Division Multiplexing (MBoSDM), and SDM-PONs in transmission and switching. It also includes telemetry solutions that includes both data retrieval and protocols definition along with Intelligent data aggregation using data analytics pipeline.

Auto-configuration and self-healing objective drives the augmentation of NetDevOps tools with CI/CD to maintain the incremental network management in cases of configurations, automated testing and rollback in case of errors. These approaches are implemented and validated against the targeted KPIs to understand SEASON solution efficiency in coping with the 5G and emerging service requirements.

2.3 WP4 OBJECTIVES

Below is a list of WP4 specific objectives according to the WP description.

(O4.1) Define, fed by WP2, the dynamic operations and services offered to infrastructure operators (operational services) and end users (such as vertical industries), elaborate a list of common functional requirements.

(O4.2) Design the architecture of the system, based on a functional decomposition (e.g., micro-services approach) with open interfaces, and an optimal trade-off between centralized and distributed approaches (e.g., node local control and infrastructure central control).

(O4.3) Design and implement the different functional elements of the system, including SDN controllers and orchestration systems.

(O4.4) Increase data plane visibility by designing and implementing an infrastructure for massive and large-scale programmable monitoring/telemetry for packet/optical devices and MB-over-SDM transmission, with data analytics and hash-based data mining techniques ensuring KPI collection.

(O4.5) Integrate continuous development / continuous tools, pipelines, and processes from DevOps for small, automated incremental configuration and control tasks.

(O4.6) Enable service assurance by designing and implementing an autonomous zero touch intent-based networking system, using ML-based closed loops with continuous training and knowledge sharing.

(O4.7) Design and implement algorithms for function placement and resource allocation across the combined packet / optical PON, RIC, and MB-over-SDM transport infrastructure, accounting for radio control, impairment-aware routing and spectrum assignment and constrained optical networks.

(O4.8) Implement a proof-of-concept that demonstrates the use cases and release the source code of key components, contributing to key framework projects.

Concrete Actions and Functions:

- A distributed software system for monitoring, streaming telemetry, control and orchestration of MB-over-SDM network has been designed and developed.
- Mechanism for overall management of operational and user services covering PON, RIC and transport.
- Advanced monitoring and streaming telemetry for local and centralized control actions and operations.
- Closed loop domain control and automation, integrated end-to-end network operation, and AI/ML based service orchestration and network simulation based on digital twin.
- Centralized SDN control is augmented by novel Continuous Integration and Continuous Delivery (CD/CI) paradigms and tools for networks infrastructure configuration and control to allow incremental network change management with configuration change management, automated testing and rollback in case of errors.
- System architectural design as a modular control platform that supports the dynamic deployment and life-time management of cloud and edge services.

2.3.1 OBJ 5 - Develop a pervasive monitoring infrastructure for secure and truly self-managed networking

The requirements related to Objective 5 “Develop a pervasive monitoring infrastructure for secure and truly self-managed networking” are analyzed in SEASON Milestone MS2.1 section 3.3.

- KPI 5.1: Achieve sub-km (<500 m) and sub-dB (<0.5) resolution in the estimation of longitudinal fiber attenuation points and optical amplifier gain, respectively, using DSP-based monitoring scheme.
- KPI 5.2: Performance improvement achievable with an OSA embedded in the amplifier setup and control identified for different link designs and applications.
- KPI 5.3: OTDR Interrogator for latency / position measurement with 4 ns / < 1 meter accuracy respectively
- KPI 5.4: Applicability of modulation format insensitive OSNR measurement techniques in different scenarios determined, sources of inaccuracy identified, impact of signal distortions worked out.

From WP4 perspective, Obj. 5 is closely related to Obj. 7, which is described in the next section.

2.3.2 OBJ 7 - Control plane, Monitoring and streaming telemetry

2.3.2.1 Objective Description

Design, development and validation of a generalized telemetry, infrastructure and control service orchestration system for the SEASON MBoSDM infrastructure, able to deploy and manage the lifecycle of pluggables, network elements, integrated packet/optical systems, and services. To fully exploit monitoring and network telemetry, while reducing bandwidth utilization, SEASON develops intelligent methods to aggregate and compress data before distributing them. Development and validation of the DevOps paradigm augmenting the centralized SDN control plane with tools and processes from continuous development/continuous integration for zero-touch infrastructure configuration and network automation.

Means of validation: Lab evaluation in WP3, integration with WP4 and assessment in WP5. This includes:

- definition of data models and relevant streaming telemetry and control in a microservices architecture;
- development of NetDevOps operations for selected use cases and integration with SDN control plane;
- development of the orchestration system; contribution (in cooperation with WP6) to SDO and Open-Source projects;
- development and regression testing within WP4; Integration testing and demo in WP5.

2.3.2.2 KPI 7.1

Intelligent data aggregation to provide data compression ratio >90% without significant information loss

2.3.2.3 KPI 7.2

Reduction on the average setup time of converged connectivity service by 30% compared to serialized provisioning, exploiting approaches relying on parallelism and concurrency. Network Connectivity Service (point to point across different segments (PON/backhaul) considering control plane only < 1 second (not considering hardware configuration latencies).

2.3.2.4 KPI 7.3

Network connectivity service with creation time < 3 min combining control and data planes. In [Sha21], 3 mins were needed for network connectivity in the metro segment only, mainly due to laser configuration. In SEASON, connectivity is extended to cover end-to-end, including front-haul, PON and metro/core.

3 OVERVIEW OF SEASON CONTROL PLANE ARCHITECTURE FOR SELF-MANAGED AND AUTONOMOUS NETWORKING

Self-managed and autonomous networking refers to the ability of intelligence and self-governance of a network. In SEASON, this is addressed by proposing a Software Defined Networking (SDN) control plane architecture covering the RAN, access/metro, and core segments, in an over-arching hierarchical control. The control plane components and applications drive towards (a) automatic network configuration, (b) self-healing during failure, (c) secure access and control of the devices, and (d) optimal use of network resources to make the network truly self-managed. This chapter covers the overall SEASON control plane architecture and associated tools and technologies.

3.1 OVERALL SEASON SOLUTION – CONTROL PLANE

Figure 3.1 portrays the key network components in each segment of the SEASON reference transport network from the RAN to the core segment and, on top of it, the associated control plane technologies. It can be macroscopically described as:

- An innovative control and orchestration plane following SDN principles for overarching control of the RAN, PON and Transport Segments (Aggregation/Metro/Core). This mostly involves service provisioning, configuration, and control.
- The applicability of new control paradigms based on *NetDevOps approaches* jointly with *AI/ML in support of network operation* and network orchestration, including Multi-Agent Systems (MAS). Macroscopically, AI/ML algorithms are applied for the near-real time control of network resources and services aiming at reducing energy consumption and ensuring performance, including moving intelligence as close as possible to the data plane, and devising a MAS distributed system.
- An optical monitoring and telemetry platform using open interfaces.
- The use of *digital twins* for use cases such as fault localization.

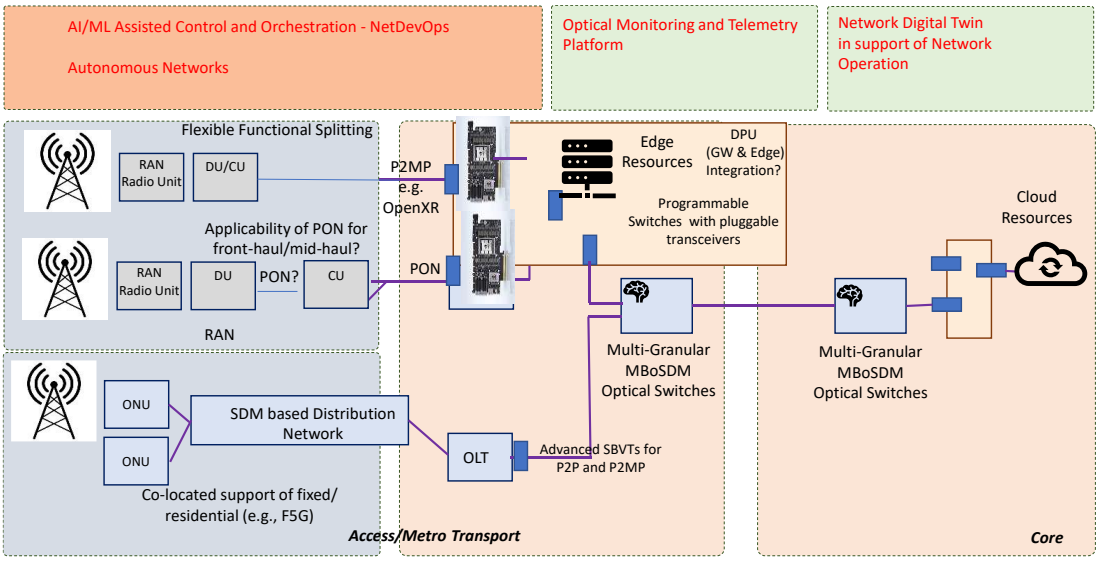


Figure 3.1: SEASON Architecture Self-managed and autonomous networking.

SEASON WP4 designs and validates an innovative transport network control and orchestration infrastructure (see Figure 3.2) aiming to support beyond 5G and new emerging services. The assessment of this novel approach is planned by considering the following activities:

- Full support of innovative Multi-band over SDM transmission and switching HW solutions.
- Integration of RAN Intelligent Controller (RIC) and access/metro SDN Control.
- Applicability of new control paradigms based on NetDevOps approaches jointly with AI/ML in support of network operation and network orchestration.

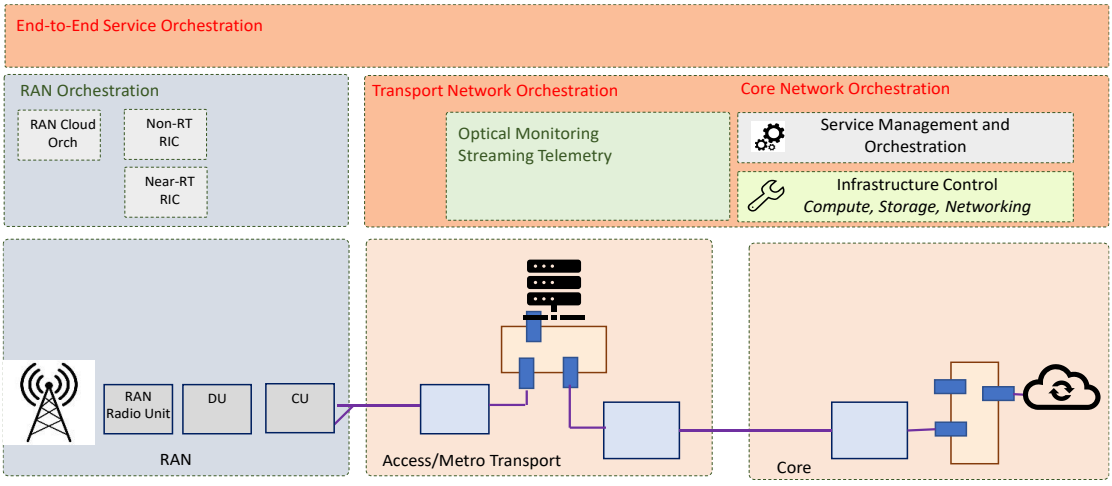


Figure 3.2: SEASON Architecture overarching control for RAN/PON/Backhaul network segments.

3.2 SDN BASED INFRASTRUCTURE CONFIGURATION AND CONTROL ARCHITECTURE

There are several key innovation technologies addressed in the project:

- Infrastructure Control for MBoSDM optical network, including New Paradigms for SDN based Control (based on NetDevOps and Continuous Integration/Continuous Development). A specific component is developed as an SDN agent for the control of flexi-grid DWDM networks on top of SDM networks.
- The usage of secure AI/ML techniques in support of network orchestration and Optical Layer Digital Twin, as mentioned above.
- Monitoring, Streaming Telemetry and Intelligent Data Aggregation in support of telemetry techniques.
- RAN Intelligent Controller (RIC) and access/metro SDN Control Integration, including SDM PON control.

Below is a list of key innovation topics that are related to WP4 tasks.

- Monitoring, Streaming Telemetry and Intelligent Data Aggregation (IDA) (T4.1)
- Infrastructure Control for MBoSDM optical networks. (T4.2)
- RAN Intelligent Controller (RIC) and access/metro SDN Control Integration (T4.2)
- New Paradigms for SDN based Control (NetDevOps) (T4.2)
- Optical Layer Digital Twin (T4.2/T4.4)
- AI/ML in support of Service Orchestration (T4.4)
- Multi-Agent Systems (MAS) (T4.4)
- Secure AI (T4.4)

SEASON defines reference architectures, but recognizing the fact that for each architectural aspect there may be different arrangement of controllers and different deployment options. Aspects like the dynamic configuration of pluggables or the control of P2MP transceivers are a subject of debate. In all, some of the key elements of the architecture can be shown in Figure 3.3 and Figure 3.4, highlighting different options. In the former figure, the Network Orchestrator is responsible for coordinating the control of the PON and the Optical controller, which, in turn coordinates the configuration of the P2P or P2MP pluggables and discrete transceivers as well as the different optical line systems.

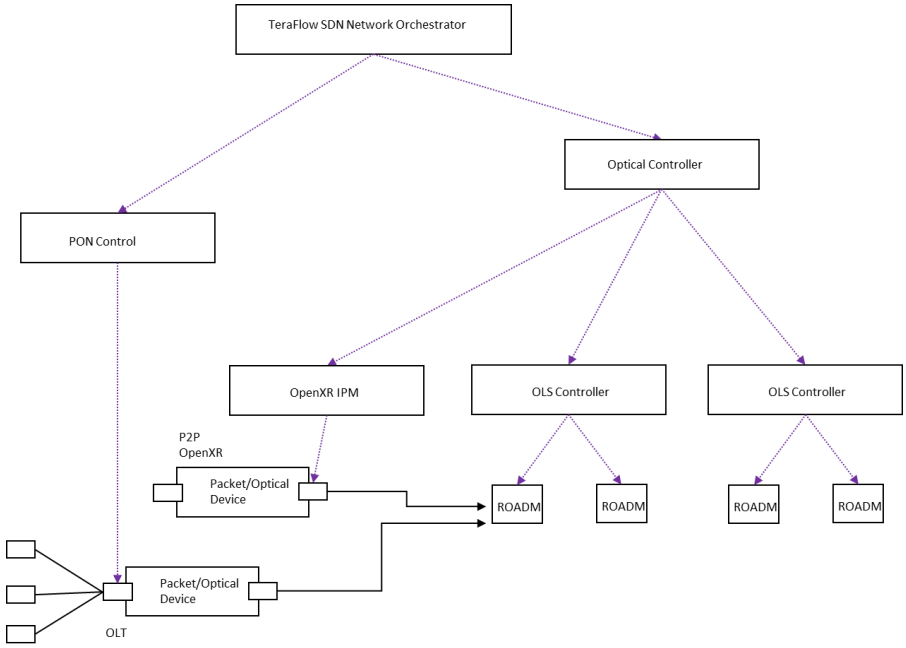


Figure 3.3: Architecture 1 – E2E SDN Control Plane Architecture for Optical Network.

In the case of the latter figure, we can see that the pluggable configuration is delegated to the orchestrator, and not the optical controller, enabling a seamless interworking with already existing optical controllers, since the configuration of the pluggables may be decoupled from the configuration of the rest of the line system.

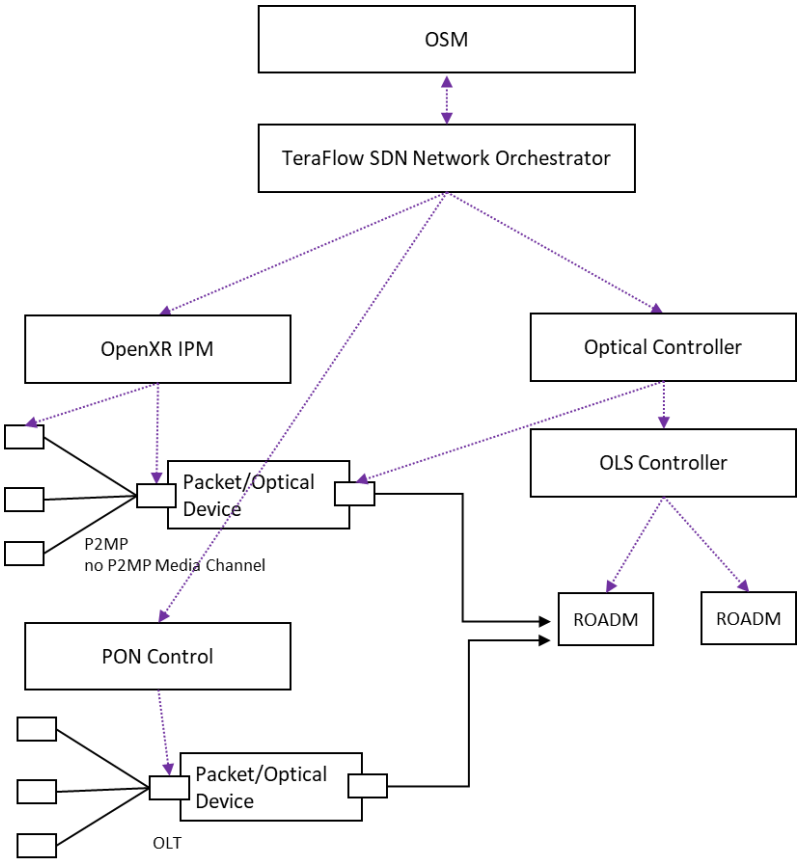


Figure 3.4: Architecture 2 - E2E SDN Control Plane Architecture for Optical Network.

3.2.1 Teraflow SDN and North Bound Interface

In the pursuit of improving network management efficiency and automation, the TeraFlow Operating System (TFS) is at the forefront of Software-Defined Networking (SDN). TFS is an open-source, microservice-based, cloud-native, and carrier-grade SDN controller that aims to revolutionize network orchestration by enabling zero-touch automation [Git22].

A crucial component of TFS is its Northbound Interface (NBI), which serves as the gateway for external systems to communicate with TFS. The NBI plays a vital role in integrating TFS with different orchestrators, such as the Open-Source Management and Orchestration (OSM) system or Kubernetes, depending on specific deployment requirements and use cases.

One of the main objectives of this integration is to implement connectivity service requests to the TFS Service Component. When an orchestrator receives a request for a new network connectivity service, it follows a defined workflow. In Phase 1, a new empty service is created in TFS to obtain a unique service identifier. In Phase 2, endpoints representing the devices that need to communicate are added to the service. The TFS Service Component fills in the required fields with default values, stores the service details in the Context database, and returns the service identifier to the orchestrator. Upon receiving the request to add endpoints to the service, the orchestrator triggers a service update request towards the Service component. This update request involves identifying the devices that own the endpoints to be connected, determining the device drivers they support, and selecting the appropriate service handler for the specific service. TFS ensures seamless communication, reliable data transmission, and optimized network resources, contributing to efficient traffic flow across the network.

Additionally, the SEASON project, in which TFS plays a vital role, places significant emphasis on energy efficiency. Through ML-AI-assisted network reconfiguration, TFS dynamically adjusts network resources (O-gNB's, CNF's etc.) to optimize energy consumption. By monitoring telemetry data at different network points, TFS can adapt capacity in near real-time, migrate functionality, and allocate resources based on traffic predictions. This approach optimizes network resource utilization and reduces unnecessary energy consumption, corresponding to the SEASON project's goal of promoting energy-efficient networking solutions.

Traffic redistribution could also be included within the TFS solution. Real-time traffic monitoring and ML-AI-based traffic forecasting enable TFS to proactively respond to fluctuating demand patterns and ensure uninterrupted network services. By redirecting traffic to different access points, when necessary, TFS demonstrates its ability to adapt to changing network conditions and optimize resource utilization.

In conclusion, the integration of the TFS NBI with orchestrators like OSM or Kubernetes represents a significant advancement towards zero-touch automation in network management. Through efficient service creation and configuration, TFS enables streamlined network operations, reliable connectivity, optimized energy efficiency, and dynamic traffic redistribution. The collaboration between TFS and orchestrators brings us closer to a more agile, scalable, and intelligent network infrastructure. The focus on energy efficiency and traffic redistribution is

vital to advanced innovative networking solutions and achieving defined key performance indicators.

4 MONITORING, STREAMING TELEMETRY AND INTELLIGENT DATA AGGREGATION

4.1 MONITORING AND STREAMING TELEMETRY

Within the course of this deliverable, a comprehensive monitoring infrastructure able to retrieve and intelligently aggregate telemetry data/metadata has been designed and validated. These data/metadata are generated by the optical data plane, including the novel monitoring information from power-efficient DSPs and pluggables, optical line systems, embedded interrogators, etc.

The optical data plane monitoring parameters are effectively combined with monitoring data originated at the packet, computing (e.g., from the DPU), and control level. A combination of in-band and out-of-band telemetry solutions are exploited, including distributed event streaming technologies.

The following elements serve as generators of telemetry data:

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

The following elements serve as consumers of telemetry data:

- Telemetry collector, a dedicated element designed to efficiently process telemetry data, typically adopted in centralized/hierarchical scenarios, collecting data from large number of producers, useful for network-level close loop operations
- SmartNIC/DPU, enabling local processing of received telemetry data, taking advantage of embedded CPU/GPU resources. It would enable decentralized scenarios, collecting data from a limited number of producers, useful for node-level (local) close-loop operations.

The following technologies were used to retrieve and exchange telemetry data:

- gRPC (using either OpenConfig or OpenROADM YANG models) for out-of-band telemetry from optical devices, packet and computing nodes. Legacy protocols like NETCONF, RESTCONF, and SNMP remain possible solutions to also feed telemetry collectors at lower pace.

- In-band and post-card telemetry, for in-band per packet statistics on QoE performance (e.g., time spent in queue on traversed IP routers)
- Kafka: Even though gRPC is very efficient to deal with data volume, its flexibility is very limited since exchanged messages need to follow predefined schemas. In view of that, Apache Kafka data streaming is an option to exchange telemetry data as simple text messages, which provides the flexibility to deal with data variety. In addition, Kafka facilitates the integration of different data sources.
- Decentralized near-real-time control based on multi-agent systems (MAS). Intelligent agents deployed as close as possible to the network devices consume telemetry measurements, communicate among them and make decisions to control the network infrastructure.

The overall telemetry architecture (see Figure 4.1) integrating pervasive telemetry and MAS decentralized control is presented next.

We assume an SDN architecture controls several optical nodes, specifically optical transponders (TP) and reconfigurable optical add-drop multiplexers (ROADM) in the data plane. Note that the SDN architecture might include a hierarchy of controllers, including optical line systems and parent SDN controllers. A centralized telemetry manager oversees receiving, processing, and storing telemetry data in a telemetry DB, which includes two repositories: i) the measurements DB that is a time-series (TS) DB that stores measurements; and ii) the events DB that is a free-text search (FT) engine. In addition, telemetry data can be exported to other external systems. Some data exchange between the SDN control and the telemetry manager is needed, e.g., the telemetry manager needs to access the topology DB describing the optical network topology, as well as the label-switched path (LSP) DB describing the optical connections.

Every node in the data plane is locally managed by a node agent. The node agent translates the control messages received from the related SDN controller into operations in the local node. In addition, the node agent includes data source adaptors that collect measurements from observation points enabled in the optical nodes or in specific optical devices, like OSAs, as well as a telemetry agent that processes and exports telemetry data to the telemetry manager. In addition, events can be collected from applications and controllers.

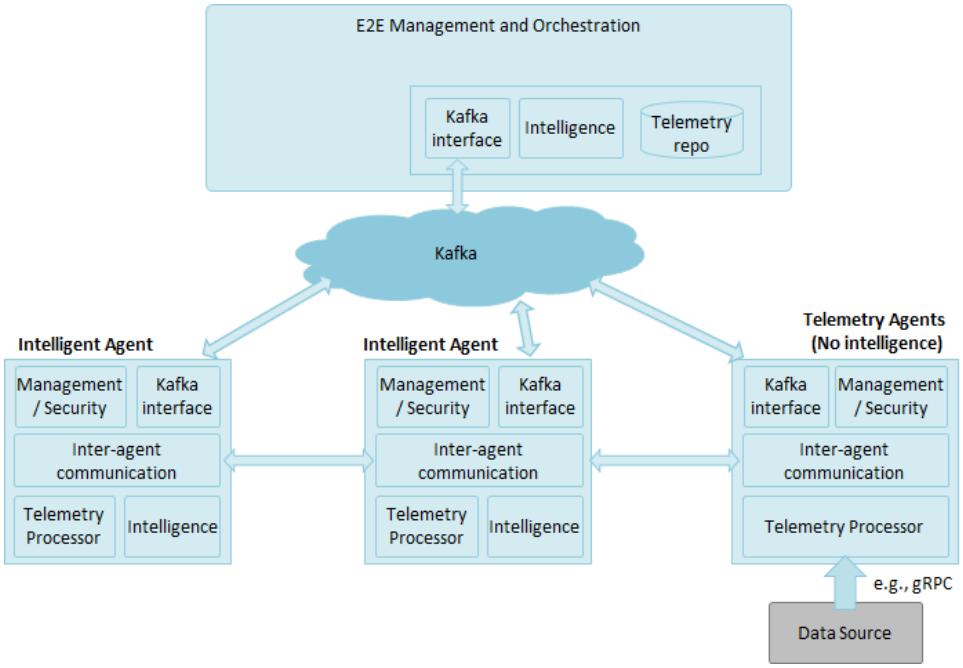


Figure 4.1: SEASON telemetry architecture

Data sources can be integrated in two different ways: i) internal data sources, i.e., those that are deployed inside the node agent and can access the Redis DB directly to publish new telemetry data (measurements or events); ii) external data sources that are connected to the telemetry agent through a dedicated interface (e.g., based on gRPC). Only trusted peers can connect externally to the telemetry agent. Kafka is used for the telemetry agents to export telemetry to the telemetry manager and to tune the behavior of algorithms in the agents.

4.2 NETWORK ANALYTICS PIPELINE

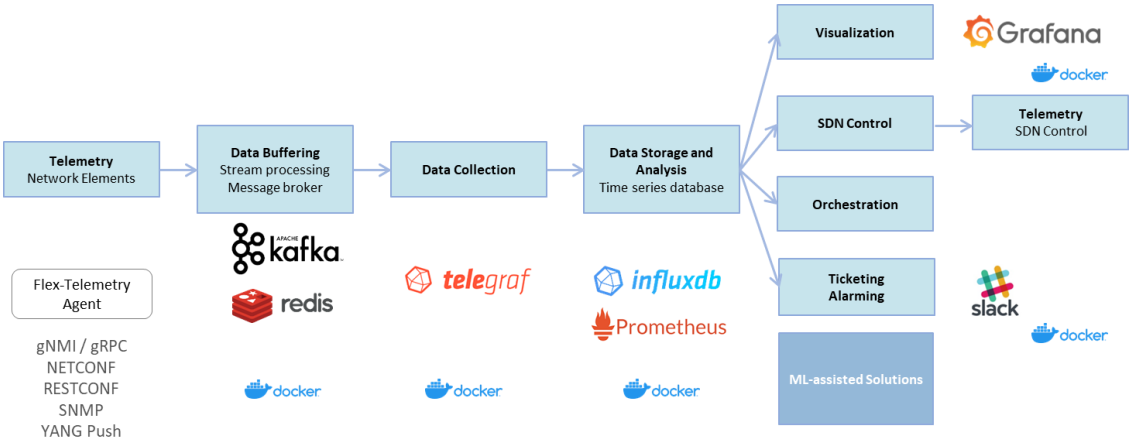


Figure 4.2: Telemetry – Data Analytics Pipeline.

SEASON WP4 envisioned the support of access/metro/core network with the proposed overall network architecture. So, an optimized telemetry analytics framework is required to support telemetry streaming from different sources using different data models and protocols. The

telemetry analytics pipeline shown in Figure 4.2 is proposed to collect, process, and expose the aggregated data to the other applications including orchestration, alarm and visualizer components. The pipeline is modeled to support collection of data from south bound data plane components and processing it with the northbound controller/orchestration components. This allows the analytics pipeline to fit well with the proposed SEASON architecture.

The following section briefly discusses the tools and technologies which are used in the analytics pipeline. These are related to retrieval, processing, and knowledge acquisition of the telemetry data from the analytics pipeline.

Telemetry agent: The telemetry agent includes the retrieval of telemetry data from the network elements (NE) serving as a mediator between the data and control plane entities. The agent implements suitable protocols to communicate with NEs in the data plane in periodic intervals, and the consumed data is exported in a suitable format for further processing. The internal working of the telemetry-agent is explained in section 4.1.3.

Apache Kafka: Apache Kafka is a open source distributed event streaming platform, which is highly scalable, and reliable for real-time data analytics. Kafka uses a publish-subscribe model, where data is organized into different “topics”. Producers are responsible for pushing data into the created topics where the application supports aggregation/processing over the published data. Consumers subscribe to these topics and receive the data for further actions. The platform allows high performance data aggregation and real-time streaming analytics for critical applications.

Redis: Redis is an open source, high performance in-memory data store providing low-latency access to the stored data. The “in-memory” characteristics in Redis differentiates it from other datastores by storing and accessing data from main memory. This makes it ideal for real-data data analytics pipeline.

Telegraf, InfluxDB, and Grafana stack: Telegraf is an open-source data collection agent where it collects data from diverse origins and transmit it to multiple destinations, playing a vital role in monitoring and gathering metrics for systems and applications. It works with a plugin-based model having input, process, aggregate and output plugins to collect, process and send the data. Influx-DB is a time-series database to store and access time stamped metrics and event data. Telegraf and InfluxDB are commonly used together making this as a ideal for monitoring and observability applications with live time-series events. Grafana is a data visualization tool which allows users to create customizable dashboards to monitor and analyze the time series data. This goes well with the time-series database to provide comprehensive solutions for monitoring and analytics of time-stamped data.

Altogether, the Telegraf-influx-Grafana stack provides a complete solution to time series data management for effective infrastructure monitoring.

From the figure, it can be seen that the telemetry agent publishes the retrieved data to the topics in Kafka. These streamed events are processed and aggregated in event streaming pipeline implemented in Kafka. The resulting data is consumed using Telegraf data collector for further storage and processing. The streamed time stamped data is exported to InfluxDB and

are visualized in Grafana dashboard based on the customized inputs. The consumed events are also exported to perform further actions including SDN control, orchestration component, and ML assisted solutions as shown in the figure.

4.3 Optical Network Element Telemetry Data Producer

The optical network elements in the data plane possess the capability to expose the configuration and operational data to reveal the current state of the device and the provisioned channels. The read-only operational data includes the performance monitoring data of the configured channels, fault monitoring data, and hardware status of the network element. This data source should be periodically retrieved and inform the control/management plane components for decision making, which is called as “telemetry data producer”.

To continuously retrieve the data from the network element, there is a need of an agent to communicate with the applied protocol. This serves as a data producer engine to the analytics pipeline which is discussed above.

The implementation of the agent focuses on the following properties:

- Enables data consistency on periodic data retrieval.
- Usage of standard protocols (NETCONF) to communicate with the network elements.
- Usage of standard data models for data retrieval.
- Consistency in data and protocol, while providing data to the analytics pipeline.

The application "Flex-Telemetry" in Figure 4.3 is a telemetry agent designed to periodically initiate requests to gather performance measurements from optical devices in the data plane. It achieves this by utilizing NETCONF along with a mix of open (OpenConfig) and proprietary data models. Moreover, the system includes a modular plugin framework that offers an NBI interface, ensuring a reliable supply of stream telemetry to various platforms, including time-series and in-memory databases, as well as International Data Spaces (IDS). Investigation includes:

- Evaluation of protocols for data exchange.
- Verifying the extent of YANG models for data retrieval from the network elements.

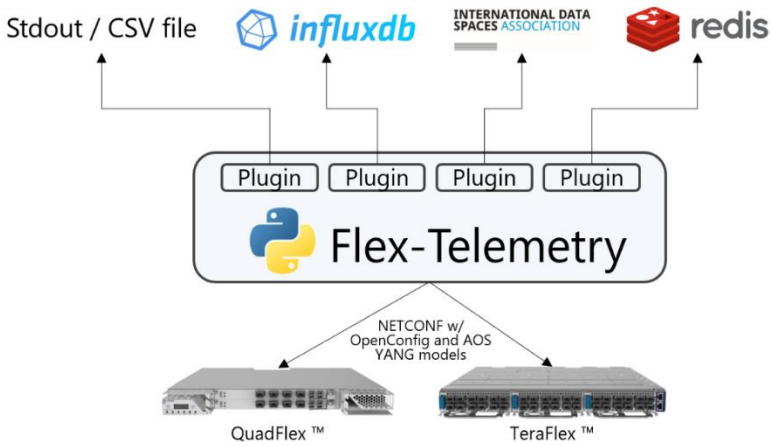


Figure 4.3: Telemetry Agent for Optical Transport Network.

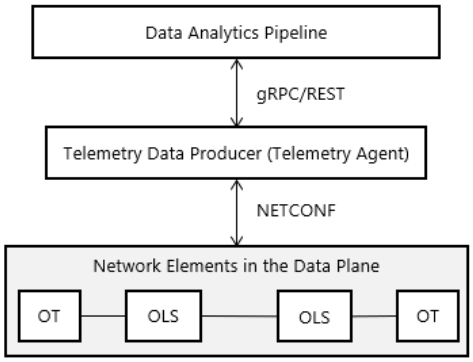


Figure 4.4: Overall Architecture - Telemetry Data Producer.

As an initial development, the flex-telemetry data producer adopted ONF-TAPI based streaming to stream the performance parameters retrieved from devices and OLS controller. The ONF-TAPI v2.5, provides recommendation that telemetry streaming should be aligned with the approach taken by GNMI community. The TAPI-server proxy application is developed which is adapts the proto file mentioned in the figure 4.5. The Figure 4.5 gives the YANG definition and respective gRPC proto file for gNMI based telemetry streaming. The TAPI-server proxy application continuously retrieves the telemetry data from devices and control plane using NETCONF/gNMI/gRPC, and streams the data with gNMI/gRPC following the ONF-TAPI recommendations.

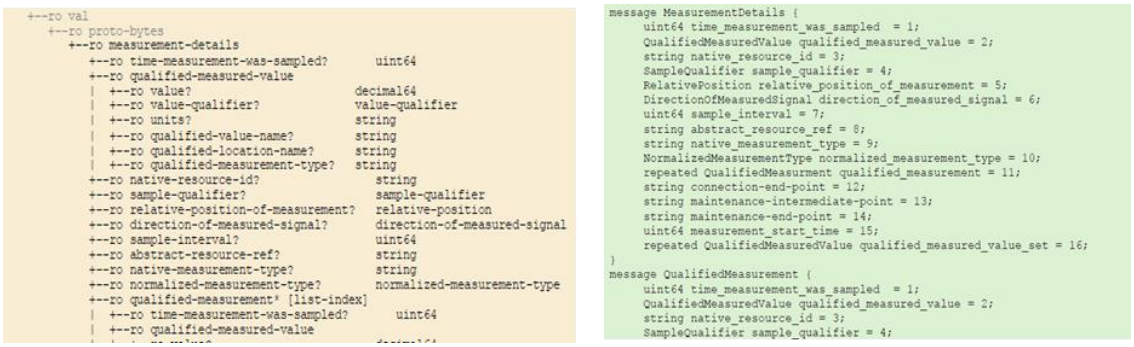


Figure 4.5: Tapi-gnmi-streaming.yang and Tapi-gnmi-streaming.proto specification defined in ONF-TAPI.

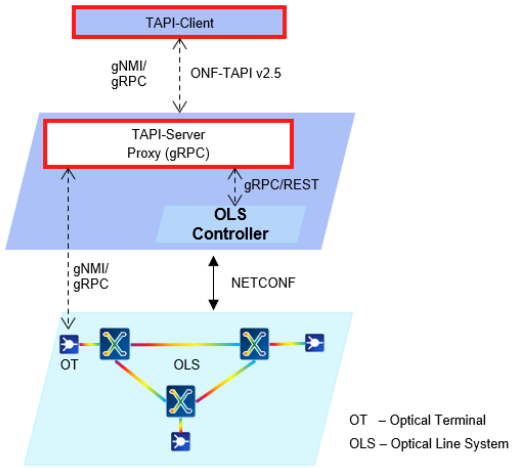


Figure 4.6: ONF-TAPI based Telemetry streaming using gNMI/gRPC is optical network.

For Example:

The TAPI client in the management plane subscribes for performance metrics from the TAPI-server using a “PATH” variable, and mentions subscription constraints along with it. The “update_only=true” helps to conserve the traffic such that TAPI-client receives the data only if there is a change in the performance data in the data plane. The Figure 4.6 shows the sample subscription request while subscribing to PM-data (signal-to-noise ratio) of a service from the TAPI-client.

```
subscribe {
 prefix {
  target: "tapi"
 }
 subscription {
  path {
   elem|{
    name: "/acor-bulk-data:bulk-data/acor-me-bulk:managed-element[entity-name=\'1\']/acor-
eqpm-data:service[name=\'service1\']/pm-monitored-entity/pm-current-data/montype-
monval/mon-type/mon-val:snr"
   }
  }
  mode: SAMPLE
  sample_interval: 20000
 }
updates_only: true
}
```

Figure 4.7: Example of a TAPI Subscription request to stream performance monitoring data of a service.

In ONF-TAPI v2.5, it is mentioned that with efficient serialization using gNMI/gRPC, the payload is 3x-1x smaller than XML payload consumed using REST. However, the packet analysis is still ongoing to investigate the latency, payload size, and data-traffic analysis, which will be reported in the upcoming meetings.

4.3.1 SmartNIC/DPU as both Telemetry Data Producer and Consumer

Nowadays, edge computing nodes are typically equipped with traditional Network Interface Cards (NIC) connected to routers which exploit long-reach transmission modules either embedded as pluggables or external as transponders. However, this type of connectivity involving multiple equipment requires expensive and power-hungry opto-electro-optical conversions. SEASON has investigated Smart NIC (smartNIC), also called Data Processing Unit (DPU), directly equipped with long-reach coherent pluggable modules (e.g., 100/400 ZR+ and 100/400 XR).

Figure 4.8 shows the proposed edge-to-edge (or edge-to-cloud) scenario where edge nodes are equipped with DPUs provided with coherent optical pluggable modules.

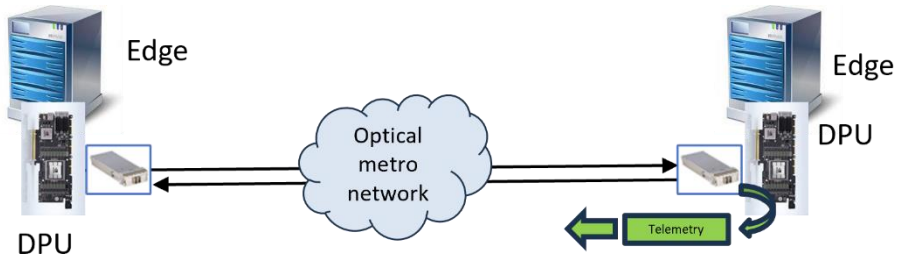


Figure 4.8: Edge-to-edge(/cloud) direct optical connectivity across metro optical networks, avoiding aggregation routers and transponders. DPUs are also used to perform in-network AI processing of received transmission parameters. Furthermore, they can stream back telemetry data to the source node.

In this proposed scenario, no intermediate routers or transponders are needed to serve edge computing nodes. Furthermore, DPUs include embedded hardware-acceleration processing solutions that can boost network telemetry and monitoring performance. This technology, at

the time of writing this document, is not commercially available, but it represents an innovative solution that has the potential to revolutionize the way edge computing infrastructures are interconnected among them or to the cloud across metro optical networks.

A smartNIC/DPU is a specialized hardware component designed to deliver fast networking and data processing capabilities. While typically utilized in data centers, servers and supercomputers, DPUs are gaining attention for their potential application in edge networking scenarios. This is due to their ability to handle data movement, storage, and processing for large datasets, performing high-speed calculations and facilitating real-time analysis and acceleration of data-intensive applications. In contrast to traditional NICs, which primarily offer low-level protocol acceleration such as Ethernet, DPUs enable programmability at higher layers and direct execution of advanced in-network functions. By offloading these functions to the DPU, processing resources are freed up for tenant and application services. The current generation of DPUs boasts several features, including up to four interfaces capable of operating at speeds of up to 400 Gb/s. They also incorporate advanced timing and synchronization capabilities, hardware encryption, and embedded security features. Additionally, these DPUs are equipped with up to 16 ARM CPUs to handle embedded computing operations.

In SEASON, DPUs equipped with coherent transceivers are used to directly perform optical transmission avoiding the interconnection trough routers. Furthermore, they serve as:

- producer of telemetry data, providing both optical data from the pluggables as well as packet telemetry data (packet statistics, QoS, computing parameters, etc.)
- consumer of telemetry data taking advantage of its embedded hardware acceleration to pre-process telemetry data in real time at wire-speed, significantly reducing scalability issues at telemetry collector.

A first validation of the DPU capabilities to process telemetry data in an efficient way is reported in SEASON D3.1 deliverable, where the DPU is used to directly process optical parameter data in the case of transmission anomalies. Preliminary results on pre-collected datasets showed that the inference time of processing N=30 optical parameters varies between 50ms and 70ms depending on the presence of specific anomalies (e.g., soft failures).

To assess the scalability performance, we also forced the DPU to simultaneously process N=5000 optical parameters. Preliminary result showed that in this case inference time always remains below 360ms.

Once transmission data are received and efficiently elaborated, they can also be forwarded to other nodes, including the transmission node. This latter scenario is currently under investigation, leveraging on the embedded HW-acceleration features to process packets with specific pre-defined fields. In the following, a specific cyber-security use case is considered. The purpose is twofold: (i) assess the hardware acceleration DPU capabilities to process packets with specific pre-defined fields be also applied to monitoring data, (ii) investigate embedded DPU cyber-security performance to provide edge computing security robustness even in the absence of an interconnected router.

Indeed, guaranteeing cyber-security to high-speed connections typically either overloads CPU performance or it requires expensive dedicated hardware components (e.g., routers or standalone firewall). The presence of hardware-accelerated security functions within DPUs provides native embedded security to edge infrastructures. A novel framework for live traffic analysis and intrusion detection on the DPU is here presented. It relies on a combination of DPDK libraries, DOCA library that provides Regular Expression (RegEx) pattern matching to DOCA applications, and the Suricata intrusion detection system. The framework leverages hardware acceleration to perform deep packet inspection (DPI) and deep learning-based intrusion detection. This enables the offloading of Distributed Denial of Service (DDoS) detection tasks to the DPU. Initially, the DPU receives incoming traffic and performs RegEx operations to detect, in this case, malicious traffic. If the traffic is benign, it is forwarded to the edge CPU for further processing; otherwise, the DPU drops malicious traffic. The framework relies on a deep learning-based Convolutional Neural Network (CNN) model to detect DDoS attacks in real-time.

The model was trained and tested using the CICDDOS2019 dataset. Figure 4.9**Errore. L'origine riferimento non è stata trovata.** shows the main performance metrics of the AI model. Furthermore, results show that DDoS attacks are detected and blocked in real-time at a rate of up to 95 Gbps of attack traffic, with 99.45% accuracy and only 27.6% CPU utilization. This represents a significant improvement over a traditional non-hardware accelerated DDoS detection method that can process only up to 20 Gbps with 90% CPU utilization.

This work has been accepted for publication at the OFC Conference: Piero Castoldi, Rana Abu Bakar, Andrea Sgambelluri, Juan Jose Vegas Olmos, Francesco Paolucci, and Filippo Cugini, “Programmable Packet-Optical Networks using Data Processing Units (DPUs) with Embedded GPU Monitoring devices”, OFC Conf. 2024 [Cas24].

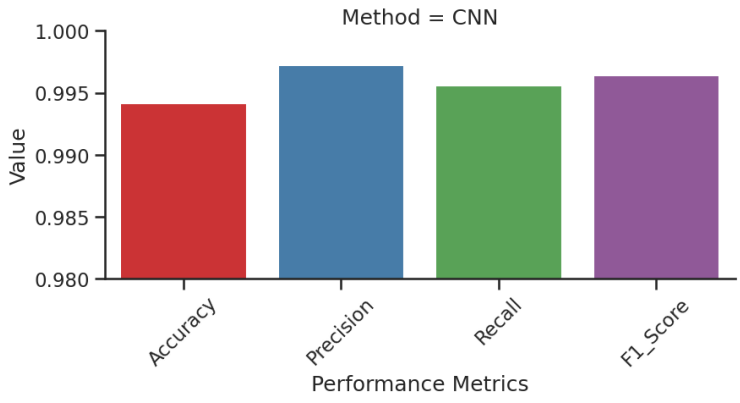


Figure 4.9: DDoS Attack Evaluation Results Using DPU

4.4 SDN Controller as Telemetry Data Producer

An SDN controller acts as a data producer since it may be able to generate network events associated with network operation. This can, for example, be used to replace legacy notification

and alarms systems (such as those based on SNMP protocol) but also to continuously report about network status changes, including topological changes and service changes.

The relevant reference point in this case is the controller NBI and targets an SDN controller that implements TAPI or similar interface. This applies equally to, for example, an Optical Controller or a dedicated Optical Line Controller.

While for regular TAPI usage the protocol is commonly assumed to be RESTCONF, in terms of Streaming Telemetry this is not specified or imposed. Implementations are free to choose transport protocols and software frameworks such as Kafka, Redis, or message buses such as MQTT or similar.

Macroscopically, streaming refers to an approach or mechanism that handles the providing of information from one system to another in some form of steady and continuous flow. In our scope, it is used primarily for the reporting (notification) of ongoing change of state of the controlled system from one Management-Control entity to another (usually superior) management-control entity [TR-548]. Such solutions should address key requirements such that peak load is averaged using some mechanism.

The focus is in streaming approaches that:

- Focus on conveying TAPI entities, i.e., yang sub-trees.
- Provide structured event reporting.
- Allow a client to achieve and maintain eventual consistency.
- TAPI notifications and migration towards TAPI 2.4

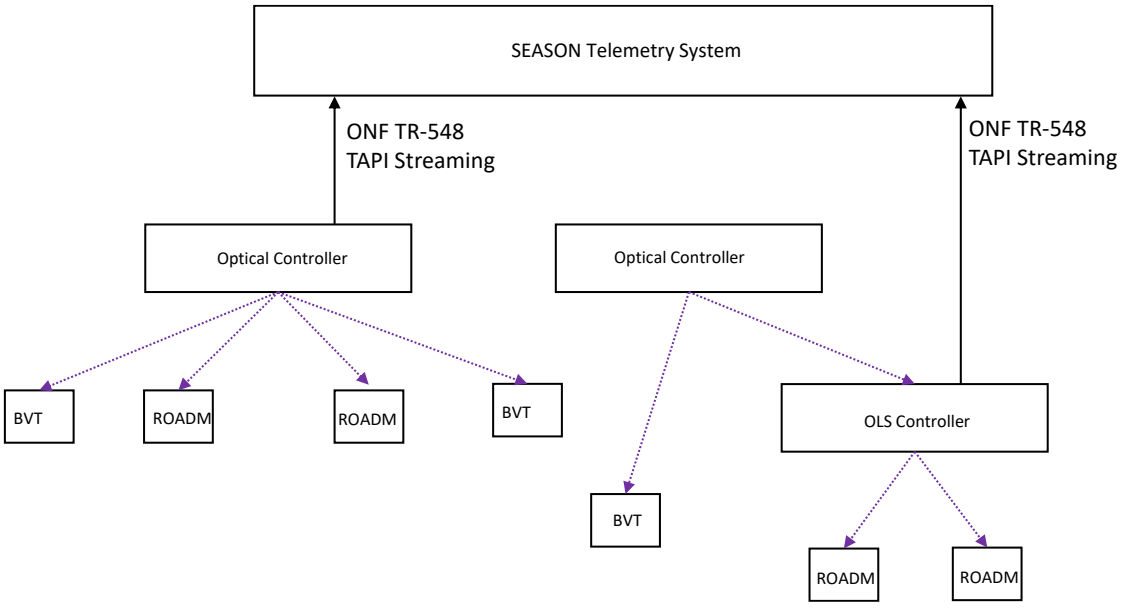


Figure 4.10: Macroscopic Telemetry architecture for TAPI TR.548 event telemetry from the optical controller NBI (or Optical Line System Controller).

4.4.1 TR-548 TAPI streaming

The proposed work is to validate ongoing work within TAPI streaming standardization and provide a set of guidelines and recommendations for use of TAPI streaming. The target architecture for TAPI is provided in [ONF TR-547] and focuses on the autonomous flow of information via TAPI from an SDN controller (e.g., OLS) to a dedicated system (e.g., Telemetry System)

In the scope of activities within WP4, we have implemented a PoC in which the SDN Controller exports TAPI Notifications based on SEASON telemetry selected platform.

4.4.2 RAN / RIC as Telemetry Data Produce

RAN: telemetry data produced by DU, CU, Antenna

CU COUNTERS

The CU-CP and CU-UP support the following telemetry data that can be used by xAPPs in Near RT RIC and rAPPs in Non-RT RIC. Highlighted ones are yet to be implemented, Table 4-1.

Table 4-1: CU-Counters.

CU-CP counters	CU-UP counters
Mean number of RRC Connections (3GPP 28.522 section 5.1.1.4.1)	UL PDCP SDU loss rate (3GPP 28.522 section 5.1.3.1.1)
Maximum number of RRC Connections (3GPP 28.522 section 5.1.1.4.2)	UL F1-U packet loss rate (3GPP 28.522 section 5.1.3.1.2)
Attempted RRC Connection Establishments (3GPP 28.552 section 5.1.1.15.1)	DL PDCP SDU drop rate (3GPP 28.522 section 5.1.3.2.1)
Successful RRC Connection Establishments (3GPP 28.552 section 5.1.1.15.2)	Average delay DL in CU-UP (3GPP 28.522 section 5.1.3.3.1)
Number of PDU sessions requested to setup (3GPP 28.522 section 5.1.1.5.1)	Distribution of delay DL in CU-UP (3GPP 28.522 section 5.1.3.3.4)
Number of PDU sessions failed to setup (3GPP 28.522 section 5.1.1.5.3)	
SS RSRP Distribution per SSB (3GPP 28.522 section 5.1.1.22.1)	

DU counters (Effnet DU)

Following PM stats are available from Effnet DU, Table 4-2.

Table 4-2: DU counters (Effnet DU).

RRU.PrbTotDl	3GPP 28.552 section 5.1.1.2.1	DL Total PRB Usage
RRU.PrbTotUl	3GPP 28.552 section 5.1.1.2.2	UL Total PRB Usage
DRB.UEThpDl	3GPP 28.552 section 5.1.1.3.1	Average DL UE throughput in gNB
DRB.UEThpUl	3GPP 28.552 section 5.1.1.3.3	Average UL UE throughput in gNB
CARR.WBCQIDist	3GPP 28.552 section 5.1.1.11.1	Wideband CQI distribution
CARR.PDSCHMCSDist	3GPP 28.552 5.1.1.11.1	Wideband CQI distribution
CARR.PUSCHMCSDist	3GPP 28.552 5.1.1.12.2	MCS Distribution in PUSCH
DlUeThroughput-Cell		Downlink UE throughput for whole cell
RACH.PreambleACell	3GPP 28.552 5.1.1.20.1	Received Random Access Preambles per cell
DRB.MeanActiveUe		Number of active UE per cell

Cell specific measurements fully supported by the DU and RIC.

RIC: telemetry data produced by RIC itself.

RIC can produce [3GPP 28.554] KPI stats in addition to [3GPP 28.552] PM stats. It can produce additional derived metrics such as predicted cell load, direction of movement etc. using machine learning apps.

4.4.3 RIC as Telemetry Data Consumer

O-RAN Near RT RIC consumes the Telemetry data from various RAN components on the southbound interface using E2 KPM Service model. It can interface with the DU, the CU, gNB or eNodeB on the southbound interface. E2 KPM service model exposes all telemetry mentioned in [3GPP 28.552] specification via E2 interface. This telemetry data can be utilized by the xAPPs to take control or config update actions to optimize RAN performance.

4.5 INTELLIGENT DATA AGGREGATION

As Big Data, optical network telemetry data are a collection of data from many sources, can be described by means of characteristics, known as the 5 V's, standing for volume, velocity, variety, veracity, and value. Such characteristics can be seen as different tiers of a pyramid: *i*) at the bottom of the pyramid, *volume* refers to the size and amount of data that needs to be collected and analyzed; *ii*) *velocity* refers to the speed at which data are collected, stored and managed. Volume and velocity together impose requirements that need to be carefully considered, e.g., sometimes it is better to have limited data in real time than lots of data at a low speed; *iii*) *variety* refers to the diversity and range of different data types and data sources; *iv*) *veracity* is related to the quality, accuracy, and trustworthiness of data and data sources and it is the most important factor of all the 5 V's for business success; and *v*) *value*, at the very top of the pyramid, refers to the ability to transform data into useful insight.

In general, telemetry measurements are collected from observation points in network devices and sent to a central system running besides the SDN controller. This defines a telemetry pipeline with basically two elements: data collectors that gather measurements from observation points in devices and send them to a centralized telemetry system that stores and processes the received data. In parallel, events generated by applications/platforms (e.g., SDN controllers and management systems) can be used to keep consistency among systems. In fact, an event streaming mechanism is made available as an alternative to traditional notifications.

As a result of volume, velocity, and variety characteristics of telemetry data, efficient and flexible mechanisms need to be considered to convey measurements and events from network devices and other systems (producers) to consumers, e.g., in a central location. We have designed a telemetry architecture to support intelligent data aggregation nearby data collection, thus

extending the telemetry pipeline. The main target is to provide a solution that is able to deal with the 5 V's of telemetry data, while providing scalability, efficiency, flexibility, easy integration of different data sources with a variety of measurements and events, and facility for turning data into useful insight that can be used for network automation.

We next introduce techniques to greatly reduce the impact of both volume and velocity of telemetry data. In particular, we analyze: i) supervised FeX; ii) data compression using autoencoders (AE); and iii) data summarization using the arithmetic mean of a number of observations obtained when variation is stable. We focus on two examples of telemetry measurements, optical spectrum and IQ constellations from a m-QAM signal, which are by far the cases where collected samples are larger.

A. Supervised feature extraction

A simple but effective dimensionality reduction technique is supervised FeX. This technique is intended to generate the set of features $\Phi(M)$ that characterize a measurement sample M. As an example of Φ, a module can pre-process the optical spectrum of a signal, i.e., an ordered list S of frequency-power pairs, i.e., S=[<f, p>] (see Figure 4.11a). After equalizing power, the module characterizes the mean (μ) and the standard deviation (σ) of the power around the central frequency ($fc\pm\Delta f$), as well as a set of primary features computed as cut-off points of the signal with the following power levels: *i*) equalized noise level, denoted *sig* (e.g., -60dB + equalization level); *ii*) a family of power levels computed with respect to μ minus $n\sigma$, denoted $n\sigma$ (e.g., 3 and 5σ); and *iii*) a family of power levels computed with respect to μ minus a number of dB (e.g., -3 and -6 dB), denoted *dB*. Each of these power levels generates a couple of cut-off points denoted $f1_{(\cdot)}$ and $f2_{(\cdot)}$. In addition, the assigned frequency slot is denoted $f1_{\text{slot}}$, $f2_{\text{slot}}$. Then, the input list with 75 <f,p> pairs representing the spectrum of a 75GHz channel is processed to generate a set Φ_S with 13 features that can be easily transformed into value, e.g., for failure detection and identification, in the telemetry agent or the manager.

Another example is for IQ constellations, where we assume that the observation point is in a TP that gathers the received optical symbols of a m-QAM signal. The related data source periodically retrieves a constellation sample X (a sequence of k IQ symbols as represented in Figure 4.11b for a 16-QAM signal) and publish it in the local Redis DB. We apply Gaussian Mixture Models (GMM) to characterize each constellation point of an optical constellation sample as a bivariate Gaussian distribution. Therefore, each constellation point i is characterized by 5 features, the mean position in I and Q axes [μ^I,μ^Q], as well as the I and Q variance and symmetric covariance terms that the symbols belonging to the constellation point i experience around the mean [σ^I,σ^Q,σ^{IQ}]. Therefore, for a m-QAM signal, a set Φ_X with m*5 features need to be propagated from the telemetry agent to the manager.

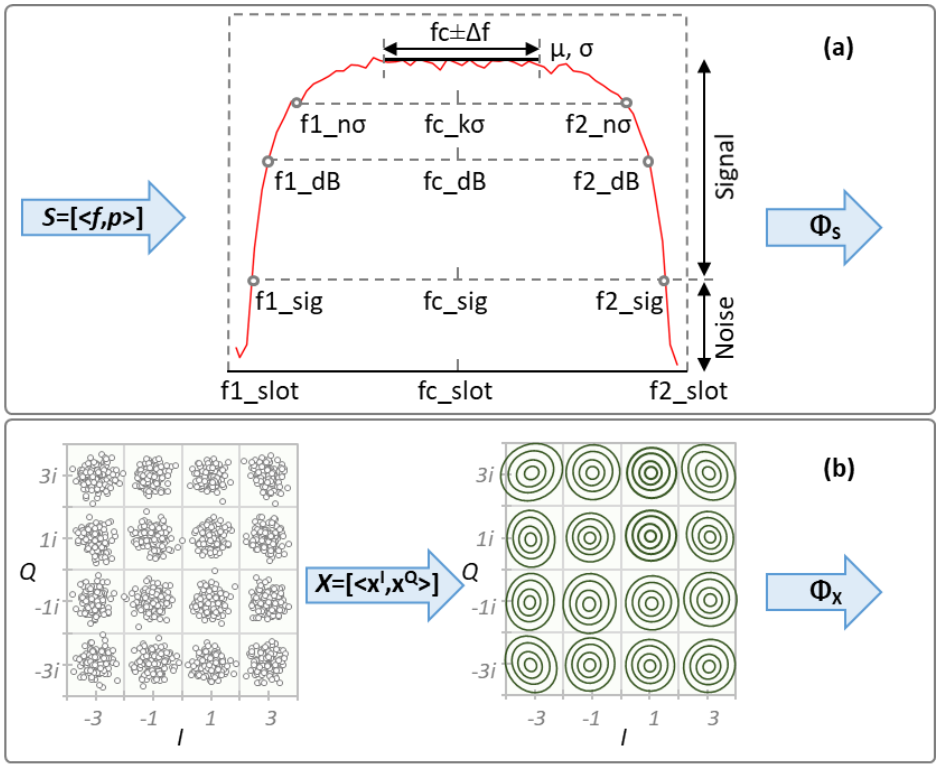


Figure 4.11: Optical spectrum (a) and IQ constellation (b) samples and FeX.

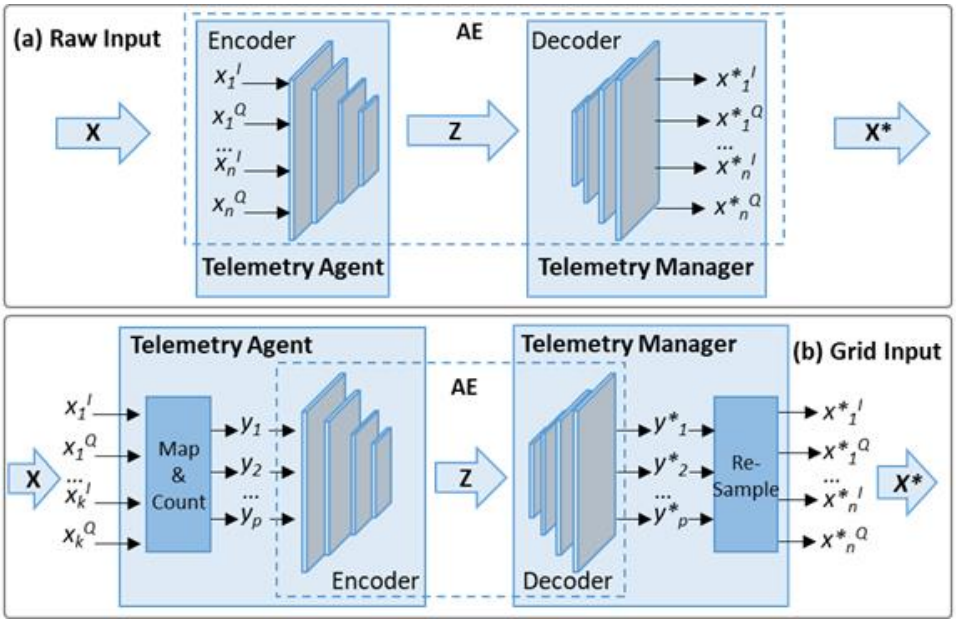


Figure 4.12: IQ constellation sample compression using autoencoders.

B. Data compression

Let us now focus on intelligent telemetry data compression performed at telemetry agents before data are sent to the telemetry manager through the gRPC interface. In this subsection, we target the compression of IQ constellations, since every sample *X* might include a large number of symbols, i.e., complex numbers. Note that the proposed compression method is compatible with any data serialization and compression engine built on the gRPC interface,

which applies additional compression to any data being sent, including raw samples and features.

We propose using AEs for intelligent IQ constellations compression. An AE is a type of neural network with two components: the *encoder*, which maps input data into a lower-dimensional *latent* space, and the *decoder*, which gets data from the latent space and reconstructs the original data back. Once trained, an AE takes as input k IQ symbols from the received constellation sample X, i.e., $[x_1^I, x_1^Q, \ldots x_k^I, x_k^Q]$ and generates latent space $Z=[z_1, \ldots, z_L]$, where the size of Z is significantly lower than that of X (see Figure 4.12a). Although such approach (hereafter, referred to as *raw input*) shows remarkable performance in terms of compression rate and average reconstruction error, it requires IQ constellations to contain a fixed number of symbols and the same proportion of symbols per constellation point. In addition, the size of both input and output layers and consequently, the complexity of the AE, depend on the number of symbols. This lack of flexibility reduces noticeably the applicability of this approach.

In order to overcome the aforementioned issues, we have proposed an alternative AE-based IQ constellation compression method (hereafter referred to as *grid input*), which is sketched in Figure 4.12b. Firstly, the whole IQ constellation is split into p regular grid cells, where each cell covers a small quadrant of the IQ constellation. Then, the input sample X is processed to generate vector $Y=[y_1, y_2, \ldots, y_p]$ containing the count of symbols that fall into each grid cell. Note that the length of Y only depends on p, which represents the *resolution* of IQ constellation pre-processing. Therefore, once p is fixed, the AE is trained to compress and reconstruct Y, which enables the AE to compress samples with different number of symbols and variable proportion of symbols per constellation point.

In consequence, the algorithm module in the telemetry agent runs both the map and count and the encoder, and exchanges Z for every input sample X with the decoder running in the telemetry manager through the gRPC interface. The algorithm in the telemetry manager uses the decoder to reconstruct the count of symbols in each grid cell (Y^*) and then, generates X^* by re-sampling Y^*, i.e., the number of symbols in each grid cell is generated by randomly choosing I and Q components within the range of the grid cell. Once generated, the sample X^* is stored in the telemetry DB to be subsequently analyzed. Note that reconstruction can be performed also in the telemetry agent, e.g., for veracity checking purposes, like detecting outliers and/or anomalies.

C. Data Summarization

In the two previous techniques, telemetry data are propagated from the observation point to the telemetry manager with the same frequency, i.e., every time a new sample M is collected from the observation point, a subset of data representing it is generated and conveyed to the telemetry manager. Assuming a high collection frequency, this policy entails large volume of data being conveyed. However, this is not needed in general in normal conditions. Hence, we could measure variations in the computed features to decide whether a new sample M or a representation of it needs to be sent to the telemetry manager. In case of no significant variations are found, the telemetry agent can send averaged values of the features with a much lower frequency, thus reducing the volume of telemetry data being conveyed.

Algorithm 1 presents the proposed data summarization procedure. The algorithm receives the set of computed features Φ and returns whether features need to be sent (Boolean variable *send*) and if needed, the set of features Φ' that can be either those received as input or averaged ones. To that end, the algorithm maintains and updates the following internal data, which are assumed to be initialized beforehand: *i*) *H* is a time series database containing the last *w* values of each feature; *ii*) Φ^{avg}, with the average value of the features stored in *H*; *iii*) *R*, with the range of variation of each feature, computed as Φ' (+/-) α times the standard deviation of values in *H*; and *iv*) *count*, with the number of consecutive telemetry periods where all features remain within the range *R*. Besides, *maxcount* defines the interval to convey averaged features.

Algorithm 1. Data Summarization.

INPUT: Φ
OUTPUT: *send*, Φ'

1: *out* ← False
2: **for each** $\varphi \in \Phi$ **do**
3: **if** φ.value < *R*[φ.id].low OR φ.value > *R*[φ.id].high **then**
4: *out* ← True
5: *H*[φ.id].update(φ.value)
6: Φ^{avg}[φ.id] ← avg(*H*[φ.id])
7: *R*[φ.id].low ← Φ^{avg} [φ.id] - α· std(*H*[φ.id])
8: *R*[φ.id].high ← Φ^{avg} [φ.id] + α· std(*H*[φ.id])
9: **if** *out* = True **then**
10: *count* ← 0
11: **return** True**,** Φ
12: *count* ← *count* + 1
13: **if** *count* = *maxcount* **then**
count ← 0
return True**,** Φ^{avg}
return False, -

Before starting with feature analysis, we assume that all features stay within the range defined in R, by setting auxiliary variable out equal to False (line 1 in Algorithm 1). Then, input set Φ is firstly processed to find any feature that is out of the range R. If so, out is set to True (lines 2-4). After this, H, average features Φavg, and range R are updated accordingly (lines 5-8). Once all features have been processed, the output of the algorithm is prepared, which leads to three different cases. In case that at least one feature is out of range, count is reset and the input features Φ are returned (lines 9-11). Otherwise, count is increased and, if maxcount is reached, it means that a period of low frequency collection has been achieved, so count needs to be reset to 0 and averaged features Φavg are returned (lines 13-15). Note that in both previous cases, send is True in order to indicate that features must be conveyed. However, if all features are within the range and maxcount is not reached, then there is no need to convey any feature from agent to manager (line 16).

Algorithm 2. Main Procedure.

INPUT: *sample*
OUTPUT: *res*

1: Φ ← FeX(*sample*)
2: *send*, Φ' ← summarization(Φ) (Algorithm 1)
3: **if** *send* = False **then return** ∅
4: **if** *isIQ*(sample) = True **then**
5:

```
6:      Z ← compression(sample)
7:      return {Φ', Z}
        return {Φ', sample}
```

Algorithm 2 shows the main process that needs to be performed every time a new measurement becomes available at the telemetry agent. The output of this algorithm is the data that needs to be conveyed through the gRPC interface to the telemetry manager. Note that the result can be empty, i.e., no data need to be conveyed. The first step is to compute features $\Phi_{(\cdot)}$ from the input sample (line 1 in Algorithm 2). Then, the data summarization procedure (Algorithm 1) is executed and, in case that there is no need to send data, empty set is returned (lines 2-3). Otherwise, if the sample is an IQ constellation, the AE-based compression is applied, and both the features after data summarization (Φ') and latent space (Z) are returned (lines 4-6). On the contrary, i.e., if no compression is needed, e.g., the sample is an optical spectrum, features Φ' and original sample are sent (line 7).

5 CONTROL OF ACCESS INFRASTRUCTURES

5.1 CONTROL OF SDM PON

The management of SEASON PON architecture involves the control of optical devices like the Optical Line Terminal (OLT) and spatial devices such as spatial aggregation/disaggregation elements. Compared to state-of-the-art PON architectures the targeted solution addresses the need for capacity scaling thanks to spatial parallelization while at the same time taking into account the orthogonal requirement of sustainability. This is achieved by adapting the number of active spatial channels to the fluctuating demand of traffic typical of optical access scenarios. The section discusses the controller components and interfaces.

5.1.1 PON Infrastructure Control

SEASON targets the SDN control of Passive Optical Networks combining SDM switching, i.e. SDM-PON. In the scope of SEASON, SDM is exploited through the aggregation/disaggregation of spatial channels which may be deployed as single core fibers in multi-fiber scenarios of cores in multi-core fibers. As the utilization of SDM-PON within SEASON targets energy saving through components activation/deactivation associated with spatial aggregation/disaggregation, the control layer is also responsible to perform energy consumption measurements.

5.1.2 Design

The design approach follows SDN principle of logical centralized control and information model abstraction. An optical access controller is responsible for setting up connectivity with targeted performance for specific services combining resource accommodation at PON elements (OLT and ONU) and spatial fiber infrastructure (spatial switch and multi fiber or MCFs).

- The north bound interface (NBI) is based on REST APIs to allow the interaction with higher hierarchical elements (e.g. TeraFlow controller)
- The south bound interface (SBI) is based on existing data models (e.g. CALIX YANG data models) as well as new data models purposely built in the scope of SEASON SDM-PON Control.

The SDN control has to be able to consider multiple resource allocation /multiplexing levels, namely, time, spectrum, and space. It is part of the controller tasks to perform resource allocation considering the restriction imposed at each level. Regarding this aspect, it is crucial to integrate the output from WP3 regarding SDM-PON capabilities and control restrictions. Design decision in WP3 also impact energy saving aspects. The controller exposes power monitoring data through REST API NBI.

5.1.3 Data Plane capabilities

WP3 carries out the SDM-PON design and address technological decisions which outlines flexibility and restrictions in resource allocation in PONs, including:

- Time domain resource allocation, including the ability to dedicate resources or prioritize specific services to offer low-latency connectivity.
- Spectrum resource allocation, which affects the OLT design and, in turn, controller design. This aspect is essential in the investigation of integration on point-to-multipoint optics and SDM-PONs
- Space resource allocation, which impacts control strategies in terms of switching time and power loss.

5.1.4 Implementation

The work in WP4 includes the implementation of an SDN-controller supporting REST-API as north bound interface. The controller can dynamically provide connectivity with target performance for a specific service. The controller supports also the capability to export energy consumption information through the above-mentioned NBI. Additionally, future direction of WP4 includes the development of software agents at data plane nodes. These agents relies on NETCONF and introduce or enhance (if already present in infrastructure components) remote controllability and energy monitoring. A graphical overview of the control architecture is given in

Figure 5.1.

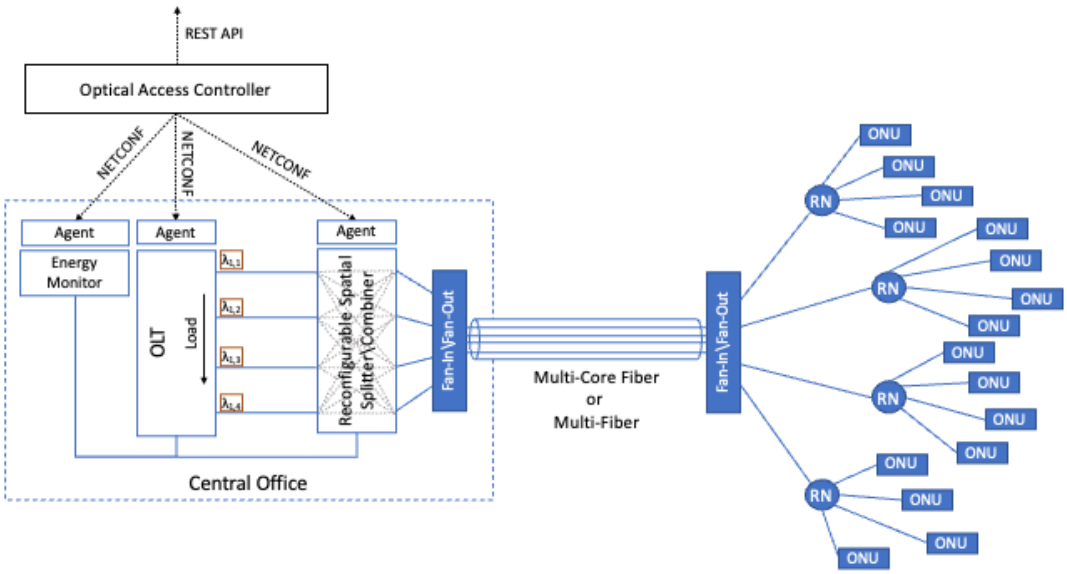


Figure 5.1: Graphical overview of the controller architecture

For the implementation of the southbound interface, we relied on the Calix YANG model. A graphical excerpt being given in Figure 5.2. This model identifies `ONTs` by their serial numbers and profiles and includes `Ethernet` (e.g. `1/1/x1`) and `ONT-Ethernet` (e.g. `1/x1`) interfaces. These interfaces bridge ONTs with Ethernet interfaces, apply ingress and egress rules, and specify VLAN mappings. Traffic profiles are associated with Ethernet traffic `policy-maps` and `class-map-ethernet`. These configurations are managed via the PON (e.g. `1/p1`) interface, which handles scheduling directives and dynamic bandwidth allocation.

```
ont 1
 profile-id    801XGS
 serial-number 47846e
 pon-us-cos user-1 pon-us-cos-
profile ont1_assured
 pon-us-cos user-2 pon-us-cos-
profile ont1_expedited
!
```

```
pon-cos-profile ont1_assured
 prio     2
 bw type explicit maximum 1000000 minimum
1000000
 cos-type assured
!
pon-cos-profile ont1_ef
 prio     4
 bw type explicit maximum 4000000 minimum
4000000
 cos-type expedited
!
```

Figure 5.2: Calix modeling examples.

Within WP4 the effective provisioning of services with expedited forwarding by configuring profiles that define parameters such as name, priority, bandwidth type, maximum and minimum bandwidth, and class of service (cos) type. These parameters are crucial for meeting the network's service levels and for accommodating various traffic requirements. An excerpt of configuration of an expedited forwarding service is given in **Errore. L'origine riferimento non è stata trovata.** 5.3.

```
<config>
       <config xmlns="http://www.calix.com/ns/exa/base">
           <profile>
               <pon-cos-profile xmlns="http://www.calix.com/ns/exa/gpon-interface-base">
                   <name>ont1_ef</name>
                   <prio>4</prio>
                   <bw>
                       <type>explicit</type>
                       <maximum>4000000</maximum>
                       <minimum>4000000</minimum>
                   </bw>
                   <cos-type>expedited</cos-type>
               </pon-cos-profile>
           </profile>
       </config>
   </config>
```

Figure 5.3: Expedited forwarding service provisioning via NETCONF.

The REST APIs exposed by the optical access controller are thoroughly documented according to the OpenAPI specification, with Swagger used for detailing and navigating the API's capabilities. This comprehensive documentation approach includes clear definitions of API endpoints, models, and operations, thus simplifying the management and configuration of the network (Figure 5.4). Focused on essential functionalities, the development of REST APIs has emphasized CRUD operations—GET, POST, PUT, and DELETE, all showcased within the Swagger UI. The integration of these APIs with NETCONF continues to be refined, highlighting the project's commitment to enhancing network control and management. As the SEASON project

progresses, there is an ongoing effort to expand the API set, informed by the collaborative contributions of partners and the evolving needs of the network infrastructure.

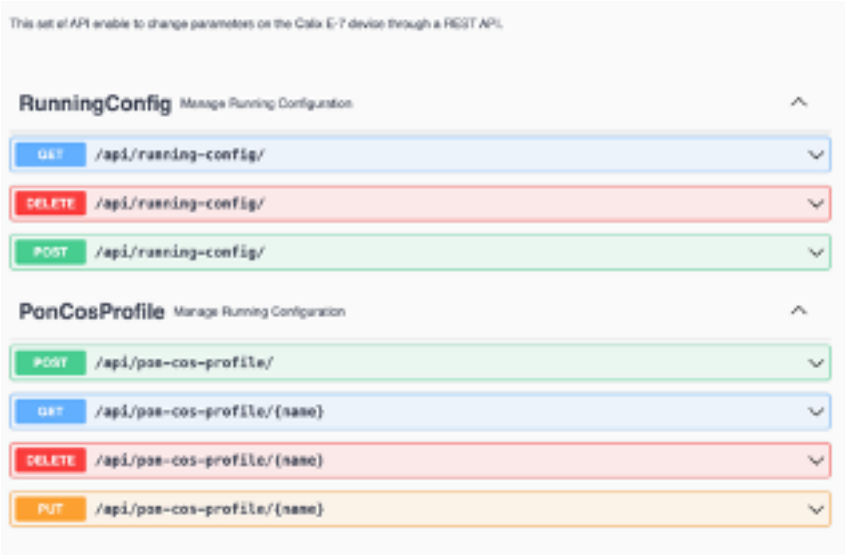


Figure 5.4: Optical Access Controller REST-APIs.

5.2 RAN INTELLIGENT CONTROLLER (RIC)

5.2.1 Architecture

Figure 5.5 shows the ONAP interfaces to manage, monitor and deploy various network functions in mobile network and gather telemetry data. The telemetry data(3GPP 28.552 stats) is available at both near RT RIC for xAPPs(via E2) and non-RT RIC for rAPPs(via ONAP High Volume-VES interface i.e. dotted purple arrows). Based on the telemetry data the ML rAPPs can instruct the SMO to optimize the configuration of compute resources or network functions, RAN nodes(RUs) and also the transport Network resources via SDN controller to save energy.

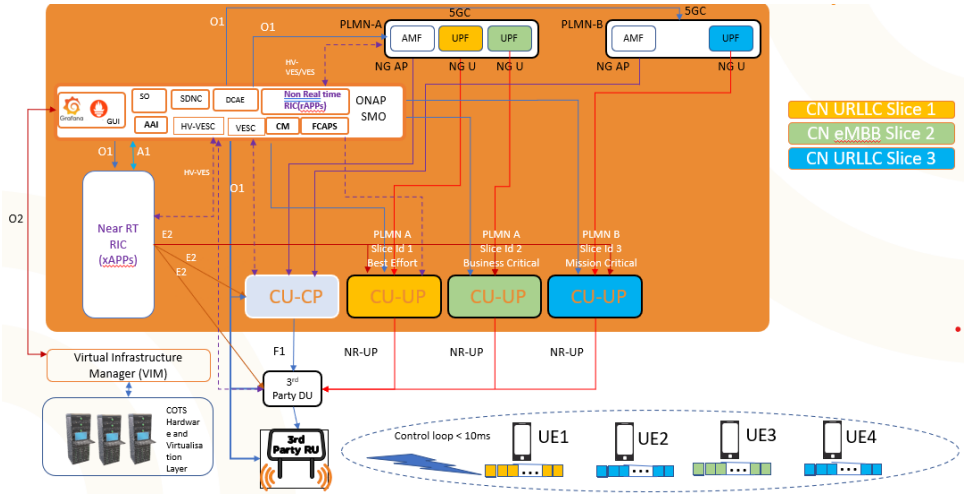


Figure 5.5: ONAP interfaces.

xApps or rApps can use Kafka bus telemetry received from the E2 termination nodes. The telemetry data is abstracted in standard JSON format. The SDN controller can use the telemetry data that is exposed by the data bus to monitor the traffic between the two components of a particular network slice and adjust the bandwidth of the flow rules accordingly.

For example, if the traffic between two components (gNB-DU and a gNB-CU-UP belonging to a slice) increases, the SDN controller increases the bandwidth of the flow rules to ensure that the traffic can flow smoothly. The telemetry data captured by the gNB-DU, gNB-CU-UP/CP can be based on counters that are available in each E2 node like number of RRC connections, number of PDU sessions, average DL delay, UL/DL SDU loss rate, average DL/UL throughput, total PRB usage, active UE per cell, ...

Energy saving can be achieved at multiple levels in the RAN by switching OFF the cells and slices that do not carry significant traffic based on AI/ML techniques. At cell level this can be done by estimating the predicted cell load based E2 KPI stats (PRB utilization in UL and DL, Number of PRACH preambles, Cell throughput, Number of active UEs, CQI distribution of UEs, Timing advance distribution of UEs) and UE measurements (RSRP, RSRQ and SINR) using an AI/ML model. If the predicted cell load is below a certain threshold for a configurable OFF period, the cells can be switched OFF. If the predicted cell load is above a certain threshold for configurable ON period, the cells can be switched ON. The CU-UP and transport network resources for high latency network slices can be dynamically switched ON and OFF by rAPPs in the non-RT RIC. The resources can be switched OFF when there are no calls and switched ON when there are active calls admitted for a slice by control plane during Admission control. This can result in significant energy and cost savings during quite periods when user plane activity is very low for such slices. In dual frequency networks with umbrella coverage provided by the Macro, small cells (DU and RU) can be completely switched OFF based on activity patterns during certain time of the day to save energy.

5.2.2 ORAN Evolution

Figure 5.5 shows the ORAN evolution towards 6G. It includes a new producer-consumer interface like Y1 which is still evolving and provides RIC telemetry exposure to 3rd party components like SMO in a standard way. The 6G architecture allows for full user plane, control plane disaggregation up to DU level. It also introduces 6G RAN technology enablers like Cell-free mMIMO at DU and RU level. The CU-UP and the other user plane components can be implemented in P4 switches. The control plane components with high latency expectations can be implemented as Kubernetes services on x86 or ARM platforms. The non-RT RIC accumulates the data from near-RT RIC and other non-3GPP interfaces into a data lake that can be used to train ML models that optimize RAN configuration. This data can also be fed to 6G Network digital twins.

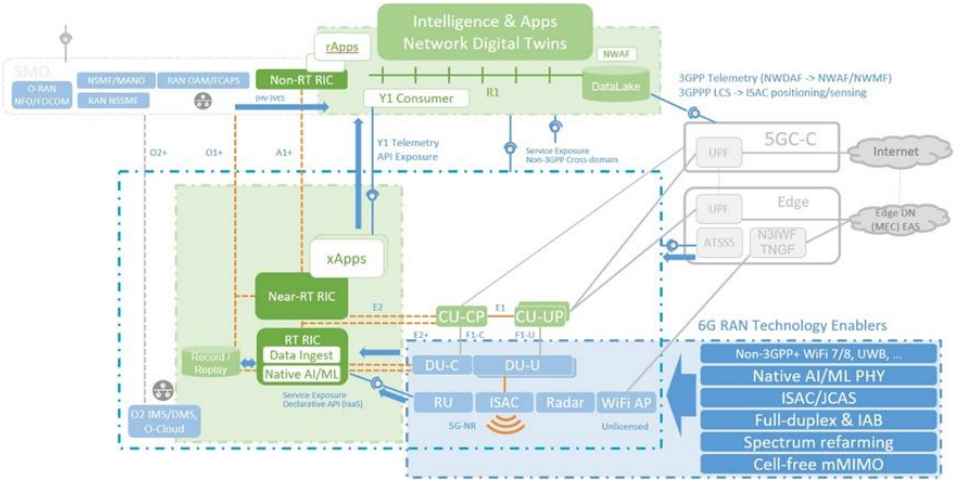


Figure 5.6: ORAN evolution towards 6G

The user plane network functions can be deployed by the SMO on virtualized hardware or P4 programmable SMARTNICs based on network slice type. xURLLC slices can be deployed on P4 programmable switches at network edge to achieve less than 1ms latency. eMBB user plane slices that require a lot of CPUs and are delay tolerant can be deployed on general purpose CPU infrastructure such as Kubernetes services. The SMO can be onboarded with both CPU based and P4 compiled variants of user plane network functions implementation. Depending on the slice SLAs specified, the SMO make use of AI/ML techniques to deploy the appropriate implementation of user plane network function and configure it accordingly. The SMO uses P4 runtime to deploy, configure and run P4 implementations of Network functions dynamically on P4 switches.

Based on KPIs received from CU-CP, CU-UP and the DU a rAPP in the SMO can optimize the transport network to be more energy efficient and reduce the operational expense by minimizing the number of switches connected to the mesh network.

6 TRANSPORT NETWORK CONTROL

The transport network management involves the control of optical and packet/optical devices in the underlying data plane. SEASON WP4 focuses on model-driven approach, and this depends on the YANG model adapted by the optical or packet/optical entities. Among the optical domain components, OpenConfig and OpenROADM YANG models are more prevalent where OpenConfig is used in Packet/optical devices. However, the scope of the work also considers evaluation of OpenROADM to control pluggables in packet/optical devices. Further details on the YANG model usage are discussed in the following subsections.

The section discusses the hierarchical controller components with ROADMs managed by OLS controller, which in turn controlled by an optical SDN controller. This optical SDN controller also controls packet/optical devices. This accelerates the evaluation of ONF-TAPI [Onf21] for usage between the optical SDN and OLS controller. As a result, the fully or partially disaggregated network control on the transport network can be verified.

6.1 INFRASTRUCTURE CONTROL FOR MBOSDM OPTICAL NETWORK

SEASON targets the SDN Control of an Optical Line System / Optical Network combining wide-band flexi-grid switching as well as SDM switching.

SDM switching, in the scope of SEASON, addresses either the Bundle of Fibers (BoF) deployments with fiber switching or basic core-switching. More complex SDM switching is left for further study. In this sense, the activity covers both the design and the implementation, along with the integration within WP5 with Data Plane nodes conceived in the scope of WP3.

6.1.1 Design

The design and development of the SDN controller for Multiband over SDM follows Software Defined Networking (SDN) principles and model driven development. A centralized SDN Controller is responsible for setting up and releasing flexi-grid connectivity services across an optical line system that combines both technologies.

- The north bound interface (NBI) is based on Open Networking Foundation (ONF) Transport Application Programming Interface (TAPI), version 2.1.3 but we are currently considering additional extensions.
- The south bound interface (SBI) is based on existing data models (notably for the flexi-grid layer) as well as on new data models based on SEASON WP2 and WP3 developments. The node capabilities and functions are developed in WP3.
- The SDN Controller manages an internal representation of the network tracking, notably, flexi-grid and SDM resources (such as a virtual topology), along with the state of each link and node (e.g., existing cross-connections).

- The topology may involve nodes that are single-level switching (e.g., pure ROADM or SDM switching nodes) and hybrid nodes combining both levels.

The SDN controller can consider multiple switching levels (we avoid the term layer, since these are aspects of the photonic or L0 layer) within the optical domain. Macroscopically, flexi-grid connectivity services may be multiplexed in one or more band or SDM services. It is part of the controller to be able to perform path computation and resource allocation, considering the restrictions imposed at each level.

6.1.2 Device Model

The preliminary "device model" for the node architectures under consideration along its initial support in the controller uses the term "core" and "core selective switch" but it applies to "SDM spatial lanes" in general, Standard Single Mode Fibers in a bundle or cores in a multi-core network. For a given SDM spatial lane, the spectrum is modelled as a contiguous set of 12.5 GHz basic slots, and spectrum allocation implies the allocation of a contiguous number "k" (e.g., k=4 for 50 GHz). The representation is a variable sized "frequency bitmap" and its size represents considered spectrum. A canonical representation of the node is shown in Figure 6.1, while overlay WDM over SDM network in Figure 6.2.

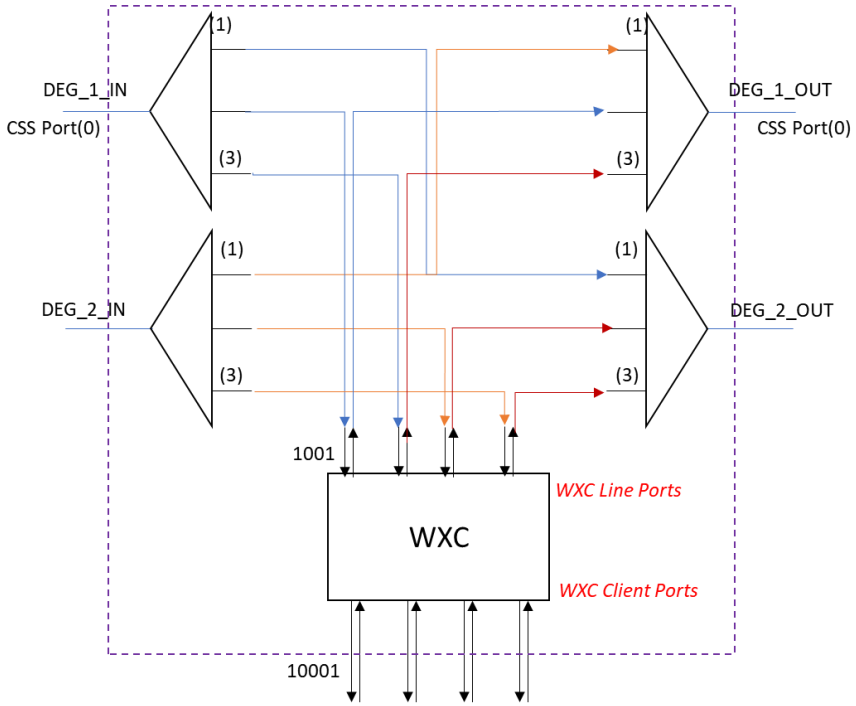


Figure 6.1: Sample Node Architecture showing a multi-granular WDM over SDM switching.

The canonical model relies on the entities of Core Selective Switch (CSS) and an ideal WXC (CDC). In this model, the common parameters are:

- A configurable number of supportable Cores (e.g., 2, 4, 5, 12, etc.)
- Number of Degrees of the node
- Number of CSS ports (assuming one CSS at the input degree and one CSS at the output degree).
- Number of WXC line ports and client ports (WDM services start and end at client ports).

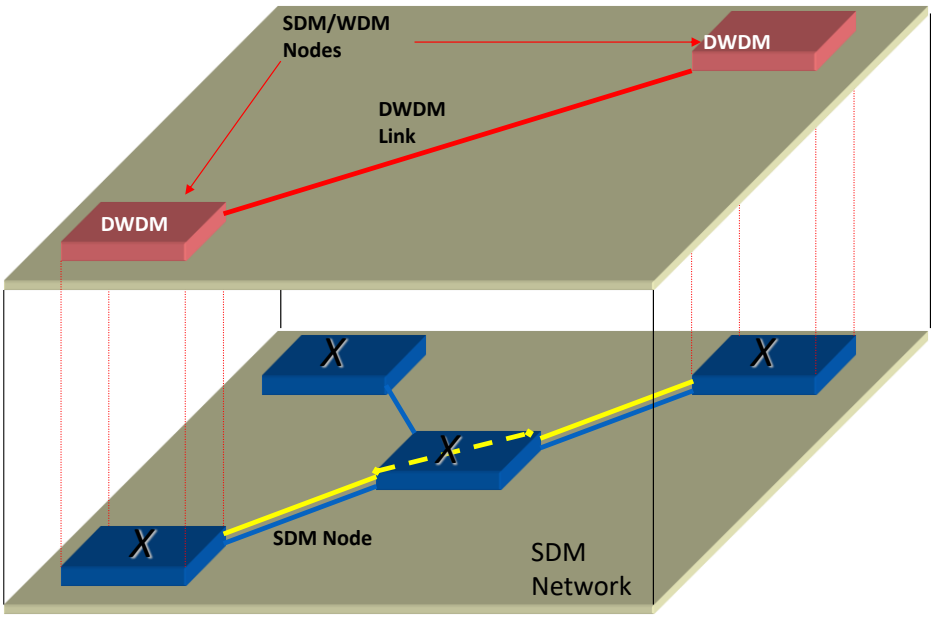


Figure 6.2: Overlay WDM over SDM network view. The SDM connection supports a virtual link at the DWDM layer.

6.1.3 Data Plane node capabilities

WP4 gathers the final WP3 input regarding the node design and to have a clear view on the hardware restrictions and how they affect to the flexibility of resource allocation algorithms, including:

- Flexi-grid switching capabilities, including the ability to configure media-channel cross-connection at the device level (e.g., a flexi-grid ROADM device)
- Wide Band switching capabilities, that is, the ability to switch, transparently, all the signals in a band, where the band is defined as an arbitrary frequency range (which may or may not correspond to a classical band, such as the C-band, L-band, etc.)
- SDM Core Switching or Fiber Switching in terms of aggregation / disaggregation (as in a Fan-In, Fan-Out)

Figure 6.3 shows the architecture of the MBoSDM Control Plane.

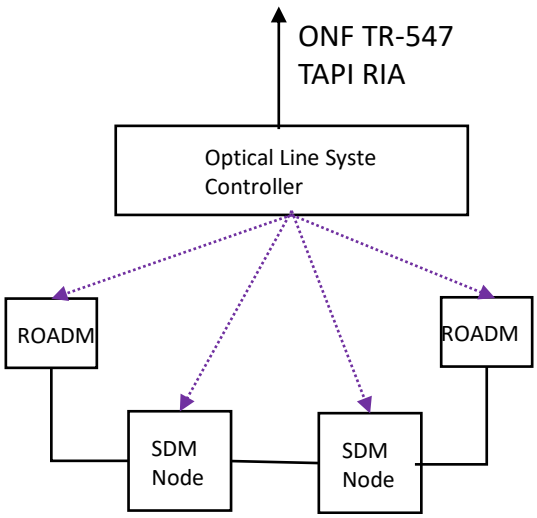


Figure 6.3: Architecture of the MBoSDM Control Plane.

6.1.4 Implementation

The work in WP4 includes de implementation of a Proof-of-concept, from multiple perspectives:

- A TAPI enabled SDN Controller able to dynamically provision and release connectivity services. This SDN Controller exports a Telemetry North Bound Interface, as explained above.
- Emulated networks to test the approach on a defined topology, in which the hardware is emulated.
- Software agents co-located in data plane nodes. This software agents relies on either NETCONF/RESTCONF (or similar) and enable the remote programmability of the nodes defined in WP3.

6.1.5 Simple 4-node scenario

A scenario has been defined to validate the provisioning of WDM services over dynamic core services. The demand requests a WDM Media Channel (Flexi-Grid), with, as input the requested slot width in GHz and between 2 node client ports (WXC client ports of the SDM/WXC node). The network state is defined in terms of its topology, including augmented nodes and links, as well as existing SDM connections (and supported virtual WDM links) and existing WDM connections and WDM links. The algorithm must provide, for a given demand: i) SDM connections to establish (will appear as virtual links) with allocated core for each SDM connection and ii) the WDM connections to establish and allocated frequency slot. Figure 6.4 shows the approach by manually requesting a service between two given service interface points (client ports).

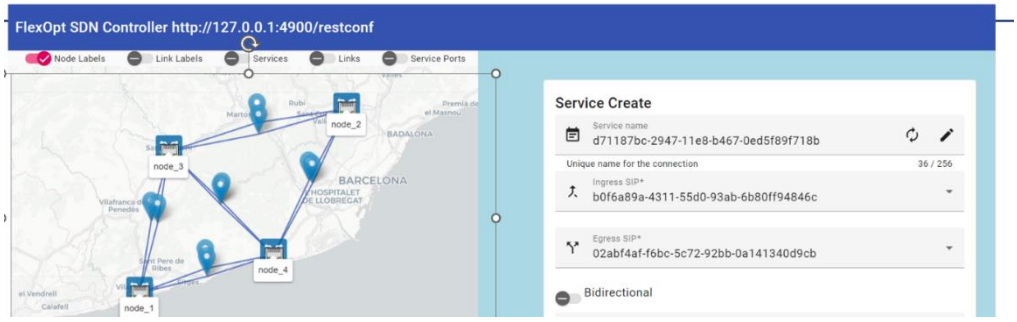


Figure 6.4: manually requesting a service between two given service interface points (client ports).

The controller proceeds to compute a path (since the network is empty, the path allocates a core between the source and the destination node which becomes a DWDM link between such nodes on top of which the WDM signal is routed).

The algorithm involves two steps: i) find a path using Yen k-shortest path and perform RSA allocating X slots as requested by the demand, ii) if step one fails, find a new core service between source and destination nodes, and perform RSA over the new allocated core (virtual DWDM link). If a core service cannot be provisioned the request fails.

As it can be seen in the Figure below, to support a WDM service, a core service (purple) has been setup across N1, N3 and N2 becoming a virtual DWDM link (grey) on top of which a 50 GHz DWDM service has been allocated. Figure 6.6 shows the virtual DWDM link between two nodes supported by an SDM connection.

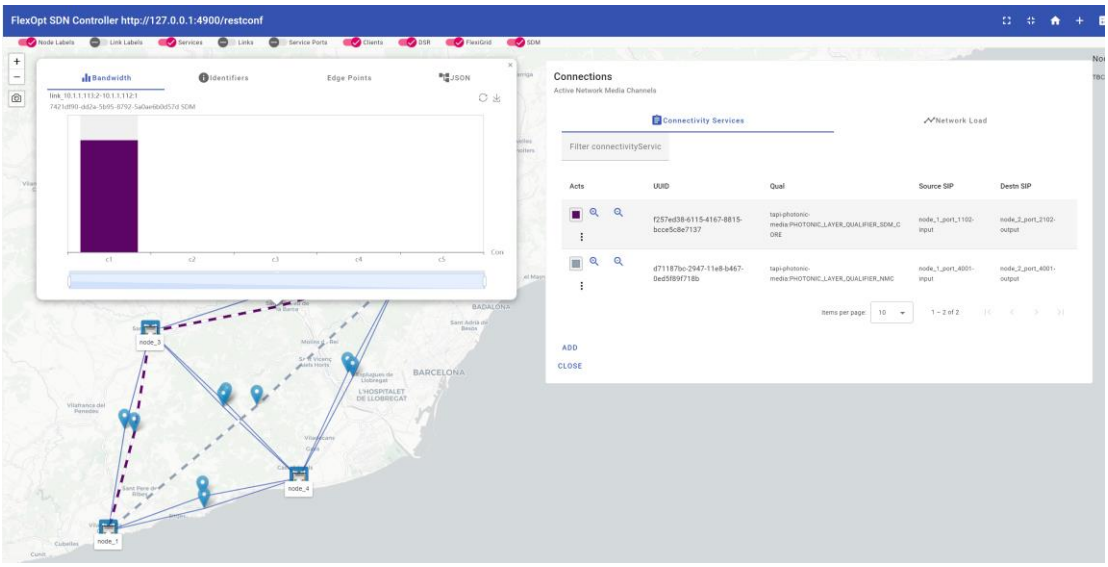


Figure 6.5: Dynamic provisioning of a fiber/core service in the multi-level node architecture.

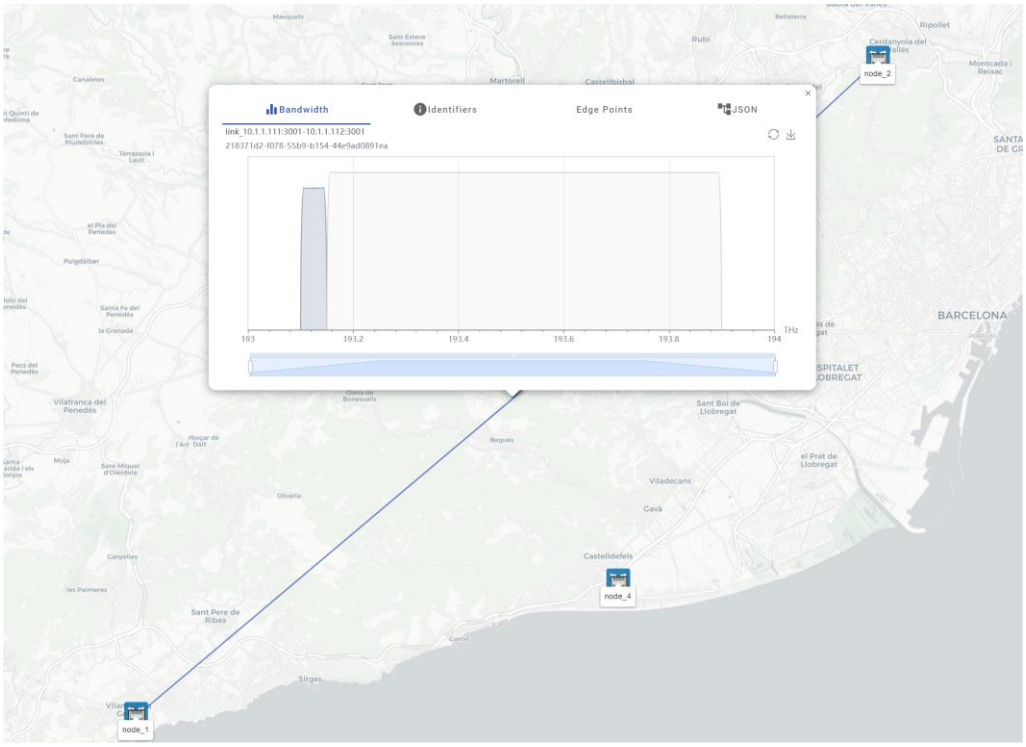


Figure 6.6: Virtual DWDM link between two nodes supported by an SDM connection.

6.2 PACKET / IP/OPTICAL CONTROL (SMARTNICS, WHITE BOX SWITCH)

SEASON has targeted the SDN Control of two types of packet/optical nodes, IPoWDM white box and smartNIC/DPU, equipped with coherent pluggable transceivers. The main features are:

- SmartNIC/DPUs makes use of OpenConfig control to configure the optical parameters of the coherent transceivers and possible IP-related parameters
- IPoWDM white box makes use of OpenROADM control to configure the optical parameters of the coherent transceivers. Telemetry data, according to the entire OpenROADM initiative, uses the OpenConfig solution.
- Packet configuration in SEASON makes use of the ETSI TeraFlowSDN Controller.
- SEASON solution uses open research questions related to the control of coherent optical pluggables in packet/optical nodes, e.g.using OpenROADM data models, taking into account current activities from standardization bodies like TIP and IETF. Two models are investigated:
    - one device – one controller, MANTRA control options (i.e. pluggable control under the domain of the packet controller
    - access from an agent to the specific device (i.e. pluggable control under the domain of the optical controller tunneled into the packet/optical device).

The ability to setup pluggable transceivers is critical. The TeraFlowSDN controller examines the OpenConfig control for the smartNIC/DPU to set the optical parameters of the coherent transceivers, as well as potential IP-related parameters. Furthermore, the SDN TeraFlow controller enables the configuration of packet processing rules for many types of traffic flows, including IP services. Depending on characteristics as source and destination addresses, port numbers, protocol types, or packet content, these rules can specify actions such as forwarding, rejecting, switching, or diverting packets. The driver features an easy-to-use graphical user interface (GUI) and a RESTful API for changing these packet processing rules. The GUI employs a drag-and-drop interface to allow users to rapidly create, change, delete, and inspect rules, and the API enables dynamic interaction with the controller using standard HTTP methods.

Moreover, the TeraFlowSDN controller supports the OpenFlow protocol, allowing for easy connection with other SDN devices and applications.

6.3 INFRASTRUCTURE CONTROL FOR TRANSCEIVERS

For the control of transceivers, namely advanced optical pluggable module (ZR/ZR+), SEASON has investigated the usage of the OpenROADM models implementing an agent able to support pluggables hosted in a SONiC based switch for the creation of optical channels on the collection of performance data to be sent using streaming telemetry.

The agent is based on ConfD that allows a good integration of gNMI based streaming telemetry code with the code performing the configuration on the device. The agent decouples the OpenROADM model processing from the action required by the underlying hardware exploiting the Linux dynamic libraries subsystem to load specific drivers at runtime. The drivers are associated to *circuit-packs*, an OpenROADM entity used to model atomic elements inside a device. Since, according to the model, every circuit-pack must have a *type* attribute, the agent requires that every *circuit-pack-type* have its specific driver and loads it whenever an *edit-config* rpc creates the first circuit-pack of that type.

The driver module must implement the functions to perform actions on the circuit-packs composing the device (e.g. frequency setting) and to get their features (e.g. supported frequencies and powers). Specifically, the agent that has been developed in SEASON models the SONiC switch as a device holding as many transponders as fitted pluggables and, taking advantage of the container-based architecture of SONiC, is implemented as a docker container.

As show in the following picture, the OpenROADM agent container exposes a NETCONF NBI based on OpenROADM YANG models and it is interfaced with the Redis database for the configuration of the transceiver's frequency and transmit power, leaving to the pmon container the task of the actual module configuration. Instead, performance parameters are directly collected from the module by means of the CMIS interface exploiting the *optoe* library, part of the SONiC software suite.

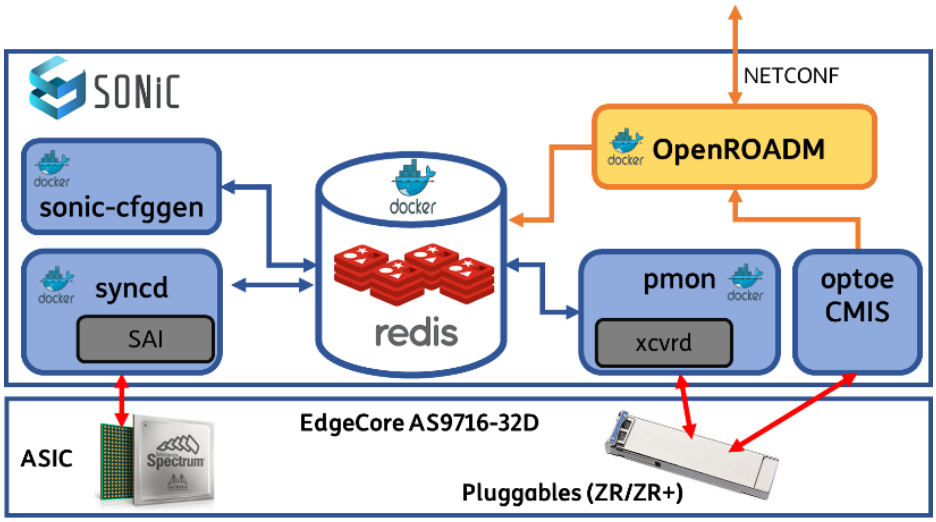


Figure 6.7: SEASON driver module for Control for transceivers control.

Monitored data are sent to a collector via gNMI based streaming telemetry using the Dial-out mode, i.e. the agent establishes the telemetry session with the collector using configuration parameters provided by the SDN controller.

6.4 NetDevOps and Continuous Integration / Continuous Development

NetDevOps ensures that network changes are small and frequent but also performed in a much more automated, efficient, and reliable way. DevOps is a software development strategy and culture that bridges the gap between the Development (Dev) and Operational teams (Ops), to build, test and release software faster and more reliably. NetDevOps is applying tools, concepts, and methodologies from DevOps and Infrastructure-as-Code (IaC) to network operation, allowing network devices and infrastructure configuration and operation in a Network-as-Code (NaC) paradigm.

The approach also increases automation and monitoring to increase efficiency and reduce errors. NetDevOps applies a number of strategies from DevOps to address this issue. They mainly are:

- Automation: It takes what traditionally are manual procedures in network infrastructure and applies the principles of automation and scalability.
- Frequent but smaller updates. They are incremental and make each deployment less risky.
- Reduce manual intervention with IaC: Under this paradigm, network devices are provisioned using machine-readable definition files rather than physical configuration files.

CI/CD: A CI/CD pipeline (see Figure 6.8) is a series of steps to facilitate integration, running test scenarios and deployment of the application (see Figure 6.9). Initially it creates the compilation environment in the "Runner machine" according to the nature of the application (Java/Python/Docker). The typical pipeline has processes like code compilation, artifact generation, deployment, testing, monitoring and feedback. This shrinks iteration times to insignificance. It involves the following use cases,

- Device provisioning: the first step is to create the configuration file and then pushing the configuration onto the device;
- Data collection and telemetry using NETCONF and gRPC or custom-built code using various libraries;
- Configuration management: it involves actual deployment and management of configuration files to networking devices, e.g., for configuration of flexible transponders with optimal modulation format;
- Service provisioning and service lifecycle management.
- Automation: Telemetry monitoring and feedback, to trigger on actions including upscale/downscale/reconfiguration based on the performance values.

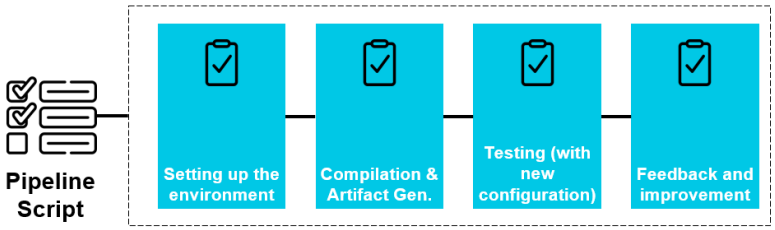


Figure 6.8: Example of a CI/CD Pipeline.

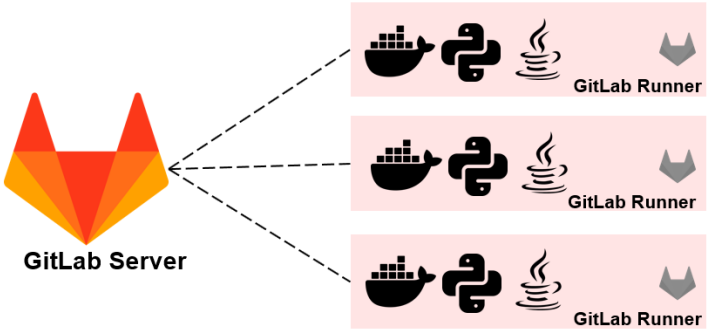


Figure 6.9: Example of Application deployed in CI/CD Paradigm.

6.5 TERAFLOW SDN CONTROLLER AND ORCHESTRATION

TeraFlowSDN (TFS) Controller is an ETSI-hosted open-source cloud-native SDN controller led by the Open Source Group for TeraFlowSDN (OSG TFS) [TFS23]. The TFS architecture, as depicted in Figure 6.10, serves as the foundation for the SEASON project, in which TFS plays two essential roles: IP/MPLS (packet) controller and Transport Network Orchestrator.

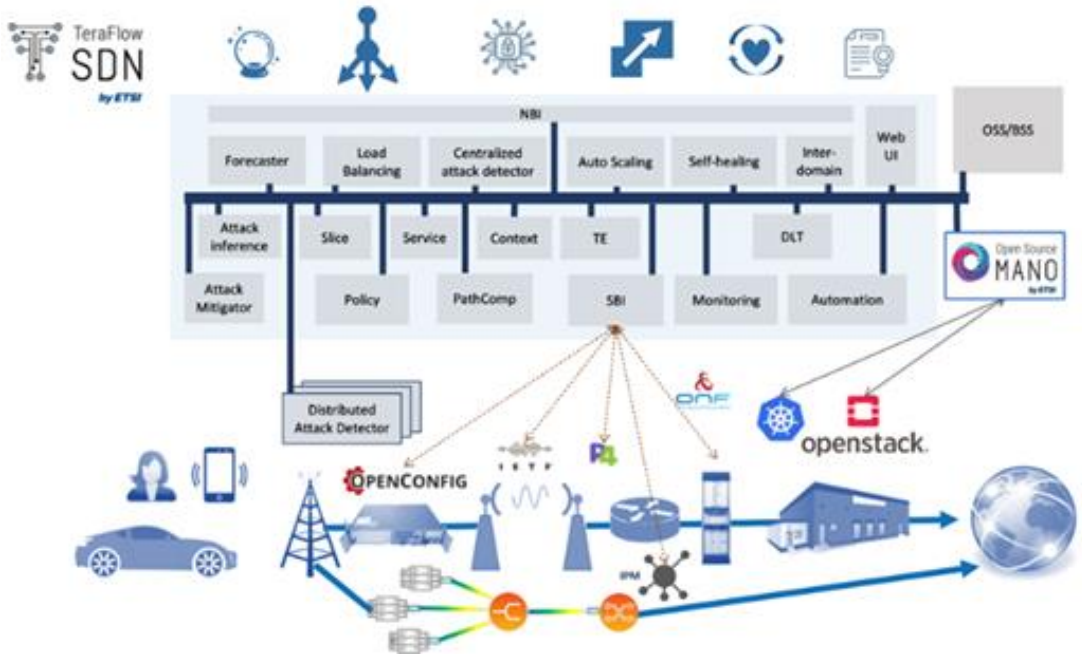


Figure 6.10: TFS architecture.

TFS as Packet Layer Controller: TFS's current release (release 2.1.0) already has several features, such as Layer 3 VPN service provisioning, L2 VPN, ACL management for security, and inventory information using Netconf/OpenConfig as the Southbound interface. TFS uses conventional L3NM/L2NM protocols in the northbound direction. It can also gather telemetry data from packet devices using gNmi/OpenConfig.

To support the SEASON packet/optical/DPU and colorful pluggable modules, the OpenDevice SBI driver module, which is responsible for establishing connections with network devices and controllers, needs to be upgraded. This update, as reported in the following, provides complete control over these aspects. Furthermore, the Northbound interface, which is now housed in the computation component, needs to be enlarged to include the interactions required to create optical communication from the packet/optical devices.

Several situations for controlling pluggable interfaces are examined in the SEASON project. Notably, much work is devoted to the investigation of the SINGLE management MANTRA situation, in which the packet controller serves as the lone entity accessing the packet box [Gon21]. The process for this scenario is being pursued for IETF standardization, as stated in the

draft paper[1]. It is important to remember that SEASON is still open to various architectural approaches and situations, such as the integration of several controllers.

TFS as a Network Orchestrator: The TFS context component serves as the central entry point for executing actions such as reading, updating, or removing components from the TeraFlowSDN controller database inside a multi-domain, multi-technology context. This component efficiently manages a wide range of objects, including topologies, devices, linkages, services, and connections. The context component of the SEASON project is improved to contain critical information essential for integrating varied transport sectors.

The TAPI (Transport API) SBI driver, which interfaces with the optical controller, is improved to access the increased TAPI context supplied by the MBoSDM (Multi-domain Bus and Topology Model) optical controller more effectively. This innovation guarantees that the TFS controller and the optical domain communicate and coordinate in real time. In addition, a new SBI driver has to be designed expressly for the PON (Passive Optical Network) domain, including the appropriate access domain abstract representation into the Context module.

TFS now offers an OSM-TFS interface based on the IETF L2VPN Yang service model for northbound requests. However, the viability of using the more generic IETF Slice model as the SEASON Orchestrator NBI (Northbound Interface) is fully studied within SEASON project. Customers will be able to request intent-like connections using the IETF Slice model, which will then be translated into the exact domain-specific resources necessary to complete the requested connections. This method improves the SEASON Orchestrator's flexibility and agility, allowing it to accommodate a wide range of client requests while also supporting optimal resource allocation within the network infrastructure.

6.5.1 TFS Controller for IPoWDM

In SEASON, the TFC SDN Controller has been extended with an OpenConfig SBI driver module. The objective is to support OpenConfig white boxes as well as the the novel concept of DPU/SmartNICs equipped with coherent pluggable modules.

The new driver successfully enables the topology discovery ad the onboard of the devices. Figure 6.11 and Figure 6.12 show an example with two OpenConfig NETCONF transceivers interconnected through two ROADMs that are successfully discovered and onboarded. Then, the TFS driver discovers automatically the device endpoints and its components.

As shown in the subsequent figures, the device is able to successfully configure the transceiver parameters, such as frequency and operational mode.

[1] https://datatracker.ietf.org/doc/html/draft-poidt-ccamp-actn-poi-pluggable-02.

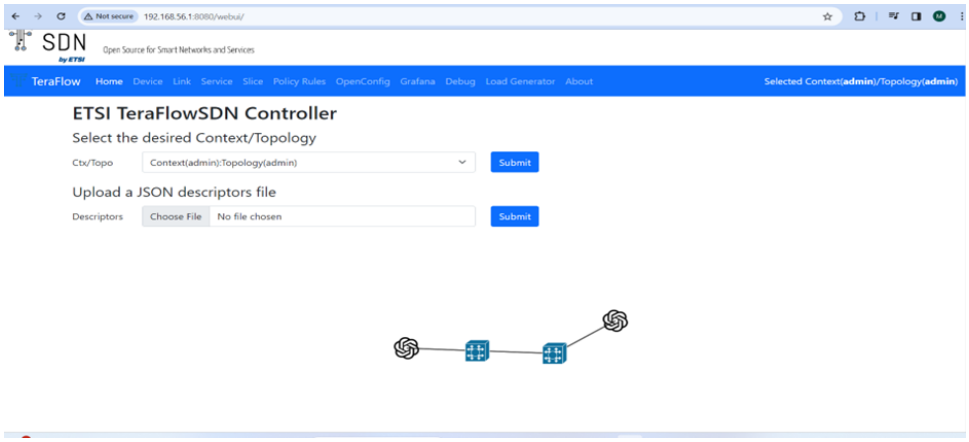


Figure 6.11: TFS topology discovery.

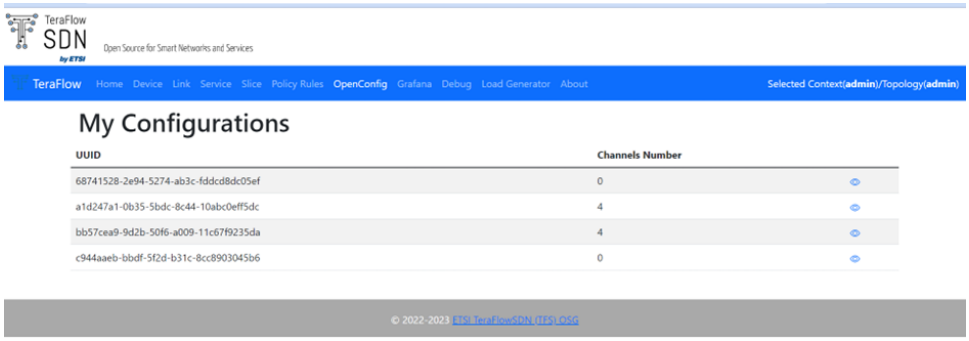


Figure 6.12:TFS device onboarding.

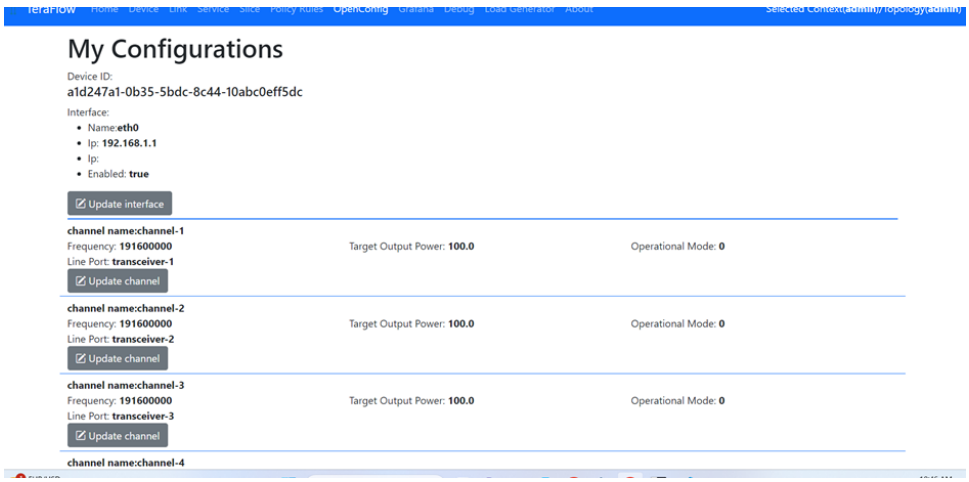


Figure 6.13: TFS list of components.

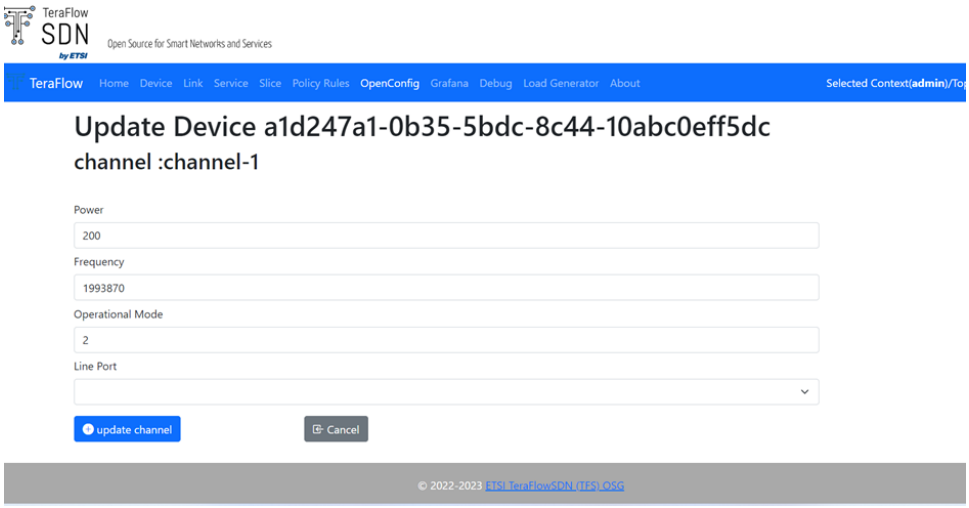


Figure 6.14: parameter configuration in terms of frequency, power and operation mode related to channel 1.

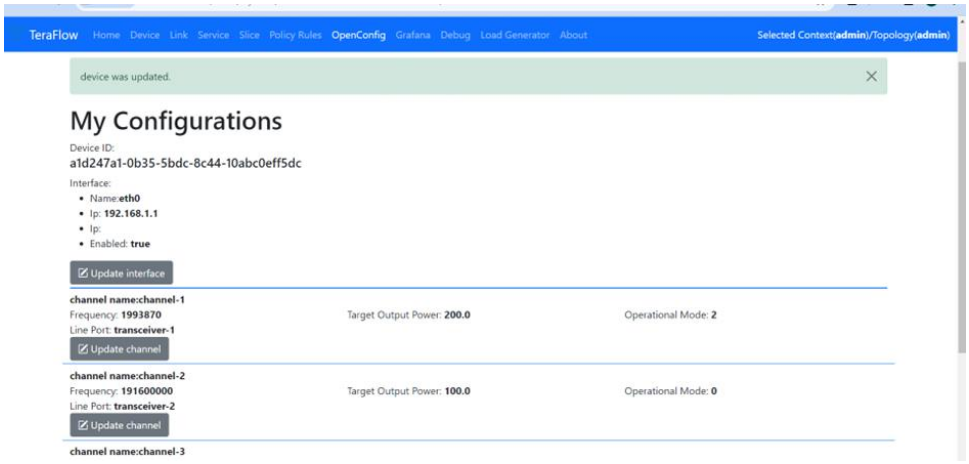


Figure 6.15: TFS driver successfully updated upon configuration changes.

6.5.2 TFS Orchestrator

SEASON benefits from several new features and changes to the TeraFlowSDN (TFS) controller technology. Pay special attention was given to the integration and functioning of the North Application Programming Interface (NBI).

The NBI serves as TFS's entry point, letting it communicate with an external NFV Orchestrator. This implies that TFS can handle the lifecycle management of network connection services across remote data centers or cloud locations automatically. In other words, the NBI serves as a front-end interface for the NFV Orchestrator to receive, process, and trigger the creation/update/deletion of network connections that fulfill the needs of the network services (using both cloud and network resources). To fulfill required activities (such as adding,

modifying, or removing connectivity links), the NBI interacts with other components such as the Service Component.

The NBI bridges the gap between the OSM WIM connector and the Service Component through translation and API mapping. OSM uses a REST API to register and request different network connectivity actions including creation, update, and deletion. The NBI can now receive and process requests for network connectivity services with specific endpoints, deploy energy target-oriented network services, and request discontinuous (primary and backup) network connectivity services thanks to the implementation of TFS version 2.1.0.

For northbound requests, TFS now provides an OSM-TFS interface based on the IETF L2VPN Yang service model. The SEASON project is thoroughly investigating the practicality of employing the more general IETF Slice model as the SEASON Orchestrator NBI (Northbound Interface). Customers are then able to request intent-like connections via the IETF Slice model, which subsequently is translated into the exact domain-specific resources required to complete the requests. This strategy increases the SEASON Orchestrator's flexibility and agility, allowing it to respond to a wide range of client demands while also facilitating effective resource allocation within the network architecture.

In addition, to the enhancements mentioned above, SEASON introduces several new features and changes to the TeraFlowSDN (TFS) controller technology. These updates further enhance the integration and functionality of the North Application Programming Interface (NBI). The NBI acts as the entry point for TFS, enabling communication with an external NFV Orchestrator.

The new features are the following:

- Multivendor behavior: The OpenConfig driver is enhanced to support multivendor configuration. The motivation is that each vendor (and even each O.S.) in some cases, the YANG model is similar to the one used by other vendors, but deviations are applied (ideally documented in a deviations file, in the worst case, detected in tests) and, sometimes, it is necessary to use proprietary rules (for example, the mandatory presence of some parameter, or a particular name or a fixed value in a field).
- Hardware inventory: Currently, the hardware inventory of equipment is supported according to [RFC8348]. This is extended to support the hardware inventory of optical devices.
- Pluggable control: this functionality starts from the orchestration from a hierarchical controller, separating the calls to the OLS to request a channel, and the calls to the IP controller for the configuration of the IP/Optical link. To this enhanced by adding the creation of an NBI with the IETF format (an RFC will be published soon).

The Open-Source Management and Orchestration (OSM) system serves as a platform enabling effectively managing and orchestrating network services. This is made up of various elements, each having a distinct function:

1. Device (South-Bound Interfaces - SBIs): This component communicates with network equipment via South-Bound Interfaces via pluggable drivers. The drivers oversee putting various network protocols and data models into action. The NetConf/OpenConfig Driver

API, for example, is used for packet routers, but the Transport API (TAPI) is used for the Optical Line System (OLS).

2. Context: The Context component saves network settings (for example, topologies, devices, connections, and services) in a No-SQL database (for example, Redis). It improves concurrent access and includes a Database API for switching between backends. It also offers publish-subscribe protocols to disseminate change events to other TeraFlow OS components using Google Remote Procedure Call (gRPC).
3. Service: The TeraFlow OS's Service component oversees the life-cycle of various connection services. It supports a variety of services and has a Handler API that allows network operators to integrate and fine-tune service behavior.
4. Automation: The Automation component supports Event-Condition-Action (ECA) loops for building network-wide automation methods. These loops carry out activities such as booting up new devices and setting interfaces and forwarding tables. Automation is triggered by specified events, follows preset conditions, and responds with a set of actions.
5. Monitoring: The Monitoring component maintains the many measurements (Key Performance Indicators - KPIs) that have been established for network equipment and services. It records monitoring data relating to these KPIs in a time-series database (e.g., InfluxDB) and makes the data available to other components.
6. Compute (North-Bound Interface - NBI): The Compute component provides a North-Bound Interface (NBI) based on Representational State Transfer API (REST-API) to external systems such as NFV (Network Functions Virtualization) and MEC (Multi-access Edge Computing) frameworks. It converts their queries into TeraFlow OS requests.
7. Web-based User Interface (WebUI): The WebUI component offers a graphical user interface via which users may observe network condition and submit operational requests to TeraFlow OS components. It interacts with TeraFlow OS components via gRPC-based APIs and uses InfluxDB querying capabilities to supply Grafana dashboards for viewing network state and important KPIs.

In conclusion, the TeraFlow OS components are built as microservices, largely in Python (except for Automation, which is built in Java). They operate in a Kubernetes-based environment and communicate with one another using well-defined TeraFlow-specific gRPC-based messages and services.

6.6 Centralized and Distributed Approaches to Control Optical Point-to-Multipoint Systems Near-Real-Time

Optical networks provide huge bandwidth and thus, they can support the expected increment in capacity and reduce delay, while saving capital (CAPEX) and operational (OPEX) expenditures for network operators. However, such requirements make it so that optical solutions brought

from the core to the metro, cannot be extended further to the access segment without important upgrades. On the one hand, current capacity allocation (e.g., 400Gb/s) is too coarse for the access, whereas finer allocation (e.g., 25 Gb/s) will help to reduce overprovisioning, and thus CAPEX. On the other hand, the control plane of optical networks is based on the centralized SDN model, which was designed to support dynamic provisioning, but not near-real-time operation (e.g., in a few sec.) to cope with highly dynamic traffic scenarios.

A solution that provides fine capacity granularity is that of Digital Subcarrier Multiplexing (DSCM), which allows for one single laser to digitally generate multiple Nyquist subcarriers (SC) that can be activated or deactivated independently. DSCM has shown important CAPEX savings by exploiting point-to-multipoint (P2MP) connectivity. In a P2MP connection, a hub node communicates with multiple leaf nodes. The way to implement P2MP connectivity on DSCM systems is to assign a number of SCs contiguous in the optical spectrum (referred to as a *spectral allocation*) to each leaf node, so each one can communicate with the hub node independently of the others.

In addition, OPEX savings can be obtained by exploiting DSCM capability for operating the different SCs independently, so just enough capacity to support the input traffic is provided, which reduces energy consumption. However, that requires near-real-time operation, as well as some anticipation to give time to timely activate the SCs, so that they become available when it is required. To provide near-real-time operation and capacity anticipation to deal with highly dynamic traffic intelligent agents running on top of the transponders can manage SC activation / deactivation autonomously, hence relieving the SDN controller from near-real-time operation.

In P2MP systems, near-real-time SC operation can increase the number of leaf nodes that can participate in a P2MP tree with respect to the static counterpart. For instance, assuming that every leaf node is equipped with a four-SC optical transmitter (Tx) and the hub node with a sixteen-SC receiver (Rx), the P2MP tree can accept up to 4 Txs. However, if SC activation could be controlled near-real-time as a function of the input traffic at every leaf node, the number of leaf nodes can be increased to 6 or 7, which would entail a large CAPEX reduction.

As previously introduced, DSCM facilitates the deployment of P2MP optical connectivity, since SCs sourced from a single hub node can be assigned to different leaf nodes. In the reverse direction (MP2P), SCs generated from different leaves can merge to connect the source nodes to the hub node. For illustrative purposes, Figure 6.16 presents an example where 4 leaf nodes are connected to a hub node in an MP2P connection. In the example, the hub node can support 16 SCs and each leaf node is assigned 4 contiguous SCs, while ensuring that each SC is assigned to one single leaf node, so as to avoid SC overlap (also referred to as *oversubscription*) since it leads to data loss. Comparing the MP2P optical connection in Figure 6.16 to a regular point-to-point (P2P) one, we observe the gain in the total number of transponders that are required, i.e., 4 Txs for the leaf nodes and 1 Rx for the hub node in the case of the MP2P connection in contrast to 4 Txs and 4 Rxs, respectively in the case of optical P2P connectivity.

The above observation can be also stated in a slightly different way. In the example in Figure 6.16, 4 leaf nodes are serviced since DSCM Txs can generate up to 4 contiguous SCs and the DSCM Rx can process up to 16 SCs. The number of leaf nodes in a P2MP connection can be

further increased with dynamic SC management, which can assign SCs dynamically to the leaf nodes, so those not requiring the full capacity of the transponder to support the local traffic can give one or more of their SCs up to other leaf nodes with higher capacity requirements. This can result in cost savings, as more leaf nodes (e.g., 5 or 6) can be serviced, as well as in power savings, as not every SC might be required to be active to support the current traffic. For these gains to be fully realized, a control mechanism is necessary to ensure the proper MP2P connection operation, i.e., to avoid oversubscriptions and to assure that the capacity needs of every leaf node are met.

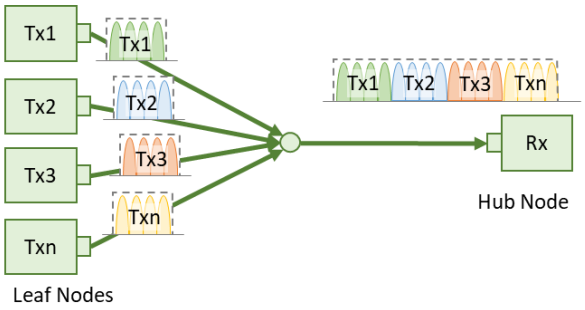


Figure 6.16: Example of an MP2P connection with 4 leaf nodes.

A classical approach is to deploy a *centralized intelligent SC manager* (CiMa) in the SDN controller (or in any other centralized control plane system) that gathers all necessary information and makes decisions on the spectrum that is allocated to each leaf node. The CiMa module might implement an ILP or heuristic algorithm for decision making, so that the optimal spectral allocation for the system as a whole is computed and communicated back to the leaf and hub nodes. Figure 6.17a shows the communication workflow for the centralized approach. Each agent collects traffic measurements (step 1 in Figure 6.17a) and uses them for traffic estimation, e.g., using some threshold, for the next time window, e.g., 1 min. The estimated traffic value is used to compute the estimated capacity that is be required for the next period, and both estimations are then sent to the CiMa module (2). Once all the estimations for the next period are received from the leaf nodes, the CiMa module computes the optimal spectral allocation to assign each leaf node that maximizes the overall traffic serviced by the leaf nodes. The CiMa module then communicates its new spectral allocation to each leaf node agent (3) and to the hub node agent the global spectral allocation (4). In summary, the number of messages, M_{Cent}, exchanged every operation cycle in a MP2P connection connecting a set T of leaf nodes under this centralized approach is defined in Eq. 1.

$$M_{Cent} = 2 \times |T| + 1 \qquad \text{Eq. 1}$$

The drawback of the centralized approach is that it requires near-real-time decision making to be performed by a centralized system in the control plane. Alternative solutions could rely on moving the intelligence to the node agents in the hub and leaf nodes to create a *distributed MAS*, thus relieving the centralized control system from near-real-time decision making. However, such distributed approaches entail that the agents need to coordinate among themselves, which increases complexity. In consequence, careful design of communication,

coordination, and decision-making capabilities of the agents needs to be carried out. In this work, we explore two main alternative approaches for the distributed solution: *i*) based on randomized MSG models; and *ii*) based on a collaborative MAS.

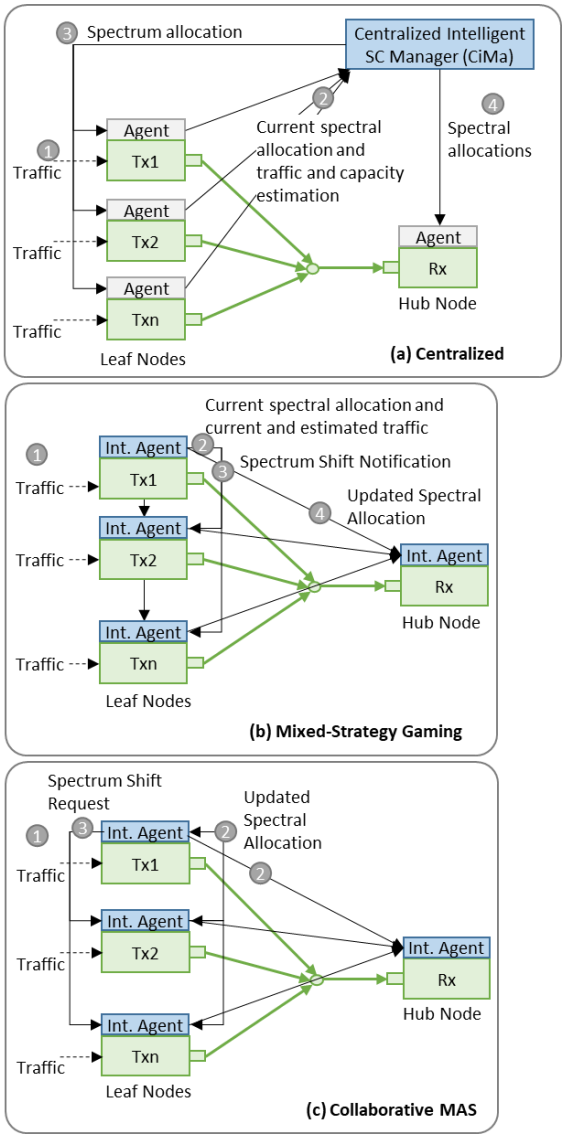


Figure 6.17: Communication workflow for the centralized (a), decentralized mixed strategy game (b), and collaborative multi-agent system (c) approaches.

The MSG approach (Figure 6.17b) includes a model within each agent that is used for local decision making based on the knowledge of the traffic and the spectral allocations of all the agents in the MP2P system. In this approach, each agent collects traffic measurements (step 1 in Figure 6.17b) and shares them with the other agents along with a traffic prediction and the current spectral allocation (2). Each agent then uses these data to calculate the modified spectral allocation using the MSG model. Once a new allocation is determined, it is locally implemented, and in case that a spectrum shift has been implemented, a notification is sent to the agents in the leaf nodes with neighboring spectral allocations. Finally, spectral allocation changes are shared with the agent in the hub node (4). Then, the number of messages, M_{MSG}, exchanged every operation cycle under the MSG approach can be computed as Eq. 2, where H is the

number of spectrum shift operations to be performed in the cycle, and variable r_t is 1 if the spectral allocation of leaf node *t* has changed; 0 otherwise.

$$M_{MSG} = |T| \times (|T| - 1) + 2 \times H + \sum_{t \epsilon T} r_t \quad \text{Eq. 2}$$

Equation (Eq. 2) results in an exponential increase in the number of messages under this approach. In addition, both the MSG and the centralized approaches, require strict synchronization among the nodes, as decisions are only made once all the required data is received. Contrarily, in the collaborative MAS approach, decision making is carried out considering the spectrum information currently available in the agent. Figure 6.17c shows the communication workflow, where each agent collects traffic measurements (step 1 in Figure 6.17c) and uses them to predict the traffic and the capacity for the next time period. The agents then decide their own spectral allocation based on such prediction and from the spectral allocations previously shared by the agents and update their allocations to the rest of the agents when some change (SC activation/deactivation) is performed (2). However, it might happen that the spectral allocation cannot be extended to satisfy its capacity needs because the neighboring SCs are already part of the spectral allocation of other leaf nodes. To solve that, agents have coordination abilities to change spectral allocations among them. Thus, agents can ask other agents with neighboring spectral allocations to shift their spectral allocation in order to release one neighboring SC (3). Note that only one SC shift is allowed, which is implemented by activating one non-active neighboring SC, thus enlarging the current spectral allocation, and then releasing one SC on the opposite side of the allocation (*make-before-break*). We consider a collaborative scenario where spectrum shift requests are always accepted if possible. If the shift request cannot be fulfilled, the agent would simply wait for the next time interval and traffic loss will happen. Note that in this approach, messages are exchanged when some change in the spectrum happens, which limits the total number of messages sent for the sake of scalability. Therefore, the number of messages M_{Det} exchanged every operation cycle under the collaborative MAS approach can be computed as Eq. 3. Note that messages are exchanged only in case of spectral allocation changes.

$$M_{Det} = \sum_{t \epsilon T} |T| \cdot r_t + 2 \times H \quad \text{Eq. 3}$$

Agents' functionality should be separated into specific components. Figure 6.18a presents the main functional modules of the Tx agents and their relationships for the centralized approach: *i*) the traffic predictor module is the only intelligent element in the Tx agents in this approach. It receives traffic monitoring from the transponder node and uses them for traffic prediction; *ii*) the SC manager coordinates SC activation/deactivation with the transponder node to enforce the spectral allocation received from the CiMa module; and *iii*) the communications module works as a proxy between modules inside the Tx agent and the CiMa module.

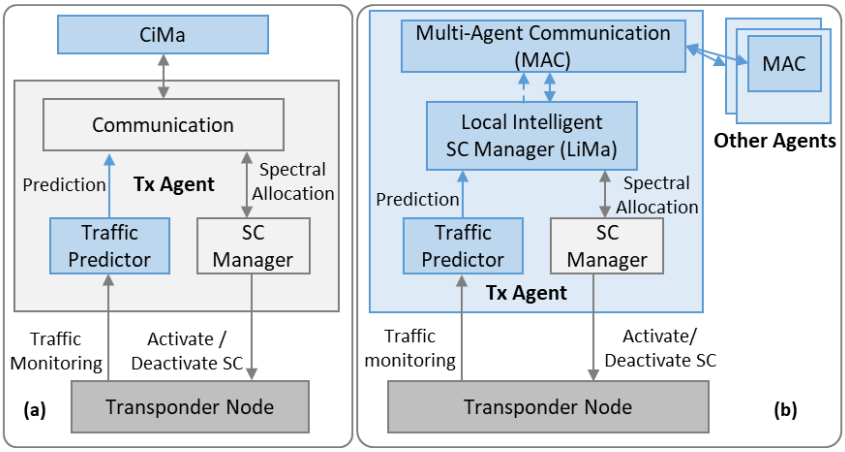


Figure 6.18: Agent design for the centralized (a) and distributed (b) approaches.

The architecture of the Tx agent in the distributed approaches is more complex (see Figure 6.18b). Extending the previous architecture from the traffic predictor and the SC manager modules, Tx agents include two additional modules: *i*) the Local intelligent SC Manager (LiMa) is in charge of making local decisions for the spectral allocation; and *ii*) the multi-agent communications (MAC) module that coordinates operation with the rest of the agents in the MP2P system. The MAC module is in charge of coordinate spectrum operations with the neighboring Tx agents if applicable. Similar to the Tx agents, the Rx agent also contains a MAC and an SC Manager that maintains a table <SC, Tx>. The architecture in Figure 6.18b needs to be complemented with the interactions among the agents in the MAS (Figure 6.17). Specifically, we consider two approaches based on collaborative MAS, where: *i*) traffic prediction is threshold-based, named distributed deterministic MAS (DD-MAS); and *ii*) an RL algorithm makes decisions on whether the capacity of the leaf needs to be increased or decreased based on the traffic, named multi-agent RL (MARL).

Four approaches supporting near-real-time SC operation in a P2MP connection have been presented and their general architectures and communication among the different elements has been discussed. In the centralized approach, traffic measurements are collected from the transponder in the leaf nodes. To anticipate traffic changes, a simple prediction based on a threshold-based approach is implemented. Such traffic prediction is sent to the centralized element (named CiMa), where an ILP modeling the spectral allocation problem is solved to the optimality. Then, a good performance can be anticipated. However, implementing the centralized approach for near-real-time operation requires that the whole process is executed in a limited time, which results in strict synchronization requirements and scalability issues if, potentially, hundreds of P2MP connections are operated.

Table 6-1: Pros and cons of every approach supporting near-real-time SC operation.

Approach	Advantages	Drawbacks
Centralized	• Optimal SC allocation provides good performance with minimal overprovisioning.	• Requires strict synchronization since leaf nodes need to collect traffic measurements and send them to the centralized module making decisions.
Mixed-Strategy Gaming	• Distributed decision making to liberate the SDN controller from near-real-time operation.	• Requires strict synchronization of the leaf nodes since they need to share traffic and spectral allocation. • Random loss even under low load and a huge number of spectrum operations (no cooperation among the leaves).
Collaborative Multi-Agent Systems (DD-MAS / MARL)	• Distributed decision making to liberate the SDN controller from near-real-time operation. • MARL can anticipate better for traffic changes. • Only spectral allocation is shared, so decisions can be made asynchronously, which also reduced the number of messages exchanged. • Few spectrum shifting operations (cooperation).	• Requires communication and collaboration among the leaf nodes. • The performance is (slightly) below the centralized approach.

Distributed approaches eliminate the need for centralized decision making by moving decision making to the agents of the transponders participating in the specific P2MP connection. Such isolation makes such approaches more scalable than the centralized one. Three different approaches have been studied: *i*) in the mixed-strategy gaming (MSG) approach each agent runs a model for allocating SCs. Agents compete for the available SCs to satisfy their individual capacity needs and release them as soon as they are not needed; *ii*) in the distributed deterministic (DD-MAS) approach, each agent makes SC decisions locally using a deterministic algorithm and can collaborate with other agents to perform spectrum shifting if necessary; and *iii*) the multi-agent reinforcement learning (MARL), which improves traffic /capacity prediction using a DRL algorithm.

A summary of the pros and cons of each approach is presented in

Table 6-1 Table 6.1.

6.7 CONTROL OF DSCM SYSTEMS

The main approach to manage pluggable models, followed in section 6.2, is the use of register-based information models, defined in Multi-Source Agreements (MSA) and recently in standards such as OIF-CMIS 2 [Oif22] and C-CMIS [Oif23] for coherent pluggable modules. This approach requires a tight integration between the host (router) and the pluggable module. When a new DWDM technology is introduced, as happens in the case of Digital Subcarrier Multiplexing (DSCM) modules, it requires updates in the relevant MSA/standard, which requires time and consensus across the industry.

In SEASON, we have investigated the use of a dual management approach proposed by Open XR Forum in [Ope22] and [Open23], by which advance optical functionalities of the optical modules are managed via a separate dedicated controller and is independent of the software in the host. Hence, there is no need to develop additional functions in the node agent, beyond those required for the basic CMIS-based access. To perform a proof of concept in SEASON, both whiteboxes and commercial routers were considered.

In the dual management approach, the physical interface of the DSCM pluggable module is split into several VLAN-based sub-interfaces. There is one sub-interface for each leaf (destination) and one sub-interface dedicated to the management of the optical module. The configuration of the packet interface is provided via the regular register-based CMIS interface, while the optical advanced functions are provided via the dedicated management ethernet-based interface.

6.8 INTEGRATION OF COMPUTING AND NETWORKING

The convergence of computing and networking is a fundamental change in information technology that integrates the capabilities of the networks and the underline infrastructure, and the computing systems [Tan21], [Hon16]. Driven by technologies such as the Service-Based Architecture (SBA) in the 5G Core and the deployments like the Multi-access Edge Computing (MEC) it allows dynamic allocation of the resources, improved performance, and seamless scalability and flexibility of the networks. In this regard, networking becomes more software-oriented, and computation more distributed and networked [Dua12]. SEASON takes advantage of this convergence so as to construct a resilient and enhanced architecture beyond of the current state of the art which is capable of meeting the standards set for the next generation networks. Central to this progression is the implementation of Kubernetes, which has emerged as a cornerstone in the orchestration and management of the underlying infrastructure. Through the integration of Kubernetes, SEASON has realized enhanced automation capabilities, facilitating the deployment, scaling, and management of cloud-native network functions and applications of the underline infrastructure with high efficiency and ensures seamless

coordination of the different network segments such as the RAN, transport and optical. This automation extends to the robust handling of dynamic service demands, showcasing Kubernetes' pivotal role in optimizing network operations and fostering a resilient, scalable service management platform owing for proactive action before minor faults escalate into major issues [Suk19].

Furthermore, the integration with TeraFlowSDN represents a notable advancement in SEASON's infrastructure management capabilities. The development of a refined interface for TeraFlowSDN facilitates a seamless acquisition of Key Performance Indicators (KPIs) from the network, enhancing the visibility and control over network operations. This interface enables the precise monitoring and management of network performance, ensuring the agility and flexibility of the network in response to evolving demands and conditions. Moreover, TeraFlowSDN allows for direct programmability of network control, which offers an advanced layer of network management via the abstraction of the underline infrastructure, further enhancing efficiency, and creating a more streamlined system for the CNFs and applications [Vil21]. By combining the computational offloading abilities of SmartNICs with the direct programmability offered by TeraFlowSDN, SEASON advances a more agile, efficient, and responsive network system, capable of adapting to the dynamic requirements of modern network environments.

Another key contributor to this synergy is the rise of AI-empowered network configuration. By leveraging advanced AI/ML algorithms SEASON has developed sophisticated mechanisms that guide the orchestration layer in making informed decisions about applications' instances placement. This optimization is driven by policies that account for various network parameters and service requirements, ensuring an optimal allocation of resources that aligns with operational efficiencies and performance benchmarks. Comprehensive information on this topic can be found in Chapter 8.2.1. Such innovations underscore SEASON's commitment to maximizing network functionality and responsiveness through the strategic application of AI/ML insights. These AI applications cover a wide range of tasks such as resource management, network topology optimization, optimization of the SDN, proactive fault detection and so on, thus contributing to the seamless coordination of the diverse network domains and the optimized management of the overall network traffic [Nac21]. In line with the ongoing declarative intent-based approach, the environments dynamically adjust to the computational needs defined by AI/ML decision-making processes, thereby ensuring optimal resource scaling.

7 DIGITAL TWIN

Optical networks play a central role in operators' networks, being a key part not only of core and metro transport networks, but also in supporting the development of 5G and beyond networks. In the last decades, optical networks have been enhanced using several technological innovations (e.g., the use of improved digital signal processing (DSP) and higher-order quadrature amplitude modulation (QAM) formats enabled by coherent-detection). Nevertheless, the continuous maximization of spectral efficiency and reduction of operating margins is fundamental for cost-effective data transmission. In addition, network operators are pushing for the creation of an open telecom market. Following a disaggregated approach, they can be split into transponder nodes, reconfigurable optical add-drop multiplexer (ROADM) built by combining degrees (each composed by wavelength selective switches (WSS) and erbium doped fiber amplifiers (EDFA)), and intermediate optical amplifiers, from different vendors. Nonetheless, disaggregation can add complexity to the optical network.

The digital twin (DT) is a paradigm that has been adopted across various industries to virtually model physical entities and understand their behavior under various scenarios. DT has emerged as a popular research topic in the telecommunications industry, where it is used to digitally replicate network infrastructure for the purpose of management, automation, and assessment. The development of DTs for Optical Transport Network requires the creation of a virtual representation of optical network elements (NEs) that can configure, monitor, and replicate the real equipment behavior.

7.1 OPTICAL NETWORK DIGITAL TWIN

A critical aspect of Network DT design for an OTN is the close integration between the DT and physical NEs, with synchronized communication to ensure continuous exchange of configuration and operation data. This integration is essential to enable effective network management while dynamically provisioning and reconfiguring optical services. In addition, by abstracting the optical components, the DT can provide valuable insights into the network's performance, helping operators to make informed decisions and optimize network operations. This ensures that the DT can accurately represent the behavior of the network, allowing for dynamic service configuration and management of the OTN [Kar23]. The proposed DT possesses the following key characteristics:

- Device abstraction - This is accomplished by capturing complete operational data, including hardware and optical service state data, using YANG data models. OpenROADM an open-source YANG specification for optical networks, is used to model the DT for each optical component type. This allows remote configuration using an SDN controller during service provisioning.

- Scalability - The DT's importance is better understood when assessing service provisioning in a larger OTN topology. Each DT can be deployed to scale and replicate with respect to individual components present in the OTN. Existing methods with emulators require more computing resources and lack real-time data for estimation, making them less scalable.
- Model-driven - A Model-driven approach is utilized in the DT-based environment for service configuration, and operational data retrieval using the NETCONF protocol. Each device twin exposes the NETCONF manager service with all protocol operations, similar to the real environment.
- Network state prediction - The DT is capable of integrating with real NEs, ML models, or estimation tools like GNPy to generate NE data for unseen network states. This is necessary to evaluate planned changes to the network using the DT.
- With the mentioned characteristics and behavior, the network layer DT has the following use cases in OTN,
- Service Provisioning: Optical Channel and network configuration assessment is carried out based on the outcome from the DT network before provisioning into the real network. [Karu23]
- Closed-loop automation: End-to-end network automation involving components from (Data, control and management planes) can be carried out as DT interacts with the components like a real NE.
- Analysis of YANG models: Upgrade in the YANG revisions, and applicability of YANG models from different working groups (OpenConfig, OpenROADM, ONF-TAPI) can be tested in network layer DT network.
- ML model testing: KPI parameters observed from the real network can be modeled and can be implied in DT network for forecasting the network state.

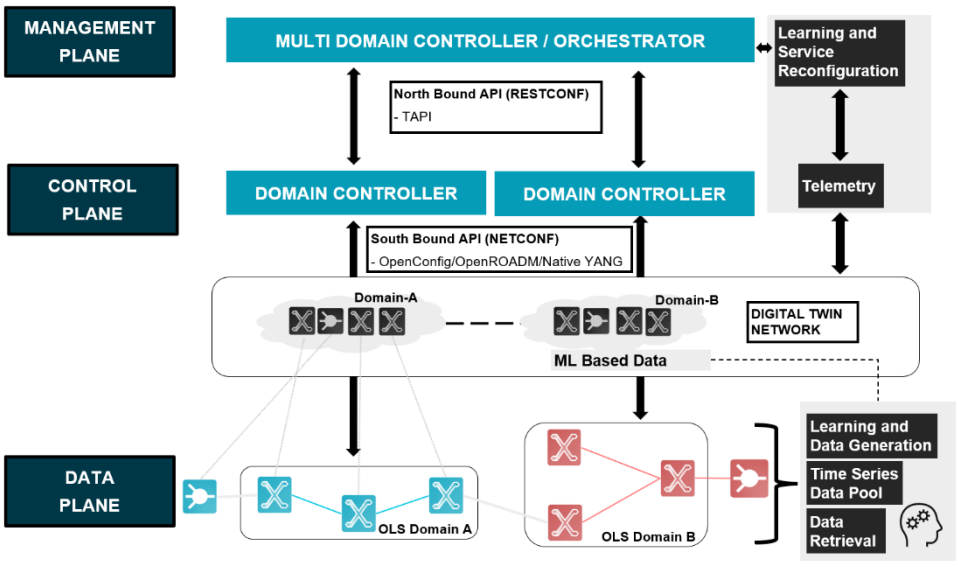


Figure 7.1: System Overview: Optical Network Digital Twin based Transport Network.

The optical network digital twin is implemented with the scope of following important characteristics applied to its software stack (i). model-driven, (ii). synchronous interaction with NE, and (iii). forecasting next state behavior. This helps to map a real network with a DT based network as shown in Figure 7.1, giving a complete provision to configure like a real network. It is also seen in Figure 7.1 that digital twins can connect with real vendor network devices and completely capture the current configuration in each of the OTN elements. Data modeling is carried out to mirror the optical network service behavior in digital twins.

The existing assessment is carried out with various simulators and emulators, but the outcomes deviate from the real network behavior. There are various reasons behind this disparity. So, the DT paradigm is appreciated as it is merely a real device. This is on the grounds that there is a communication process (i). Between physical NE and DT, (ii). Between DTs, and (iii). Between DT and control plane entities. In this manner, the software stack of DT exhibits a real physical NE with the possibility of external communication as shown in Figure 7.2. To ensure the real NE behavior, Machine Learning (ML) based data models are used to generate metrics, which is observed from the real network, and can be utilized to forecast the data such that various alarms can be prevented by knowing the network behavior in prior.

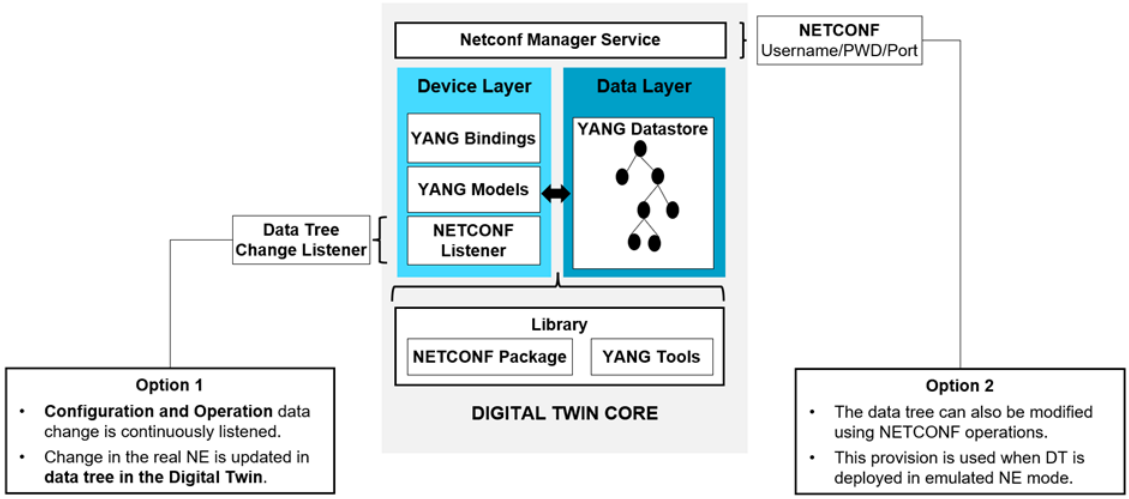


Figure 7.2: Internal Architecture: Optical Network Digital Twin.

The evaluation is performed between the following platforms: (a) An OpenROADM-based network simulator, (b) A vendor-based network emulator, and (c) An OpenROADM or a vendor-based optical network digital twin. An abstract comparison of the capabilities of the three different platforms is given in Table 7-1.

Table 7-1: Platform Capabilities – Comparison.

Property	Simulator	Emulator	Digital Twin
NE Abstraction	No	Yes	Yes
Scalability	Yes	No	Yes
Model-Driven	Yes	Yes	Yes
Forecast Network	No	No	Yes
Connected Operations	No	No	Yes

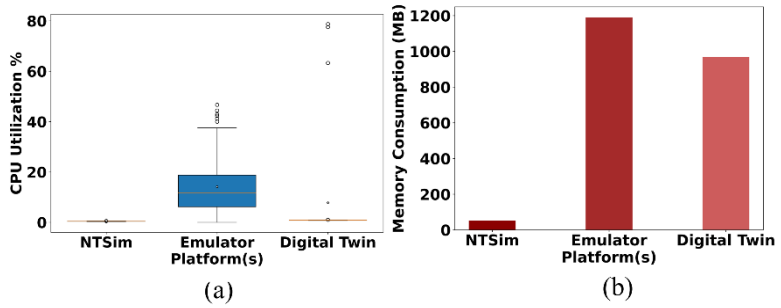


Figure 7.3: Resource Usage Comparison: (a) CPU (b) RAM.

A simple network topology is constructed with different platforms (Simulator/Emulator/Digital-Twin). Each instance in the network is assigned with a CPU of 1 core and enough main memory for their functioning. An initial evaluation is carried out from the scalability perspective, by monitoring the resource usage behavior of the instances over time and is plotted as shown in Figure 7.3. We have implemented the DT with NE functionalities with the scope of having a time-correlated behavior rather than implementing an entire stack of NE as in the emulator. This reduces the processing power required for the instance and this is evident in Figure 7.3a, where the emulator in Figure 7.3(b) requires more %CPU, when the ROADM instance is ideal. Digital twin has comparatively less %CPU utilization as it possesses a lightweight software stack than the emulator. On the other hand, NTSim simulator has a very small %CPU utilization as it mainly focuses on NETCONF manager service and lacks other network assessment functionalities. Digital Twins abstracts the entire NE using NETCONF and continuously monitors the real NE to mirror the change in the device configuration. Also, ML-based data models were implemented by observing the real network and were used to simulate data specific to optical network services. With the requirement of continuous listening to configuration and operation data changes in the real NE, there is an increased memory requirement for DT which can be seen in Figure 7.3b, where it has a memory usage of about 900MB. Whereas Emulator has a higher memory requirement due to the application running behind each pluggable of the NE. The outcomes provide confirmation of Digital Twin's scalability for deployment in a larger network and its extent for the research.

Since the Optical Network Digital Twin is not fully developed, the basic assessment is carried out on simulators, and investigation on network management and operations is planned in a future work.

7.2 Optical Time Domain Digital Twin and Applications

In the context of network automation, solutions, e.g., based on Machine Learning (ML) techniques, can be adopted to facilitate network operation. Two critical network operation tasks are quality of transmission (QoT) estimation and failure management. QoT estimation can be used to: *i*) check lightpath feasibility, e.g., during the provisioning phase; *ii*) estimate the QoT of a lightpath already established; *iii*) enhance existing analytical models; and *iv*) improve model generalization.

Regarding failure management, tools should include: *i*) the *detection* of degradations (*soft-failures*) before they have a major impact on the network (*hard-failures*); *ii*) *severity estimation*, i.e., estimate whether soft-failures would become hard-failures and when this would happen. Note that such estimation is of paramount importance to plan network maintenance activities; *iii*) *identification* or classification of the failure; and *iv*) *localization* of the malfunctioning device/element that produces the observed degradation.

In this section, we introduce applications for the OCATA time domain DT, which models the propagation of optical signals from the transmitter to the receiver in the optical time domain. Such an approach allows extracting additional information from the optical signal, which can be used, e.g., to find characteristics of network components that an optical signal has traversed, e.g., distances, number of ROADMs, and others. In addition, because OCATA generates also the *expected* optical signals in the time domain, they can be used to detect degradations easily and quickly when they are compared to the real samples collected from the network.

Applications of the OCATA digital twin for QoT estimation and soft-failure management are proposed with the aim to demonstrate that optical signal time domain analysis can greatly complement other techniques. QoT estimation is used during lightpath provisioning and as part of failure management, e.g., for severity estimation, i.e., to predict when a degradation becomes a hard-failure together with the detection, identification and localization of soft-failures. Soft failures might result from misconfiguration, component aging or malfunctioning, and from environmental events. For instance, the aging of EDFAs leads to the worsening of its noise Figure (NF), i.e., to the increase of the amplified spontaneous emission (ASE) noise. Differently, component malfunctioning includes filter tightening (FT), filter shift (FS) and power fluctuation of the transceiver that might have been caused by environmental events like high temperature. In this regard, optical signals generated at high baud rates and using wide channel spacing (75 GHz) are more impaired by penalties caused by filter narrowing due to cascading and/or failures.

Specifically, this section: *i*) extends and validates BER estimation by means of simulation and experimental measurements; *ii*) provides a comprehensive assessment of failure detection and severity estimation algorithms; and *iii*) provides new methods for failure identification.

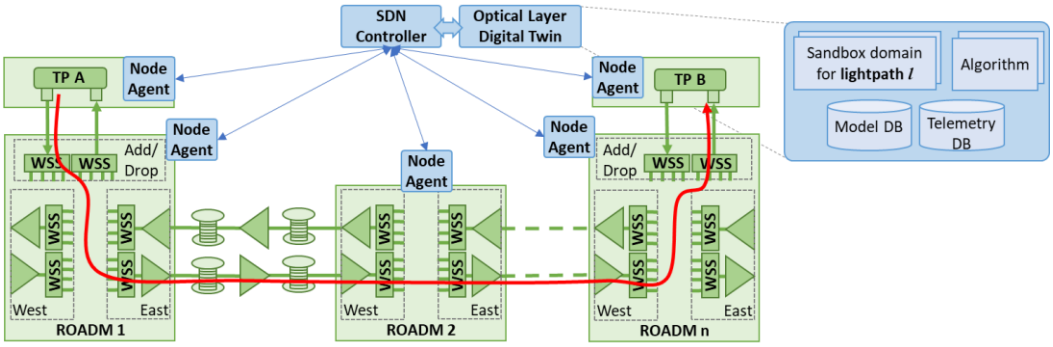


Figure 7.4: Overview of the envisioned optical network scenario.

Figure 7.4 presents an illustrative scenario with a lightpath connecting two end locations equipped with transponder nodes (TP A and TP B) through an optical network consisting of several ROADMs and optical links. Every ROADM includes WSSs and EDFAs, whereas every optical link is composed of EDFAs and single mode optical fiber spans. Transponder nodes and ROADMs are controlled by a local node agent that configures the underlying optical devices and collates telemetry measurements from them. On top of the architecture, a SDN controller connects to the node agents for network programmability and measurements collection. In addition, an optical layer digital twin modeling the data plane connects to the SDN controller and includes (or it has access to): *i*) a telemetry database (DB), where collected measurements are stored; *ii*) a model DB that includes models for several purposes ranging from physical layer to QoT estimation and failure identification; *iii*) a sandbox domain that is used for composing models representing lightpaths or for offline training of ML models. Training datasets in the sandbox can be populated initially based on the results from simulations and lab experiments, and then from the received telemetry measurements. They can also be augmented to include not-yet-considered patterns (e.g., related to soft-failures), which are added as soon as they are detected; and *iv*) a set of algorithms that are used to analyze the measurements received and compare them to those generated by the models.

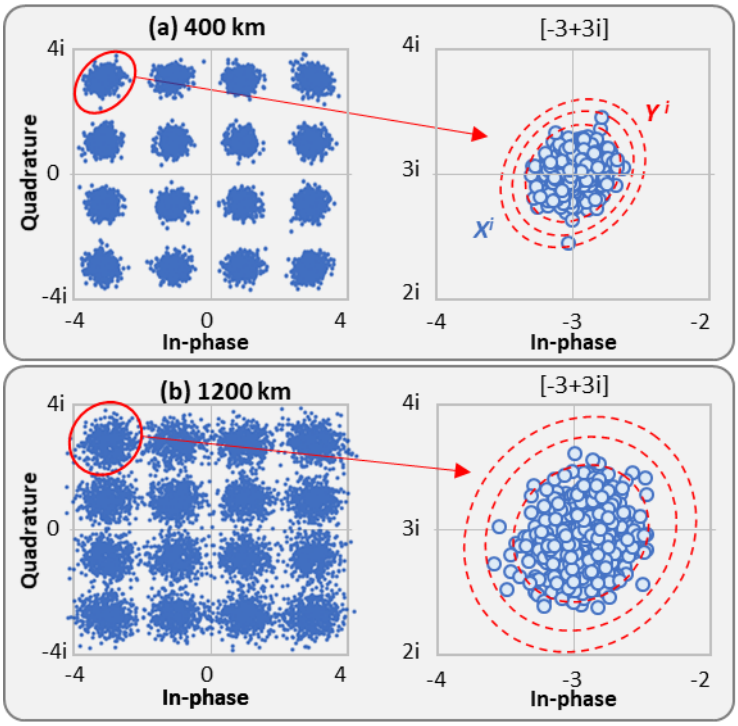


Figure 7.5: GMM fitting for feature extraction (Fex).

We concentrate on the measurements collected by the coherent receivers, which include in-phase (I) and quadrature (Q) optical constellations and QoT-related figures of merit, such as the pre-FEC BER and SNR. IQ constellations, denoted X, are defined by a sequence of complex symbols $x \in X$, where the real and imaginary parts represent the I and Q components of the optical signal, respectively. In an m-QAM optical signal, every symbol belongs to one of the m possible constellation points. Given an optical IQ constellation sample X, we apply Gaussian Mixture Models (GMM) to characterize every constellation point as a set of bivariate Gaussian distributions. In the example in Figure 7.5, GMM fitting has been applied to characterize the constellation point [-3+3i] of an m=16 QAM optical signal after transmission along 400 and 1200 km. Then, for every input sample X, a set of semi-supervised constellation features Y that characterizes X is generated. In particular, the feature extraction (FeX) procedure uses GMM fitting to characterize every constellation point i by means of vector Y_i with 5 features representing the real and imaginary mean position in the constellation (μ) and the real and imaginary variance and symmetric covariance terms (σ), i.e., $Y_i = [\mu^I,\mu^Q,\sigma^I,\sigma^Q,\sigma^{IQ}]_i$. Note that the higher the LI and NLI impairments affecting the optical signal, the more scattered the symbols are, which makes constellation characterization more challenging.

Among the applications of OCATA, we specifically focus on two quite useful ones: *i*) QoT estimation, or more specifically, pre-FEC BER evaluation; and *ii*) failure management, including soft-failure detection, identification, and severity estimation. For those applications, the algorithms in the OCATA time domain digital twin extract and analyze the features of the IQ constellation samples received from the TPs and compare them to those generated by the models.

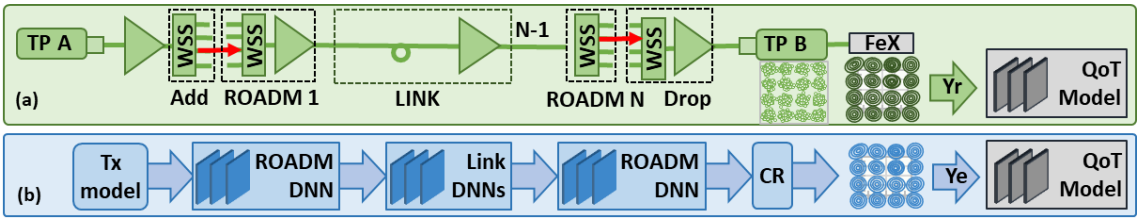


Figure 7.6: Use cases for BER estimation: QoT monitoring (a) and lightpath provisioning (b).

Figure 7.6 shows two different use cases that can take advantage of pre-FEC BER prediction: (a) during *lightpath operation* for QoT monitoring purposes, where the BER is estimated from the features Y_r extracted from optical constellations collected from the TPs; and (b) *lightpath provisioning*, where BER prediction is used to assess the feasibility of the selected route before establishing the lightpath. To this end, OCATA can be used to generate the expected features Y_e that are expected for a specific route and from there, the expected BER can be estimated. In this case, OCATA builds a virtual replica by concatenating DNN models which approximate the degradation resulting from the signal propagation along the ROADMs and optical links in the lightpath. In this way, generated input features, representing a TP, can be propagated through the concatenated DNN model and predict features Y_e. In both applications, a DNN model is needed to estimate BER values in a meaningful range, e.g., from 10^{-5} to 10^{-2}.

We proposed the function $diff(X_r, X_e)$ to compare received signals (X_r) and the expected ones (X_e), by computing the Euclidean distance of the difference between the features extracted from X_r and the expected ones, i.e.,

$$diff_Y(X_r, X_e) = \|Y_r - Y_e\|_2 \qquad \text{Eq. 4}$$

An application case for Eq. 4 is failure detection. In this section, we target not only failure detection but also its identification and severity estimation. Figure 7.7 illustrates several possible failures affecting a lightpath. Figure 7.7a corresponds to the regular network operation, i.e., when the lightpath is not affected by any failure or misconfiguration. Figure 7.7b illustrates a soft-failure in the transmitter, e.g., an extra gain in its booster amplifier leading to an extra transmission power (eTxP). Figure 7.7c illustrates the impact of an EDFA in the first ROADM with an increased noise Figure (iNF). Finally, Figure 7.7d corresponds to a failure in a WSS in the first ROADM that produces FS or FT on the optical signal. Note that some of the failures can already be detected and identified by analyzing the optical spectrum of the signal only. Nevertheless, here we target at analyzing the signal in the time domain also, which greatly complements the frequency domain analysis.

Let us now present new features specifically designed to estimate the BER of a lightpath and to do a more precise modelling of optical constellations, which enables failure management. Next, we propose algorithms for failure detection, identification and severity estimation (i.e., time for the degradation to become a hard-failure) using these features.

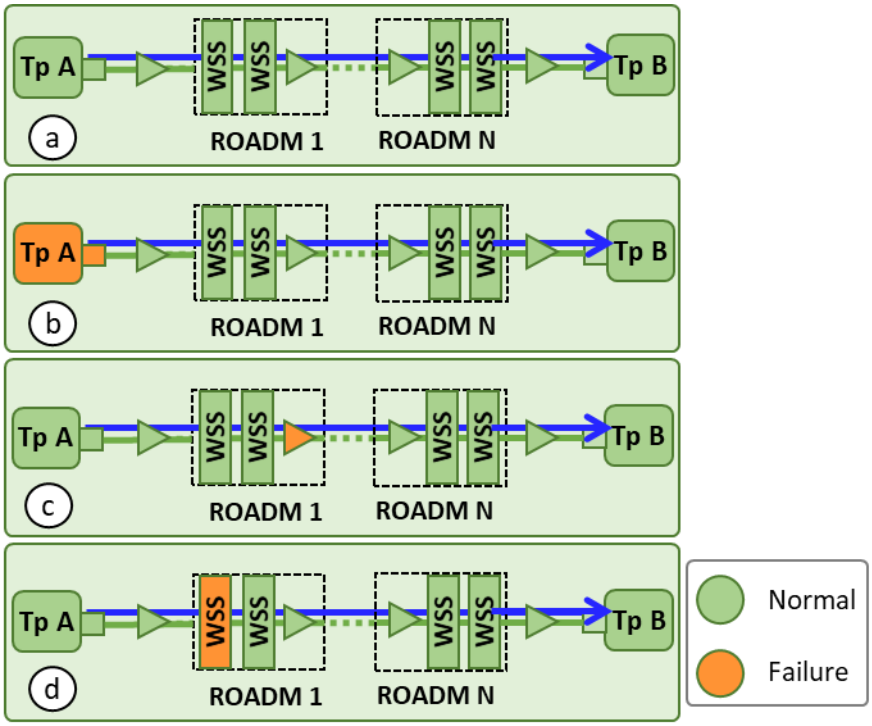


Figure 7.7: Considered use cases for failure management: (a) normal operation, (b) TP failure, (c) EDFA failure, and (d) WSS failure.

The basic features *Y* defined above can be used as inputs of a DNN model that estimates the BER in a meaningful range. Note that *Y* characterizes the scattering of symbols around their mean using bivariate Gaussian distributions and, in turn, such scattering is related to the pre-FEC BER of the signal. Nevertheless, the characterization of *Y* and its relationship with the BER is not trivial. In view of that, we propose adding a new feature, denoted Φ^i_{out}, that computes the probability that a symbol initially transmitted in constellation point *i* is detected at the receiver out of the detection area of such constellation point, denoted as A^i. Φ^i_{out} is computed under the assumption symbols corresponding to constellation point *i* follow the bi-variate Gaussian distribution characterized by Y^i.

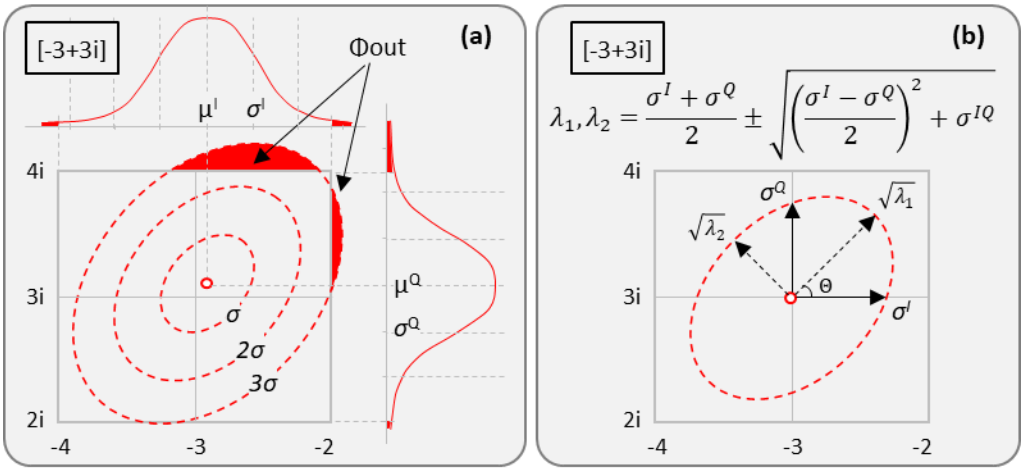


Figure 7.8: 17Example of Φ_{out} (a), Π and Θ (b) features for constellation point [-3+3i].

Figure 7.8a shows an example of Φ^i_{out} feature for constellation point [-3+3i]. The contours represent the different levels of the bi-variate Gaussian distribution that characterize this constellation point (CP) for a given lightpath. For the sake of clarity, we depict σ, 2σ, and 3σ levels only; univariate marginal distributions are provided for both I and Q axes. The area highlighted in red in both bi-variate and marginal distributions represents the region that falls out of A^i, i.e., the square delimited by vertices [-4+4i] and [-2+2i]. Hence, Φ^i_{out} is formally defined as follows:

$$\Phi^i_{out} = 1 - P(x \subset A^i, x \sim N(Y^i)) \quad \text{Eq. 5}$$

It is worth noting that this feature is clearly related to errors in the receiver, and therefore a DNN can be trained taking this feature as input for different CPs and produce a QoT estimation as output.

As for failure management, we rely on the $diff_Y(\cdot)$ function defined in Eq. 4 for failure detection, whereas for severity estimation and failure identification and localization, we rely on a combination of features Y together with the QoT estimation based on the new feature Φ^i_{out} defined in Eq. 5. Additionally, we introduce new features to capture the effects of NLI noise on the shape of constellation points. Specifically, we consider the following new features of the bi-variate Gaussian distribution that characterizes CP i (Figure 7.8b): i) the Π^i, which measures the degree of ellipticity. Π^i **Errore. L'origine riferimento non è stata trovata.** from the elliptical major and minor axes is defined by the 1×σ contour, denoted sqr(λ_1) and sqr(λ_2) to compare the received and expected values of a specific feature.

$$\Pi^i = \frac{\sqrt{\lambda_1} - \sqrt{\lambda_2}}{\sqrt{\lambda_1}} \quad \text{Eq. 6}$$

$$\Theta^i = \begin{cases} 0 & \text{if } \sigma^{IQ}=0 \text{ and } \sigma^I \geq \sigma^Q \\ \pi/_2 & \text{if } \sigma^{IQ}=0 \text{ and } \sigma^I < \sigma^Q \\ atan2(\lambda_1 - \sigma^I, \sigma^Q) & \text{otherwise} \end{cases} \quad \text{Eq. 7}$$

$$\mathrm{A}^i = \pi \cdot \sqrt{\lambda_1} \cdot \sqrt{\lambda_2} \quad \text{Eq. 8}$$

$$l^i = \frac{\lambda_2}{\sqrt{\lambda_1}} \quad \text{Eq. 9}$$

$$delta_{y^i}(y_r, y_e) = \left| y_r^i - y_e^i \right| \quad \text{Eq. 10}$$

By analyzing the evolution of the features defined above and $diff_Y(\cdot)$ and $delta_y(\cdot)$ functions, the required failure management functionalities can be implemented from observations collected from the network and from the OCATA digital twin working at the time domain.

The pseudocode for the general failure management algorithm is presented in Algorithm 3, which is executed in OCATA when a new constellation sample X_r is stored in the telemetry DB.

The algorithm receives: *i*) parameters and models for the lightpath (*ml*) that were trained in the sandbox domain and stored in the model DB; *ii*) historical observations and computed *diff*(·) values for the lightpath (*hl*), processed from the telemetry DB; *iii*) a list with operational parameters (*O*); and *iv*) a timestamp with the current time *t*. The algorithm first obtains the last sample X_r from *hl*, computes features Y_r and stores them in *hl* (lines 1-3 in Algorithm 3). Next, algorithms for degradation detection (Algorithm 4), failure identification (Algorithm 5) and severity estimation (Algorithm 6) are called (lines 4-7). The failure management algorithm returns whether a degradation has been detected and if so, the cumulative probabilities *p* of the considered types of soft-failures, i.e., p_{eTxP} for extra transmission power, p_{FF} for filter failure (FF) (either FS or FT), and p_{AF} for amplifier failure, as well as a prediction of the time at which the soft-failure may lead to service disruption (line 8).

Algorithm 3. ***Failure Management.***

INPUT: *ml*, *hl*, *O*, *t*

OUTPUT: *degradation*, *p_eTxP*, *p_FF*, *p_AF*, *t_disrupt*

1: $X_r \leftarrow$ *hl*.getLastX()
2: $Y_r \leftarrow$ *ml*.featureExtraction (X_r)
3: *hl*.append("Y_r", Y_r, *t*)
4: *degradation* ← Detect(*ml*, *hl*, *O*, *t*) (Algorithm 4)
5: **if not** *degradation* **then return** [false,0,0, ∞]
6: [*p_eTxP, p_FF, p_AF*] ← Identify(*ml*, *hl*, *O*, *t*) (Algorithm III
7: *t_disrupt* ← EstimateSeverity(*ml*, *hl*, *O*) (Algorithm 6)
8: **return** true, *p_eTxP, p_FF, p_AF*, *t_disrupt*

Algorithm 4 presents the pseudocode for degradation detection. The expected constellation features Y_e are generated using the model for the lightpath and the features Y_r from the received constellation are retrieved from *hl* (lines 1-2 in Algorithm 4). Next, $delta_y(\cdot)$ comparing received and expected features *A*, *l*, Φ_{out} is computed (lines 2-5). The obtained delta values of the features together with lightpath's details (i.e., number of ROADMs and total distance) are given as inputs to a binary classifier *ls* (line 6). Whenever a failure is pinpointed for more than *n_th* consecutive times, a positive detection is returned (lines 7-11).

Algorithm 4. ***Degradation Detection.***

INPUT: *ml*, *hl*, *O*, *t* **OUTPUT:** *degradation*

1: $Y_e \leftarrow$ *ml*.generateY()
2: $Y_r \leftarrow$ *hl*.getLastY()
3: A_r, l_r, Φ_r, -, - ← *ml*.featureExtraction (Y_r)
4: A_e, l_e, Φ_e, -, - ← *ml*.featureExtraction (Y_e)
5: *Delta*[] ← $\text{delta}_Y(i_r, i_e)$ **for** *i* **in** [(A_r, A_e), (l_r, l_e), (Φ_r, Φ_e)]
6: *pred*← *ml*.Ensemble(*ml*.normalize(*Delta*[], *ml.N*, *ml.D*))
7: **if** *pred* **then**
8: O.*n* ++
9: **if** O.*n* > O.*n_th* **then return** true
10: **else** O.*n* ← 0
11: **return** false

Algorithm III. ***Failure Identification.***

INPUT: *ml, hl, O, t* **OUTPUT:** *p_eTxP, p_FF, p_AF*

1: $Y_e \leftarrow ml$.generateY()
2: $Y_r \leftarrow hl$.getLastY()
3: $A_r, l_r, \Phi_r, \Theta_r, \Pi_r \leftarrow ml$.featureExtraction (Y_r)
4: $A_e, l_e, \Phi_e, \Theta_e, \Pi_e \leftarrow ml$.featureExtraction (Y_e)
5: *Delta1*[] ← $\text{delta}_Y(Y_r, Y_e, i)$ **for** *i* **in** [Θ, Π]
6: *Delta2*[] ← $\text{delta}_Y(Y_r, Y_e, i)$ **for** *i* **in** [*A, l*, Φ, Θ, Π]
7: *eTxP* ← *ml*.Ensemble(*ml*.normalize(*Delta1*[], *ml.N*,
8: *ml.D*))
9: **if** NOT *eTxP*
10: *AF*←*ml*.DNNedfa(*ml*.normalize(*Delta2*[], *ml.N*,
11: *ml.D*)))
12: *FF*←*ml*.DNNwss(*ml*.normalize(*Delta2*[], *ml.N*,
13: *ml.D*)))
14: *unknown* ← *AF* = *FF*
15: *hl*.append("failureIdentif", [*eTxP,FF,AF,unknown*], *t*)
eTxP[],FF[],AF[],- ← *hl*.getLastN("failureIdentif")
p_eTxP ← *ml.*cumulative_prob(eTxP[], [*t* - *O.Δde*, *t*])
p_FF ← *ml.*cumulative_prob(FF[], [*t* - *O.Δde*, *t*])
p_AF ← *ml.cumulative_prob*(*AF*[], [*t* - *O.Δde, t*])
return *p_eTxP, p_FF, p_AF*

Algorithm 5 presents the pseudocode for failure identification. The algorithm targets at identifying soft-failures by returning their cumulative probability based on hierarchical binary classifiers trained off-line and it consists of: *i*) the identification of a transponder failure due to eTxP; *ii*) the identification of optical filter failures (i.e., either FS or FT) or amplifier failures (i.e., iNF). The rationale behind this structure is that we found that the impact of eTxP soft-failures on the time-domain is clearly different from the one of optical filters and amplifiers. Lines 1-6 in Algorithm 5 are similar to the first ones in Algorithm 4, except that here $delta_y(\cdot)$ functions are computed for all the extended features, i.e., *A*, *l*, Φ_{out}, Θ, and Π. As in Algorithm 4, a hybrid ensemble is employed as a binary classifier to identify the eTxP based on the *flattening* and *rotation* of the IQ constellations (line 7). eTxP soft-failure is detected after majority voting classification. If eTxP is not detected, then two DNN models are used to identify filter and amplifier soft-failures. A positive identification happens when only one of the two models detects a soft-failure whereas, when either both models detect a soft-failure or none of them do, the soft-failure remains unidentified. This usually happens, when the magnitude of the soft-failure is still too small for a positive identification (lines 8-11). Finally, Algorithm 5 stores the output of the ML inference (line 12) and returns the cumulative probabilities of each potential soft-failure from the detection time (lines 14-17).

Whenever a failure is detected, Algorithm 6 is used to predict when the degradation results into a hard-failure. This result is achieved by exploiting the time evolution of the BER (estimated using Φ_{out}). Initially, the features Φ^i_{out} are extracted from the received constellation and used as input of a DNN that estimates *BERe*, which is the stored in *hl* (lines 1-4 in Algorithm 6). The moving average of the *BERe* is calculated over a time window ranging from an interval *T* prior the detection time to the actual time *t* (lines 5-6). Then, the future time series evolution of the *BERe* is extrapolated based on the accurate Holt-Winters exponential smoothing (HWES) (lines 7-13).

The model's hyperparameters are optimized at each iteration of the loop through Bayesian optimization. Then, the performance of these hyperparameters are evaluated implementing time series cross-validation and the one leading to the best performance (i.e., lower error) are chosen as optimal (line 14). Afterwards, the model is fitted and fine-tuned and the evolution of the time series is extrapolated over a time window Δ*fr* (lines 15-16). The algorithm estimates the time for service disruption when the forecasted BER, *BER_fr*, exceeds the pre-FEC BER threshold, *BER_th* (line 17). Otherwise, no service disruption is found (line 18).

Algorithm 6. ***Severity Estimation.***

INPUT: *ml,hl*, *O* **OUTPUT:** *time_disrupt*

1: $Y_r \leftarrow hl$.getLastY()
2: -, -, Φ_r, -, - ← *ml*.featureExtraction (Y_r)
3: *BERe* ← *ml*.QoTestimator(Φ_r)
4: *hl*.append("*BER", BERe*, *t*)
5: *BERe*[]← *hl*.getLastN("*BER*")
6: *maBERe*[]← movAvg (*BERe*[], [*t*- *O*.Δde -*O.T*, *t*])
7: **for** *O.n_it*:
8: *hp* ← *ml*.BayesianSearch(search_space)
9: **for** train, test **in** TSSplit(*maBERe*[])
10: *model* ← *ml*.HWES(*hp*).fit(*BERe*[train])
11: *pred*← *ml.model*.predict(test)
12: *error[]* ← error_function(Pred, *maBERe*[test]))
13: *AvgError*← { *hp,* mean(*error*[]) }
14: *hpo* ← minimize(*AvgError*).*hp*
model ← *ml*.HWES.tune(*hpo*).fit(*maBERe*[])
[*BER_fr, t_fr*] ← *ml.model*.predict(*O*.Δfr)
if *BER_fr* > *O.BER_th* **then return** *t_fr*
return ∞

To conclude, two main applications of the OCATA digital twin have been extensively investigated and evaluated in this work: QoT estimation and failure management. New features that characterize IQ optical constellations, as well as algorithms and ML models are proposed.

For the QoT application, the pre-FEC BER estimation accuracy has been evaluated in terms of relative error for a comprehensive list of scenarios. Features extracted from IQ optical constellation samples generated by simulation and from experiments have been used to train and fine-tune prediction models based on DNNs. Specifically, the proposed feature with the probability of detecting a symbol at the receiver out of the detection area of the original constellation point (Φ^i_{out}) exhibited high correlation with the BER. Overall, the models converge toward an average relative error not exceeding 5%, for both experimental and simulated samples, independently on the signal format, transmission power and lightpath distances.

As for failure management, a framework based on the analysis of the received IQ constellations has been presented. With this objective, algorithms for soft-failure detection, identification and severity estimation have been proposed, leading to noticeable accuracy. These algorithms

exploit the comparison of features extracted from the received IQ constellations, with those extracted from OCATA-generated constellations for the lightpath under analysis.

Finally, it is worth noting that the OCATA applications proposed in this paper complement (i.e., do not necessarily replace) the analysis that can be carried out using other tools. For instance, the proposed QoT estimation fits very well for making decisions during lightpath provisioning and for severity estimation, where the level of accuracy of OCATA is enough to support such decisions. Clearly, for those use cases where very accurate QoT estimation is required, other tools or methods should be considered. Another example is on failure identification, where the optical time domain analysis that OCATA provides can be complemented with other tools that analyze the frequency domain to achieve higher accuracy in failure identification.

7.3 Extending the OCATA Digital Twin for Digital Subcarrier Multiplexing

As technology advances, new generations of TPs become available, e.g., a new generation of flexible TP capable to transmit both single carrier (SC), as well as digital subcarrier multiplexed (DSCM) signals is becoming available. Compared with the SC counterpart, DSCM transmission have better nonlinear tolerance and allows each subcarrier to be independently operated. DSCM systems will also be deployed on ROADM-based optical networks, which can impose significant optical filtering penalties as a result of narrowing the transmission passband bandwidth, an effect that becomes severer as ROADMs are cascaded in an optical connection. In this context, optical network DTs need to be upgraded to be able to predict the physical layer impairments affecting not only SC but also DSCM signals. Such technology would complement DTs in all their applications. E.g., during lightpath provisioning phase, DTs can be used to predict the expected QoT of an optical signal before the lightpath is setup, and thus the optimal format of the signal to be transmitted, in terms of spectral-efficiency and/or resiliency, can be found.

In this section, we extend OCATA time domain DT to model DSCM signals. OCATA includes DNN-based models and algorithms for QoT estimation (specifically for the pre-forward error correction (FEC) bit error rate (BER)) and failure management.

Figure 7.9 18introduces the considered scenario, where OCATA models an end-to-end lightpath between TP-A and TP-Z traversing *n* ROADMs and *n*-1 optical links. Route-and-select ROADMs with WSS and EDFA as booster and pre-amplifiers are considered. In this section, we extend OCATA with models for DSCM signals. OCATA models are based on the concatenation of DNN, each modelling a single ROADM or a single optical link (Figure 7.9b). For illustrative purposes, the spectrum of the generated 16-subcarrier DSCM signal after TP-A and the spectra after traversing several ROADMs are shown in Figure 7.9a-c, as well as for the SC signal. As expected, we observe that the external digital subcarriers (DSC) suffer different filter penalties than the internal ones, which becomes more evident as soon as the number of cascading filters increases. Ultimately, the spectra of the most external DSCs are mostly filtered and those of the neighboring DSCs are also severely affected. However, the spectra of the internal DSCs do not

look impacted and can be used for reliable data transmission. Therefore, differently to the SC transmission, where the whole signal degrades by physical layer impairments, in DSCM transmission, such degradation is gradual.

In view of the above, it seems important to provide models for the transmission of the different DSCs, particularly for ROADM propagation. Specifically, we consider differentiated DNN models for the ROADMs: *i*) for the two *external* DSCs (i.e., DSCs 1 and 16); *ii*) the two *intermediate* DSCs 2 and 15; and *iii*) *internal* DSCs 2-14. With such models, we target to (a) analyze the tolerance of SC and DSCM signals to filter penalties; and (b) to verify the suitability of OCATA for lightpath provisioning to decide whether to use DSCM or SC for a given route (ROADMs and optical links) computed over the optical network topology.

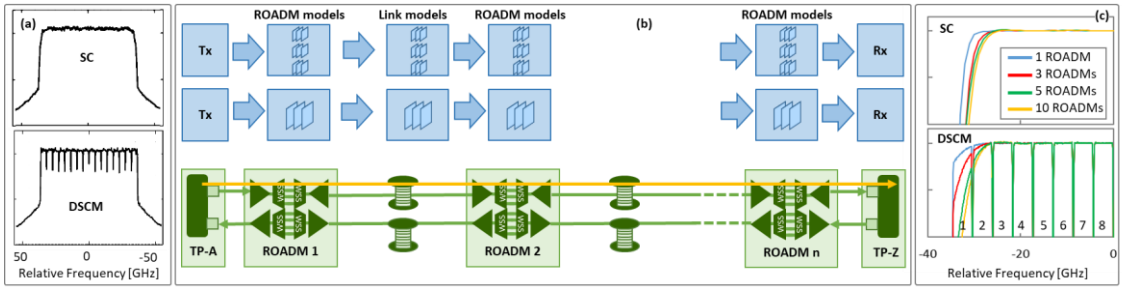


Figure 7.9: Overview of the envisioned scenario. Optical data plane (in green) and OCATA DT (in blue) modeling SC and DSCM signals.

8 AI/ML IN SUPPORT OF SERVICE MANAGEMENT AND ORCHESTRATION

The use of AI/ML in service management and orchestration plays a key role in the transformation towards a more intelligent and autonomous network operation. It empowers the network to be responsive, predictive, and adaptive, thus, significantly improving its efficiency and flexibility. The orchestration which makes use of fS widely adopted technologies such as containerized applications, VNF's and SDNs relies on AI/ML algorithms to manage and optimize the end-to-end network services in an automated and self-adaptive way achieving seamless interoperability and integration.

AI/ML techniques are deployed to address complex tasks such as failure prediction, traffic monitoring and analysis, anomaly detection, and analyzing traffic patterns. These in turn lead to proactive rather than reactive resource management, increasing in this way the overall network performance while reducing the operational costs. Moreover, AI/ML models can help in predicting network faults and anomalies before they impact service quality. By leveraging the power of AI/ML for predictive maintenance, network downtimes can be reduced, and service quality can be assured.

Some examples in the context of service management and orchestration that AI/ML can be applied:

- Assisting on specific autonomous orchestration and control operations such as network service provisioning/deployment fulfilling stringent requirements (e.g., bandwidth, latency, resiliency, etc.). The AI/ML mechanism tackles diverse selected objectives such as improving resource utilization (e.g., optical spectrum), compute resource placement, reducing overall power consumption, etc.
- Failure Prediction: By utilizing AI/ML models, historical data can be analyzed to forecast potential failures in network components or services. These models are used at identifying patterns and trends thus enabling proactive maintenance actions and minimizing service disruptions [Ts22].
- Traffic Monitoring and Analysis: Real time monitoring and analysis of network traffic patterns is made possible through AI/ML algorithms. This capability allows for the identification of congestion, bottlenecks, or abnormal behaviors ultimately leading to optimized resource allocation and improved quality of service [Aqu21].
- Anomaly Detection: Unusual network behavior such as atypical traffic patterns or unexpected system performance can be effectively identified through AI/ML techniques. By promptly recognizing and flagging these anomalies service providers can take corrective actions to prevent service degradation or any possible security issue. [Rat22].
- Predictive Maintenance/Autoscaling: Through the utilization of historical data and predictive analytics service providers can proactively schedule maintenance activities

such as autoscaling regarding computer resources (VM's, containers etc.), network resources (load balancers, bandwidth allocation etc.) or Services and Applications running in the cloud infrastructure, thereby reducing network downtimes and ensuring high availability of their services [Sub21].

8.1 COORDINATION BETWEEN RAN SLICES AND NETWORK TRANSPORT

Future RAN will operate with massive and heterogeneous small-cell deployments in support of diverse beyond 5G (B5G) and 6G use cases demanding high bandwidth and stringent latency requirements. To cope with such demand, RAN cells need to be planned with high number of base stations (BS) per cell, which anticipates both large overprovisioning and energy consumption. To reduce both, the operational mode (active - sleep) of BSs can be dynamically managed as a function of current user equipment (UE) traffic requirements.

Thanks to RAN and 5G core *virtualization*, functional splits can be used to distribute the signal processing chain between a distributed unit (DU) and a centralized unit (CU) in the RAN, and the user plane function (UPF) in the core, which can be deployed at different sites of the network. The adoption of *flexible function split* is a promising solution that allows adapting to different quality of service (QoS) requirements dynamically, which substantially improves RAN efficiency. RAN slices can be created and managed to exploit the capabilities of dynamicity and adaptability, as well as achieving a virtualized, interoperable RAN among multiple vendors. Hence, *smart slice operation* can be achieved by combining dynamic RAN resource allocation and slice management with flexible functional split management.

The deployment of multilayer optical networks in access and metro segments plays a fundamental role in meeting the end-to-end (e2e) requirements of RAN slices. Similarly, to smart slice operation, *autonomous network operation* is required to allow fixed (optical) networks to operate efficiently. Such operation paradigm is typically based on autonomous control loops, where monitoring data is continuously gathered and analyzed by means of AI/ML models and algorithms that trigger actions to be performed in the network, e.g., to adapt optical capacity to current and expected traffic demand. This is particularly interesting when DSCM technology is used, as sub-carriers can be activated and deactivated in near real-time.

The main challenge for an autonomous transport network operation is to deal with highly variable traffic, which also becomes unpredictable because of smart slice operation. In a classical 4G scenario, the traffic injected by RAN cells to the fixed network typically fluctuates with smooth patterns highly correlated with UE demand. However, depending on the functional split and DU/CU virtual function placement, B5G slices carry a mix of front-haul (F-H), mid-haul (M-H), and back-haul (B-H) traffic that depends not only on UEs demand, but also on slice operation. Thus, actions such as activating a new BS and changing the functional split or the placement of virtual functions, introduce large and sudden changes in the traffic of slices and consequently, in the underlying optical connections supporting them.

In view of the above, smart RAN slice operation and autonomous fixed network operation must coordinate among them to achieve the required e2e performance. We propose the coordination between RAN and fixed networks to enable e2e smart operation by acompassing operation of slices and the fixed network in an effective and privacy-preserving way. The key concept is defining context variables passed from the slice manager to the SDN control performing dynamic capacity resource allocation in the fixed transport network. Context variables contain relevant information about the slice configuration in an aggregated way to preserve the privacy of individual services and UEs. Moreover, context is updated asynchronously, e.g., before a significant slice reconfiguration is performed. In this way, the frequency and volume of data exchanged between domains is minimized.

B5G Reference Scenario

In the RAN, we consider that a cell consists of a single macro BS (MBS) and a number of micro BSs (µBS). MBSs provide full coverage within their cells and provide the minimum capacity to absorb users' traffic, whereas µBSs complement the capacity of the MBS within a limited area of the cell. We assume that µBSs provide two operational modes: *i*) *active*, where the µBS is switched on and fully operational; and *ii*) *sleep*, where the µBS is switched off. Without loss of generality, we consider that radio units (RU) on both MBS and µBSs provide support for e2e traffic flows. RAN cells provide radio connectivity to UEs requiring one of the following main service classes: *i*) enhanced Mobile BroadBand (eMBB); *ii*) Ultra-Reliable Low Latency Communications (URLLC); and *iii*) massive Internet-of-Things (mIoT). It is worth mentioning that eMBB typically requires a large capacity (~150 Mb/s per UE and service) with relaxed e2e latency requirements (~4 ms from the UE to the core). On the opposite, the URLLC service has very stringent latency requirements (~ 1 ms) and reduced capacity. Finally, mIoT is typically highly distributed, which entails managing a large number of UEs injecting moderated bandwidth (in the order of tens of Mb/s) with intermediate target e2e latency assurance (~2 ms).

Figure 8.1a illustrates the 5G high-level reference architecture considered in this work, where the traffic generated by UEs in a cell sequentially traverses a number of functions, namely, RU, DU, and CU, until reaching the UPF serving as breakout point of the 5G core. Thus, the resultant *graph* can be split in four different *slice links*, characterized by the RAN segment, i.e., radio (between UE and RU), F-H (between RU and DU), M-H (between DU and CU), and B-H (between CU and UPF). All these functions can be virtualized and run on the computing resources (servers, virtual machines, or containers) available at the different sites of the network. The B5G architecture is supported by resources in the fixed network infrastructure, for both connectivity, i.e., capacity and ensured latency. and computing. The e2e B5G reference topology assumed in this work is depicted in Figure 8.1b, where the main network segments connecting sites and Central Offices (CO) are sketched. Therefore, the traffic of a cell enters the fixed network. Specifically, an access optical network connects cell sites with their reference *access CO* (ACO). Typically, the distances between RAN cells and their ACO site are short, i.e., from a few to tens of km. Besides optical transport and switching capabilities, ACO sites are small datacenters equipped with computing and storage resources that enable the deployment of virtualized DU/CU functions, as well as other UPF functions. Typically, ACOs aggregate traffic from various RAN cells in proximity, as well as from other access technologies, such as residential gateways

or customer edge premises. ACOs are interconnected among them and with *regional COs* (RCO) by metro-aggregation networks. RCOs are farther from UEs (around hundreds of km) and larger and more complex than ACOs and hence, they can host more virtualized functions and achieve higher efficiency. Finally, RCOs are interconnected with *national COs* (NCO) by means of a meshed metro-core network, which provides large computational capabilities and serves as a gateway to other networks.

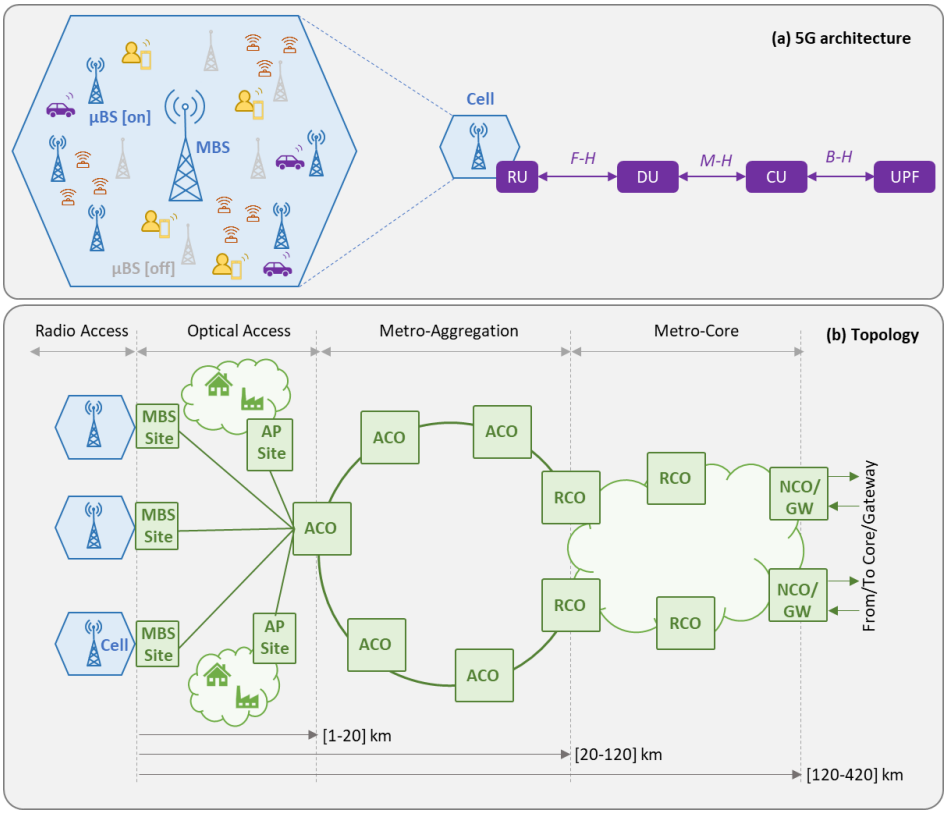


Figure 8.1: Reference 5G architecture (a) and topology (b).

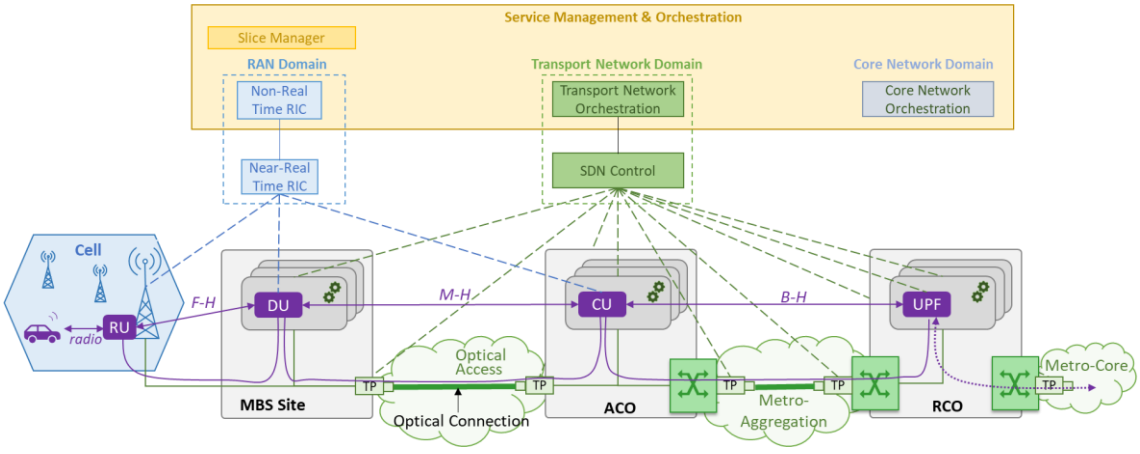


Figure 8.2: High-Level Architecture.

Figure 8.2 illustrates the overall architecture considered in this work, including the control and orchestration planes, which is an adapted version of the O-RAN architecture. The main entity

responsible for RAN domain management is the RIC in charge of a wide set of actions, such as QoS-based resource optimization, traffic steering, and RAN energy efficiency. The RIC is divided into near-real time RIC and non-real time RIC. The near-real time RIC controls RAN elements and their resources by means of local control loops that typically run in the range of 10 ms to 1 second; it receives policies from non-real time RIC, running in the service management and orchestration system, that enables wide control loops requiring execution time above 1 second. For the sake of simplicity, hereafter we refer to simply RIC as the unified RAN control entity that combines near-real time and non-real time operation. Specifically, we assume that the RIC deals with cell configuration, e.g., BS on/off switching, as well as it manages DU/CU placement for each slice. The core network orchestrator is responsible for the core functions and specifically, we assume that it manages UPF placement for each slice. A slice manager is in charge of making decisions of the configuration of each slice for service level agreement assurance. Finally, in the transport network domain, the orchestrator coordinate actions with the SDN control plane. It is worth noting that that the orchestrator layer provides O-Cloud functionality, i.e., it manages the computing nodes running in each site, as well as the connectivity between sites.

Without loss of generality, sites are equipped with optical transponders (TP) that allow connecting them to remote sites by establishing an optical connection. Here, we assume DSCM TPs, which can allocate a variable number of sub-carriers to adapt the capacity to the traffic needs.

The mapping of slice links connecting functions onto optical connections depends on the slice configuration (capacity and placement of virtual functions) managed by the slice manager, which in turn, consumes resources (computing and connectivity). Note that the placement of the functions cannot be done in any potential location site due to constraints of each RAN segment, such as distance between sites and latency requirements. Table 8.1 summarizes the mapping of virtual functions and site types, based on a typical network operator configuration. In the case of DU and assuming split 7.2 for F-H, only MBS and ACOs are suitable for its deployment. However, M-H latency can be relaxed by means of split 2, which allows extending its placement to RCO if suitable, i.e., for eMBB services. Regarding UPF, without loss of generality, we assume that they consist of processes that require more intensive computation and centralization than those of DU/CU. Therefore, due to the very limited availability of resources at MBS, the placement of such functions is avoided at the very edge of the network. In addition, although functions placement is allowed in ACOs, their computational resources are reserved for URLLC and mIoT services due to their limited capacity.

Table 8.1: Virtualized function placement constraints

Function	MBS	ACO	RCO	NCO
DU	*Yes*	*Yes*	*Yes (eMBB)*	*No*
CU	*Yes*	*Yes*	*Yes*	*No*
UPF	*No*	*Yes (URLLC, mIoT)*	*Yes*	*Yes*

B5G RAN and Slice Operation

As introduced above, smart slice operation is built upon three main pillars: *i*) *dynamic µBSs management,* by switching on/off *µBSs* with the objective of reducing energy consumption in the RAN, while ensuring the minimum capacity needed to support UE traffic; *ii*) *dynamic RAN capacity slicing*, with the aim of managing physical radio blocks (PRBs) to assign resources to each of the different slices in order to provide the required QoS; and *iii*) *flexible functional split operation*, where the placement of virtual functions (DU/CU) is adapted to match the requirements of the UEs served by each BS in a cell. In this section, we aim at illustrating how that smart slice operation dramatically affects the traffic supported by the underlying transport network in each of the segments of the reference topology.

Figure 8.3a shows the RAN state at a given time t_a of an example consisting of one cell with one active MBS that provides connectivity to a mix of UEs from different services. For the sake of simplicity, we assume that one slice per service type is deployed. Let us assume that the core network orchestrator decides, at slice provisioning time, the placement of UPF according to slice type and QoS requirements. This UPF placement remains fixed during slice lifetime. Moreover, each slice has its own placement of DU/CU along the different CO sites (see Figure 8.1b). In this case, DU/CU placement can be dynamically reconfigured by the slice manager according to current and expected UE traffic conditions in order to guarantee that e2e latency (i.e., from UE to UPF) meets the requirements of the slice service type. The Figure also shows a simplified view of the PRBs used by each of the slices. Table 8.2 shows the RAN segment per service class that is transported in each of the network segments. Note that the traffic in each segment is a heterogeneous mix of F-H, M-H, and B-H traffic, depending on the slice configuration. The selected time instant t_a illustrates a scenario where radio resources are reaching a point of saturation that is negatively affecting services e2e QoS (represented by colored gauges). In particular, URLLC service is strongly affected by such saturation, even when DU/CU functions are currently placed as close as possible to UEs to reduce e2e latency. In view of this, let us imagine that such QoS degradation is detected by the slice manager and, after analysis, some *slices reconfiguration* have been identified, which entail several actions to be performed. First, the slice manager triggers the activation of an available µBS (in light grey in Figure 8.3a). Due to the physical location of the antenna and its proximity to the majority of URLLC UEs, activating such a new antenna relieves the MBS from serving most of URLLC traffic. Figure 8.3b shows the RAN state after activating such µBS at time instant t_b. Since the activated µBS (now in green) captures most of the URLLC traffic, the overall RAN load is reduced and, consequently, the delay introduced by the RAN segment, which in turn makes that the e2e QoS of all services reaches the desired target performance.

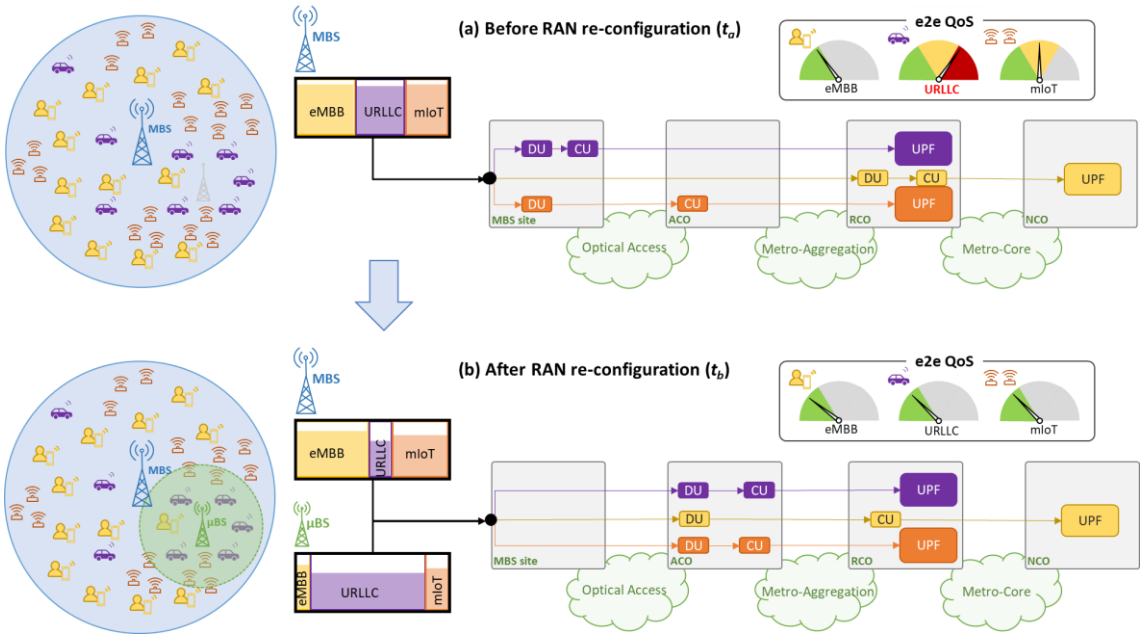


Figure 8.3: Example of RAN reconfiguration: before (a) and after (b) BS activation and function placement reconfiguration.

Table 8.2: Network Traffic Before Reconfiguration (time t_a).

Service Class	MBS <-> ACO [Optical Access]	ACO <-> RCO [Metro-Aggregation]	RCO <-> NCO [Metro-Core]	Service Class
URLLC	B-H	B-H	-	
eMBB	F-H (7.2)	F-H (7.2)	B-H	
mIoT	M-H (2/4)	B-H	-	

Table 8:3: Network Traffic After Reconfiguration (time t_b).

Service Class	MBS <-> ACO [Optical Access]	ACO <-> RCO [Metro-Aggregation]	RCO <-> NCO [Metro-Core]
URLLC	**F-H (7.2)**	B-H	-
eMBB	**F-H (7.2)**	**M-H (2/4)**	B-H
mIoT	**F-H (7.2)**	B-H	-

Nonetheless, smart slice operation goes beyond µBS activation. For instance, in order to reduce the cost associated with virtual function placement, URLLC and mIoT functions can be now located far from the edge (where available resources are typically cheaper) without mayor impact on the QoS of those services. This action might require re-allocation of some functions of other slices, for the sake of global optimally (as illustrated with the re-allocation of eMBB functions). Therefore, because of this smart slice reconfiguration (both µBS activation and virtual function re-allocation), the traffic supported in fixed network segments sharply changes. Table

8.3 updates table 8.2 after slice reconfiguration, where we observe segments that are greatly affected, e.g., traffic in the optical access sharply increases due to the addition of large F-H traffic volumes. It is worth noting that the change in the transport network traffic between time t_a and t_b cannot be predicted by typical monitoring and data analytics control loops in the fixed network, since the reason for that change is uncorrelated with past observed traffic. Hence, some *contextual information* about slice operation needs to be provided to the transport network orchestrator before it actually happens, so as the latter can prepare the fixed transport network accordingly.

In view of the above, a *context-aware autonomous network operation* solution based on sharing contextual information between the slice manager and the fixed transport network orchestrator is proposed in order to allow AI-based autonomous operation to efficiently control the optical capacity allocated to the optical connections supporting e2e slice connectivity.

8.2 AI/ML SERVICE ORCHESTRATION

AI/ML Service Orchestration is the process of integrating AI/ML capabilities into the orchestration layer of network management to automate and optimize service delivery across different network domains. By leveraging techniques such as Support Vector Machines (SVM), Deep Neural Networks (DNN), Reinforcement Learning (RL), and Long Short-Term Memory (LSTM), AI/ML models ingest operational data, understand complex patterns, make predictions, and take actions in near real-time. This approach is crucial for managing and optimizing end-to-end service orchestration, including resource optimization, provisioning, and other tasks. The integration of AI/ML into service orchestration goes beyond connectivity-focused management and is particularly valuable in enabling vertical solutions across diverse industries.

AI/ML Service Orchestration in the SEASON project represents a transformative approach in network management, integrating AI/ML capabilities across RAN, transport, optical, and core networks to automate and optimize service delivery. This integration leverages advanced techniques like Support Vector Machines (SVM), Deep Neural Networks (DNN), Reinforcement Learning (RL), Long Short-Term Memory (LSTM), and genetic algorithms. AI/ML models process operational data to understand complex patterns and make predictions, facilitating actions in near real-time. Such an approach is indispensable for managing end-to-end service orchestration, covering aspects such as resource optimization, provisioning, and predictive maintenance. It plays a pivotal role in the RAN layer, focusing on predictive maintenance within the RAN Cloud Orchestrator to ensure high availability, which is crucial in real-time applications where minor disruptions can have significant impacts. At the Transport Network layer, AI's analytical prowess extends to optical monitoring and streaming telemetry analysis, key for uninterrupted real-time service delivery. In the Core Network, AI/ML addresses diverse requirements such as high data compression and ultra-low latency, crucial in sectors like precision manufacturing and emergency response systems.

SEASON leverages artificial intelligence and machine learning methodologies to effectively manage network resources and services in near-real-time, with the goal of minimizing energy consumption while upholding optimal performance. The utilization AI/ML techniques hold great significance in extracting meaningful Key Performance Indicators (KPIs) and enhancing the Quality of Service (QoS) for users. The latter is feasible as Machine learning algorithms offer an opportunity to measure and evaluate vital performance indicators associated with the telemetry, infrastructure, and control service orchestration system in the SEASON infrastructure. These indicators include achieving high data compression ratios with minimal loss of information, the reduction of setup time for converged connectivity services, and the optimization of network connectivity creation time.

To effectively monitor and analyze network performance, data visualization tools such as Grafana and data monitoring tools as Prometheus are widely employed. These tools offer comprehensive and user-friendly visual representations of network metrics (Bandwidth, Latency, Packet Loss etc.) and service metrics (Resource Utilization, Response time, availability etc.) empowering network operators to gain valuable insights into performance patterns, identify potential bottlenecks, and make informed decisions based on data. Grafana, with its customizable dashboards and real-time monitoring capabilities, enables the visualization of diverse network data sources. Conversely, Prometheus serves as a robust monitoring and alerting toolkit that enables the collection and storage of time-series data, facilitating in-depth analysis and troubleshooting. By integrating AI/ML techniques with data visualization tools, network operators can seamlessly monitor and optimize network performance, ensuring that the QoE expectations (such as availability, scalability, security) of users are met.

AI/ML can play a pivotal role in the orchestration and inter-communication of RAN, transport, optical, and core domains of SEASON. Reinforcement Learning and genetic ML algorithms can be used for this. This integration would enable intelligent decision-making and coordination among various management functions and components. It facilitates proactive anticipation of future traffic loads. This enables dynamic adjusting based on real-time demand of Virtual Network Function (VNF) instances for seamless scaling and efficient resource allocation.

Moreover, ML/AI techniques, when combined with open-source tools such as Apache Kafka facilitate the analysis of monitoring data. This analysis enables coordinated network service scaling operations across domains and enhances overall network performance.

Kubernetes serves as an interface between different domain managers and transport SDN controllers, enabling the efficient integration and deployment of AI/ML techniques in the network architecture. By positioning Kubernetes on top of the transport, optical, and core domains, it acts as a bridge that facilitates seamless communication and coordination between these domains. This allows for the smooth deployment, scaling, and management of AI/ML workloads, ensuring optimal utilization of network resources and improved service delivery. Additionally, technologies like Transport API (TAPI) can be leveraged to enhance the interface between Kubernetes and the various domain managers and controllers, further enhancing the orchestration capabilities in the network architecture.

This innovative approach leverages AI/ML techniques in SEASON to optimize network performance, reduce energy consumption, and enhance user experience. By integrating AI/ML with tools such as Grafana and Prometheus operators gain insights from real-time network data for informed decisions.

8.2.1 AI/ML-assisted Control for service orchestration and application placement

In the recent research conducted within the SEASON framework, an innovative approach has been adopted, integrating artificial intelligence and machine learning techniques. This approach is aimed at optimizing the allocation and utilization of network applications, a pivotal aspect in the evolving domain of network orchestration. The methodology involved the simulation of a multi-layered network environment, designed to reflect the complex nature of contemporary network topologies.

The implementation of a genetic algorithm was central to this study. This algorithm was developed to optimize the placement of application instances across the network, involving the careful selection of nodes based on their ability to meet the application's requirements. A custom fitness function was developed, evaluating the placement of application instances while considering factors such as latency, energy consumption, data exposure, and resource utilization. This comprehensive approach ensured a holistic optimization of network resources.

Reference Scenario

We focus on a network topology comprising two important layers: the Edge and Extreme Edge. The Extreme Edge mainly hosts Distributed Units (DUs) and Radio Units (RUs), which are essential for eMBB services due to their proximity to end users, ensuring lower latency and high data throughput. The Edge layer, characterized by Centralized Units (CUs), plays a crucial role in managing network operations and maintaining connectivity with the core network. This topology aligns with 5G NR standards, ensuring compliance with industry specifications [3GP23]. The primary challenge addressed in this scenario is the efficient allocation of network resources for a set of applications, each with unique demands in terms of latency, data processing, and storage capabilities.

Implementation of AI/ML Techniques

In response to this challenge, AI/ML algorithms are employed. These algorithms are tasked with the optimal placement of application instances across the network’s nodes. The focus is on achieving a balance between resource availability, latency requirements, energy efficiency, and data security considerations for the eMBB scenario. For instance, the latecy for eMBB

applications typically should be under 10ms whereas the throughput can reach up to 10Gbps for extreme cases in 5G networks.

It is observed in this scenario that network demands and resource availability are subject to fluctuations. Consequently, the AI/ML models are designed to continuously process operational data, adapt to these changing conditions, and make predictive adjustments to service orchestration. This dynamic capability is crucial for maintaining an optimal balance in network performance and resource allocation.

Anticipated Outcomes

Several key outcomes are expected from the application of AI/ML-assisted control in this scenario:

- Increased Efficiency: Resource allocation optimized through AI/ML results in reduced latency and lower energy consumption, leading to heightened efficiency in network operations.
- Enhanced Quality of Service: The adaptive nature of AI/ML models ensures rapid response to changing network demands, thereby maintaining high service quality.
- Optimal Resource Utilization: Predictive analytics and intelligent decision-making contribute to superior resource utilization, ensuring balanced network load.
- Cost-Effectiveness: Optimizing resource allocation and energy consumption translates into reduced operational costs, benefiting both providers and users.

To validate the efficacy of our algorithm, we conducted a comparative analysis against a random placement strategy. This involved assessing crucial performance metrics such as latency, energy consumption, data exposure, and resource utilization, across a range of application instances. The simulations, which serve as a higher-level abstraction of numerous potential real-world network configurations, demonstrated that the genetic algorithm yielded more stable results, balancing latency, energy consumption, data exposure, and resource utilization effectively as seen in Figure 8.4. The genetic algorithm's adeptness in allocating resources efficiently across the network was particularly notable in its stable performance trends, underscoring the potential of AI/ML methodologies in enhancing network management systems. This contrasted with the random strategy's more unpredictable performance, which did not exhibit the same level of optimization. Overall, these findings advocate for the integration of intelligent, adaptive algorithms in network orchestration to address the dynamic and complex demands of modern network infrastructures.

Figure 8.4 shows the comparative analysis where the results show that the genetic algorithm consistently outperformed the random placement strategy across all evaluated metrics. We observed a significant reduction in latency and energy consumption, indicative of the efficient utilization of network resources. Furthermore, our approach demonstrated a marked decrease in data exposure, enhancing the security and integrity of data transmission within the network.

Lastly, the genetic algorithm achieved a more balanced and optimal utilization of resources across nodes, a testament to its effectiveness in managing complex network environments.

In the simulation studies conducted, it was discerned that the genetic algorithm provided a more consistent optimization of network performance when contrasted with a random placement strategy. These metrics included latency, energy consumption, data exposure, and resource utilization and validate the effectiveness of the method in an eMBB context.

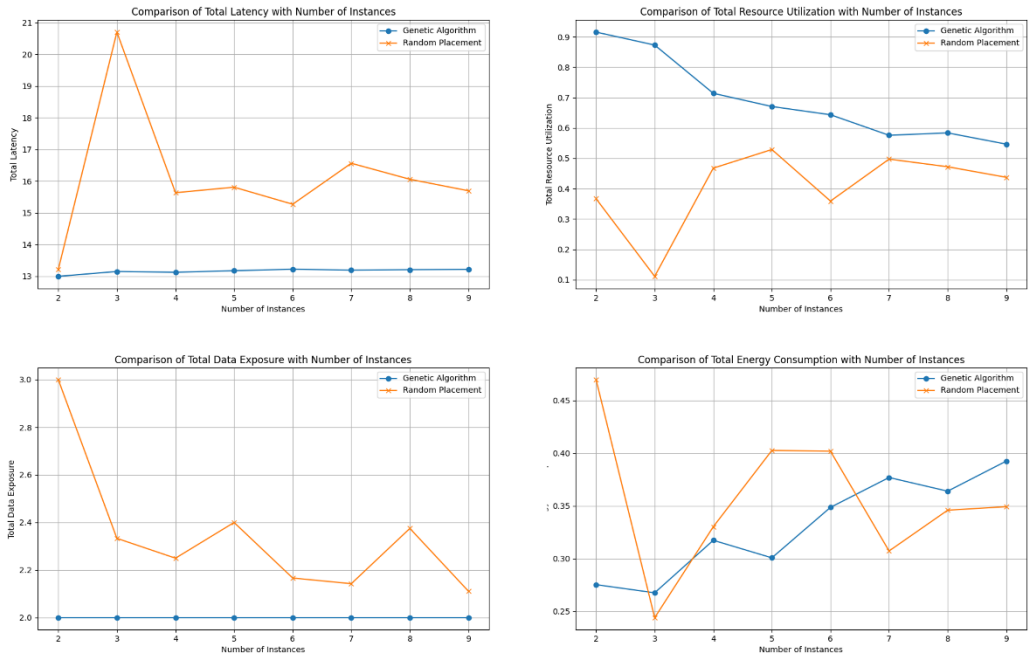


Figure 8.4: Comparative analysis among various KPIs regarding the application placement on a multi-layer network.

8.3 AI/ML-ASSISTED CONTROL FOR ENERGY EFFICIENT OPTICAL NETWORKS

Nowadays, the ICT industry consumes 5%-9% of total electricity [Mag23]. To improve the sustainability from both economic and energy perspectives, diverse approaches are being explored: i) deploying greener devices; ii) adaptive power management strategies which are adopted according to the traffic volume; and iii) advanced energy-aware routing and resource selection algorithms.

For the latter, the objective is to provide optical connectivity services not only fulfilling their requirements (e.g., bandwidth, end-to-end latency, etc.) but also conserving the overall network power consumption [Xio18]. To this end, energy-aware routing and spectrum assignment (EA-RSA) algorithms are used [Cos21]. These algorithms have been widely investigated mostly exploiting heuristics, which attempt to minimize activating optical devices, i.e. cross-connects, optical amplifiers, transceivers, etc. Nevertheless, minimizing network elements activation may lead to degrade the overall network performance, e.g., worsening the optical spectrum

utilization. In other words, a trade-off arises between power consumption and network performance.

Bearing this in mind, this activity focuses on leveraging the advantages of adopting AI/ML trained models to improve the above tradeoff with respect to heuristic approaches. Before delving into the specifics of the devised AI/ML approach, it is beneficial to provide a comprehensive overview of the reference scenario, i.e., transport network infrastructure, integrated dedicated AI/ML server to assist SDN control decisions, and energy model.

Reference Scenario

Figure 8.5 depicts a generic (multi-technological) optical transport infrastructure made up of optical switches, optical line amplifiers, packet optical nodes with pluggable transceivers, and/or dedicated transponders (e.g., Sliceable Bandwidth Variable Transponders, SBVTs). The optical connection service requests arrive to a higher layer optical controller via a NortBound Interface (NBI) specifying the endpoints, data rate (in b/s), maximum end-to-end tolerated latency (in ms), etc. Such a NBI can be implemented based on TAPI. The optical controller takes over programming both the terminal devices (e.g., pluggables) and line devices (i.e., optical line system, OLS) comprising the ROADMs and OLAs. The Optical Controller relies on an OLS controller to handle the configuration of the OLS. To this end, the OLS controller delegates the path and the resource selection to an externalized path computation server. The interaction between the OLS controller and the path computation server relies on a TAPI-enabled interface providing Context (topology and resource) information and exchange of Path Computation Request/Response. As shown below, the path computation server offers different routing computation mechanisms:

- Heuristics algorithms tackling different switching technologies and/or configurable parameters/operational modes (e.g., Routing and Band, Core, Modulation, and Spectrum Assignment)
- Pre-trained AI/ML models. In both mechanisms are considered to output the optical resources meeting the request demands and targeting an efficient use of resource and energy consumption.

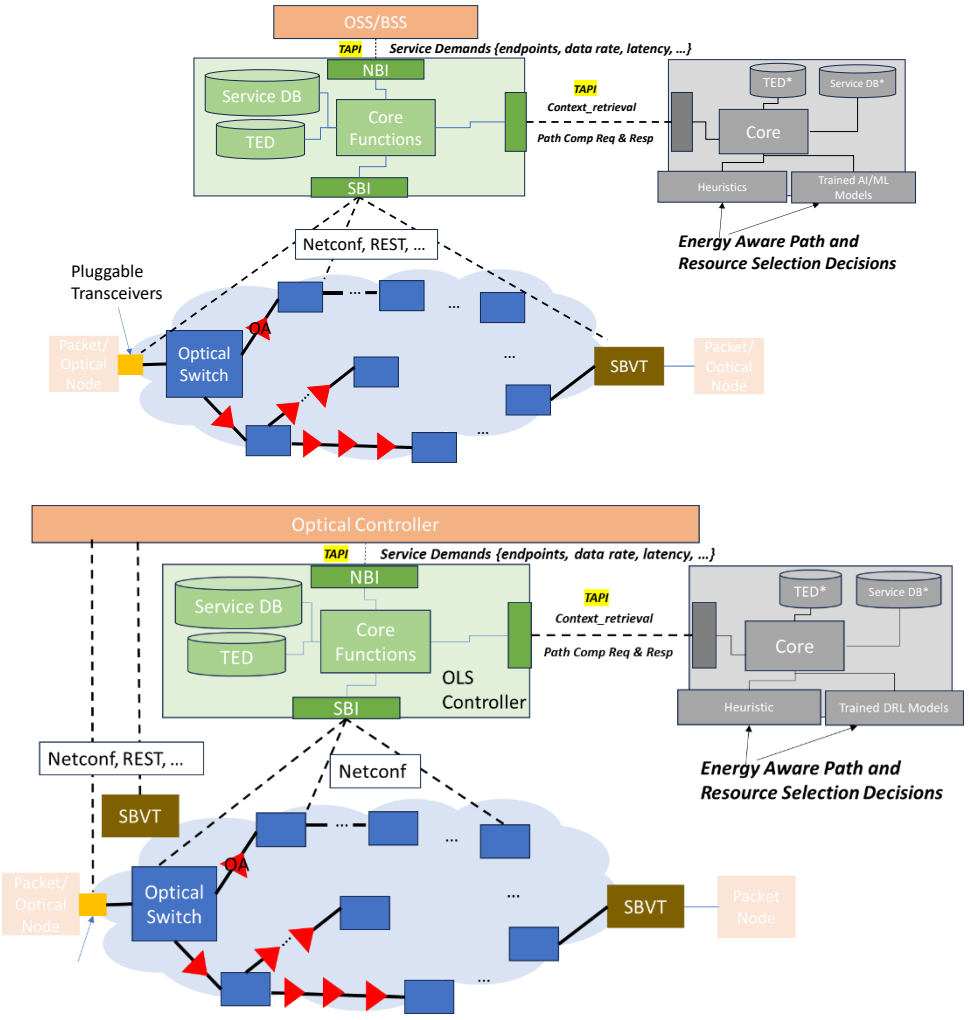


Figure 8.5: Adopted General Architecture: AI/ML-Assisted SDN Controller for Energy-Efficient Optical Networks.

Regardless of the adopted mechanism, the aim of this work is to attain a reduction of the energy/power consumption, associated to activated optical network elements, while setting up new optical connections. To do that, an energy/power consumption model needs to be determined. In the literature, the energy model for optical networks have received notable attention in the past [Elm14] [Zha15] [Viz12] [Kyr19] [Dur15]. In a nutshell, it is listed the "consumers" of energy/power, which are: optical transponders (i.e., pluggable, SBVTs), optical switches, and optical amplifiers. Herein, we start particularly focusing on the network elements within the Optical Line System (OLS) for an Elastic Optical Network provisioning connections in the C-band. That is, it is only considered the power consumption for both optical switches, and optical line amplifiers (OLA). In the same activity, in upcoming deliverables, it will be reported the considerations with respect to terminal devices such as SBVTs supporting different operational modes along with multi-band transport technologies.

Power Consumption Model for the OLS

The optical line device, e.g., ROADM / BV-OXC is formed by a set of line ports and tributary ports which are interconnected by an optical backplane enabling the cross-connection among these ports, i.e., adding/dropping a client/tributary signal or switching between two line ports. Line

ports, also referred to as *optical line boards*, are formed by diverse devices such as WSSs, optical amplifiers (Booster and pre-amplifiers), mux and demux.

The power consumption of an active/powered-up optical switch (i.e., $PC_{BV\text{-}OXC}$) depends on:

I. the environment power (i.e., P_{idle}) which determines the dissipated power caused by fans, control system, optical backplane, etc. Thereby, note that this power is consumed regardless of the number of established optical connections through the ROADM.

II. the consumed power bound to activated optical line boards/ports (P_{OLP}). That is, if no connection flows are using a ROADM port, no power is consumed by that port.

Therefore, the total power consumption for an active optical switch can be given by $PC_{BV\text{-}OXC} = P_{idle} + \Sigma P_{OLP}$, considering only the active OLPs.

Besides the ROADM power consumption, there is also the power consumption associated to optical fiber links, i.e., P_{link}. This is mainly associated to the accumulated consumed power by all OLAs forming the link. Then, the consumed power by an optical link can expressed as $P_{link} = \Sigma P_{OLA}$. It is important to note that when a network device or element, such as BV-OXC, ports, and OLAs, is powered up to support an optical flow, adding another flow using the same device does not result in increasing the power consumption. Consequently, a device has two states: *active*, when it consumes power while transporting one or more optical flows; and *slept down*, where no flow services are utilizing the device, resulting in no power consumption.

Figure 8.6 illustrates an example of a new optical flow between nodes ROADMs S1 and S14. The Figure shows the power consumption for each device involved in the path S1-S14-S14, as well as the total computed power when the optical flow is successfully established.

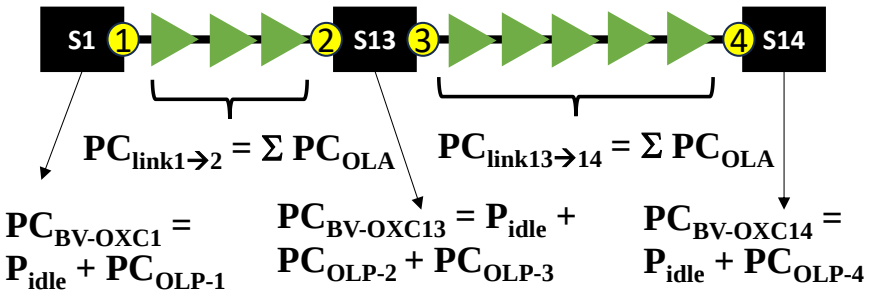


Figure 8.6: Example of OLS Power Consumption for an Optical Connectivity Service.

The values used to calculate the network power consumption are shown in Table 8.4.

Table 8.4: Considered Power Consumption values.

Notation	Description	Values
P_{OLA}	Power Consumption of an activated Optical Line Amplifier	12.0 W
$P_{BV\text{-}OXC}$	Power Consumption of an active BV-OXC 2 components: P_{idle} + set of active P_{OLP}	$P_{idle}+\sum P_{OLP}$
P_{idle}	Tied to an active BV-OXC environment (e.g., control, fans, etc.) regardless of the connection services	150.0 W
P_{OLP}	Power Consumption of an optical line board /port (Tx and Rx) integrating WSSs board, optical amplifier, mux/demux	85.0 W

Energy-aware routing algorithm: Devised DRL-trained model

Once the power consumption model is defined, the following tackles the energy-aware routing algorithm problem, i.e. to dynamically establish connectivity services, while simultaneously optimizing resource utilization and minimizing network power usage. This is a complex problem, commonly solved by heuristics. In EON, RSA algorithms are trigged seeking for spatial paths and spectral resources (Frequency Slot, FS) that meet optical flow demands along with fulfilling spectrum continuity and contiguity constraints. RSA strategies can be based on well-known modified K-Shortest Path (KSP) approach. The output yields (up to) K feasible paths, e.g., sorted by the number of hops, each specifying the set of eligible FSs. Next, a First-Fit solution can be used to pick the first available FS to provision the optical flow, known as *KSP-FF*, leading to efficiently utilize the whole optical spectrum. An energy efficient-oriented RSA algorithm extends KSP-FF, which is referred to as *EA-KSP-FF*. This outputs (up to) K paths pursuing network power consumption minimization. That is, for each k^{th} path, the rise of the network power consumption is calculated if the optical flow is provisioned (based on the above energy model). The K paths are sorted in ascending order based on their respective power consumption increase. Thus, EA-KSP-FF favors routing upcoming optical flows through already powered-up/active devices to save power, rather than activating slept down devices. However, this might result in provisioning optical flows over longer paths (in terms of hops and links) to utilize active devices. Consequently, resource (spectrum) utilization is increased, potentially degrading overall network performance.

To attain a better balance between network performance and energy efficiency, an offline-trained Deep Reinforcement Learning agent is proposed (named *EA-DRL*) to offer an optimized policy/model, enabling efficient online energy-aware RSA decisions. The agent, a *Deep Neural Network* (DNN), iteratively updates its parameters across multiple steps/flow requests to maximize cumulative rewards. To do that, a *network state/observation* is presented to the agent, which then selects an *action* from a specific space and receives a *reward*. For each step/optical flow request, a state is constructed, incorporating source and destination nodes, required data rate, and latency, alongside selected features from the K paths generated by the EA-KSP-FF algorithm. These features include end-to-end unused NCF count, average eligible FS size, initial eligible FS position, and current power consumption by existing connections. The

action space is a discrete set, where each action is linked to a candidate k[th] path from the EA-KSP-FF algorithm computation for a given flow request. The reward takes the form of a piecewise function designed to promote both efficient utilization of optical spectrum and energy consumption. Particularly, it consists of two components. Firstly, a normalized average eligible FS size (*sizeFS*) to encourage actions/paths with more available NCFs. Secondly, an inversely normalized value of power consumption increase (*pwPath*) is introduced to prioritize optical flows with lower power consumption. In the event of a failed flow request allocation, due to either lack of spectral resources or inability to meet spectrum constraints, the reward is set to a negative value of -1 to penalize the selected action/path.

$$\boldsymbol{Reward} = \begin{cases} (\boldsymbol{sizeFS} + \mathbf{1}) + \dfrac{\mathbf{1}}{\boldsymbol{pwPath} + \mathbf{1}}, & \boldsymbol{if\ allocation} \\ -\mathbf{1}, & \boldsymbol{otherwise} \end{cases}$$

DRL Training and First Performance Evaluation

The devised and implemented EA-DRL training model follows the workflow depicted in Figure 8.7. The DNN agent approach comprises 3 fully connected layers. An input layer consisted of 23 neurons, encompassing source, destination, requested data rate, along with the 4 features of each of the K= 5 paths. The output layer has five neurons representing the probability of choosing one of the K paths. The hidden layer has 128 neurons. The agent relies on a policy optimizer named maskable Proximal Policy Optimization using a learning rate of 10^{-6} and discount factor of 0.95. It undergoes 10^{6} training steps with episodes length of 5k. The training steps are generated using a Poisson process with inter-arrival time set to 50. The holding time (HT), exponentially modelled, is set to 2500. The requested flow data rates are uniformly distributed within [200, 400, 800] Gb/s, corresponding to FSs within [2, 4, 8] NCFs, respectively.

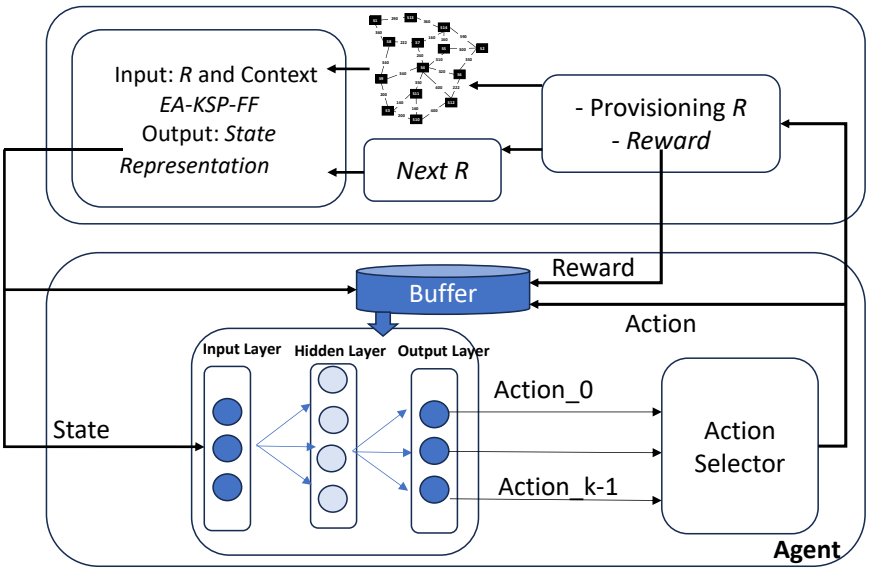


Figure 8.7: EA-DRL Agent Training.

For the numerical results, we have considered the transport EON shown in Figure 8.8. This comprises 14 bandwidth-variable optical cross-connects (ROADMs) and 22 bidirectional optical links with 100 NCFs spaced 6.25GHz. The optical fiber distance (in km) is labeled on each edge between a pair of nodes/ROADMs. In each link OLAs are placed every 80 km.

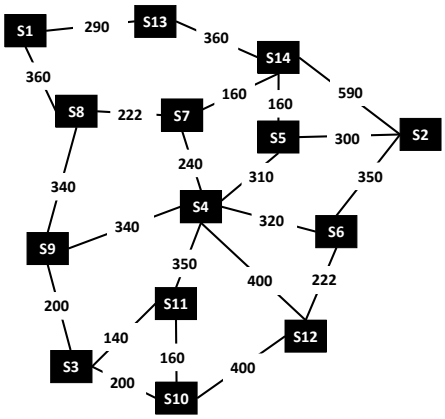


Figure 8.8: EON Topology.

The performance evaluation compares the three approaches, i.e., KSP-FF, EA-KSP-FF, and EA-DRL, across different traffic loads with HT ranging from 1500 to 3500. Each data point is derived from 50k flow requests. Figure 8.9a illustrates the obtained Blocked Bandwidth Ratio for the three RSA approaches. The KSP-FF approach consistently achieves the best (i.e., lowest) BBR performance for all traffic loads, albeit at the cost of performing the worst in terms of the average network power consumption (see Figure 8.9b). The reason is that KSP-FF prioritizes optimizing spectrum resource utilization to accommodate more flows, disregarding power consumption. By adopting the EA-KSP-FF strategy, a notable reduction in power consumption is achieved, particularly under low/moderate traffic loads. Compared to KSP-FF, the EA-KSP-FF approach accomplishes up to 15% of power reduction for HT=1500. Recall that EA-KSP-FF is devised to set up optical flows through active BV-OXCs and links (i.e., OLAs) to save network power. However, this results in longer paths to utilize active devices, increasing spectral usage and complicating spectral constraint fulfillment. The trained EA-DRL model significantly enhances BBR in comparison to EA-KSP-FF at moderate to high traffic loads, reaching an 11% improvement at HT=3000. More noticeable is that applying the EA-DRL model outperforms both KSP-FF and EA-KSP-FF heuristics in terms of network power consumption for all traffic loads. This improvement is achieved as the DRL agent learns optimal strategies for varying network conditions. In essence, EA-DRL optimizes RSA decisions for different network states, striking a balance between network performance and power consumption when dynamically provisioning flows.

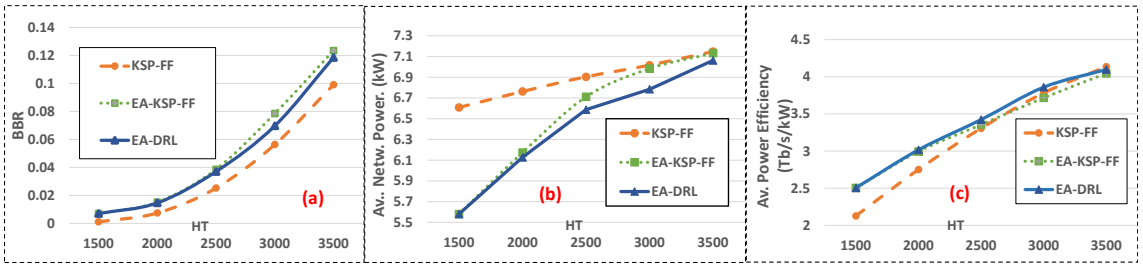


Figure 8.9: Numerical results: (a) BBR; (b) Av. Network Power Consumption (kW); (c) Av. Energy Efficiency (Tb/s/kW).

To illustrate that trade-off, the energy efficiency metric (in Tb/s/kW) is defined as the ratio of average network throughput to power consumption (depicted in Figure 8.9c). A higher value of this metric indicates more energy-efficient RSA performance. The KSP-FF strategy achieves the highest average network throughput, ranging from 14.06 to 29.56 Tb/s, but its high network power consumption negatively impacts the energy efficiency metric. In contrast, while EA-DRL achieves a slightly lower average throughput (13.97 – 28.94 Tb/s), its reduced power consumption results in the best energy efficiency performance. It is important to note that as traffic load increases, more network devices become activated to accommodate optical flows. Consequently, the benefits of adopting energy-aware RSA approaches diminish, and KSP-FF becomes the preferred solution.

This contribution, as mentioned above, is planned to be extended to cover the network power consumption within the OLS and terminal devices supporting diverse operational modes and the multi-band transport infrastructures. Both aspects may lead to increase the complexity when solving the energy-aware routing problem. Therefore, adopting trained DRL policies seems a plausible solution to further enhance the tradeoff between power consumption and network performance when compared to heuristic-based solutions.

8.4 GNN AND DRL FOR ONLINE CONNECTIVITY SERVICES

Orchestration of computing and network resources across multiple technology layers is critical for the autonomous roll-out of network services that fulfill the heterogeneous requirements of vertical industries (e.g., Industry4.0, automotive, etc.). The concept of network slicing enables the flexible and dynamic provisioning of network services which map into low-level computation and networking resources those high-level requirements demanded by vertical services. A network slice/service consists of a set of Virtualized Network Functions (VNFs) that can be distributed geographically in several cloud sites/data centers (DC).

Thanks to its offered high transmission capacity and efficient use of the optical spectrum, Elastic Optical Networks (EON) are the most promising transport network technology for the data center/cloud site interconnections [Tor21]. In this scenario, the dynamic provisioning of network services across DCs interconnected by an EON is a challenging problem, as it requires the effective allocation of computing resources in DCs and spectrum resources on transport EON. Therefore, the provisioning of such services entails the selection and allocation of both

computational and network resources that meet the requirements of the VNFs (i.e., CPU, RAM, storage) and the links (e.g., bandwidth and latency) inter-connecting those VNFs. This problem, commonly referred to as VNF placement, has been notably addressed from different approaches, ranging from optimization problem solving to heuristics and machine learning. Among ML techniques, Deep Reinforcement Learning (DRL) has recently received more attention because learning is performed by direct interaction with the system environment.

In DRL, it is well known that the input state representation and feature extraction are essential for the resulting trained model performance. Simple state representation or manual feature extraction are not suitable for graph-structured data, such as an EON. Graph Neural Networks (GNNs) are a type of neural network dedicated to graph-structured data. GNNs have demonstrated good performance in a wide range of applications, including node classification, graph classification, and link prediction [Xu22].

In this context, this activity aims to exploit GNN and DRL to build ML trained models that assist in the placement of VNF and improve the performance of existing heuristic algorithms and other DRL-based solutions.

For the deployment of VNF-oriented network services, a cloud and transport EON network infrastructure is considered. Considering the scenario described in the previous section, each packet optical node offers computing resources (i.e., CPU, RAM, and storage) to host the VNFs of every incoming network service. Given this scenario, the goal is to dynamically select and provision incoming network service requests fulfilling their requirements, i.e., computing and networking resources (i.e., CPU, RAM, bandwidth, etc.) and satisfying QoS demands (e.g., maximum latency).

GNN-DRL Model for Online Service Provisioning

Our DRL-based solution is designed to manage the deployment of network services in realistic scenarios, where service requests have stochastic arrival and departure and are represented as complex meshed VNF-Forwarding Graphs (VNF_FG). The DRL agent makes the selection of a suitable DC to host the VNF. For each service request, the agent tries to place every VNF into a DC with sufficient unused resources to allocate a VNF. The *observation space,* which is the input for the neural networks of the agent, is composed of the network topology that embeds the information of the DC computational resources and the spectrum usage of the optical links. A GNN takes the topology information as input and extracts features from it. Then, a fully connected neural network uses the GNN output and the spectrum and latency features of the *k* candidate paths between the DCs as input. The input state of the fully connected neural network is complemented by the VNF and virtual link requirements consisting of: the compute demand of the VNF; and the bandwidth and latency requirements of the virtual link. The agent outputs the selected DCs for the VNF placement, which represents the *action space*. For every successful VNF allocation, the agent receives a *reward* equal to the computing capacity allocated. If the whole network service (i.e., VNF-FG) is deployed, the reward is equal to the aggregated computing capacity successfully allocated. This favors the deployment of complete network services. Otherwise, the reward is negative when a VNF cannot be deployed since either

constraints or requirements cannot be met. Figure 8.10 depicts the scheme of the proposed DRL-based solution.

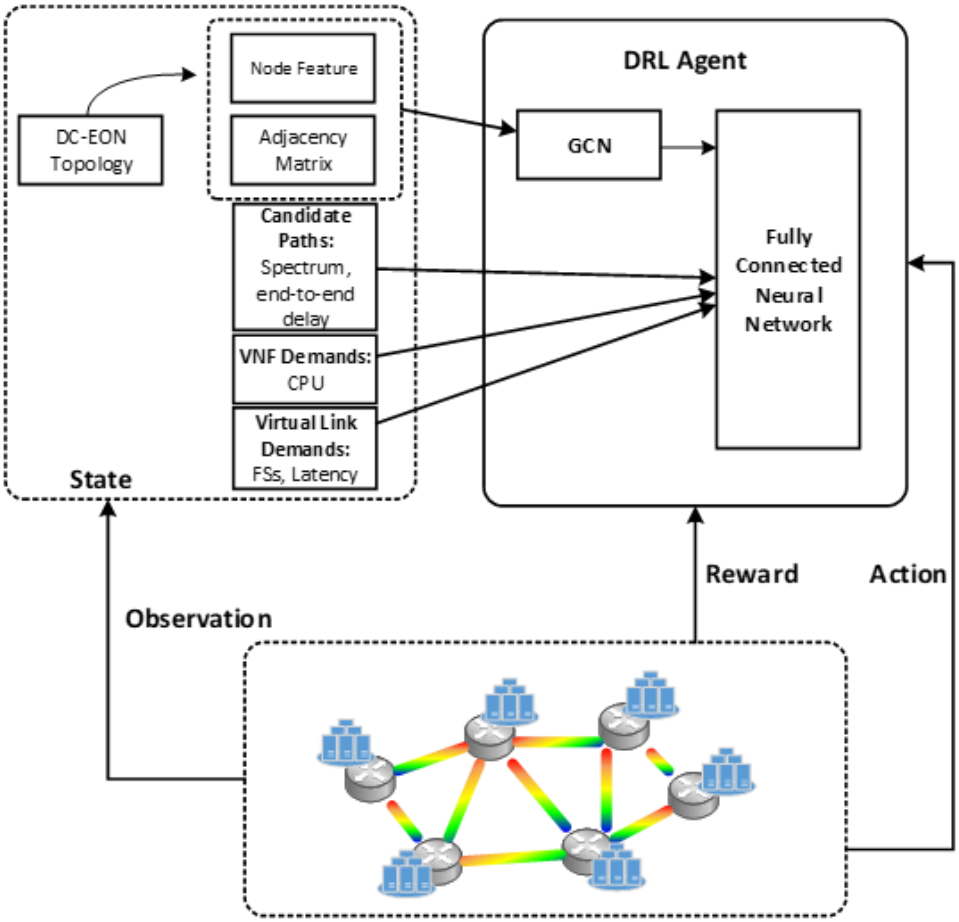


Figure 8.10: GNN DRL-based solution.

Preliminary Performance Evaluation

The simulations were conducted over a network topology consisting of 14 optical nodes and 21 bidirectional fiber links, as shown in Figure 8.11. In this topology, each optical node is attached to a DC with a total CPU capacity of 100 cores for VNF deployment. Each optical link can accommodate 200 FSs. Three different network service types were defined, each with different numbers of VNFs (2, 4, or 6), virtual links (2, 10, or 20), latency constraints (5, 10, or 15 ms), and bandwidth requirements (1, 2, 3 or 4 FSs). The number of contiguous FSs required is determined by the modulation format in use (BPSK, QPSK, 8-QAM or 16-QAM), which in turn is based on the physical distance of the path. Each VNF instance can demand [1, 5, 10] CPU cores. Our simulations assume that the service requests arrive dynamically according to Poisson process, whilst the lifetime of a successfully deployed network service follows an exponential distribution. The inter-arrival time is set to a fixed value and the lifetime is varied to produce different loads. For the training, our DRL agent was implemented using the Proximal Policy Optimization (PPO) algorithm with a learning rate of 10^{-5}, a discount factor of 0.95 and a batch

size of 256. We used a graph convolutional network (GCN) with 14 message passing layers as GNN. This ensures that information from all the nodes that make up the topology is aggregated. The fully connected neural network has 5 hidden layers with 128 neurons. A training episode consists of the provisioning of 5,000 service requests.

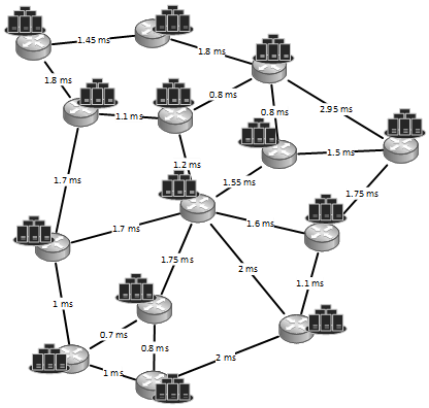
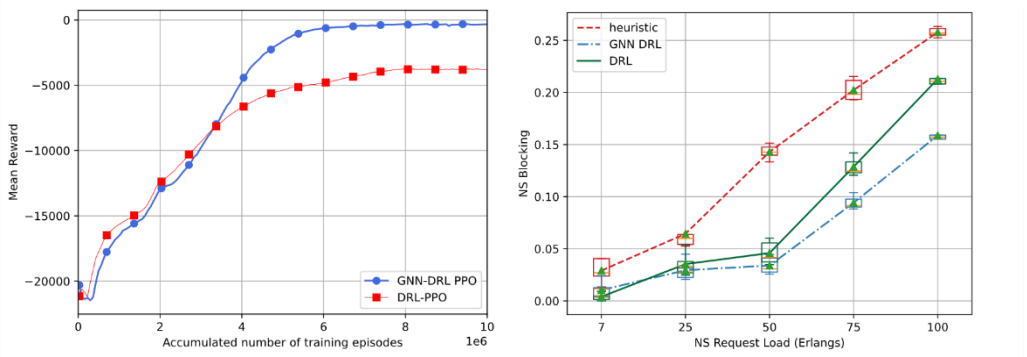


Figure 8.11: DC-EON network topology.

We first evaluate the training performance of our GNN DRL agent and compare it with a previously proposed DRL agent [Her23]. The DRL agent employs a traditional DNN based on linear structures, and thus it cannot operate directly on graph-structured data. This agent relies on hand-crafted feature extraction to build the observation space. Figure 8.12(a) shows the evolution of the mean reward of the episode of both agents. Our GNN DRL solution overcomes the other DRL agent by attaining a higher episode reward and converges faster than the traditional DRL architecture, because it directly operates on graph-structured data.



(a) (b)

Figure 8.12: Network Service Blocking for heuristic, DRL and GNN-DRL DC-EON network topology and Training for DRL and GNN-DRL approaches.

Then, online provisioning of service requests was conducted with the trained model. We compare the performance of our proposed GNN DRL approach with respect to a heuristic algorithm and the DRL agent under different traffic loads. The heuristic benchmark algorithm works by preferring DC with higher spare computing resources, if they meet the latency and bandwidth requirements. Then, the k-shortest path algorithm is used to compute the lightpaths. Figure 8.12(b) presents the service blocking when provisioning 5,000 network services using the

three approaches. This process was repeated 10 times to ensure the statistical accuracy of our results. Note that our GNN DRL agent clearly outperforms (i.e., obtains lower blocking) the other two solutions regardless of the requests load. For instance, under 75 Erlangs of requests load, the GNN DRL attains the lowest blocking rate of 0.09, reducing the blocking rate by 10.8 % and 3.5 % when compared to the heuristic algorithm and the DRL agent, respectively. This improvement is attributed to the GNN's ability to capture a comprehensive view of the network topology, allowing the agent to make more informed and optimal placement decisions.

8.5 Privacy Preserving Digital Twin Knowledge Sharing for Multi-domain Networks

Optical layer digital twin applications generally focus on the operation of single domain networks, where the digital twin has full network visibility. However, future 6G networks are envisioned to support a large number of services with stringent performance spanning multiple domains and therefore, meeting such e2e requirements requires tight coordination among domains. To create e2e models, the different domains supporting an e2e lightpath can share models trained for the intra-domain network. However, distributing such intra-domain models is not secure, as they can include details of the intra-domain network, e.g., the number of hops, the distance of the optical links, or the configuration of optical devices. Note that such information could be of interest to craft specific attacks in case of eavesdropping. Therefore, privacy of the internals of each domain must be enforced when intra-domain models leave the security perimeter of the domain when are shared.

In this section, we assume the OCATA optical time domain digital twin that relies on deep neural networks (DNN) to model the expected effects of optical devices (optical filters and amplifiers) and fibers on in-phase and quadrature (IQ) optical constellations. By concatenating DNNs for the elements in the intra-domain route of a lightpath, expected QoT, such as the pre-forward error correction (pre-FEC) bit error rate (BER), as well as other metrics, can be computed. As intra-domain models are concatenations of DNNs, transformations are proposed to secure sharing intra-domain models used for the modelling of e2e multi-domain lightpaths.

The proposed solution is to create *exportable* intra-domain DNN models that preserve privacy at the required level. Such models are created on-demand, e.g., every time a new inter-domain lightpath is provisioned. Exportable models are built from already trained ones and shared among domains supporting an e2e lightpath in order to build the e2e model. In this work, domain boundaries are defined by the network elements under the control of a single SDN domain controller. We assume that every domain includes an instance of OCATA that is fed with DNN models of the different TPs, ROADMs, and fiber links (referred to as *components*) in the route of a lightpath in the domain (intra-domain route or segment).

Figure 8.13 illustrates an e2e multi-domain network scenario with three domains (labeled *D1*, *D2*, and *D3*). Without loss of generality, we assume that the route between sites *A* and *Z* (in orange color in Figure 8.13 represents either: *i*) a new multi-domain lighpath which e2e QoT

needs to be evaluated during provisioning time; or *ii*) an already established multi-domain lightpath which e2e model needs to be built for computing expected QoT metrics. The main workflow for the e2e model generation is also included in Figure 8.13. OCATA first obtains the intra-domain route segment *p* of the lightpath from the local SDN controller (position (1) in Figure 8.13) and builds a *disaggregated model* η that characterizes the segment by concatenating trained component models (2). Such models propagate the set of features *F* that characterize IQ constellation points as bi-variate Gaussian distributions with its mean position (μ_I and μ_Q), its variance (σ_I and σ_Q), and covariance (σ_{IQ}), i.e., five features per constellation point.

To compose the e2e model of the lightpath, let us assume that source and intermediate domains of the lightpath share intra-domain models with the destination domain through the network orchestrator. Therefore, *intra-domain model synthesis* is carried out in the domains sharing their models to generate exportable segment models φ from the disaggregated ones (3). Models φ hiding the internals of the domain for privacy preserving, are sent to the domain SDN controller (4) and shared with the destination SDN controller through the orchestrator (5). Finally, OCATA in the destination domain generates the e2e model from the received φ models (6-7). Note that the e2e model is a DNN-based model that concatenates the intra-domain models of the domains in the path (φ_{D1} and φ_{D2}), together with the disaggregated model for the local domain (η_{D3}). By propagating features *F* from source to destination, expected IQ constellations can be obtained (8) and used for the selected application.

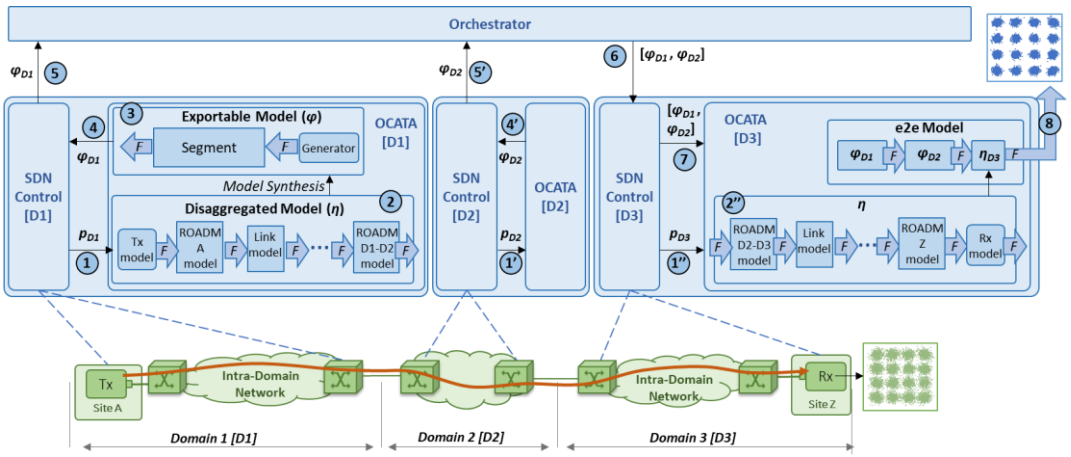


Figure 8.13: Multi-domain scenario and proposed workflow.

Let us detail the procedure carried out by source and intermediate domains of an e2e lightpath to generate exportable intra-domain models φ from disaggregated models η. To this aim, two different sets of component models are available in OCATA. On the one hand, the *symmetric non-linear biased* (SNLB) set contains highly accurate DNNs with: *i*) equal number of neurons per layer; and *ii*) non-linear activation functions (e.g., hyperbolic tangent, tanh) and non-zero bias for every hidden and output neuron. This set is used to build η, which is expected to provide the highest fidelity to model the intra-domain lightpath segment. On the other hand, the *asymmetric linear un-biased* (ALUB) set contains component models where: *i*) each layer has a different number of neurons; and *ii*) the activation function of the first hidden layer and output

layer is linear, and the bias of the first hidden layer and output layer is zero. This set is used to generate an alternative disaggregated model *η'*. Although *η* is more accurate than *η'*, the latter exhibits good properties for security that allow easily hiding the concatenation of consecutive components by merging layers, thus obtaining a layered intra-domain model whose components cannot be isolated.

Finally, an *obfuscation* layer makes it more difficult to get lightpath details by model inspection. This is implemented by randomly shuffling neurons within their layer and weights and biases are truncated to 0 if they are lower than *thr*.

The composition of e2e models for multi-domain lightpaths has been proposed to increase security during model sharing. OCATA builds exportable models for intra-domain lightpaths segments that are shared.

9 GLOSSARY

Acronym	Description
2G	*Second Generation*
3GPP	*Third-Generation Partnership Project*
4G	*Fourth Generation*
5G	*Fifth Generation*
6G	*Sixth Generation*
AI	*Artificial Intelligence*
AP	*Access Point*
API	*Application Programming Interface*
BER	*Bit Error Rate*
BH	*Backhaul*
CD	*Continuous Delivery*
CI	*Continuous Integration*
CEP	*Customer Engagement Platform*
CMIS	*Common Management Interface Specification*
CNF	*Cloud Native Function*
CPU	*Central Processing Unit*
CU	*Centralized Unit*
DD	*Double Density*
DPU	*Data Processing Unit*
DSP	*Digital Signal Processing*
DSR	*Data Signaling Rate*
DT	*Digital Twin*
DU	*Distributed Unit*
DWDM	*Dense Wavelength Division Multiplexing*
EDFA	*Erbium Doped Fiber Amplifier*
EPC	*Evolved Packet Core*
ETSI	*European Telecommunications Standards Institute*
FEC	*Forward Error Correction*
FH	*Fronthaul*
FL	*Federated Learning*
FTTH	*Fiber To The Home*
GE	*Gigabit Ethernet*
GPON	*Gigabit Passive Optical Network*
gRPC	*google Remote Procedure Call*
GTP	*GPRS Tunneling Protocol*
HTTP	*Hyper Text Transport Protocol*
HW	*Hardware*

I/O	*Input/Output*
IaC	*Infrastructure as Code*
IDA	*Intelligent Data Aggregation*
IETF	*Internet Engineering Task Force*
IP	*Internet Protocol*
IPoWDM	*In-phase Quadrature*
ITU-T	*International Telecommunications Union – Telecommunications sector*
KPI	*Key Performance Indicator*
LSTM	*Long Short-Term Memory*
LTE	*Long Term Evolution*
MAS	*Multi-Agent System*
MBoSDM	*Multi-Band over Space Division Multiplexing*
MEC	*Mobile Edge Computing*
MH	*Madhul*
ML	*Machine Learning*
NBI	*Northbound Interface*
NE	*Network Element*
Near-RT	*Near-Real Time*
NETCONF	*Network Configuration protocol*
NetDevOps	*Network Development Operations*
NFV	*Network Function Virtualization*
NGMN	*Next Generation Management Network*
NIC	*Network Interface Card*
Non-RT	*Non-Real Time*
ODU	*Outdoor Unit*
OLS	*Optical Line System*
OLT	*Optical Line Terminal*
ONAP	*Open Network Automation Platform*
ONF	*Optical Networking Foundation*
ONT	*Optical Network Termination*
ONU	*Optical Network Unit*
OPEX	*Operational Expenditure*
O-RAN	*Open RAN*
OSA	*Optical Spectrum Analyzer*
OSM	*Open-Source MANO*
OSNR	*Optical Signal-to-Noise Ratio*
OSS	*Operations Support Systems*
OTN	*Optical Transport Network*
P2MP	*Point-to-Multipoint*
P2P	*Point-to-Point*
PON	*Passive Optical Network*

QAM	*Quadrature Amplitude Modulation*
QoS	*Quality of Service*
QoT	*Quality of Transmission*
RAN	*Radio Access Network*
RIC	*Radio Intelligent Controller*
RL	*Reinforcement Learning*
ROADM	*Reconfigurable Optical Add/Drop Multiplexer*
Rx	*Receiver*
RSCA	*Routing and Spectrum and Core Assignment*
SBI	*Southbound Interface*
SBA	*Service Based Architecture*
SDM	*Space Division Multiplexing*
SDN	*Software Defined Networking*
SLA	*Service Level Agreement*
TAPI	*Transport API*
TFS	*TeraFlowSDN*
VNF	*Virtual Network Function*
WDM	*Wavelength Division Multiplexing*
WSS	*Wavelength Selective Switch*

10 REFERENCES

[Aqu21] Ons Aouedi, Kandaraj Piamrat, Salima Hamma, J K Menuka Perera. Network Traffic Analysis using Machine Learning: an unsupervised approach to understand and slice your network. Annals of Telecommunications - annales des télécommunications, 2021.

[Cas24] Piero Castoldi, Rana Abu Bakar, Andrea Sgambelluri, Juan Jose Vegas Olmos, Francesco Paolucci, and Filippo Cugini, "Programmable Packet-Optical Networks using Data Processing Units (DPUs) with Embedded GPU Monitoring devices", OFC Conf. 2024 [Cas24]

[Cos21] L. Costa, et. al., "Energy Efficiency in Sliceable-Transponder Enabled Elastic Optical Networks", IEEE Trans. Green Commun. Netw., 2021

[Dua12] Q. Duan, Y. Yan and A. V. Vasilakos, "A Survey on Service-Oriented Network Virtualization Toward Convergence of Networking and Cloud Computing," in IEEE Transactions on Network and Service Management, vol. 9, no. 4, pp. 373-392, December 2012, doi: 10.1109/TNSM.2012.113012.120310.

[Dur15] F. Durand, et. Al., "Energy Efficiency Analysis in Adaptive GEC-based Lightpath Elastic Optical Networks", J. of Circuits, Sytems, and Computers, vol. 24, no.9, 2015.

[Elm14] J.M.H. Elmirghani, et. al., "GreenTouch GreenMeter Core Network Power Consumption Models and Results", IEEE (online) GreenComm 2014

[Git22] L. Gitre et al., "Demonstration of Zero-touch Device and L3-VPN Service Management using the TeraFlow Cloud-native SDN Controller," 2022 Optical Fiber Communications Conference and Exhibition (OFC), San Diego, CA, USA, 2022, pp. 1-3.

[Her23] C. Hern´andez-Chulde et al., "DRL for VNF placement in Inter-Data Center Elastic Optical Networks," in 2023 Optical Fiber Communications Conference and Exhibition (OFC), 2023, pp. 1–3.

[Hon16] Z. Hongbo, Y. Longxiang, Z. Qi and J. Shi, "Ubiquitous Information Service Networks and Technology Based on the Convergence of Communications, Computing and Control," in Journal of Communications and Information Networks, vol. 1, no. 1, pp. 98-110, June 2016, doi: 10.11959/j.issn.2096-1081.2016.012.

[Nac21] A. Nacef, M. Bagaa, Y. Aklouf, A. Kaci, D. L. C. Dutra and A. Ksentini, "Self-optimized network: When Machine Learning Meets Optimization," 2021 IEEE Global Communications Conference (GLOBECOM), Madrid, Spain, 2021, pp. 1-6, doi: 10.1109/GLOBECOM46510.2021.9685681.

[Oif22] OIF, "OIF-CMIS-05.2: Common Management Interface Specification (CMIS) Revision 5.2", April 2022

[Oif23] OIF, "IA OIF-C-CMIS-01.3: Implementation Agreement for Coherent CMIS", October 2023

[Onf20] ONF Transport API SDK Version 2.1.3. Available at https://github.com/OpenNetworkingFoundation/TAPI/releases/tag/v2.1.3. 2020.

[Onf21] ONF TR-547: "TAPI Reference Implementation Agreement", Available at "TR-548-TAPI_ReferenceImplementationAgreement-Streaming_v1.1.pdf (opennetworking.org)", Version 1.1. Dec, 2021.

[Ope22] Open XR Forum "OXR.NETMGMT.01.1" Open XR Management Architecture Specification", March 2022

[Ope23] Open XR Forum "OXR.POC.02.1 Dual Management of Open XR pluggable modules in P2MP Applications Hosted in Various Routers with Transmission over Multiple Line Systems", September 2023

[Kal20] (Kaloxylos et al., AI and ML – enablers for beyond 5G networks 2020 5G PPP Technology Board).

[Kar23] V Karunakaran, J Muller, S Zimmermann, A Autenrieth, J Elbers, and T Bauschert, "Digital Twin for Optical Transport Network: Implementation, Comparison and Evaluation", Optica Advanced Photonic Congress (APC), 2023.

[Karu23] V Karunakaran, J Muller, F Slyne, K Kaeval, S Troia, A Autenrieth, D Kilper, T Bauschert, and M Ruffini, "Model-based Service Provisioning in Optical Networks", European Conference on Optical Communication (ECOC), 2023.

[Kyr19] C. Kyriakopoulous, et. al., “Fast Energy-Efficient Design in Elastic Optical Networks Based on Signal Overlap”, IEEE Access, Aug. 2019.

[Mag23] C. Magnouche, et. al., “Safe Routing in·Energy-Aware IP Networks”, in Proc. of Design of Reliable Communication Networks 2023.

[Rat22] G. Rathinavel, N. Muralidhar, N. Ramakrishnan and T. O'Shea, "Efficient Generative Wireless Anomaly Detection for Next Generation Networks," MILCOM 2022 - 2022 IEEE Military Communications Conference (MILCOM), Rockville, MD, USA, 2022, pp. 594-599, doi: 10.1109/MILCOM55135.2022.10017520.

[RFC8348] IETF, “A YANG Data Model for Hardware Management”, March 2018, online: https://datatracker.ietf.org/doc/rfc8348/

[Seq23] D. Sequeira, M. Ruiz, N. Costa, A. Napoli, J. Pedro, and L. Velasco, "OCATA: A Deep Learning-based Digital Twin for the Optical Time Domain," IEEE/OPTICA Journal of Optical Communications and Networking (JOCN), vol. 15, pp. 87-97, 2023.

[Sub21] T. Subramanya and R. Riggio, "Centralized and Federated Learning for Predictive VNF Autoscaling in Multi-Domain 5G Networks and Beyond," in IEEE Transactions on Network and Service Management, vol. 18, no. 1, pp. 63-78, March 2021, doi: 10.1109/TNSM.2021.3050955

[Suk19] N. Sukhija and E. Bautista, "Towards a Framework for Monitoring and Analyzing High Performance Computing Environments Using Kubernetes and Prometheus," 2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), Leicester, UK, 2019, pp. 257-262, doi: 10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00087.

[Tan21] X. Tang et al., "Computing power network: The architecture of convergence of computing and networking towards 6G requirement," in China Communications, vol. 18, no. 2, pp. 175-185, Feb. 2021, doi: 10.23919/JCC.2021.02.011.

[Tor21] M. Tornatore et al., “Guest Editorial Latest Advances in Optical Networks for 5G Communications and beyond,” IEEE J. Sel. Areas Commun., vol. 39, no. 9, pp. 2667–2671, Sep 2021.

[Ts22] “TS 128 105 - V17.0.0 - 5G; Management and orchestration; Artificial Intelligence/ Machine Learning (AI/ML) management (3GPP TS 28.105 version 17.0.0 Release 17)”, Jul, 2022.

[Vil21] R. Vilalta et al., "TeraFlow: Secured Autonomic Traffic Management for a Tera of SDN flows," 2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit), Porto, Portugal, 2021, pp. 377-382, doi: 10.1109/EuCNC/6GSummit51104.2021.9482469.

[Viz12] J. L. Vizcaino, et. Al., “Energy Efficiency Analysis for Dynamic Routing in Optical Transport Networks”, in ICC, 2012

[Xio18] Y. Xiong, et. al., “Lightpath Management in SDN-Based Elastic Optical Networks with Power Consumption Considerations”, JLT, 2018

[Xu22] L. Xu et al., “Deep Reinforcement Learning-Based Routing and Spectrum Assignment of EONs by Exploiting GCN and RNN for Feature Extraction,” JLT, vol. 40, 2022.

[Zha15] J. Zhang, et. al., “Energy-Efficient Traffic Grooming in Sliceable-Transponder-Equipped IP-Over-Elastic Optical Networks”, J. OPT. COMMUN. NETW./VOL. 7, NO. 1/JANUARY 2015.

[Zsm19] “GS ZSM 002 - V1.1.1 - Zero-touch network and Service Management (ZSM); Reference Architecture” 13 Aug. 2019.

[3GP23] "3GPP TS 22.261 V19.5.0 - Service requirements for the 5G system; Stage 1 (Release 19)," 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects, Dec. 2023.